# Elements of Computational Fluid Dynamics

M. Ramakrishna
Department of Aerospace Engineering
Indian Institute of Technology Madras
Chennai 600 036 India

A Golden Jubilee Publication

Published 2011
Elements of Computational Fluid Dynamics
Ramakrishna Mokkapati, PhD
Department of Aerospace Engineering
Indian Institute of Technology Madras
Chennai 600 036

ISBN 978-93-80689-05-0

Revision Number: TAB20160217

To

everyone who has taught me

# Contents

# Preface

Information regarding most topics of interest is available on the Internet. There are numerous wonderful sources of CFD paraphernalia and knowledge. This book sets out to cut a path through this jungle of information. In spirit, it is a journey rather than a systematic exposition of CFDiana. This is not a historical monograph of the author's personal journey. It is more of a guided tour that the author has been conducting into the realm of applied mathematics. The target audience has typically been engineering students. The pre-requisites are calculus and some matrix algebra.

Computational fluid dynamics[CFD] requires proficiency in at least three fields of study: fluid dynamics, programming, and mathematics. The first two you can guess from the name CFD. The last two are languages, the last is a universal language.

This is a book for an introductory course on CFD and, as such, may not be that demanding in terms of fluid mechanics. In fact, it is possible to pitch this material as an introduction to numerical solutions of partial differential equations[**Smi78**]. A good bit of the fluid mechanics used here will be basically one-dimensional flow of gases. There is a lot of CFD that can be learnt from one-dimensional flows, so I would suggest that a review of gas dynamics and the attendant thermodynamics would help, especially in chapter 4.

That was about fluid mechanics, next is programming. Clearly, programing skills are required, though in these days of canned packages one may hope to get away without acquiring the programming skills. However, I feel that one learns best from first-hand experience. So, there are programming assignments in this book, and it would really help the student to be able to write programs. I strongly urge you to do all the assignments in the book. They are not an addendum to the book. They are an integral part of the book. Never ask someone whether something can be done when you can try it out and find out for yourself.

Fluid mechanics and programming done, finally, we come to mathematics. "Acquisitive" is the word that comes to mind when I think of what should be the attitude of a student to mathematics. The following will see you through most of your mathematics requirement as far as the "manipulation" part goes: product rule, chain rule, integration by parts and implicit function theorem.Most importantly, a review of linear algebra [**Str06**][**Kum00**] would help, at least that of matrix algebra.

As we go along, I will indicate material that you may choose to revise before proceeding[**Str86**].

An unsolicited piece of advice on learning a new field: Learn the lingo/jargon. You get a group of people together to work on some problem, after a time everyone knows that the context of any conversation typically will be around that problem. They get tired of describing things all the time. So, they could make a long winded statement like: those lines in the flow field to which the velocity vector is tangent at every point seem to indicate the fluid is flowing in nice smooth layers. They will eventually say: the **streamlines** seem to indicate that the flow is **laminar**. It is important to pick up the jargon. Many text books will supply the jargon in terms of definitions. Other bits of jargon you may have to pick up from the context. Try to understand the definition. Definitely commit the definitions along with the context to memory.

This book is biased. There are times when I try to remove some of my personal bias, for the most part however, it reflects a lot of my opinions on how things should be done. It will not reflect all of the things that are done in CFD. I have however, tried to give enough information for the student to pursue his or her own study.

I hope we lay the foundation for the analysis and intelligent use of any new technique the student encounters. I hope students takes away an understanding of the subject that will allow them to read other material intelligently.

Throughout the book I will try to suggest things that you can do. The objective is not so much to provide answers as it is to provoke questions.

I originally had a chapter on visualisation. I then realised that a chapter at the end of the book on visualisation was not a help to the student. I have distributed the material through the book. Any data that you generate, learn to plot it and visualise it. Try to get a better understanding of what is happening.

It is clear from what I have written so far that I do not, in my mind, see this book as being restricted to providing some skills in CFD. That is the primary objective of the book. However, it is not the only objective. I truly believe that while trying to study the specific one should keep an eye on the big picture.

Finally, there are other books that the student can looks at for a variety of related, advanced and background material[**Ham73**],| and so on.

This is not a comprehensive book on CFD. It is not a reference. It meant to be a journey.

Over the years, students have asked me why they are learning something. I will give an example here. "Why am I learning about the second moment of inertia in statics? What use is it?"

Engineering curricula have been designed by engineers. We have abstracted out bits that are fundamental and invariant and moved them to a point in the curriculum where they can be taught. Sometimes the student wonders why they are learning something. They do not see any immediate application. Teachers then jump to some future course as a justification, buckling of columns is easy to illustrate with a yardstick and can be presented as evidence for the use of the second moment of inertia to a student who has not studied differential equations and eigenvalue problems. Spinning the yardstick about various axes is also useful in the illustration as students can actually perform these experiments to see a correlation with the second moment of inertia. The example, though not satisfactory, at least allows the teacher to move on gracefully rather than bludgeoning the student with authority.

Computational fluid dynamics is, fortunately, an advanced course and I am able to determine the content and sequence in which the material is presented to the student. In my courses, I have tried to arrange the material in a fashion that I create the need before I present the solution. However, some of the strategies that I employ in the class room do not translate well to paper.

This book is meant to be an introductory one. Though it contains some advanced topics, most of the emphasis is on why a few simple ideas actually work. Some of the analysis techniques do not carry over to complex problems that you may encounter later; the questions that these tools answer do carry forth. It is very important indeed to ask the right questions.

We will start chapter 1 with the question: what is CFD? We will expand the idea of modelling and show that we have a need to represent various mathematical entities on the computer.

Having determined this need, we will demonstrate the representation of numbers, arrays, functions and derivatives in chapter 2. We will also answer the question: How good is the representation?

In chapter 3 we will look at some simple problems as a vehicle to introduce some fundamental ideas. Laplace's equation is a good place to start as a confidence builder then we go on to the wave equation and the heat equation. Through these equations we will study the ideas of convergence, consistency and stability of schemes that we develop to solve them.

Having established a foundation in modelling (the student hopefully has written a variety of codes), we look at systems of non-linear equation in chapter 4. We use the one-dimensional flow equations as a vehicle to extend the ideas from chapter 3 to fluid flow equations. A little background in gas dynamics would help. However, in my experience, students without the background have managed. We will look at some details of the algorithms and application of boundary conditions.

Now, up to this point, we have been looking at a hierarchy of problems which we posed in the form of differential equations and boundary conditions. In chapter 5, we look at tensor calculus and derive the governing equations in a general framework. This material is independent of the preceding material and can be read at any time by the student. On the other hand, the first few chapters can also be read independent of chapter 5. Tensor calculus and the derivation of the general form of the governing equations is important for the rest of the book.

Chapter 6 deals with flows in multiple dimensions, techniques of representing the equations and applying the boundary conditions. It also gives an overview of grid generation. We will look at just enough of grid generation so that you can solve problems using grids other than Cartesian coordinates in two-dimensions.

A taste of variational techniques, random walk, multi-grid acceleration and unsteady flows is given in chapter 7. The book ends with a closure in chapter 8. This is meant to round off the discussion started in chapter 1.

There are a few appendices at the end. The first deals in a simple manner with computers and programming. The rest provide the jargon and an exposition of the minimal manipulative skill required for complex variables, matrices and Fourier series.

I would suggest that a student work through the book in a linear fashion. For the teacher there is more choice. I tend to start with Taylor's series, representation of derivatives and quickly get to the iterative solution to Laplace equation. This allows the student to start coding Laplace equation in the first week. I then start each class with a conversation on the numerical solution to Laplace equation make a few suggestions based on the conversation and then proceed with the rest of the material. Sometimes, the conversation ties material to be discussed in the rest of the class: for example, when to stop iterating ties in with machine epsilon and so on. I revisit Laplace equation at various points in the course, when I am dealing with stability, variational methods, multigrid techniques, and finally random walk. It also figures prominently in grid generation.

I need to end this with a vote of thanks. I have been writing bits and pieces of this over the years. I had figures in colour and programs in various programming languages. Most demos now use python. I was, in some vague fashion, thinking of putting this material out for free. Then, IITM as part of its Golden Jubilee celebration came out with a book writing scheme. For a while, I thought it would be published in paper, though deep down inside, I was sure that the electronic medium was the way to go. However, in the duration that it was to be paper, I changed the figures to black and white to cut down the cost of printing.

I thank the institute for the support of a semester for writing this book, this includes colleagues who saw me around and did not get upset for not showing up at the progress meetings of their students. This gave me the push to complete. It also resulted in a lot of topics collapsing into the advanced topics chapter.

I thank my colleague, Prof. Vinita Vasudevan of Electrical Engineering who read through the book atleast twice, M. Manoj Kumar, who slogged through the first three chapters. This was for the early stages of the book. Prof. S. C. Rajan and Prof. Luoyi Tao read through chapter 5 before I put it up. The students in my 2009 Computational Aerodynamics course also sent me some corrections to that chapter. Prof. Joel George gave me corrections and conversation in regard to the first chapter and the appendices. Prof Santanu Ghosh read through parts of chapters on representations, one-dimensional flows and grid generation. My thanks to all of them. Inspite of all of this, my PhD students Mythreya and Siva Prasad found enough to fix. They also whetted the tablet / phablet version. Thanks guys.

The last reading by my students forces me to write: Despite their help, I am sure there are corrections to be made. I sincerely apologize if you have helped me with the book and I failed to acknowledge it here.

I thank my wife for her continuous support and my immediate family and friends that lived with “busy with the book” for a while and then helped out with “Is it out yet?”.

# Introduction

In this chapter, we try to find the motivation for all the things that we do in this book. Further, we will try to lay the foundation on which the rest of this book depends. At the end of the chapter, I hope you will have an idea of what you will get out of this book.

## What is Computational Fluid Dynamics?

We will start by putting computational fluid dynamics, CFD as it is often called, in context. We will try to see what it is and what it is not. Here we go.

An initial attempt at definition may be along the lines: **the use of computers to solve problems in fluid dynamics.** This unfortunately is too broad a definition of CFD. Let us see why. We will first check out what CFD is not.

The general attempt at understanding and then predicting the world around us is achieved by using three tools: experiment, theory and computation. To understand what computational fluid dynamics is not, remember that computation is used along with experiment and theory in numerous ways. Experiments can be automated. Raw experimental data can be reduced to physically meaningful quantities. For example, one may have to do some post-processing to "clean up" the data, like de-noising and so on. Data can be processed to convert many measurements to useful forms: like obtaining streamlines or master curves that capture behaviour so as to aid design. All of this can be done with the help of computers.

Similarly, computers can be used to perform symbolic manipulation for a theoretician. They can be used for visualising closed-form solutions or solving intermediate numerical solutions. So, I repeat, defining computational fluid dynamics as using computers to solve fluid dynamic problems is too broad a definition.

On the other hand, computational fluid dynamics is the use of computer algorithms to predict flow features based on a set of conservation equations. However, you may have encountered computer packages that simulate flow over and through objects. One could classify the use of these packages as experimental computational fluid dynamics (ECFD). After all, the we are using something like a simulated wind tunnel to understand the problem at hand. In order to be good at ECFD, some knowledge of CFD in general and the algorithms used in that package in particular would help. Though, an understanding of the physical principles and fluid mechanics may often suffice.

We have seen some of the things that I would not consider to be CFD. So, what is CFD? We use the conservation laws that govern the universe to build computer models of reality. We want to understand the computer model and would like to predict its behaviour. How faithful is it? How robust is it? How fast is it? All of these questions are asked and answered in the context of fluid dynamics, so the discipline is called Computational Fluid Dynamics. By the end of this chapter, you should have a better understanding of these questions and some answers to them by the end of the book. Let's now take a very broad view of affairs, before we start getting into the nitty-gritty details of modelling/representing things on the computer and predicting their behaviour.

## Modelling the Universe

Modelling the universe is a very broad view indeed. The idea is that we are interested in modelling any aspect of our surroundings. To illustrate the breadth of application and the importance of these models, consider the following scenarios.

**Scenario I**:

It is 3:30 in the afternoon in Chennai. The sea breeze has set in and is blowing in from the east. The residents heave a sigh of relief as they recover from the hot afternoon sun. Leaves flutter in this gentle breeze. Some flap up and down as though nodding in assent. Some sway left to right. If it cools down enough, it may even rain in the night. This is nature at work.

**Scenario II**:

The oil company has been prospecting. They have found oil off the coast and want to pipe it to a refinery north of the city. The crude oil that comes out of the ground is a strange concoction of chemicals. We may lay roads with some of the chemicals, we may drive on the roads using other chemicals. The flow of the oil in the pipeline is clearly very important to the economic well being of a nation.

**Scenario III**:

> "At the mark the time will be $T - 10$ seconds" says the voice over the loud speaker and then proceeds to tick off the count down 9, 8, 7, 6, 5, 4... As the count down proceeds, many scheduled events in the launch sequence take place automatically. The strap-on rocket motors are fired. All of them reach a nominal value of thrust, the main rocket motor is fired. The thrust generated by the hot gases rushing out of the nozzles of all the rocket motors is greater than the weight of the launch vehicle and it lifts off slowly from the pad. There is a gentle breeze from the sea, enough to make the vehicle drift ever so little. This is not a concern, however. The vehicle accelerates quite rapidly as it thrusts its way to its destination of delivering a communication satellite or a remote sensing satellite; all part of the modern way of life.

All of these scenarios involve fluid dynamics. These are all situations where we would like to predict the behaviour of our system. To add a bit of urgency to the state of affairs, we suspect that the second and third scenario may be having an effect on the first one.

We will use the first scenario to indicate that we may have issues of interest that are small in scale or we could have questions that are enormous in scale. In the first scenario, consider one small puzzle. Why do some leaves flap up and down, while the others sway left to right? Call it idle curiosity, but one would like to know. Is the direction of the breeze with reference to the leaf important? One would think so. How about the shape of the leaf? Yes, I would think the shape of the leaf would be a factor. What about the weight of the leaf and the stiffness of the stem to which it is connected? All determining factors and more. The first scenario also has a "problem in the large" embedded in it. Why does the sea breeze set in? Is it possible that it sets in at a later time in the day, making Chennai an uncomfortable city? Can we predict the weather somehow? Can we see the effect our activity on the weather? Can we do anything to avert a disaster or to mitigate the effects of our activity? These are all very relevant questions and "hot" topics of the day.

In the second scenario, fluid mechanics and modelling again play a very big role. The oil field was likely located by using technology similar to echo location. The oil company used a combination of explosions to infer the substructure of the earth and drilled some sample locations for oil. There is still some guess work involved. It is, however, better than wildcat prospecting where you take a random shot at finding oil. Having struck oil or gas, we then come to handling the material. Pumping to storage yards. How large should the pipes be? How much power is required to pump the crude oil? How fast should we pump? Is it possible that the pipeline gets clogged by the constituents of this crude?

Having fetched the crude where we want, we now process it to generate a wide variety of products from fabrics to plastics to fuels. Are there safety related issues that should worry us?

What happens to the reservoir of oil (or the oil field as it is normally called) as we pump out the oil? How do we ensure we maximise the amount that we can get out economically? If we are removing all of this material, what happens to the space left behind? Do we need to fill it up?

The final scenario is one of exploration on the one hand and lifestyle on the other. Access to space, especially low-cost access to space is something for which we are striving. Putting something in orbit was traditionally done using rocket motors. A rocket motor basically consists of a chamber in which the pressure of a working medium is raised and maintained at a high level, say sixty times the atmospheric pressure at sea level. This working medium, which may in fact be made up of the products of combustion that led to the increased pressure, is then vented out through a strategically placed nozzle. A nozzle is a fluid-dynamic device that causes fluid that flows through it to accelerate while its pressure drops. The high pressure acts on the interior of the rocket motor walls. This high pressure results in a net force acting on the rocket motor walls. This is the thrust generated by the rocket. Meanwhile, the launch vehicle powered by this rocket motor is accelerating through the atmosphere. The flow over the vehicle becomes important.

Finally, it takes ages for the carbon in the atmosphere to be trapped by trees. It takes very little time for us to chop the tree down and burn it. The consequences of scenarios II and III on scenario I may be quite severe. Instead of the sea breeze coming in, the sea itself may rise and come in if the polar ice melts and the sea water expands, like all things do as they warm. With this in mind we can now ask the question: How do we develop these models?

## How do we develop models?

Let us look at the process that leads to a model. We observe the world around us. It seems complex and very often unpredictable. Scientist and engineer take a sceptical view of an astrologer[**GU72**], all the same, they would like to predict the behaviour of any physical system. In seeking order in our universe, we try mirroring the universe using simpler models. These models necessarily exclude

(1) things that we do not perceive,
(2) features that we deem unimportant,
(3) phenomena that are too complex for us to handle.

We have no control over the first reason. How can we control something we cannot sense? We can only try to keep our eyes open, looking for effects that we are not able to explain. In this fashion, we can start coming up with ideas and theories that explain the behaviour. We may even employ things that we cannot see; like molecules, atoms, subatomic particles...

The last two reasons for excluding something from our model we handle by making "assumptions". In fact, a feature we deem unimportant is typically an assumption that it is unimportant. We sometimes confuse the reasons for our assumptions. These come in two forms.

- We make assumptions that certain effects are unimportant. These can and should always be checked. They are not as easy to check as you may think. Let us look at a problem that has a parameter in it, say $\epsilon$. Consider a situation, where assuming $\epsilon$ small makes the problem easier to solve. So, we solve the resulting "simpler" problem. If $\epsilon$ is indeed small, the assumption is consistent, but may be self fulfilling. We do not know if it makes physical sense and will actually occur. If $\epsilon$ turns out to be large, our assumption is definitely wrong. (There is of course the more complicated possibility of $\epsilon$ being small when it is considered in the model and turning out to be large when neglected.)

A more subtle kind of an assumption comes from assuming that a small parameter leads to small effects. In this context, the reader should check out the origins of the theory of boundary layers. The tale starts with the assumption that viscosity is very small and the creation of D'Alembert's Paradox and the final resolution with the realisation that small viscosity does not always mean small viscous effects.

- We make assumptions that make our lives easier. Whether we admit it or not, this is the fundamental drive to make assumptions. We can and should try to find out the price of that ease.

There can be a difference of opinion as to whether an assumption is worthwhile or too expensive.

Having made all the necessary assumptions, we come up with a model. We now study this model and try to make sense of the world around us and maybe try to predict its behaviour. If we can analyse a physical system and predict its behaviour, we may be able to synthesise a physical system with a desired behaviour and enter the wonderful world of design.

So, what is the nature of our model? Usually, in an attempt at precision, these models are couched in the language of mathematics. This may sound like a contradiction. After all, we have been harping about the model being approximate. However, we want a precise statement of our approximation so that we know exactly what is included and what is excluded. Also, the mathematical description only prescribes the necessary properties that the phenomenon satisfies. It is up to the individual to infer the behaviour of his/her particular system from the mathematical expression of its behaviour and any other information that may be available.

To understand how this process works, let us consider two examples. At this point, we will not take it all the way to a mathematical model. We will look at the continuum model for a gas first. We will then look at that useful contraption called a nozzle.

**Example I - Air.** We could start by looking at air, which is made up of molecules of various types. To keep life easy, we could assume that air, instead of being a mixture, is made up of only one constituent element with the molecular weight corresponding to the average molecular weight of air. This makes our lives simpler. Now, what are these molecules doing? They could be just moving around colliding with the walls and each other. If we are going to study these collisions, we would further assume that the molecules are hard spheres and that all collisions are elastic.

We pause here to take look at what we have done so far. You can see that we are constantly trying to make things easier for ourselves by throwing away what we consider to be inessential details. All we need now is the initial position of every one of these particles and their velocities and we are all set. We just integrate our usual equations of motion: Newton's laws applied to particles and we can predict the motion of all the particles.

Once we realise that we cannot really keep track of all these hard spheres—there are after all an Avogadro number of them in every mole of gas—we start looking only at statistical quantities like means and variances. These quantities are obtained through the discipline of statistical mechanics. If we are willing to step away from the need of discrete molecules and assume that air is a continuum, then life seems to becomes easier. I say "seems to become easier", as this book is about CFD. When you are done with the book, you will see a certain irony in replacing a large, albeit finite number of molecules with a continuum of uncountable infinity of points. While we are looking at this, ask yourself this question: Is there an approximation involved in assuming we are dealing with a continuum instead of a finite number of molecules? We will address this question at the end of the book.

A few points to note here. When we went from a variety of molecules to one type of molecule, we threw away detail, as we did when we went on to the hard sphere. The internal structure of the sphere is forced on us only when we get to temperatures where that structure becomes significant or worse still, molecules start dissociating into constituent elements. When we get to the continuum from hard spheres, many of the processes that we have ignored must be accounted as properties of the continuum. For example, a mean quantity may be identified as the velocity at a point in the continuum. From this velocity and density at that point, we have an associated kinetic energy density at that point. The kinetic energy density that we calculate this way does not account for all the energy of the molecules. There is still a random motion about this mean "drift" speed. The random motion about the mean may have to be accounted through an associated new variable which is "internal" to our model, called the temperature.

This is what we have managed to do. We have come to a continuum. We have field properties like mass, momentum and energy. We also have their densities associated with every point in that continuum. Now, we can generate equations to help us track the evolution of these properties. We do this by applying the general principles of conservation of mass, conservation of momentum and conservation of energy. The equations associated with these balance laws can be derived in a very general form. We typically use the set of equations known as the Navier-Stokes equations. These equations are derived in chapter 5. We will now look at a typical application.

**Example II - A Nozzle.** We have seen that a series of assumptions led us to the Navier-Stokes equations. These assumptions were quite general in nature. We will proceed with that process in the context of an actual problem and see where that takes us.

Let us consider the problem of the flow through a converging-diverging nozzle. This is a very popular problem and these C-D nozzles, as they are called, are cropping up everywhere. They are used to accelerate fluid from subsonic speeds to supersonic speeds, when employed as nozzles. They will also function as supersonic diffusers and decelerate supersonic flow to subsonic flow. They are used in Venturi meters to measure fluid flow rate and as "Venturis", just to create a low pressure region in a flow field. The fact is, they have been studied in great detail and you may have spent some time with them too. C-D nozzles continue to be of great interest to us.

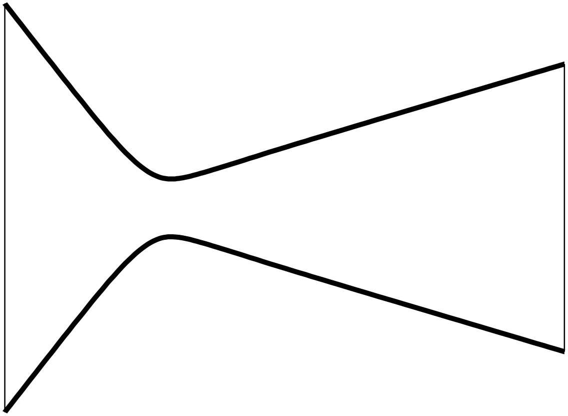

**Figure 1.1. A converging-diverging nozzle**

A cartoon of a C-D nozzle is shown in Figure 1.1. If we now draw a cartoon of the possible realistic flow through this nozzle, we could get something like Figure 1.2. Possible realistic? Well, my experience tells me it could look like this. Anyway, you should go and check out a real nozzle. [1]

[1] As a general exercise, you can see how we model various things and what the actual physical system looks like. As you build up your intuition, it is nice when you expect something and it works out that way. After you have built up a lot of experience, its a lot more fun when it does not work out the way you expected.

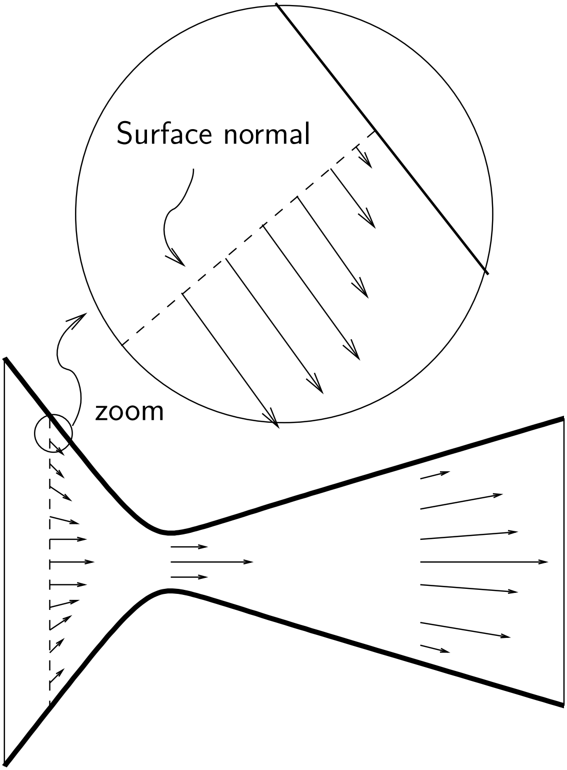

**Figure 1.2. Nozzle with an imagined realistic flow. I say imagined since I just made this up. A zoomed in view of part of the flow field is also shown. In this case the velocity variation along a direction perpendicular to the wall is indicated.**

So, we continue with the process of making assumptions in the "solution" to the flow through a nozzle. In the context of fluid mechanics, for example, the flow of a fluid in the region of interest, may be governed by the Navier-Stokes equations. This is an assumption. ( In fact a whole set of assumptions ) I am not going to list all the assumptions in great detail here because there are too many of them. We will look at a few that are relevant. The governing equations will capture the laws of balance of mass, momentum and energy. Our objective is to predict the parameters of interest to us using these general principles.

The walls of the nozzle are assumed to be solid. In this case, from fluid mechanics, we know the following three statements are valid and equivalent to each other.

(1) The fluid cannot go through the wall.
(2) The velocity component normal to the wall is zero.
(3) The wall is a streamline

This is often referred to as the **no-penetration condition** or the **solid wall condition**.

For a viscous fluid, we would also assume that at the wall, the fluid adheres to the wall. That means that the tangential component of the velocity is also zero near the wall. This is referred to as the **no slip condition**. Figure 1.2 shows such a flow conforming to the wall of the nozzle.

We may simplify our model by assuming the flow to be uni-directional, or quasi-one-dimensional, or one-dimensional flow. Please note that these are three different assumptions.

Figure 1.3 shows a flow that has all the velocity vectors pointed in one direction. The flow is uni-directional. However, it is more than that. A closer examination of the figure shows

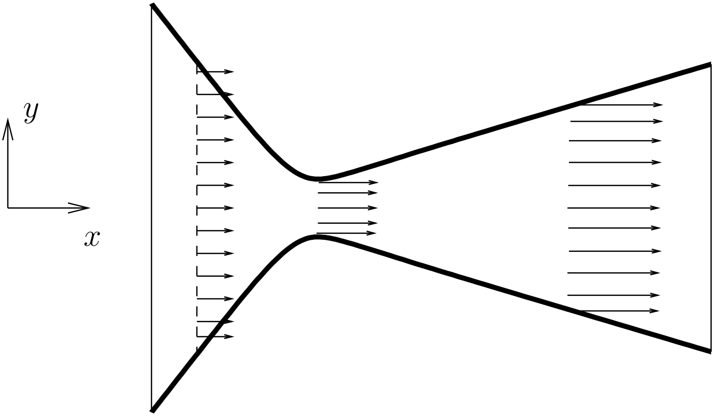

**Figure 1.3. Nozzle flow with the assumption that the flow is quasi–one–dimensional**

that the speed of the flow seems to change along the length of the nozzle. So, one would think that the flow depends only on one dimension. It is directed along only one coordinate direction and therefore should be one-dimensional flow. Why does the speed change? Well, the area changes along the length of the nozzle and that results in a speed change along the length of the nozzle. To differentiate this situation from one where there is no such geometrical change in the other coordinate directions, we call this a quasi-one-dimensional model.

We see that we can make a variety of assumptions that lead to models of different complexity. The validity of our assumptions needs to be determined by us. For instance, in the quasi-one-dimensional assumption, we assume the velocity vectors are pointed along the axis and that the area variation alone contributes to a change in the magnitude of the velocity. Our intuition tells us that the nozzle problem will be better represented by a quasi-one-dimensional model as the area variations get smaller.

So, from these two examples we see that we take a physical system of interest, and we then come up with a model that we hope captures all the important features of this system, at least those in which we are interested. Having come up with a mathematical model, we need to study how it behaves. To this end, we turn to mathematical analysis and the computer. Before we go on, let us summarise what we are up to.

(1) We have the "real" world system. We would like to predict its behaviour.
(2) To this end, we come up with an abstract model. The behaviour of the model, we hope, is the same as that of the real world system. So, we have reduced our original problem to understanding our abstract model.
(3) Since the abstract model may not be amenable to direct prediction of its behaviour, we create a computer model of the abstract model. Again, we hope this computer model captures the behaviour of our abstract model. So, we have reduced our problem to running the computer model.

I repeat, this book is about understanding the computer model and trying to predict some of its more generic behaviour. The things we would like to know are

(1) How faithful is the model? This property is called **fidelity**. You have a recording of a concert hall performance. Is it Hi-Fi, meaning is it a high fidelity recording? Does it reproduce the performance?

(2) How robust is it? Does the model work for simple problems and not work at all for other problems? A model which fails gently is said to be **robust**. This means that the answer you get degrades (it is not as faithful as we like it to be, the model fidelity degrades), however, you still get an answer. Small changes in parameter values do not cause drastic changes to the quality of the model.

(3) How fast is it? Will I get an answer quickly? Meaning: will I get the result in time to use it?

All of these questions about our computer model are asked and answered in the context of fluid dynamics, so the discipline is called CFD.

## Modelling on the Computer

Now that we understand where CFD fits in the big picture, we can focus on the computer model. In order to model something on the computer, we must first be able to represent it on the computer. If we are not able to represent something exactly on the computer, we approximate it. Even though we make an approximation, which may or may not be the best approximation, we shall still refer to it as our representation on the computer or simply the computer representation.

This whole book is about representations, especially that of mathematical entities related to fluid mechanics, on the computer. Somethings we can represent exactly, some not so well.

In the next chapter, we will look at how to represent mathematical ideas on the computer. Before we do that, let us try to see what we need to represent. To answer this, we take a step back to see what we actually want to do.

Computational fluid dynamics falls into the general realm of simulation. The object of simulation is to create an automaton that behaves like the system we are simulating. We would like to create an automaton that can solve the flow past an object or a flow through a system and give us the data that is of interest to us. We may use the data for design or for a better understanding of the behaviour of our models and from there infer/predict the behaviour of our system.

Directly or indirectly, it involves the computational solution to a variety of equations, especially differential equations. Which leads to the immediate question: What does it mean to ask:

"find the solution of a differential equation?"

We will try to answer this question throughout this book. Here is the first shot at it. A differential equation is a description of a function in terms of the derivatives of that function.

A "closed-form"[2] solution is a description of the same function/behaviour in terms of standard functions. A numerical solution could be a description of the function by tabulated data, say.

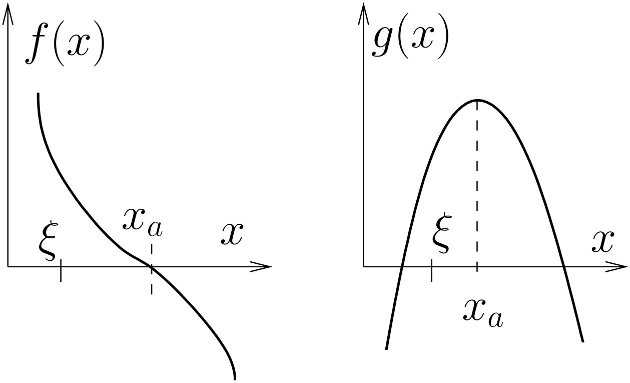

**Figure 1.4. A function $g(x)$ and its derivative $f(x) = g'(x)$. The extremum of the function $g(x)$ occurs at $x_a$. Calculus tells us that $f(x_a) = 0$. So finding the extremum involves finding the zero of $f(x)$.**

What does it mean to find the solution of any equation? We will look at an example that just involves elementary calculus. Suppose you are told that the profit that your company makes is determined by a function $g(x)$ where is $x$ is some parameter that determines operation of your company. You have a simple question that you ask. What is the value for $x$ for which your profit $g(x)$ is a maximum? You have learnt in calculus that given

[2] I refrain from using the term analytic or analytical solutions so as not to confuse with the technical meaning in other disciplines of mathematics

a function $g(x)$, one may try to obtain its maxima or minima by setting the derivative $f(x) = g'(x)$ to zero and solving for $x$. The " $'$ " here indicates differentiation with respect to $x$. As I have picked an example where $g(x)$ is differentiable, we are reduced to finding a solution to the equation $f(x) = 0$. Let us see how this works. Figure 1.4 shows the scenario we just discussed. If I were to give you a $\xi$ and claim that it is an extremum for $g(x)$, how would you test it out? Well, you could find $f(x) = g'(x)$, if that were possible, and then substitute my candidate $\xi$ and see if $f(\xi)$ was indeed zero.

So there you have it. You have a predicate: "$g'(x) = 0?$". This will tell you, given a value for $x$, whether it is a stationary point or not. You just need to come up with ways by which you can generate candidate values for $x$. Most of the problems/assignments that you have encountered so far would have involved finding $x$. Maybe even simpler than the "find the maximum" problem given here. Finding the square root of two can be posed as finding the zero of a function $f(x) = x^2 - 2$. Ultimately, we find an $x \approx 1.414$ which is a zero of this function. There are two points to take away from this simple example.

(1) Given a candidate solution we can substitute back into the equation to determine whether it is a solution or not. This is an important idea and there are times later on in the book where we may have to take a "weaker" stand on the substitute back into the equation part.
(2) The important lesson to take from the simple example is that the problems that we have been solving so far have solutions which are numbers.

Let's study situations that are a little more complicated. We need to find the zeros of functions of the form $f(u, x)$ meaning

$$f(u(x), x) = 0 \tag{1.1}$$

and the objective is to find the function $u(x)$ that satisfies this equation. We could call this a functional equation as we are trying to find a function. For example we may seek a "solution" $u(x)$ such that

$$u^2(x) + x^2 - R^2 = 0 \tag{1.2}$$

A solution[3] to this equation is **a function** given by

$$u(x) = \sqrt{-x^2 + R^2} \tag{1.3}$$

It is important to note here that one is trying to find the function $u(x)$. The equations (1.1), (1.2) are called implicit equations, as the entity we want to determine is embedded in an expression from which it needs to be extracted. As opposed to equation (1.3), which is an explicit equation. The term we seek, in this case $u(x)$, is available on one side of the equals sign and does not appear on the other side. The implicit function theorem tells us when we can extract out the $u(x)$ and write the implicit equation in an explicit form[**CJ04**].

So far we have looked at algebraic equations. Let us now consider a differential equation that is, I hope, familiar to all

[3] We have a situation here that has more than one solution

$$u(x) = \pm\sqrt{-x^2 + R^2}$$

of us: Bernoulli's equation. Consider the streamline shown in Figure 1.5. "$s$" a measure of length along that stream line from some reference point on the streamline. On this streamline, we have for the incompressible flow of an inviscid fluid

$$\textbf{(1.4)} \qquad \frac{1}{\rho}\frac{\mathrm{d}p}{\mathrm{d}s} + \frac{\mathrm{d}}{\mathrm{d}s}\left(\frac{u^2}{2}\right) = 0,$$

where $\rho$ is the density, $p(s)$ is the static pressure and $u(s)$ is the speed at the point $s$. Given $u(s)$, one can actually solve the equation for $p(s)$ as

$$\textbf{(1.5)} \qquad \frac{p(s)}{\rho} = C - \frac{u(s)^2}{2}$$

Equation (1.5) is obtained by integrating equation (1.4). The result is a function $p(s)$. Again, it must be emphasised that the solution we seek is a function and not just a number.

A third problem we will consider is even more explicit in its application. Say you walk from your room/home to the classroom every day. There are many possible routes that you could take. Which is the shortest? Which is the fastest? Which is the safest? There can be any number of "selectors" or predicates that we can invent in order to pick an "optimal" path. Remember, we are picking a path and not some point in our three spatial dimensions. We are picking a path that we may be able to define as a function. Figure 1.6 shows this scenario in two dimensions. It shows three possible paths between the two points A and B. The path we select depends on which property

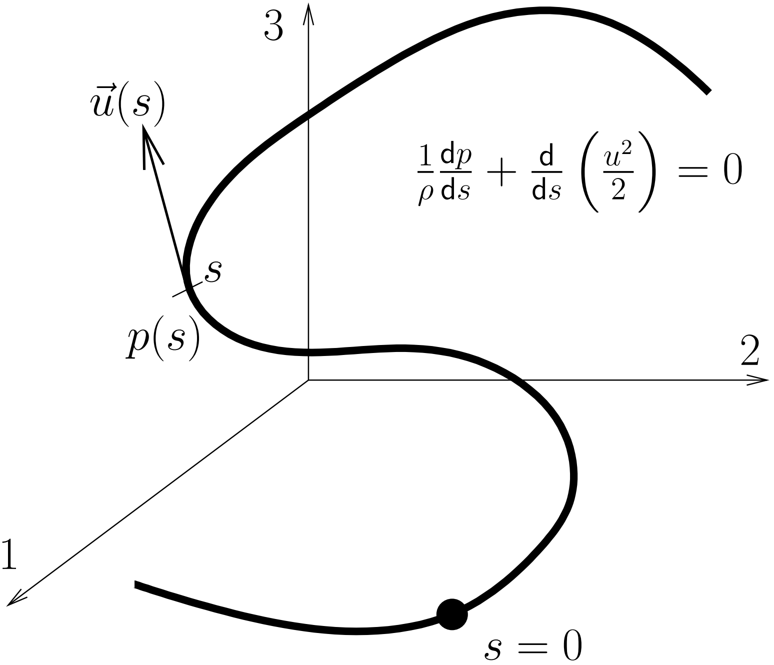
3
$\vec{u}(s)$
$\frac{1}{\rho}\frac{\mathsf{d}p}{\mathsf{d}s} + \frac{\mathsf{d}}{\mathsf{d}s}\left(\frac{u^2}{2}\right) = 0$
$s$
$p(s)$
2
1
$s = 0$

**Figure 1.5.** The segment of a streamline. $s = 0$ is the origin from which we measure length along the streamline. In incompressible flow, the speed of flow is a function of $s$, $u = u(s)$, as is pressure. By the definition of the streamline, the direction of flow is tangent to the streamline and is also a function of $s$

of that path is important to us. However, it is clear from the figure that we are talking about three different functions.

This then is the focus of this book: We are looking for solutions which are functions. We will spend a little time looking

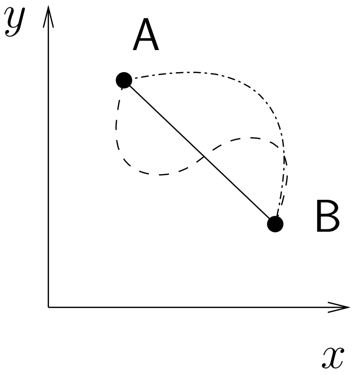

Figure 1.6. Different paths to go from point A to B. Some could be shortest distance, others could represent shortest time or least cost.

at how to organise functions so that we can search in a systematic fashion for solutions. We will generate algorithms that will perform just such a search and study these algorithms on how well they hunt for the solution.

When we do a search, it is clear that we will pick up candidates that are actually not solutions. We check whether a given candidate is a solution to the equation by actually substituting it into the equation. If it satisfies the equation, the candidate is a solution. If it does not satisfy the equation it leaves behind a **residue**. This residue is a signal that we do not have a solution. It can be used effectively in the search algorithms of our computer model.

We see now, when we model physical systems on the computer, very often, we are trying to find solutions to equations.

Whether the solutions are functions or otherwise, we need a mechanism to represent them on the computer. Why should we be able to represent them on a computer? Does it sound like a stupid question? Obviously if you want to solve a problem on the computer you need to represent both the problem and potential solutions on the computer.[4]

This is not quite the whole story. You may have heard of equations having a closed-form solution. Let us find out a little more about closed-form solutions.

What exactly is a closed-form solution? Think of all the functions that you learnt in your earlier classes. You learnt about the functions shown in Table 1.1. You know polynomials. You know of a class of functions called transcendentals: trigonometric, exponentials and logarithms. Typically you know how to construct new functions by taking combinations of these functions. This could be through algebraic operations performed on functions or by compositions of these functions when possible.

Think about it, Table 1.1 about sums it up. Yes, there are a category of functions called special functions that expand on this set a little. You can take combinations and compositions to form new functions. If we are able to find a solution to our problem in terms of these primitive functions, we say that we have a closed-form solution. If I make up a function from these primitive functions, it is likely you will be able to graph it on a sheet of paper (This idea was first proposed by Descartes). You may have difficulty with some of them; $\sin(1/x)$ would be nice to try and plot on the interval $(-\pi, \pi)$. So, there are some functions

[4] sentence included with with permission of a reviewer

| Basic Function | Examples of Combinations |
|---|---|
| Monomials,<br>$f(x) = ax^n$ | $g(x) = ax^2 + bx + c$,<br>$h(x) = \dfrac{ax^2 + bx + c}{dx^2 + ex + f}$ |
| Transcendentals,<br>$f(x) = \sin x, \cos x, e^x, \log x$ | $g(x) = e^{-x} \sin x$,<br>$h(x) = \log(\lvert \cos x \rvert)$ |

**Table 1.1. Table of basic functions and sample combinations to generate new functions**

that we can write down which are difficult to graph. How about the other way around. Does every graph that you can sketch have a closed-form representation? No!! That is the point. Look at Figure 1.6 again. Can every possible path from your home to the classroom be represented in combinations of these simple functions? Maybe, it is possible to use a combination of an infinite number of these functions and approximate the graph or the path.

This difficulty exists not just for a graph. Now consider a differential equation. It too is a description of some function for which we seek a simpler description. The simplest differential equation with which most of us are familiar is something of the form

**(1.6)**
$$\frac{dy}{dx} = f(x), \quad x = a \Rightarrow y = c$$

This a fairly simple equation. You think it is easy to integrate? How about

**(1.7)**
$$f(x) = \exp(-x^2) \ ?$$

Even the simplest differential equation that we write may not have a closed-form solution. The definite integral of the function given in equation (1.7) is extremely important in many fields of study and is available tabulated in handbooks. The indefinite integral does not have a closed-form.

"Okay, big deal, there is no closed-form solution." you say. Why do you want a closed-form solution? Closed-form solutions very often give us a better handle on the solution. We can perform asymptotic analysis to find out how the solution behaves in extreme conditions. We can use standard calculus techniques to answer a lot of questions regarding maxima, minima, zeros and so on. Lastly, closed-form solutions are great to check out computer models. More about this later. So, we would like to have closed-form solutions. Failing which, we would like to represent the solution on the computer.

Many fluid flow models that we use involve differential equations and finding a solution is basically integration. Before we go further, we will take a closer look at the process of integration. Integration is more difficult to do than differentiation. You most probably knew that already. Why is this so? We have a direct method by which we get the derivative at a point through its definition. Bearing in mind that we are talking of a closed-form expression, the process of differentiation is in fact pretty easy to automate. Integration is the inverse of this process.[5] This means that in order to integrate a function, we come up with a candidate integral and differentiate it. If the derivative turns out to be right, we have the integral, otherwise we try again. The process of integration fundamentally involves guessing. Either we are good at guessing or we look up tabulations of guesses of others. So, again, we need to be able to hunt in a systematic fashion for these functions.

This brings us back to our original **need**: The equations that govern the behaviour of our systems typically have functions as their solution. We need a mechanism to represent functions on the computer and algorithms to hunt for solutions in a systematic fashion. Please note, there are many a response that will answer this need. We will look at one class of answers.

First, we will see how to represent numbers on the computer. Normally, in calculus we construct the real line so as to solve problems like "find the zero of the function $f(x) = x^2 - 2$". Here, we will start at the same point and then we can get on with vectors, matrices and functions. We will go through this exercise in chapter 2.

## Important ideas from this chapter

- We make assumptions to generate models that capture what we want or can of the actual real world system: Always check your assumptions. This is the general

[5] Integration predates differentiation by millennia, however that it is the anti-derivative has been an extremely useful property and discovered much later.

idea of abstraction. We remove the nonessential (or assumed nonessential) and leave only the abstract model.

- Our models are mathematical and the solutions we seek here are likely to be functions and not just numbers. We do not have a whole host of functions that we can use to get closed-form solutions and therefore have to consider, carefully, how we are going to represent functions on the computer.
- Finding the solution to a differential equation is like integration. Integration is a process of guessing at a function whose derivative would be the function at hand.
- Finally, remember, that if you are fortunate to have your computer model agree with your experiments, it may be that the errors in both of them have gone in the same direction. A pinch of scepticism is always good.

We started this chapter talking of the sea breeze and pumping oil. By the time you are done with this book, the hope is that you will have an idea as to how to model them. At the least, you should have an idea of some of the difficulties that you will encounter.

# Representations on the Computer

> **There are 10 kinds of people in this world: those who understand binary, and those who don't -** Anonymous[1]

> **... there is nothing either good or bad, but thinking makes it so...** - Hamlet

We have seen that very often, we are forced to build models on the computer. Ultimately, we hope that these models will represent the real world. A study of a typical model will show that it is made of parts. These parts need to be represented on the computer so that the model can be assembled from the representation of the parts. Where we cannot represent something exactly, we try to approximate it.

This chapter will provide the preliminaries in representing / approximating things on the computer. We could address the questions:

(1) How accurate is a certain representation?
(2) How efficient is the representation in the use of computer resources?

[1] One could go on – There are 10 kinds of people in this world: those who understand ternary, those who don't, and those who mistake it for binary...

(3) How fast can the particular representation be processed? or how efficient is our model in the use of this representation?

Somethings that we represent on the computer can be viewed as "data". This data may be represented in many different ways as we shall see soon. For this reason, the particular type of representation is called a "data structure". We use algorithms to process these data structures, resulting in our computer models.

In the context of CFD, our models are often made up of differential equations, integral equations and integro–differential equations. We want to represent our differential equations on the computer. We have seen that the solutions to these equations are functions. These functions need to be represented on the computer. This is our objective. We will start small, with a simple mathematical entity: the number.

## Representing Numbers on the Computer

As was pointed out earlier, the whole book is about representing things on the computer. We will do this in a hierarchical fashion. We will begin with simple mathematical entities and ideas. We will then use these simple entities to represent more complex mathematical entities and so on. To start with, we are interested in representing the real line on the computer[**Gol91**]. For a quick overview of computers see the Appendix A[**HP03**].

We have designed computers to use Binary digits (**bit**) to represent things including numbers. A single binary digit has two states. We could use these two states to represent a zero or a one. We can of course use this bit to represent other ideas and states like (off, on), (black, white), and so on; anything that can be captured by just two states.

If we consider two binary digits, we have the combination $00, 01, 10, 11$, which gives us $2^2$ possible states. Four bits, sometimes called a nibble, can represent sixteen possible states. As an example, this is enough for a four function calculator: ten numerals, four operations $(+, -, \times, \div)$, and two keys for clearing and obtaining the result of a calculation. We get a sense of the power that flows from our ability to represent things on the computer.

If we were to use 32 bits, we could represent $2^{32}$ different symbols. The easiest would be whole numbers. So, we could simply count from 0 to 4294967295. Or, We could represent integers from $-2147483648$ to 2147483647. That's a lot of counting. However, if you needed larger integers then you would need to employ more bits. 64 bits will get you almost 20 digits.[2]

**Note**: We can count. We can count to large numbers. We can't count to infinity!

Nice conclusion, nice words. We will always have only a finite number of symbols that we can use to represent things. The problem: if we cannot even represent all integers, how are we going to represent the real line? We will do the best we can and try to approximate it. The real line is made up of rational and irrational numbers. There are an uncountable infinity of irrational numbers. We will thank Cantor, Dedekind [**Ded63**], Dirichlet, and Weierstrass and all the mathematicians that helped construct the idea of a continuum and abandon

[2] It really is worthwhile remembering the powers of 2

irrational numbers. After all, they guarantee that we will find a rational to approximate any irrational to whatever degree of accuracy that we choose. Consequently, we try to represent only rationals as best as we can.

There are a countable infinity of rationals, the same as the set of integers. Just as in the case of whole numbers, we are not going to be able to represent all of the rationals. In fact, the way we have represented integers gives us an idea as to how we could try to represent fractions. If we decide and are able to scale our problem so that all the computations we perform are likely to generate numbers in the open interval $(-1, 1)$, we could take our integer representation and imagine in our minds that there is a decimal point placed to the left of the integer. That is, we have a mantissa made up of about 9 digits. This is called **fixed-point** arithmetic since the location of the decimal point is fixed. At first glance this may look like a good scheme. There are many applications where people have made very clever use of fixed point arithmetic. The problem is that different numbers on the interval $(-1, 1)$ will have a different accuracy to which they are represented. For instance, a largish fixed-point number may be $0.5$. If we consider in comparison $1.57 \times 10^{-10}$, we may be only able to represent $0.0000000001$. We realise that this problem occurs because the decimal point is fixed and we do not have an exponent to shift it around. The alternative is to go for something known as **floating-point** arithmetic. We will now look at floating-point representation.

Again, we remind ourselves that we will confine our attempts to representing rationals and that there are a countable infinity of rational numbers. We look for some guiding principles to help us decide which numbers to keep and which to throw away. We keep $0$ and $\pm\infty$, symbols that are an anchor for the construction of the real line. We throw away everything between some yet to be determined largest number $L$ and $\infty$. So, we have something that looks like

$$\{-\infty, -L, \ldots, 0, \ldots, L, \infty\}.$$

With one swipe we eliminated an infinity of rationals. Of course, the slippery part of infinity is that we still have an infinite number of rationals from which to choose, leaving us with only one question. Now, how do we distribute the numbers that we choose to represent between $[-L, L]$? It may strike you that one may as well distribute them uniformly. It turns out most people like to know what's happening around zero. So, we pack more points close to zero. In the bad old days (good old days?) everybody represented numbers in their own way. Now we have standardised this representation on the computer. It is called the IEEE754 standard. A lot of CPU (Central Processing Unit) manufacturers conform to this standard. In the IEEE754, 32 bits are allocated to a **floating-point** number as follows:

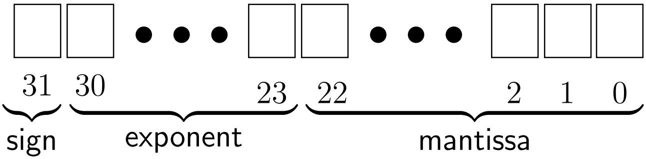

**Figure 2.1. Allocation of bits to various parts of number in the IEEE754 little-endian standard. Each box represents a bit. Notice that the count starts at zero.**

In Figure 2.1, we see that the last bit, bit 31, is indicated as the sign bit. This determines whether the number is positive or negative. This leaves $2^{31}$ symbols to represent any unsigned number. The mantissa is allocated 23 bits. Since, we will always adjust the exponent so that there are no leading zeros in the mantissa, the first digit of the mantissa will, under the normal circumstances, be 1. In which case, we might as well assume the first digit is a 1 without actually storing it. This gives us an effective mantissa of 24 bits indicating that we can have a resolution of 1 in $2^{24}$. The exponent (bits 23 to 30) determines the total dynamic range of the numbers that we can represent. Quite often, this may just not be enough for our computation. We may need better resolution or a greater dynamic range. The IEEE754 also defines a double precision representation. Here, as you would have guessed from the name, we use 64 bits to represent a floating-point number. The mantissa is 53 bits. Finally, the IEEE854 defines a quad precision representation. This may or may not be supported by the particular computer on which you are working. You can find out more about it.

You have some information on how numbers are represented on the computer. We have not seen all the details. This is just enough to get on with our investigations. Due to various constraints, we have the representation of “floating-point” numbers on the computer as given by the IEEE754 standard. We need to understand what we have actually achieved going through this whole process. We will do this by looking at something we call the epsilon of the machine.

**Machine Epsilon.** The epsilon of a floating-point data type on a computing platform is defined as follows.

It is the smallest instance of that data type such that

$$1.0 + \epsilon_m \neq 1.0 \tag{2.1}$$

on that particular machine. It can also be defined as the largest instance of the data type such that

$$1.0 + \epsilon_m = 1.0 \tag{2.2}$$

on that particular machine. These two definitions will not give us the same value. They will differ by a tiny amount. (can you guess by how much they will differ?) We will use the definition given in equation (2.1).

With hardware becoming compliant with the IEEE standards the $\epsilon_m$'s are gradually turning out to be the same across vendors. However, with optimising software and hardware architectures it requires more effort to actually measure the $\epsilon_m$.

Before you do the assignment, Given what we have seen about representing numbers using the IEEE754, can you guess what $\epsilon_m$ should be?

---

**Assignment 2.1**

Here is a simple algorithm to evaluate the $\epsilon_m$. Implement it in your favourite programming language. Try it for single precision, double precision and long double if it is available. Are you surprised by the results? Fix the problem if any.

```
Candidate_$\epsilon_m$ = 1.0
while 1.0 + Candidate_$\epsilon_m$ $\neq$ 1.0
    Candidate_$\epsilon_m$ = Candidate_$\epsilon_m$ * 0.5
endwhile
print "epsilon_m = ", Candidate_$\epsilon_m$
```

---

Did you get the $\epsilon_m$ the same for float and double? A float will typically occupy four bytes in the memory. A double will occupy eight bytes in memory. Now, computers usually have temporary memory called registers with which they perform their mathematical operations. So,

Candidate_$\epsilon_m$ = Candidate_$\epsilon_m$ * 0.5

will result in Candidate_$\epsilon_m$ and 0.5 being stored in two registers. A multiply operation is then performed employing the contents of this register and should in theory be stored back in memory as Candidate_$\epsilon_m$. The registers are usually at least as a large as the double precision variable, that is eight bytes in size. So, even the single precision computations are performed in double precision. However, the minute the Candidate_$\epsilon_m$ is stored in memory it goes back to being a single precision value since you have set aside only four bytes for it.

Here is one other test that you can try out. What happens if the test is $1.0 - \epsilon_m \neq 1.0$ in equation (2.1), that is we use a minus instead of a plus. You can rework the assignment 2.1 with this test.

What can we conclude from the definition of $\epsilon_m$ and the study we have made so far? It is clear that any number that lies in the interval $(1 - \frac{1}{2}\epsilon_m, 1 + \epsilon_m)$ will be represented on the computer by 1.0. So, here is the fundamental point to be remembered when dealing with floating point numbers on the computer. Each number represents an interval. This is shown in Figure 2.2.

The arithmetic we do on the computer is interval arithmetic. If some computation generates a number $b$, which falls into the

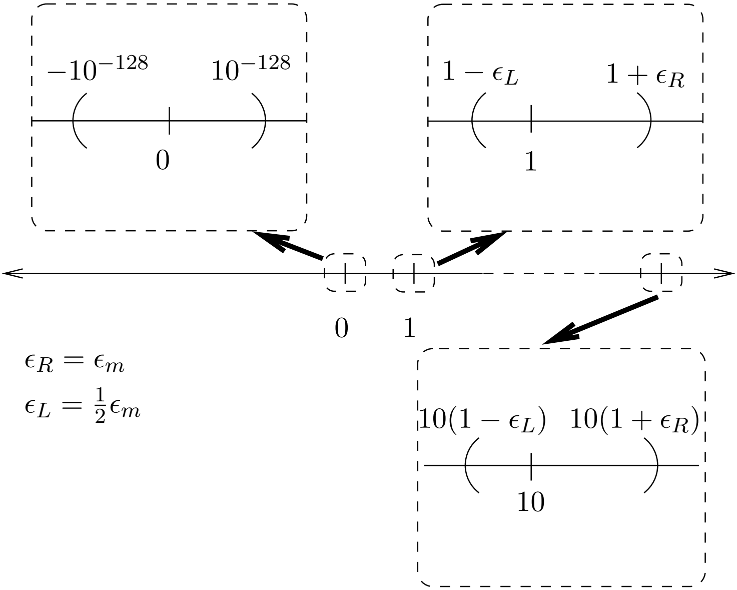

**Figure 2.2. The real line and some sample points on it that are represented on the computer along with the intervals that they represent. Zero actually represents a smaller interval than shown in the figure if we allow leading zeros in the mantissa. As the points are clustered towards the origin, given a representation $r$, the next representation to the right is $s$ away and the next available representation to the left is $\frac{1}{2}s$ to the left**

interval represented by the number $a$, then that number will become $a$ on the computer. The difference $r = b - a$ is the roundoff error.

**Definition**:

The difference between a number and its representation on the computer is called **roundoff error** in that representation.

---

### Assignment 2.2

Redo assignment 2.1. Find the epsilon of the machine using the test

(1) $1. + \epsilon_m \neq 1.$
(2) $10. + \epsilon \neq 10.$, and values other than 10 that you may fancy.

and generating a new candidate $\epsilon$ using

(1) $\epsilon = 0.5 * \epsilon$
(2) $\epsilon = 0.1 * \epsilon$, and any values other than 0.1 that you may fancy.

---

How does roundoff error affect us and why should we be worried about it? First, it is clear from the definition and everything we have seen so far that the act of representing a number on the computer with finite precision is very likely to result in roundoff error. If you combine two numbers with roundoff error in any particular arithmetic operation, you may have a significant error in the result which is greater than the original roundoff error due to the representation of the two numbers. That is, if $c = a + b$ the error in $c$ may be much more than the roundoff error in $a$ and $b$. Unfortunately, colloquially, this error in $c$ is often referred to as roundoff error. We will call it the **cumulative roundoff error** or accumulated roundoff error.

Every numerical computation we perform, we are actually doing operations between intervals and not numbers [**Ded63**], [**Moo66**]. On the computer, we start with an uncertainty when we represent our number. As we perform operations upon these intervals... if the intervals grow, our uncertainty grows. We should be concerned that the roundoff errors in our computations may accumulate over numerous such operations.

We define **cumulative roundoff error** as the net error in a numerical value which is the result of a sequence of arithmetic operations on numbers that may have either roundoff error or cumulative roundoff error.

Before we go on with our discussion we will look at cumulative roundoff error a little more closely as a justification for the remarks that we have just made. It seems that cumulative roundoff error is an accumulation of roundoff errors. Is it just a matter of a lot of $\epsilon$'s accumulating? Actually, it can be worse than that.

**Case 1**: Consider the difference between two positive numbers that are very close in value. This is the whole point of the $\epsilon$ calculation. We try to compute $1 + 2\epsilon - 1$. If all the digits in our original number are significant, how many significant digits do we have now? Very often not all the digits are significant digits. We happen to have a number $A$ which is $1 + 2\epsilon$ which we have arrived upon after many calculations. It is possible that it should actually have been $1$. Maybe the actual answer to $A - 1$ should have been zero and not $2\epsilon$.

**Case 2**: There is a second kind of a problem that can occur. Bear in mind our experience with polynomial algebra. We can add the coefficients of two terms only when the terms are the same degree in the variable. That is $3x^2 + 2x^2 = (3 + 2)x^2 = 5x^2$. The same holds when we do floating-point arithmetic on the computer. After all a floating-point number on the computer is really a signed mantissa coefficient multiplying $x^n$, where $n$ is the exponent part of the floating-point number and in the binary system $x = 2$. So, when we combine two numbers with addition or subtraction, we need to make sure that the exponent $n$ is the same for the two numbers before we actually perform the operation. Which means, one of the mantissas will have as many leading zeros added to it as required to make the exponents equal. So, in performing this operation we have fewer significant digits. In fact, you could possibly have none at all! You may protest saying "this is why we dropped fixed-point arithmetic in the first place". This is not quite right. If we had a situation where, $n = -20$ for the first number and -25 for the second, fixed-point arithmetic will not be able to represent either, especially if the nominally expected values are near one. In the case of floating-point arithmetic, we have possibly a more accurate representation for both the numbers, and one of them looses accuracy when combined with the other in an addition or subtraction.

**Case 3**: The two cases we have seen just now deal mainly with errors caused by the limited mantissa size. There is one more situation where the mantissa limit will haunt us. We may encounter numbers where the exponent cannot be adjusted and we are essentially stuck with fixed-point arithmetic in the representation of the numbers.

If you have had a course in probability and statistics you can have a little fun playing around with this. Most programming languages will allow you to roundoff numbers to a certain number of decimal places. For example **round**$(x, r)$ may round $x$ to $r$ places. We will use this to perform an experiment[**KPS97**].

**Assignment 2.3**

First let us cook up some iterative scheme to generate a sequence of numbers $\{x_0, x_1, x_2, \ldots, x_n, \ldots\}$.

**(2.3)**
$$x_{n+1} = \alpha x_n, \quad \alpha = 1.01, \quad x_0 = 1$$

Let us decide to round to four places, that is, $r = 4$. Call the rounded number $\tilde{x}_n$. Then the roundoff error that we have made is

**(2.4)**
$$\mathcal{E} = x_n - \tilde{x}_n$$

What are the extreme values (the maximum and minimum values) that $\mathcal{E}$ can take? Plot a histogram to get an idea of the distribution of roundoff error.

Did the histogram look about uniform? That is not good news. It is not bad news either. We would have been happy if the histogram had a central peak and died off quickly since that would have meant that our roundoff error is not as bad as we had feared. On the other hand it could have been bad news and the histogram could have had a dip in the middle with most of the results having the largest possible error. We can live with a uniform distribution. In fact this is the basis of generating pseudo random numbers on the computer.

We have seen two possible structures or schemes by which we can represent numbers. Generally, floating-point arithmetic tends to be more resource hungry. However, it is very popular and it is the representation that we will use routinely. Now, we go on to look at representing other data, which are made up of multiple numbers. The preferred data structure in mathematics for this entity is a matrix. We will look at the representation of matrices and multidimensional arrays in the next section.

## Representing Matrices and Arrays on the Computer

Matrices are easy to represent on computers. Many of the programming languages that you will learn will allow you to represent matrices and you should find out how to do this. Some programming languages may call them arrays. You should be aware that the programming language may not directly support matrix algebra. We just need to implement the necessary matrix algebra or use a matrix algebra library and we should be done. Into this happy world we now shower a little rain.

Your favourite programming language may allow you to represent a square matrix. However, the memory in the computer where this matrix is actually "stored" seems linear [3]. Meaning,

[3] Actually memory is very often organised at the chip level as a two-dimensional array which makes it possible to reduce the number of physical electrical connections to the chip. This is a very serious cost consideration. This reduction happens as the address of any memory location can now be split into two: a row address and a column address. So, one can use half the number of

we can only store vectors. Now, how does your programming language manage this trick of storing something that is two-dimensional in memory that is one-dimensional? Simple, store the two-dimensional arrays as a set of one-dimensional arrays. Do we store them by rows or by columns? They make an arbitrary decision: “Store the matrix one row after another as a vector”. Remember now that someone else can make the decision (and they did) that they will “store the matrix one column after another”. There are two reasons why we should worry.

- A lot of the mathematics that we have learnt is more easily applicable if we actually store the data as vectors. This will allow us to naturally translate the mathematics into our data structure and the algorithm.
- If we do our matrix operations row-wise and it is stored column-wise we may (and often do) pay a performance penalty.

Instead of using the word “matrix”, which has many other properties associated with it, we will use the term “array”. This is also to convey to the reader that we are talking of how the object is stored. Arrays come in different flavours. A one-dimensional array of size N made up of floating-point numbers can be indexed using one subscript: $a[i]$. These can be thought of as being written out in a row

$$a[1], a[2], a[3], \cdots, a[i], \cdots, a[n] \tag{2.5}$$

pins and pass it two pieces of data corresponding to the row address and column address one after the other.

They are stored as such on most computers. On the other hand, if we are interested in a two-dimensional array then we have

**(2.6)**

$$\begin{pmatrix} a[1,1] & a[1,2] & \cdots & a[1,j] & \cdots & a[1,m] \\ a[2,1] & a[2,2] & \cdots & a[2,j] & \cdots & a[2,m] \\ \vdots & \vdots & \ddots & \cdots & & \vdots \\ a[i,1] & a[i,2] & \cdots & \boxed{\text{a[i,j]}} & \cdots & a[i,m] \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ a[n,1] & a[n,2] & \cdots & a[n,j] & \cdots & a[n,m] \end{pmatrix}$$

Here, we have laid out the two-dimensional array in two dimensions with two subscripts $i$ and $j$. Now, most programming languages will allow you to use two subscripts to index into the array. One can continue to use this. However, we have already seen that the memory in which it is actually to be stored is typically one-dimensional in nature. Poor performance may result if one were to access the array contrary to the order in which it is stored. Let us see how storing a multi-dimensional array as a one-dimensional array works. If we were to lay a two-dimensional array out row by row and employ only one subscript $p$ to index the resulting one-dimensional array we would have

**(2.7)**

$$a[1,1], a[1,2], \cdots, a[1,m], a[2,1], a[2,2], \cdots, a[2,m], \cdots, a[i,j],$$

Since it is now a one-dimensional array we use one subscript $p$ to index into this array as follows

**(2.8)**
$$a[1], a[2], \cdots, a[p], \cdots, a[nm].$$

In a similar fashion we can rewrite the same one-dimensional array with a single subscript in the form of a matrix as shown below. This allows us to relate the two sets of subscripts to each other.

**(2.9)**
$$\begin{pmatrix} a[1] & a[2] & \cdots & a[j] & \cdots & a[m] \\ a[m+1] & a[m+2] & \cdots & a[m+j] & \cdots & a[2m] \\ \vdots & \vdots & \ddots & \cdots & & \vdots \\ a[(i-1)m+1] & a[(i-1)m+2] & \cdots & \boxed{\text{a[(i-1)m+j]}} & \cdots & a[im] \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ a[(n-1)m+1] & a[(n-1)m+2] & \cdots & a[(n-m)+j] & \cdots & a[nm] \end{pmatrix}$$

In the two methods of indexing shown in (2.6) and (2.9) the element indexed by $(i, j)$ are boxed. Clearly, the relationship between $p$ and $i$ and $j$ is

**(2.10)**
$$p(i, j) = (i-1)m + j$$

where $m$ is called the **stride**. This is the number of elements to go from $a[i, j]$ to $a[i+1, j]$. Every time you use two subscripts to access this array, your program needs to perform the multiplication and addition in equation (2.10). On the other hand, if you were to create a one-dimensional array of size $nm$ so that **you** can store two-dimensional data in it you can very often avoid the multiplication shown above. For example, if we wanted to evaluate the expression $a[i-1,j]+a[i+1,j]$, we would write $a[p-m]+a[p+m]$.

This way of storing a row at a time is often referred to as **row-major**. We could also store the data one column at a time, which would be called **column-major**. We will use row-major here.

We have seen how to represent two-dimensional arrays and matrices on the computer. What if we had data that was in three dimensions? The solution is simple: we view the three-dimensional array as a set of two-dimensional arrays that are stacked in the third dimension. So, we just end up with another stride consisting of the size of the two-dimensional array that is being stacked. If this stride is designated by $s$, then our formula relating the indices $i$, $j$, and $k$ of our array to the index $p$ of our representation is given by

$$\textbf{(2.11)} \qquad p(i,j,k) = (i-1)s + (j-1)m + k, \quad s = nm$$

In general, if we had a $d$-dimensional array with strides $s_1$, $s_2$, ..., $s_{d-1}$, the relationship between the indices $i_1$, $i_2$, ..., $i_d$ and $p$ is given by

$$\textbf{(2.12)} \qquad p(i_1, i_2, ..., i_d) = \sum_{l=1}^{d-1} (i_l - 1)s_l + i_d$$

We have not said anything about the memory efficiency of the representation so far. We will look at only one scenario here.

If we have a diagonal matrix of size $10000 \times 10000$ we would have $10^8$ elements to store. We do note that all but $10000$ of them are guaranteed to be zero. What we do in this case is just store the diagonal of the matrix as a one-dimensional array and then replace our formula given by equation (2.10) with a small algorithm.

```
ZeroElement = 10001
def p(i, j):
  if i == j:
     return i
  return ZeroElement
```

The 10001 element in the representation needs to be set to zero. There are more sophisticated ways of doing this, but they are outside the scope of this book. You can check out the section on programming A.

**Assignment 2.4**

(1) What happens to the formulas relating indices if our array subscripts started at zero instead of one as done in many programming languages.

(2) How would you modify the algorithm given above to handle tridiagonal matrices? (A tridiagonal matrix may have non-zero elements on the main diagonal and each of the diagonals above and below the main diagonal.)

We are now able to represent numbers and arrays of numbers. Let us now look at functions and the domains on which the functions are defined. The latter part is a little more difficult and we will devote a large portion of a chapter to it later. We will restrict ourselves to representing intervals on the computer and functions that are defined on such intervals.

## Representing Intervals and Functions on the Computer

We concluded at the end of Chapter 1 that we need to be able to represent functions on the computer. A function is defined on a region that we call the domain of definition. This means that at every point in the domain of definition, the function takes on a value. In one dimension, this domain of definition may be an interval.

We have already seen that floating-point numbers on a computer represent intervals. What if we want to represent a part of the real line, say the interval $[0, 1]$? Yes an array with the two values $(0, 1)$ will suffice. However, unless we have a standard function: polynomials, rational polynomials ..., we would have to prescribe a function value at every point in the interval $[0, 1]$. Mathematically, we have an uncountable infinity of points. We have already seen that even if we took all the possible numbers that a computer can represent on the interval, we would still have more numbers than we could handle. So, we follow our natural instinct and do exactly what we did to represent the real line, throw away the part that we cannot handle. For example, we will use a hundred numbers to represent the interval $[0, 1]$. The numbers correspond to points that are called **grid points** or nodes or nodal points. Can we get away with it? Frankly, most of the time, we have no choice! Typically, we do not have the computing resources to handle as many points as we can index or keep track. The resource crunch may be time, memory or both.

Right. We know that given a function say $f(x) = x^2$ on the interval $[0, 1]$, we can find the value of the function for any $x$ in the interval. How do I ensure that the behaviour of a function does not change if I am replacing the interval on which it is defined by hundred grid points? We will give a rather longish answer to this question, since the whole point of this book is to find functions. We have seen in the previous chapter that our business here very often boils down to hunting down a function that satisfies our requirement as described by, say, a differential equation.

We need to understand the meaning of "function" a little better. You are likely to have seen functions as mappings in your calculus class. Here we will look at it from a different point of view.

Consider the following expressions

$$3x^2 + 2x + 1, \tag{2.13}$$

$$3\hat{\imath} + 2\hat{\jmath} + \hat{k}, \tag{2.14}$$

$$3\cos\theta + 2\sin\theta + 1, \tag{2.15}$$

$$3 + 2 + 1, \tag{2.16}$$

$$(3, 2, 1), \tag{2.17}$$

$$321, \tag{2.18}$$

$$3dx + 2dy + dz, \tag{2.19}$$

$$3\frac{\partial}{\partial x} + 2\frac{\partial}{\partial y} + \frac{\partial}{\partial z}, \tag{2.20}$$

where $\hat{\imath}$, $\hat{\jmath}$, $\hat{k}$ are unit vectors, $dx, dy, dz$ are differentials and the last term consists of partial derivatives in the respective coordinates. What do all of these expressions (2.13-2.20) except one have in common? There is really only one case where one would actually perform the addition and simplify the expression further, that is expression (2.16), $3+2+1=6$. Of course, for $x=1$, the polynomial produces the same result.

Why would you think it crazy if I tried to do this kind of a summation with the other expressions. How come I can perform the addition when $x=1$ and not otherwise? Clearly, the $x^2$ and the $x$ prevent us from adding the coefficients $3$ and $2$. We look at this further. Let us perform an operation. How about this:

**(2.21)** $$(3x^2+2x+1)+(2x^2+5x+2)$$

How is this operation different from

**(2.22)** $$(3\hat{\imath}+2\hat{\jmath}+1\hat{k})+(2\hat{\imath}+5\hat{\jmath}+2\hat{k})$$

You see that this argument seems to work in the other cases where "something" prevents you from adding the "3" to the "2". That something is our notation and meaning that we ascribe to what we write and type. The $x^2$ and $x$ stop you from adding the coefficients together just as the $\cos\theta$, $\sin\theta$ or the $\hat{\imath}$, $\hat{\jmath}$. In fact, this is exactly the role of the "," between the 3 and 2. The "321" is in fact the polynomial with $x=10$. The decimal notation is something with which we have been brainwashed for a long time.

If you were to take a course in linear algebra they will point out to you that you are dealing with a three-dimensional linear vector space. They will proceed to talk about the properties of entities that deal with linear vector spaces. The point that we should make note of here is that

> **something that looks like a function can also be viewed as a point in some space.**

In our examples, the space looks like our usual three-dimensional space or at least promises to behave as such. The expressions

**(2.23)** $$(3dx + 2dy + dz) + (2dx + 5dy + dz)$$

and

**(2.24)** $$(3\frac{\partial}{\partial x} + 2\frac{\partial}{\partial y} + \frac{\partial}{\partial z}) + (2\frac{\partial}{\partial x} + 5\frac{\partial}{\partial y} + \frac{\partial}{\partial z})$$

seem also to behave in the same fashion. However, we will not pursue these two expressions at this point. They are stated here so that one can puzzle over them.

So what is the essence of the examples? The triple $(3, 2, 1)$. In fact, $(3, 2, 1)$ is something that can be represented by an array on the computer. It can then represent any of the functions in our list; it is our interpretation that makes it so. We make the following observations

(1) Functions can be considered as points in some space. If we were to organise this space as we did the real line it would be possible for us to come up with algorithms to hunt down a function in a systematic fashion.

(2) Right now we have one way of representing a function on the computer using arrays as long as we interpret the array entries properly. We can use the ability to combine

simple functions to represent a general function as a linear combination of simple functions.

We not only want to represent functions on a computer, we would also like to have them organised so that we can search for one of interest in a systematic and efficient fashion. We will review the process of organising vectors in vector algebra so that we understand how it works with functions. You would have done all of this before when you learnt vector algebra, where the power of algebra is brought to bear on the geometry of vectors.

**Vector Algebra.** Before we go about organising functions, let us first recollect how we went about doing it for vectors. I would recommend at this point that you refresh your memory on vector algebra. We will cover only those parts required for this discussion here.

The plane of this page is two-dimensional. We will restrict ourselves to the plane of this page for now. Let us consider two vectors in this plane $\vec{A}$ and $\vec{B}$. We know from the graphical rep-

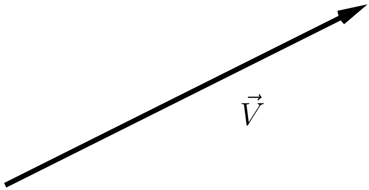

Figure 2.3. The graphical representation of a vector as a directed line segment. Notice that I have not drawn any coordinates, or provided any other frame of reference other than this page on which it is drawn.

resentation of vectors that these can be represented as directed line segments as shown in Figure (2.3). We also know that the scalar product or dot product between them is defined as

$$\vec{A} \cdot \vec{B} = |\vec{A}|\,|\vec{B}|\cos\theta \tag{2.25}$$

where, $|\vec{A}|$ is the magnitude of $\vec{A}$ and $|\vec{B}|$ is the magnitude of $\vec{B}$. $|\vec{A}|$ and $|\vec{B}|$ would be the lengths of the line segments in the graphical representation of the vectors $\vec{A}$ and $\vec{B}$. $\theta$ is the angle between the line segments also called the included angle. The dot product of $\vec{B}$ with unit vector $\hat{a}$ is shown in Figure (2.4). It shows us that the dot product is the projection of the vector $\vec{B}$ onto the line parallel to the unit vector $\hat{a}$. This definition of the dot product allows us to transition from drawing vectors to algebraic manipulation. In particular one can immediately derive an expression for the magnitude of the vector as

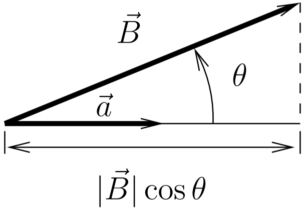

**Figure 2.4. Graphical representation of the dot product of $\vec{B}$ with unit vector $\hat{a}$. $|B|\cos\theta$ is the projection of $\vec{B}$ in the direction of $\hat{a}$.**

$$|\vec{A}| = \sqrt{\vec{A} \cdot \vec{A}} \tag{2.26}$$

We can consequently define a unit vector along $\vec{A}$ as

**(2.27)**
$$\hat{a} = \frac{\vec{A}}{|\vec{A}|}$$

For two vectors whose magnitudes are not zero, it is now clear that if $\vec{Q} \cdot \vec{R} = 0$, then the two vectors are orthogonal to each other.

**(2.28)**
$$|\vec{Q}| \neq 0,\ |\vec{R}| \neq 0, \quad \text{and} \ \vec{Q} \cdot \vec{R} = 0 \Rightarrow \vec{Q} \perp \vec{R}$$

Again, if $\vec{A}$ and $\vec{B}$ are two vectors on our page and they are not parallel to each other (if they were parallel to each other $\vec{A} = \alpha\vec{B}$, or $\hat{a} = \hat{b}$), we can represent any other vector on the page as a linear combination of these two vectors. The plane containing the page is called a linear vector space or a "Banach space". Since any vector in the plane can be generated using $\vec{A}$ and $\vec{B}$, they are said to span the plane. That is some other vector $\vec{P}$ in the plane of the page can be written as

**(2.29)**
$$\vec{P} = \alpha'\vec{A} + \beta'\vec{B}$$

Better still, we could find the unit vectors $\hat{a}$ and $\hat{b}$ and represent $\vec{P}$ as

**(2.30)**
$$\vec{P} = \alpha\hat{a} + \beta\hat{b}$$

How do we find $\alpha$ and $\beta$? The dot product maybe useful in answering this question. If we find the projection of $\vec{P}$ along $\hat{a}$

and $\hat{b}$, we would get

$$P_a = \vec{P} \cdot \hat{a} = \alpha + \beta \hat{b} \cdot \hat{a} \tag{2.31}$$

$$P_b = \vec{P} \cdot \hat{b} = \alpha \hat{a} \cdot \hat{b} + \beta \tag{2.32}$$

So, to find $\alpha$ and $\beta$ we would need to solve this system of equations. However, if $\hat{a}$ and $\hat{b}$ were orthogonal to each other, things would simplify and then $\vec{P}$ can be written as

$$\vec{P} = P_a \hat{a} + P_b \hat{b}, \quad \text{when } \hat{a} \perp \hat{b} \tag{2.33}$$

see Figure(2.5). This simplification that orthogonality gives is so great that quite often we go out of our way seeking it. If $\vec{A}$ and $\vec{B}$ started of by not being orthogonal to each other we could do the following to obtain an orthogonal set $\vec{Q}, \vec{R}$ or an orthonormal set $\hat{q}, \hat{r}$.

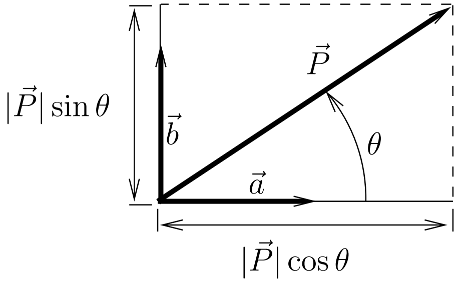

**Figure 2.5.** $\vec{P}$ **is resolved into components along** $\vec{A}$ **and** $\vec{B}$ **using the dot product and hence can be represented by these components**

(1) Set $\vec{Q} = \vec{A}$
(2) then $\hat{q} = \hat{a}$ is the corresponding unit vector

(3) $\vec{B} \cdot \hat{q}$ is the projection of $\vec{B}$ onto $\hat{q}$ and the vector component is $(\vec{B} \cdot \hat{q})\hat{q}$.

(4) $\vec{R} = \vec{B} - (\vec{B} \cdot \hat{q})\hat{q}$ is vector that is orthogonal to $\vec{Q}$. This can be easily checked out by taking the dot product.

(5) if the problem was in three dimensions and a third vector $\vec{C}$ which is independent of $\vec{A}$ and $\vec{B}$ is available, we can obtain a vector $\vec{S}$ which is orthogonal to $\vec{Q}$ and $\vec{R}$ as $\vec{S} = \vec{C} - (\vec{C} \cdot \hat{q})\hat{q} - (\vec{C} \cdot \hat{r})\hat{r}$, where $\hat{r}$ is the unit vector along $\vec{R}$

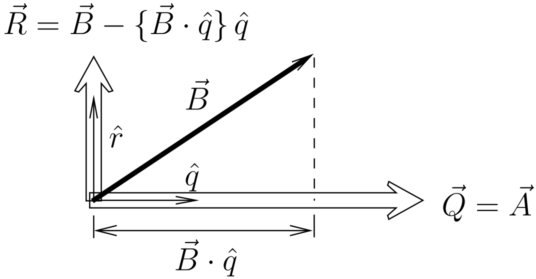

**Figure 2.6. The Gram-Schmidt process applied to two vectors $\vec{A}$ and $\vec{B}$.**

The first few steps to obtain two orthogonal unit vectors is shown in Figure 2.6. The next step to find the third vector perpendicular to the first two is shown in Figure 2.7.

This is referred to as the Gram-Schmidt process and can be carried out to as many dimensions as required. For example,

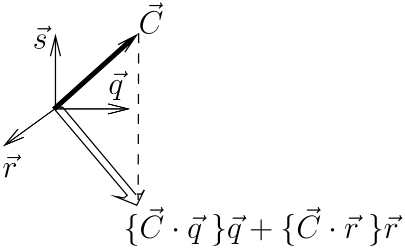
$\vec{C}$
$\vec{s}$
$\vec{q}$
$\vec{r}$
$\{\vec{C}\cdot\vec{q}\,\}\vec{q}+\{\vec{C}\cdot\vec{r}\,\}\vec{r}$

**Figure 2.7. Having applied the Gram-Schmidt process to the two vectors $\vec{A}$ and $\vec{B}$ as shown in Figure 2.6 we now include $\vec{C}$.**

if we have a set of vectors $\vec{E}_1, \vec{E}_2, ..., \vec{E}_n$, which span[4] an $n$-dimensional space, we can then generate an orthonormal set as follows.

(1) Set $\vec{e}_1 = \dfrac{\vec{E}_1}{|\vec{E}_1|}$

then $\vec{\mathcal{E}}_2 = \vec{E}_2 - (\vec{E}_2 \cdot \hat{e}_1)\hat{e}_1$

(2) Set $\hat{e}_2 = \dfrac{\vec{\mathcal{E}}_2}{|\vec{\mathcal{E}}_2|}$

$\vdots$

$\vec{\mathcal{E}}_i = \vec{E}_i - \sum_{j=1}^{i-1}(\vec{E}_i \cdot \hat{e}_j)\hat{e}_j$

[4] Remember that this means that any vector in that $n$-dimensional space can be represented as a linear combination of these vectors

(i) Set $\hat{e}_i = \dfrac{\vec{\mathcal{E}}_i}{|\vec{\mathcal{E}}_i|}$

$\vdots$

$\vec{\mathcal{E}}_n = \vec{E}_n - \sum_{j=1}^{n-1}(\vec{E}_n \cdot \hat{e}_j)\hat{e}_j$

(n) Set $\hat{e}_n = \dfrac{\vec{\mathcal{E}}_n}{|\vec{\mathcal{E}}_n|}$

Thus, we generate $n$ unit vectors that are orthogonal to one another. This technique is illustrated here as it is quite easy to appreciate. However, as has been pointed out before, it is liable to accumulate roundoff errors with all those subtractions.

There are more robust techniques to achieve the same end of obtaining an orthogonal set of vectors. We could perform rotations[**GL83**]. For the sake of our discussion here, the Gram-Schmidt process will suffice. It is clear that the process requires not only the addition and subtraction of vectors, but also the dot product. Once we had the dot product, we were able to build a basis of vectors with which we could represent any other vector in our space. With this in mind, we are now in a position to extend this idea of vectors to functions. To this end, we look for functions that will act as a basis with which we can represent functions of interest to us.

## Functions as a Basis: Box Functions

We will start off by looking at a very specific example. We will define “box” functions and check out their properties. In fact it seems, we will find an easy way to build a basis without using the Gram-Schmidt process.

**Box Functions.** Consider the two functions

$$\textbf{(2.34)} \quad \begin{aligned} \mathfrak{f}(x) &= 1.0 \textbf{ for } x \in [0, 0.5], \quad \text{support of } \mathfrak{f} \\ &= 0.0 \textbf{ for } x \in (0.5, 1.] \end{aligned}$$

and

$$\textbf{(2.35)} \quad \begin{aligned} \mathfrak{g}(x) &= 0.0 \textbf{ for } \ x \in [0, 0.5] \\ &= 1.0 \textbf{ for } x \in (0.5, 1], \quad \text{support of } \mathfrak{g} \end{aligned}$$

The **support** of a function is that part of its domain of definition where it, the function, is non-zero. The two functions $\mathfrak{f}$ and $\mathfrak{g}$ are shown in Figure (2.8).

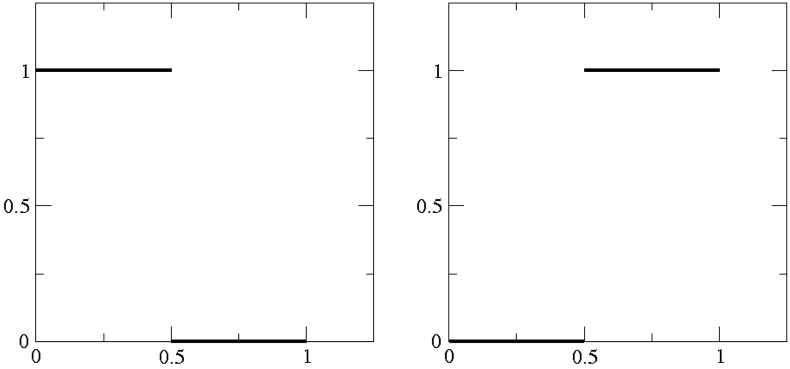

**Figure 2.8. Plots of $\mathfrak{f}$ and $\mathfrak{g}$ defined on the interval $[0, 1]$. These functions are orthogonal as the supports are non-overlapping. We will call these functions "box functions".**

It should be clear to you that we can add/subtract these functions to/from each other. That is, $a\mathfrak{f}+b\mathfrak{g}$ for two numbers $a$ and $b$ makes sense and the operation can actually be performed. As we noted earlier, the whole process works with the definition of the dot product. If we define the dot product or the inner product as

**(2.36)**
$$\langle \mathfrak{f}, \mathfrak{g} \rangle = \int_0^1 \mathfrak{f}(\xi)\mathfrak{g}(\xi) d\xi$$

we can actually take dot products of functions defined on the interval $[0, 1]$. Of course, here we assume that the integral exists. We use the $\langle , \rangle$ notation for the dot product since the $\circ$ is usually used for the composition of functions and the use of "$\cdot$" may create confusion.

In general, for functions $f$ and $g$ defined on the interval $[a, b]$, we can define the dot product as

**(2.37)**
$$\langle f, g \rangle = \int_a^b f(\xi)g(\xi) d\xi.$$

Just as a matter of general knowledge, a linear vector space of functions where the inner product is defined is called an inner product space or a Hilbert space. You will notice that even here, as with the earlier dot product, we have

**(2.38)**
$$\langle f, g \rangle = \langle g, f \rangle.$$

The definition of the inner product immediately allows us to define the following terms. The "magnitude" of $\mathfrak{f}$ (or the **norm** of $\mathfrak{f}$) is given by

$$\text{(2.39)} \quad \|\mathfrak{f}\| = \sqrt{\langle \mathfrak{f}, \mathfrak{f} \rangle} = \sqrt{\int_0^1 \mathfrak{f}(\xi)^2 d\xi} = \sqrt{\int_0^{0.5} d\xi} = \sqrt{0.5}$$

We use $\|\cdot\|$ for the norm as $|\cdot|$ is already used for the absolute value. Consequently, we have the norm (or magnitude) of a function $f$ defined on a general interval as

$$\text{(2.40)} \qquad \|f\| = \sqrt{\langle f, f \rangle} = \sqrt{\int_a^b f(\xi)^2 d\xi}.$$

Without any loss of generality we will look at examples on the interval $[0, 1]$.[5]

It is clear that $\langle \mathfrak{f}, \mathfrak{g} \rangle$ is zero. From our analogy with vector algebra, we conclude that $\mathfrak{f}$ is orthogonal to $\mathfrak{g}$. In fact, we can generalise the definition of the angle between two vectors to the "angle", $\theta$, between two of these functions as follows

$$\text{(2.41)} \qquad \theta = \cos^{-1}\left\{\frac{\langle \mathfrak{f}, \mathfrak{g} \rangle}{\|\mathfrak{f}\| \, \|\mathfrak{g}\|}\right\}, \quad \|\mathfrak{f}\| \neq 0, \|\mathfrak{g}\| \neq 0$$

In particular, as we had already noted if the $\theta = \pi/2$, then the functions are said to be orthogonal to each other. It is likely that you have seen this earlier if you have already studied Fourier series.

Look closely at the definitions of $\mathfrak{f}$ and $\mathfrak{g}$ and their graphs in Figure 2.8. Do you see why they are orthogonal? $\mathfrak{f}$ is non-zero where $\mathfrak{g}$ is zero and vice-versa. So, the support of $\mathfrak{f}$ is $[0, 0.5]$ and the support of $\mathfrak{g}$ is $(0.5, 1.0]$. Since the supports

[5] **We can map any interval $[a, b], b > a$ to the interval [0,1]**

of the two functions do not overlap, the functions turn out to be orthogonal to each other. We chose $\mathfrak{f}$ as being the constant 1.0 in the interval $(0, 0.5)$. Now we see that it could have been almost any function as long its support is $(0, 0.5)$. In a sense, the intervals are orthogonal to each other. We can pick any function on the interval; for now the constant function suffices. It would be nice if $\mathfrak{f}$ and $\mathfrak{g}$ were orthonormal, meaning their norms (or magnitudes) are one. In order to make $\mathfrak{f}$ orthonormal, we need to divide the function $\mathfrak{f}$ by its norm, $1/\sqrt{2}$. We can do the same for $\mathfrak{g}$.

We can define the set of all functions generated by taking linear combinations of $\mathfrak{f}$ and $\mathfrak{g}$ to be $S_2$. In fact we could have defined three functions $\mathfrak{f}$, $\mathfrak{g}$, and $\mathfrak{h}$ with corresponding support $[0, 1/3]$, $(1/3, 2/3]$ and $(2/3, 1]$. These form $S_3$. We can even add $3\mathfrak{f} + 2\mathfrak{g} + \mathfrak{h}$ to our list of expressions (2.13).

---

**Assignment 2.5**

(1) Given two arrays $(3, 2, 1)$ and $(5, 1, 4)$, perform the following operations
   (a) $(3\hat{\imath} + 2\hat{\jmath} + \hat{k}) + (5\hat{\imath} + 1\hat{\jmath} + 4\hat{k})$ and $(3\hat{\mathfrak{f}} + 2\hat{\mathfrak{g}} + \hat{\mathfrak{h}}) + (5\hat{\mathfrak{f}} + 1\hat{\mathfrak{g}} + 4\hat{\mathfrak{h}})$
   (b) $(3\hat{\imath} + 2\hat{\jmath} + \hat{k}) \cdot (5\hat{\imath} + 1\hat{\jmath} + 4\hat{k})$ and $\langle (3\hat{\mathfrak{f}} + 2\hat{\mathfrak{g}} + \hat{\mathfrak{h}}), (5\hat{\mathfrak{f}} + 1\hat{\mathfrak{g}} + 4\hat{\mathfrak{h}}) \rangle$

   where $\hat{\mathfrak{f}}, \hat{\mathfrak{g}},$ and $\hat{\mathfrak{h}}$ are the orthonormal basis functions corresponding to the three functions $\mathfrak{f}$, $\mathfrak{g}$, and $\mathfrak{h}$ suggested above.

(2) On the interval $[-1, 1]$ you are given a function $f(x) = x$. Find $|f(x)|$ and $\|f(x)\|$.

(3) On the interval $[0, 1]$, find the angle (see equation (2.41)) between $\sin x$ and $\cos x$.

(4) On the interval $[-1, 1]$, find the angle between the functions $f(x) = 1$ and $g(x) = x$. How about if you considered the functions on the interval $[0, 1]$.

(5) Here is a trick question for you. Is there any interval on which the functions $x$ and $1/x$ are orthogonal? Be careful with $1/x$ and where it is **defined.**

---

From the assignment, we can see that for the operations of regular vector algebra $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ could well be $\hat{\mathfrak{f}}$, $\hat{\mathfrak{g}}$, and $\hat{\mathfrak{h}}$. Great, we are able use functions to do our regular vector algebra. How would we use them to represent any given function? Let us consider the simple function $P(x) = x$ and see if we are able to use our $\hat{\mathfrak{f}}$, $\hat{\mathfrak{g}}$, and $\hat{\mathfrak{h}}$ to represent it. The graph of this function is a 45-degree line with a positive slope and passing through the origin. Well, we will project the function onto $\mathfrak{f}$, $\mathfrak{g}$, and $\mathfrak{h}$ to find the components. We are saying we can represent $P(x)$ as

$$P_{\mathfrak{f}}\mathfrak{f} + P_{\mathfrak{g}}\mathfrak{g} + P_{\mathfrak{h}}\mathfrak{h} \tag{2.42}$$

where $P_{\mathfrak{f}}$ would be the $\mathfrak{f}$-component of $P$. Likewise, the other two are the $\mathfrak{g}$-component and the $\mathfrak{h}$-component, respectively.

Taking the scalar product with $\mathfrak{f}$ we see that

$$\begin{aligned}\langle P, \mathfrak{f}\rangle &= \langle P_{\mathfrak{f}}\mathfrak{f} + P_{\mathfrak{g}}\mathfrak{g} + P_{\mathfrak{h}}\mathfrak{h}, \mathfrak{f}\rangle \\ &= \langle P_{\mathfrak{f}}\mathfrak{f}, \mathfrak{f}\rangle + \langle P_{\mathfrak{g}}\mathfrak{g}, \mathfrak{f}\rangle + \langle P_{\mathfrak{h}}\mathfrak{h}, \mathfrak{f}\rangle \\ &= P_{\mathfrak{f}}\langle \mathfrak{f}, \mathfrak{f}\rangle + P_{\mathfrak{g}}\langle \mathfrak{g}, \mathfrak{f}\rangle + P_{\mathfrak{h}}\langle \mathfrak{h}, \mathfrak{f}\rangle \\ &= P_{\mathfrak{f}}||\mathfrak{f}||^2\end{aligned} \tag{2.43}$$

Therefore,

**(2.44)** $$P_{\mathfrak{f}} = \frac{\langle P, \mathfrak{f} \rangle}{\|\mathfrak{f}\|^2}$$

With the new definition of $\mathfrak{f}$, $\mathfrak{g}$, and $\mathfrak{h}$ we have $\|\mathfrak{f}\|$, $\|\mathfrak{g}\|$, and $\|\mathfrak{h}\|$ as $1/\sqrt{3}$. We have by definition

**(2.45)** $$\langle P, \mathfrak{f} \rangle = \int_0^1 P\mathfrak{f}dx = \int_o^{1/3} xdx = \left.\frac{x^2}{2}\right|_0^{1/3} = \frac{1}{18}$$

giving us $P_{\mathfrak{f}} = 1/6$. Verify that $P_{\mathfrak{g}} = 1/2$ and that $P_{\mathfrak{h}} = 5/6$. This results in the approximation to the straight line in our "coordinate system" as shown in figure 2.10.
Clearly, the function and the representation are not identical. This, we see, is different from our experience with vectors in three spatial directions. We have here a situation very much like roundoff error. There is a difference between the original function and its representation. We should really write

**(2.46)** $$P(x) \approx \tilde{P} = P_{\mathfrak{f}}\mathfrak{f} + P_{\mathfrak{g}}\mathfrak{g} + P_{\mathfrak{h}}\mathfrak{h}$$

Can you make out that the area under the two curves is the same? We will refer to this generalised roundoff error as **representation error**. How do we get an estimation of this representation error? We need the distance between two points in our function space. We can use the norm that gives us the magnitude to define a **metric** which will give the distance between two points as

**(2.47)** $$d(F, G) = \|F - G\| = \sqrt{\langle F - G, F - G \rangle},$$

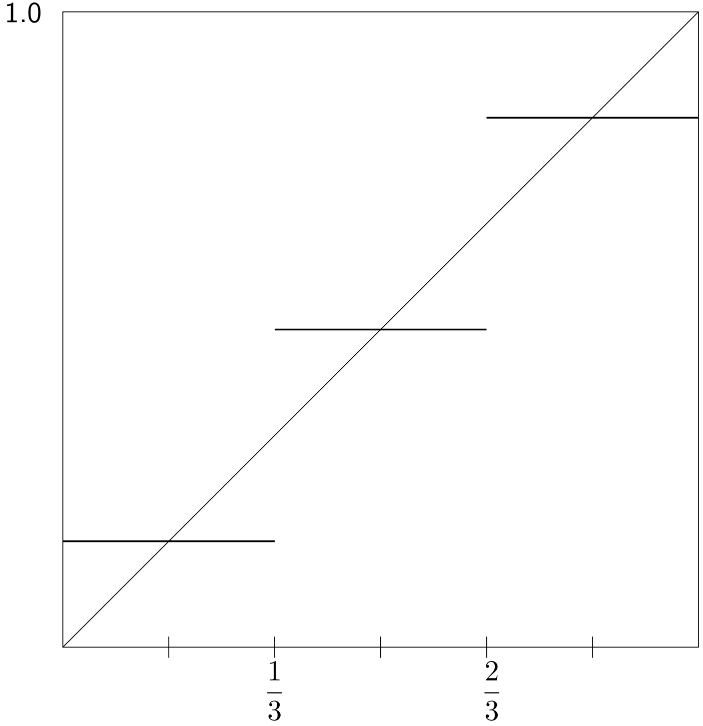

**Figure 2.9. Representing a line $P(x) = x$ using $\mathfrak{f}$, $\mathfrak{g}$, and $\mathfrak{h}$. Should we say "approximating" instead of "representing"?**

$F$ and $G$ are points in the function space. This allows us to get a measure for the error in our representation as

**(2.48)**

$$E(\tilde{P}, P) = d(\tilde{P}, P) = \sqrt{\left\{\int_a^b \left(\tilde{P}(x) - P(x)\right)^2 dx\right\}} = d(P, \tilde{P})$$

Let us now calculate $E$. From Figure 2.9, it is clear that the difference between our function and its representation over the interval $(0, 1/3)$ is identical to that over the interval $(1/3, 2/3)$ and $(2/3, 1)$. So $E$ turns out to be

$$E = \sqrt{3\int_0^{1/3}\left(\frac{1}{6} - x\right)^2 dx} = \frac{1}{6\sqrt{3}} \tag{2.49}$$

Let us step back now and see what we have managed to do so far. The situation is not as bad as it looks. We have a method now to represent functions on the computer and we have some way by which we can measure how good is the representation. The great part is that we can ask a natural follow up question: How do we make our representation of this simple function better? We will explore different answers to this question.

How about defining a whole bunch of functions $B_i^h$ whose support is $(x_i, x_{i+1})$, here $h \approx h_i = x_{i+1} - x_i$ is used as a superscript to remind us of the definition of the functions. Keeping in mind our experience with using the three functions $\mathfrak{f}$, $\mathfrak{g}$, and $\mathfrak{h}$ as a basis, we define $B_i^h$ as $1/\sqrt{h_i}$ on its support and zero otherwise.

$$B_i^h = \begin{cases} \dfrac{1}{\sqrt{h_i}} & x \in (x_i, x_{i+1}) \\ \\ 0 & \text{otherwise} \end{cases} \tag{2.50}$$

The $B_i^h$ are orthonormal. That is

$$\langle B_i^h, B_j^h \rangle = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{2.51}$$

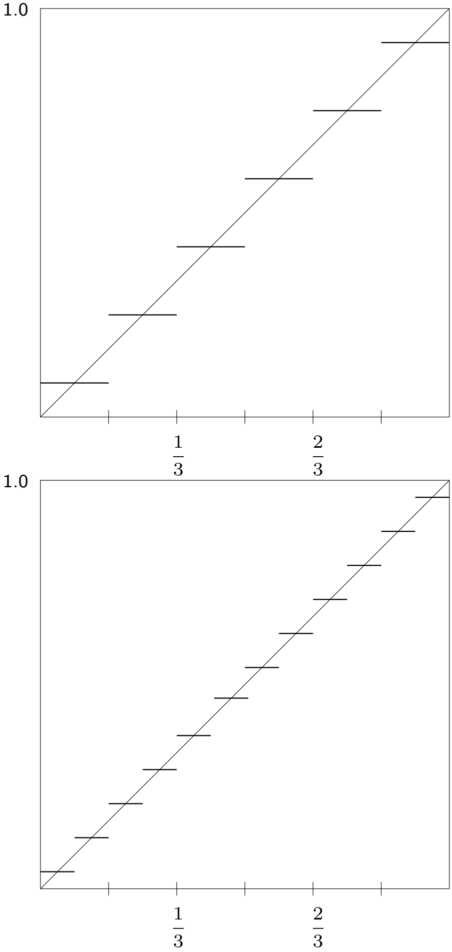

**Figure 2.10. Representing a line $P(x) = x$ using smaller intervals and consequently, more box functions**

We do not see any reason right now to use uneven intervals, so for this discussion, we use $N$ equal intervals on the interval $(0,1)$. Figure 2.10 shows the same function. $P(x) = x$, represented by a larger number of basis functions. We see the function is indeed approximated more closely. However, there is a fear that as we take more and more intervals the number of jumps increases, though the magnitude of the individual jumps can be seen to drop. Our approximation gets better in the sense that it is closer to the function values. On the other hand, it also gets "more" discontinuous in the sense that there are more points at which the function jumps.

So, in general, the function $P(x)$ can be written as

**(2.52)**
$$P(x) \approx \sum_{i=1}^{N} a_i B_i^h$$

The basis functions $B_i$ defined so far are called box functions as suggested by the graph of any one basis function. In all of the equations above, we have used the equals sign to relate the function to the representation on the right hand side. However, we see that the right hand side is only an approximation of the left hand side. Remember that $h \approx (x_{i+1} - x_i)$ really determines the definition of the box functions. We can use the symbol $P^h$ to indicate the representation on the given basis. So, we really should write

**(2.53)**
$$P^h(x) = \sum_{i=1}^{N} a_i B_i^h$$

where, $B_i^h$ are defined on a grid of size $h$. The error at any point in the representation is

**(2.54)**
$$e(x) = P(x) - P^h(x).$$

Finally, the total error on the interval is

**(2.55)**
$$E = \|e\| = \sqrt{\langle P - P^h, P - P^h \rangle} = \sqrt{\int_a^b (P(x) - P^h(x))^2 dx}$$

For the function $f(x) = x$, we can work out the general expression for the error $E$. It is given by

**(2.56)**
$$E = \frac{1}{2N\sqrt{3}} = \frac{h}{2\sqrt{3}}.$$

The reader is encouraged to verify this. The point to note is that we get closer and closer to the actual function as $N$ increases and this happens at the rate proportional to $1/N$. The approximation is said to converge to the original linearly. The error goes to zero in a linear fashion. The error is said to be of first order.

On the other hand, we can represent a constant function $ax^0$ exactly (this is $a$ times $x$ raised to the power zero). The representation is said to accurate to the zeroth order, or simply the representation is of zeroth order.

It is important to note that the first corresponds to the rate of convergence and the second represents the order of the polynomial that can be represented as exactly up to roundoff error.

As promised, we have come up with a basis that allows us to represent our functions in some fashion. However, the more "accurate" is the representation, the jumpier it gets. Instead of breaking up the region of interest into intervals, we could have tried to use polynomials defined on the whole interval of interest. These polynomials can then be used to represent functions on the whole interval $[0, 1]$.

---

**Assignment 2.6**

(1) I would suggest that you try to represent various functions using the box functions on the interval $(0, 1)$. Do this with 10 intervals and 100 intervals.
  (a) $x^2$,
  (b) $\sin x$,
  (c) $\tan x$
(2) For the first two items repeat the process for the interval $(0, \pi)$.
(3) Find $e(x)$ and $E(P, P^h)$ for all the approximations that you have obtained in the first two problems. Is $e(x)$ orthogonal to $P^h(x)$?

---

**Polynomial on the Interval $[0, 1]$.** We have already asked in an earlier assignment whether the functions $p_0(x) = 1$ and $p_1(x) = x$, both defined on the interval $[0, 1]$, are orthogonal to each other. That they are not, is clear from the fact that

**(2.57)**
$$\langle p_0, p_1 \rangle = \int_0^1 x dx = \frac{1}{2}$$

If we define

$$p_i(x) = x^i, \quad x \in [0, 1],\ i = 0, 1, 2, ... \tag{2.58}$$

you can verify that none of these are orthogonal to each other. Why not try to apply the Gram-Schmidt process and see where it takes us? For convenience we will write $p$ instead of $p(x)$. We will indicate the orthonormal basis functions as $\hat{p}_i$.

Let us find the expression for a few of these $\hat{p}_i$. The first one is easy. It turns out $\hat{p}_0 = p_0$. In order to find the second one we need the component of $p_1$ along $\hat{p}_0$. We have already taken the dot product in equation (2.57). Using this, the component of $p_1$ along $\hat{p}_0$ is given by the function $h(x) = \frac{1}{2}$. So, we get

$$\hat{p}_1 = \frac{p_1 - h(x)}{\|p_1 - h(x)\|} = 2\sqrt{3}\left(x - \frac{1}{2}\right) \tag{2.59}$$

Along the same lines we have for the third basis function We need to find the components of $p_2$ along $\hat{p}_0$ and $\hat{p}_1$. We take this out of $p_2$ and normalise to get

$$\hat{p}_2 = 6\sqrt{5}\left(x^2 - x + \frac{1}{6}\right) = 6\sqrt{5}\left(x^2 - \left[x - \frac{1}{2}\right] - \frac{1}{3}\right) \tag{2.60}$$

You can try a few more of these and other problems from the assignment.

---

**Assignment 2.7**

(1) Find $\hat{p}_3$ and $\hat{p}_4$.
(2) Repeat this process on the interval $[-1, 1]$.
(3) Find $\hat{p}_0$ and $\hat{p}_1$ on the interval $[1, 2]$.

(4) Repeat the last question with the arguments of the functions taken as $x-1$ instead of $x$. How about if we calculate the other two basis functions $\hat{p}_3$ and $\hat{p}_4$?

---

If you have had a course in differential equations, you may have recognised the polynomials that you got from problem 2 of assignment 2.7 as being the Legendre polynomials. We can clearly apply this scheme to obtain a set of basis functions on any interval $[a, b]$, bearing in mind that as $x$ increases all polynomials of degree one and greater will eventually diverge.

It looks like we have something here that we can use. The functions are smooth and right now look as though they are easy to evaluate. The scheme of using these polynomial bases does suffer from the same problem that Fourier series representation does. That is, they lack a property that we call locality. Just say you were trying to fit a function and there was some neighbourhoods where you were particular that the representation should be good. You find coefficients and pick enough terms to fit one neighbourhood and then find that at some other spot it is not that good. Adding terms and tweaking may fix the second spot but then change the representation in the first spot on which we have already spent a lot of time. By **locality**, we mean that if we were to change something (for example a coefficient) in one location it is not going to affect our representation in any other location.

On the other hand, if we think back to our box functions, we see that we did indeed have the locality property. That is, changes in coefficients of one basis function did not affect the representation elsewhere. We will now try to merge these two ideas in the following section.

### Linear Approximations: Hat Functions

As we had noted earlier, the representation using box functions gets jumpier as $N$ gets larger. On the other hand, while using polynomials on the whole interval of interest, though smooth, we lose the locality property. We will now define a new set of basis functions. Our aim is to get rid of all those jumps. So, instead of using constant functions over the interval, we will use linear functions. As long as we keep the functions defined on different non-overlapping intervals, they will be orthogonal to each other. However, we will compromise and define two functions on the interval $[x_i, x_{i+1})$. This is so that we can get a smoother representation of $P(x)$.

Consider two functions shown in Figure 2.11). They are defined as follows

**(2.61)**
$$N_i^1(x) = \begin{cases} 1 - \alpha_i(x) & \text{for } x \in [x_i, x_{i+1}), \\ 0 & \text{otherwise.} \end{cases}$$

and

**(2.62)**
$$N_{i+1}^0(x) = \begin{cases} \alpha_i(x) & \text{for } x \in [x_i, x_{i+1}), \\ 0 & \text{otherwise.} \end{cases}$$

where,

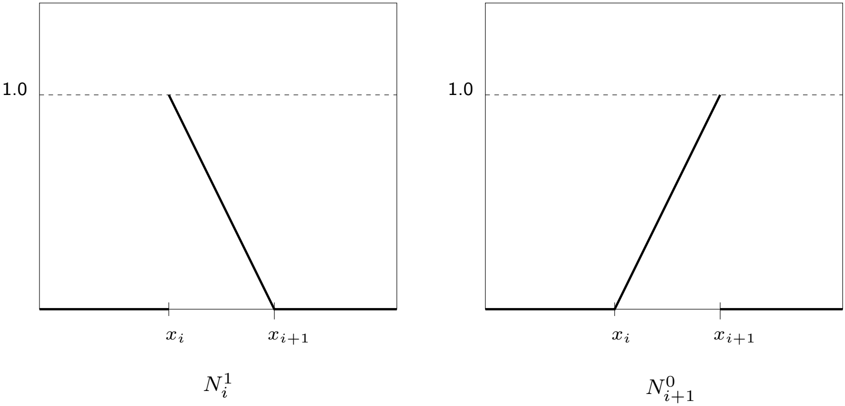

**Figure 2.11. Plots of two functions $N_i^1$ and $N_{i+1}^0$ defined on the interval $[x_i, x_{i+1})$**

**(2.63)**
$$\alpha_i(x) = \frac{x - x_i}{x_{i+1} - x_i}$$

The functions $N_i^1$ and $N_{i+1}^0$ are shown in Figure 2.11. They are first degree polynomials in the interval $[x_i, x_{i+1})$ and zero outside the interval. $[x_i, x_{i+1})$ is the support for the two functions $N_i^1$ and $N_{i+1}^0$.

What is the graph of the function

**(2.64)**
$$f(x) = aN_i^1 + bN_{i+1}^0?$$

This function is graphed in Figure 2.12. It is zero outside the interval $[x_i, x_{i+1})$. Inside the interval it is the sum of two first order polynomials and therefore is a first order polynomial. At $x = x_i$, $f(x_i) = a$, and at $x = x_{i+1}$, $f(x_{i+1}) = b$. We can

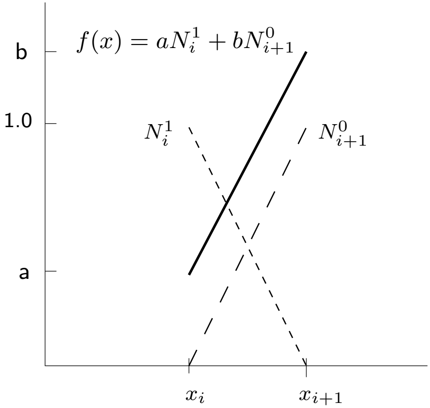

**Figure 2.12. Plot of a straight line segment as represented using $N_i^1$ and $N_{i+1}^0$**

conclude that it is a straight line from the point $(x_i, a)$ to the point $(x_{i+1}, b)$. Are $N_i^1$ and $N_{i+1}^0$ orthogonal to each other? You can take the dot product and verify that

$$\langle N_i^1, N_{i+1}^0 \rangle = \frac{x_{i+1} - x_i}{6}. \tag{2.65}$$

So, they are not orthogonal. On the other hand, with the two functions we have managed to isolate the effects of $f(x_i)$ and $f(x_{i+1})$. What do I mean by this? For the given $f(x_i)$, if we change $f(x_{i+1})$, the coefficient of $N_{i+1}^0$, $b$, alone would change. Local changes are kept local. The sum $N_i^1 + N_{i+1}^0 = 1$. So, for any point $x \in [x_i, x_{i+1}]$ the function value is a linear combination of $f(x_i)$ and $f(x_{i+1})$ (you will hear the use of the expressions "convex combination" or "weighted average" in place of linear combination).

Though the two functions $N_j^1$ and $N_{j+1}^0$ are not orthogonal to each other, they are orthogonal to the functions defined on other intervals that are non-overlapping and can be used to represent straight line interpolants over their support. We can now represent a function $f(x)$ as piecewise linear segments as follows

**(2.66)**
$$f^h(x) = \sum_{i=1}^{n} \left\{ a_i N_i^1 + b_{i+1} N_{i+1}^0 \right\}$$

$h$ is the size of a typical $x_{i+1} - x_i$ and is used as a superscript here to distinguish the representation of the function from the function. This representation is illustrated in the Figure 2.13.

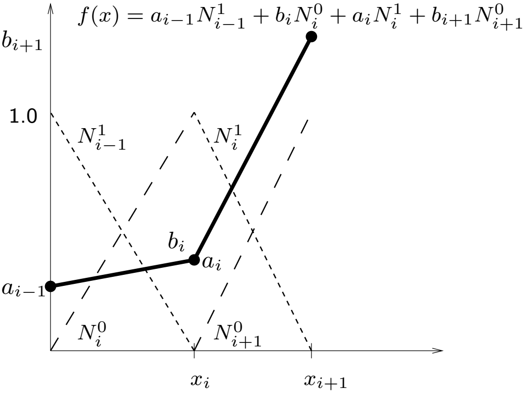

**Figure 2.13. Two adjacent intervals on which a piecewise linear function is represented as being continuous at $x_i$. Continuity at $x_i$ means $a_i = b_i$**

Look carefully at this now. The function is continuous at the point $x_i$. The value of $a_i$ must be the same as that of $b_i$. In this case, we can expand the summation in equation (2.66) to get

**(2.67)**

$$f^h(x) = a_1 N_1^0 + a_1 N_1^1 + \cdots + a_i N_i^0 + a_i N_i^1 + \cdots = \sum_{i=1}^{n} a_i \left\{ N_i^0 + N_i^1 \right\}$$

Please note, I am being a little hazy about what happens at the left end of the interval and right end of the interval. We will look at them later. Right now, what is the sum $N_i^0 + N_i^1$?

$N_i^0$ is supported by the interval $[x_{i-1}, x_i)$, whereas $N_i^1$ has support $[x_i, x_{i+1})$. The sum is zero everywhere except in the interval $(x_{i-1}, x_{i+1})$. This function, shown in Figure 2.14, is called the **hat function** about the node $i$ and we label it as $N_i$ and it has support $(x_{i-1}, x_{i+1})$. It is also called the **tent function**.

Now, we can write

**(2.68)**

$$f^h(x) = \sum_{i=1}^{n} a_i N_i$$

where the $N_i$ are hat functions that have unit value at the grid point $x_i$. Again, we should get a closer approximation to the function if we increase the number of intervals. Functions represented by the hat function can be continuous, but the derivatives at the nodal points will not be available. However, one could take as the derivative at a node, the average of the slopes of the two line segments coming into that node.

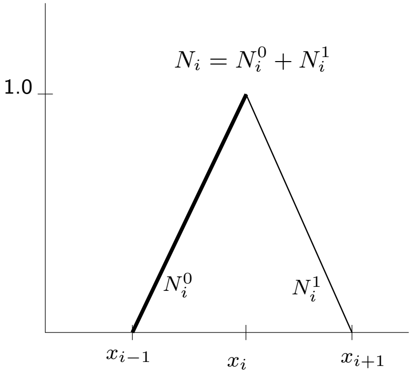

**Figure 2.14.** The sum $N_i^0 + N_i^1$ gives us $N_i$ which is called the **hat function**. It also called the **tent function**

It is clear that we can represent polynomials of first order exactly. Hat functions give us a first order representation. Consequently, the error is of second order. By checking the error in representation of a quadratic we can see that the rate of convergence also happens to of second order. That is as $h \to 0$, the error goes to zero as $h^2$. The error goes to zero in a quadratic fashion.

Is there a difference between our approach to the box functions and the hat function here? There is. In the definition of the $B_i$ of the box function, we made sure that they were orthonormal. In the case of the hat functions, we have an independent set that allows us to represent any function. We could, but do not normalise them. The advantage with defining the hat functions in this fashion is that linear interpolation of tabulated data is very easy. The coefficients $a_k$ are the nodal values taken directly from the tabulated data. No further processing is required.

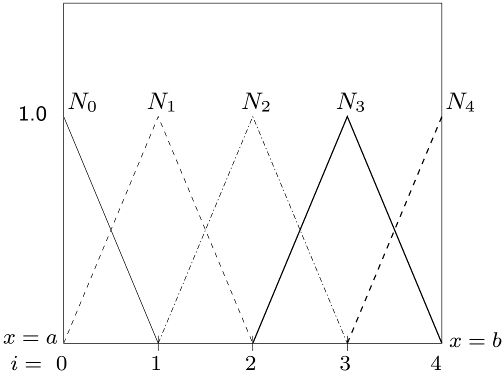

**Figure 2.15. The interval $[a, b]$ is split into four subintervals. Functions on this interval can be approximated by piecewise linear elements that are represented as a linear combination of the hat functions shown in this figure. $N_0$ and $N_4$ in this case are truncated or half hat functions.**

We need to deal with the first point and the last point carefully. Clearly, if $i = 0$ is the first nodal point, we will have an $N_0^1$ and no $N_0^0$. On the other hand if $i = n$ is the last point. We will have a $N_n^0$ and no $N_n^1$. A set of five hat functions are shown in Figure 2.15. Note that the first and last functions have only one of the two limbs that make up the typical hat function.

---

## Assignment 2.8

(1) Verify equation (2.65).

(2) Again, I would suggest that you try to represent various functions using the hat functions on the interval $(0, 1)$. Do this with 10 intervals and 100 intervals. Here are some sample functions that you can try.

(a) $3x + 2$, (b) $x^2$, (c) $\sin x$, (d) $\tan x$

(3) For the first three items, repeat the process for the interval $(0, \pi)$.

(4) Given $n$ grid point that are equally spaced on the interval $[a, b]$ so as to divide that interval into $n - 1$ subintervals and $n$ values of a function at those points do the following.

(a) Fit hat functions and graph the function.

(b) Write a function interface, $h(x)$, to your representation so that given an $x$ in $[a, b]$ the function will return $h(x)$ which is the value given by your representation.

(c) Write a function mid$(x)$ that return an array of the function values at the midpoints of the $n - 1$ intervals as obtained from the hat function representation

---

We set out to construct functions that were orthogonal to each other. We really have not quite achieved that. This can be seen easily by considering the dot product of any two hat functions.

$$\textbf{(2.69)} \qquad \langle N_i(x), N_j(x) \rangle = \begin{cases} 0 & j < i-1 \\ \dfrac{h}{6} & j = i-1 \\ \dfrac{2h}{3} & j = i \\ \dfrac{h}{6} & j = i+1 \\ 0 & j > i+1, \end{cases}$$

where $h = |x_i - x_j|$. It is clear that we do not have orthogonality. However, we do have something called compactness and locality. A change in the coefficient of the $i^{th}$ hat function will result in a change in the function $f(x)$ only in the two adjacent intervals containing the point $x_i$. Remember, this property of keeping the local change local is called **locality**! In the case of the box functions, a change in a coefficient affects only one interval.

What is the value of the derivative of the function at any point between nodes. Just taking the derivative of the representation we see that

$$\textbf{(2.70)} \qquad f'(x) = \sum_{i=1}^{n} a_i N_i'$$

where the prime indicates differentiation with respect to $x$. We find that $N_i'$ is given by

$$N_i'(x) = \begin{cases} 0 & x < x_{i-1} \\ \dfrac{1}{h} & x \in (x_{i-1}, x_i) \\ -\dfrac{1}{h} & x \in (x_i, x_{i+1}) \\ 0 & x > x_{i+1} \end{cases} \tag{2.71}$$

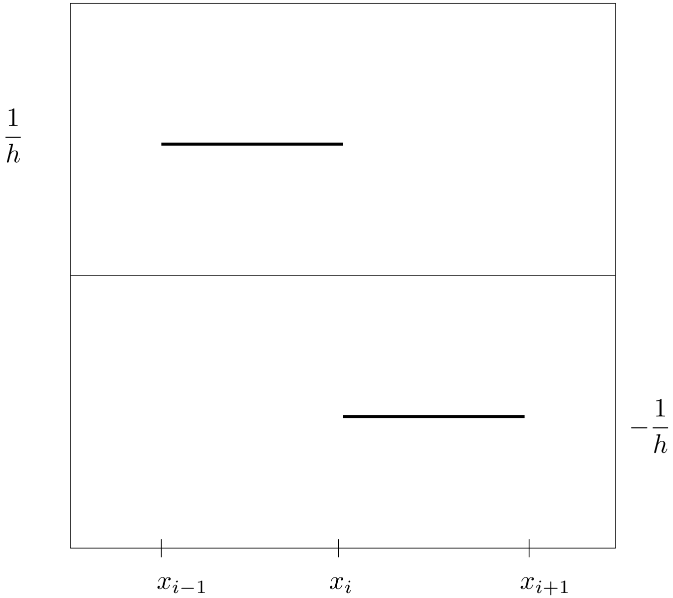

**Figure 2.16. The Haar function. It is the derivative of the hat function defined on the interval** $(x_{i-1}, x_{i+1})$

Note that the derivative seems to be made up of two of our box functions. This combination of two box functions is called a Haar function and is shown in Figure 2.16. Given the relationship to the hat functions, we should use Haar functions in preference to the box function. The derivative at some point $\xi \in (x_i, x_{i+1})$ is written as

**(2.72)**

$$f'(\xi) = a_i N_i'(\xi) + a_{i+1} N_{i+1}'(\xi) = -\frac{a_i}{h} + \frac{a_{i+1}}{h} = \frac{a_{i+1} - a_i}{h}$$

The derivative is a constant on the interval $(x_i, x_{i+1})$ as is expected and can be evaluated based on the end points. We are using the hat functions to approximate the actual function using piecewise linear interpolation and the actual derivative by a piecewise constant function. An estimate of the derivative at the nodal point $x_i$ can be obtained by taking averages from the adjacent intervals as

$$f'(x_i) = \frac{1}{2}\left\{ \frac{a_{i+1} - a_i}{h} + \frac{a_i - a_{i-1}}{h} \right\} = \frac{a_{i+1} - a_{i-1}}{2h} \tag{2.73}$$

where, $h = x_{i+1} - x_i = x_i - x_{i-1}$

So, what do we have? Here is a quick summary.

(1) We are able to get a continuous representation for a continuous function. With the box function we could only get a piecewise continuous representation.

(2) We are able to get approximations of derivatives which we did not even consider with the box functions. The derivative of the hat function suggests that we use the Haar functions as a basis instead of the box function

anytime we are tempted to employ the box function. This is especially true when we look at solving differential equations. After all, on integration the Haar function will give representations in terms of the hat function.

**Assignment 2.9**

In the previous assignment (assignment 2.8) you have used hat functions to represent various functions. For those representations do the following:

(1) Find the derivative of the representation.
(2) Plot this derivative and the actual derivative.
(3) What is the error in the representation of the functions?
(4) What is the error in the representation of the derivatives?

Now that we have these hat functions that are very encouraging, let us ask the same question that led us from the box functions to the hat functions. How good is the representation that we get? We have already seen that we can represent the function $F(x) = x$, exactly. How about a quadratic? The previous assignment gave us an answer to this. We can only represent polynomials of the first degree exactly. The hat function gives us a first order representation of any function.

We will look at a different aspect of approximation now. We have so far used polynomials and trigonometric functions to test our ability to representation functions. We have seen so far that the greater the number of intervals, the better the approximation. We cannot take an infinite number of intervals. So, we have this lingering question, how many is enough and is there a minimum number that we have to take in order to represent the essential features of a function?

Before we can answer this question, we have to understand what we mean by "how many" and "essential features". The question "how many" we can tie with the length of the interval on which the basis function is defined. So far, we have assumed the basis functions are defined on intervals of equal length. The question "how many" then translates to "how long". Which leads to the followup question: how long in comparison to what length scale? This question gives us the first essential feature of a function: an inherent length scale. The other essential feature of interest could be maxima, minima, and zero crossings. We can check out functions that are bounded and can be used to characterise a length scale.

Let us look at what happens if we try to represent a function like $\sin nx$ using hat functions. If we consider the function $\sin x$ and employ 11 grid points to represent it on the interval $[0, 2\pi]$, we see that it does not look as great as using 101 grid points (see assignment 2.8). The representation on ten intervals is shown in Figure 2.17 We will try a whole series of these functions on 11 grid points: $\sin nx, \quad n = 1, 2, 3, \ldots$. What do we observe? We see that on ten intervals the essential features are captured.

- The extrema are not captured by the representation. Since we are not sampling the $\sin x$ function at $\pi/2$ and $3\pi/2$, that is, we do not have a grid point at $\pi/2$ and $3\pi/2$, the extrema of $\sin x$ are not represented. Clearly, this problem will exist for any function. If we

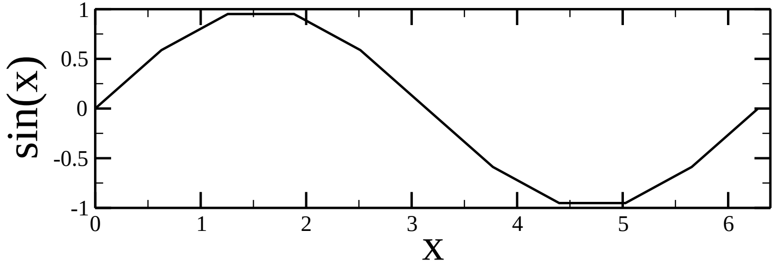

**Figure 2.17.** $\sin x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

do not sample the function at its extrema, we will loose that information.

- The zero crossings, the point at which a function traverses the $x$-axis, are quite accurate. Try to generate these graphs and verify that the zero crossings are good (see first problem of assignment 2.10).

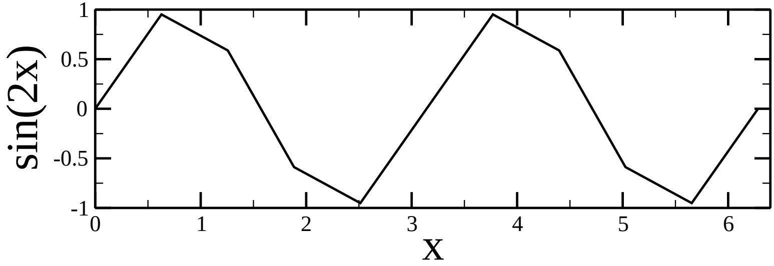

**Figure 2.18.** $\sin 2x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

In Figure 2.18, we double the frequency of the sine wave. Or, equivalently, we halve the wavelength. Again, we see that the extremal values are not represented well. With some imagination, the graph "looks" like the sine function. Again, verify that the zero crossings are fine. So, with ten intervals we are able to pickup the periodicity of the sine function along with the zero crossings. The inherent anti-symmetry of the sine function is also captured by the representation.

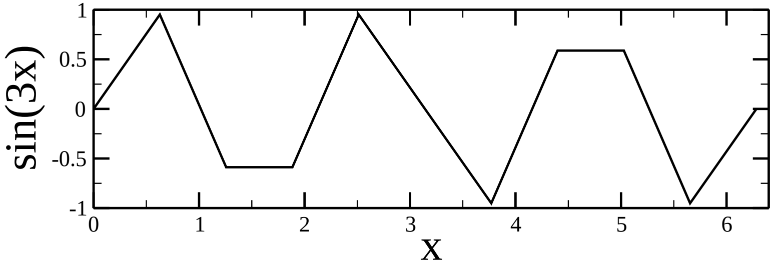

**Figure 2.19.** $\sin 3x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

Now we check out what happens at three times the fundamental frequency. The wavelength is consequently one third the original length. We have not lost the anti-symmetry in the representation. Our peaks now are quite poor. The number of zero crossings is still fine. The location of the zero crossings is now a bit inaccurate. Through the zero crossings, our estimation of the basic frequency of the signal is still accurate.

It would take some pretty good imagination to see the sine wave in our representation for the $\sin 4x$ employing ten intervals as shown in Figure 2.20. The extrema are off. The zero crossing are off. The number of zero crossings is still correct.

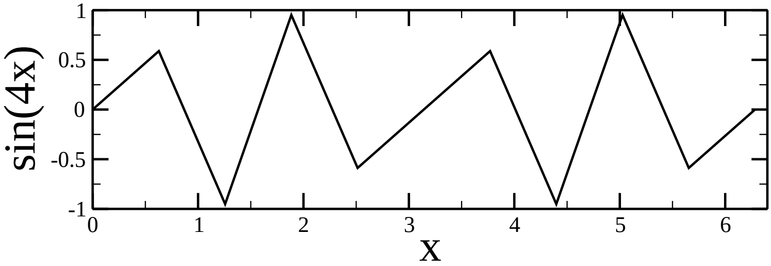

**Figure 2.20.** $\sin 4x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

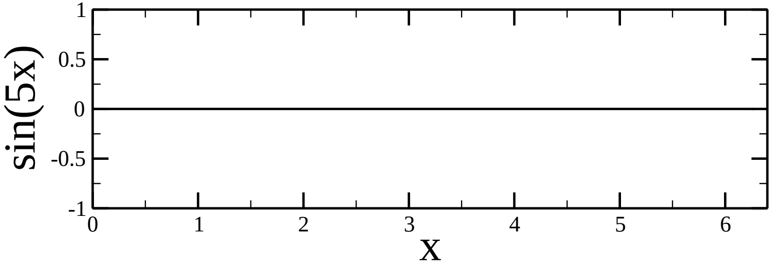

**Figure 2.21.** $\sin 5x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

The $\sin 5x$ curve represented on the ten intervals is shown in Figure 2.21. In fact you could ask: What curve? Here, we have clearly sampled the original function at exactly those points at which it is zero. We have completely lost all information on the extrema. This is a disaster as far as representing functions goes.

We have seen a continuous degeneration in our representation as the wave number increased. We are using ten intervals for our representation. At a wave number five, that is half of the number of intervals that we are using, we have seen complete degeneration. It has no amplitude information or frequency information that we can discern. We will press on and see what happens as we further increase the wave number.

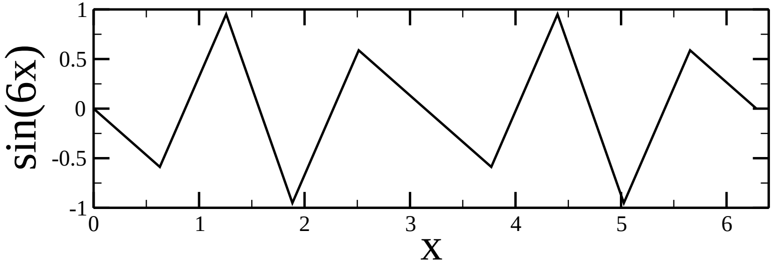

**Figure 2.22.** $\sin 6x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

We see in Figure 2.22 the representation for $\sin 6x$. Does it look familiar. Go back and compare it to $\sin 4x$. They are in fact negatives of each other.

Consider the graph of the representation of $\sin 7x$ shown in Figure 2.23 This is the negative of the representation of $\sin 3x$. We should be able to guess that the next representation shown in Figure 2.24 would look like $\sin 2x$. The most shocking of them all is the fact that the representation of $\sin 9x$ on ten intervals looks like the representation of $-\sin x$ on the same ten intervals. The order of graphs is just reversed from the fifth one. For ten intervals, wave number five is called the folding frequency.

Just for the sake of completeness we plot the representation of $\sin 10x$ on ten intervals in Figure 2.26. We are not surprised by the graph that we get and expect this whole drama to repeat

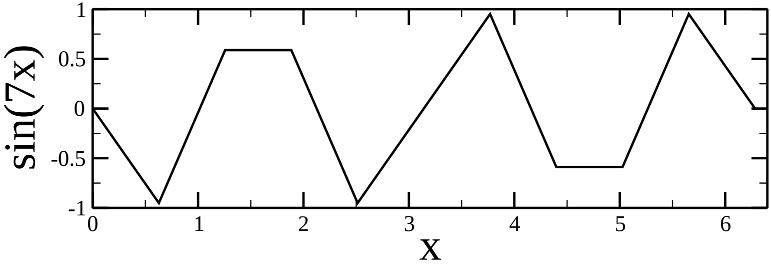

**Figure 2.23.** $\sin 7x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

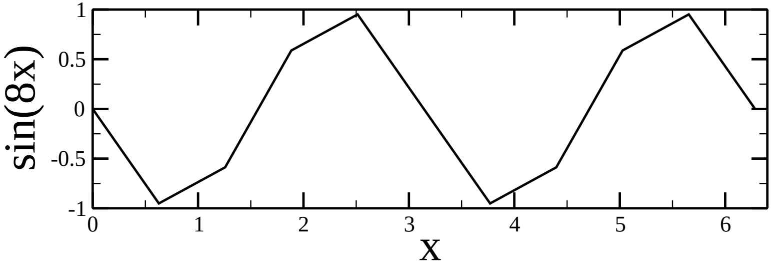

**Figure 2.24.** $\sin 8x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

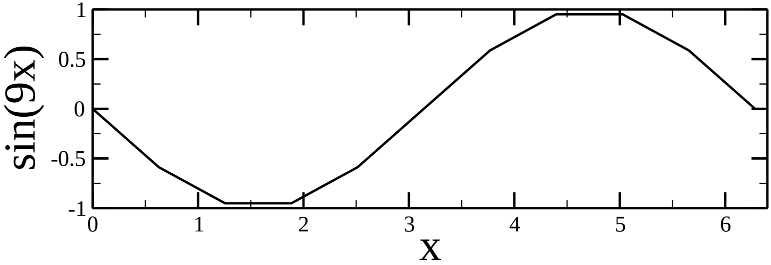

**Figure 2.25.** $\sin 9x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

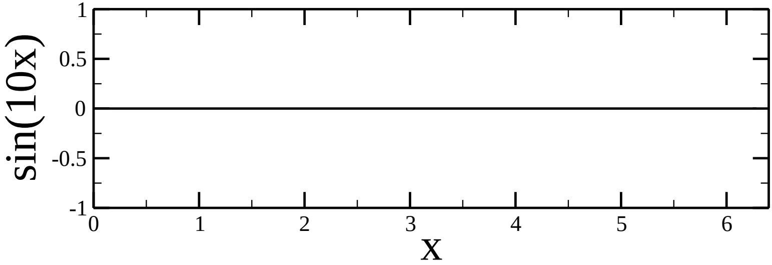

**Figure 2.26.** $\sin 10x$ **sampled at 11 grid points on the interval** $[0, 2\pi]$.

as we increase the wave number. We see that there is a highest wave number that we can represent on this grid. I repeat:

> **A given grid on which we are going to sample functions to obtain a representation of that function, has associated with the grid a maximum wavenumber that can be captured. The function representation using hat functions will not be good at the higher wave numbers.**

Looking at all the figures that we have plotted it looks as if $\sin 4x$ is the highest frequency sinusoid that can be captured by our grid of 11 points. Wave number four corresponds to a high frequency on a grid of 11 points. It corresponds to a lower frequency on a grid of 101 and an even lower frequency on a grid of 1001 and so on. The point being, when we say high frequency or low frequency, we are talking about a given frequency in comparison to the size of the grid.

---

**Assignment 2.10**

(1) Generate the graphs shown in figures 2.17– 2.26. Verify amplitudes and zero crossings.
(2) What happens if we try the same thing on 10 grid points? Is there a difference between an even number of grid points and an odd number of grid points.
(3) Try out the following. Represent the $\sin 4x$ function using 41, 81, 101, 161 grid points.
(4) Repeat the previous problem using $\sin 4x + \sin 40x$.

---

If we want to represent a function fairly accurately, it should be clear from the above, assignment 2.10, that if we have a priori knowledge of the maximum frequency that we expect, we need a grid frequency that is at least ten times larger and preferably forty times larger. Figure 2.21 and Figure 2.26 should give us pause. They tell us explicitly that there are an infinity of functions that can take on the function values $f_i$ at the nodes / grid points $x_i$. The hat function just presents us with a linear candidate representation. In fact, if we use the sinc function instead of the hat function we could have recovered a better and smoother representation of the original $\sin 4x$ instead of the graph shown in Figure 2.20. You can find out more about this in Bracewell's book on Fourier Transforms [**Bra00**]. The use of the sinc function results in a smoother representation but locality goes out the window since its support is the whole real line. This possibility, however, tells us now that we need to seek polynomials of higher order as basis functions.

## Higher Order Approximations

We can define higher order functions spanning multiple intervals that ensure that higher derivatives exist and are continuous. Having used constants and linear interpolants so far, it is clear that a quadratic should be next in line.

What kind of quadratics would we use that are akin to the functions $N^0$ and $N^1$ that made up the hat functions? A linear interpolant in two dimensions requires two points to be specified and we had two corresponding functions to make up the hat functions. In the case of the quadratic, we expect that we need to provide three pieces of information. We would, therefore, expect to have three functions to make up the final quadratic. Three candidate functions are shown in Figure (2.27). Their analytic expressions are

$$N^0(x) = \alpha_i(x)^2 \tag{2.74}$$

$$N^1(x) = 2\alpha_i(x)(1-\alpha_i(x)) \tag{2.75}$$

$$N^2(x) = (1-\alpha_i(x))^2 \tag{2.76}$$

where, again

$$\alpha_i(x) = \frac{x - x_i}{x_{i+1} - x_i}. \tag{2.77}$$

As in the case of the hat functions the sum of the three functions on the interval is always one. So any quadratic on the interval $(x_i, x_i + 1)$ can be represented as a linear combination

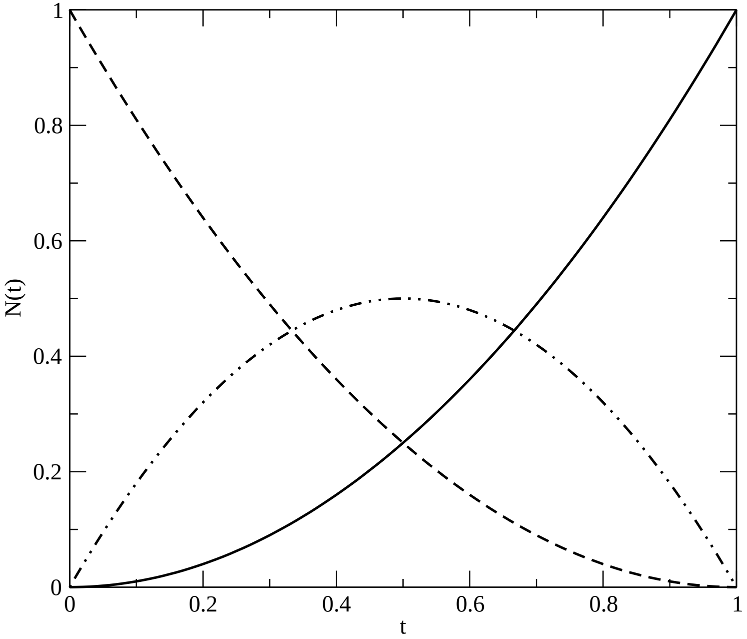

**Figure 2.27. The three quadratic functions used to interpolate on a given interval.**

of these three functions. Instead of dwelling further on the quadratic representation let us, instead skip directly to a cubic. We will follow the same process that we did with the hat function. We consider an interval and ask ourselves what kind of cubics would we use akin to $N_i^1$ and $N_{i+1}^0$.

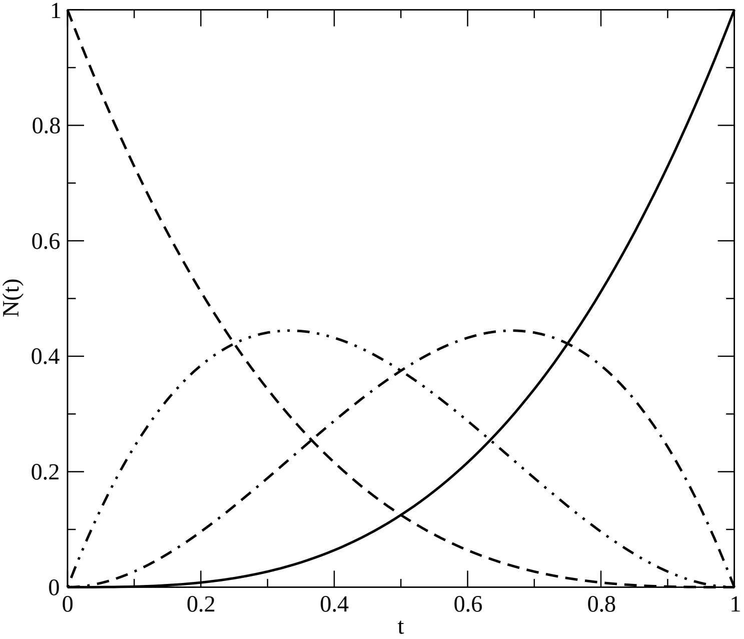

**Figure 2.28. The four cubic functions used to interpolate on a given interval.**

Their analytic expressions are

$$N^0(x) = \alpha_i(x)^3 \tag{2.78}$$

$$N^1(x) = 3\alpha_i(x)^2(1-\alpha_i(x)) \tag{2.79}$$

$$N^2(x) = 3\alpha_i(x)(1-\alpha_i(x))^2 \tag{2.80}$$

$$N^3(x) = (1-\alpha_i(x))^3 \tag{2.81}$$

where, again

**(2.82)**
$$\alpha_i(x) = \frac{x - x_i}{x_{i+1} - x_i}.$$

Before we proceed any further, it should be noted again that in all the four cases studied here,

(1) The sum of the functions in given interval add to one (go ahead and check it out.)
(2) $\alpha$ takes values in $[0, 1]$ and is kind of a non-dimensionalised coordinate on any particular interval.
(3) the coefficients seem to correspond to those of a binomial expansion of the same order. If you remember, it was just a matter of combinatorics.

On the interval $(x_i, x_i + 1)$ we can represent any function as

**(2.83)**
$$f^h(x) = \sum_{i=0}^{3} c_i N^i(x)$$

This comes in many flavours, meaning we can provide a variety of data to determine the cubic. One obvious thing for us to do is to specify the values of the function and its derivative at the grid points. Please note that for the hat function we specified only the function values. If we indicate the nodal values by $a$ and the derivatives by $d$ we can work out what the coefficients of each of our cubics given in equations (2.83). The coefficients

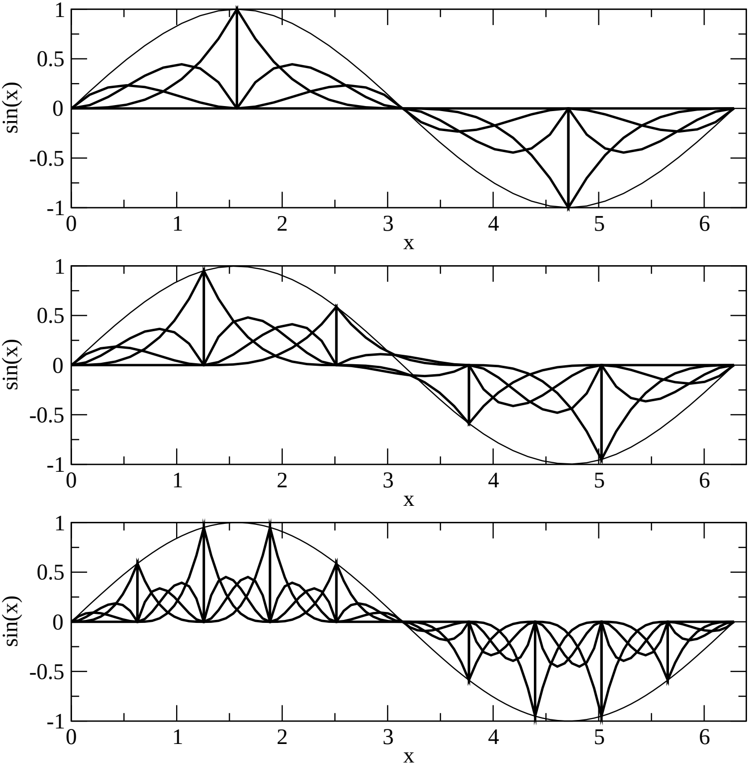

Figure 2.29. Three cubic spline representations of $\sin x$, the function was sampled at 5, 6, 11 grid points along with derivatives at those points. The constituent components that add up to the final function are also shown. The spline in each interval is sampled at ten points for the sake of plotting the graphs.

are

$$c_0 = a_1 = f(x_i) \tag{2.84}$$

$$c_1 = \frac{d_1 h}{3} + a_1, \quad d_1 = f'(x_i), \quad h = x_{i+1} - x_i \tag{2.85}$$

$$c_2 = -\left(\frac{d_2 h}{3} - a_2\right), \quad d_2 = f'(x_{i+1}) \tag{2.86}$$

$$c_3 = a_2 = f(x_{i+1}) \tag{2.87}$$

By inspection, we know that $c_0 = a_1$, where $a_1 = f(x_i)$, the value of the function on the left hand end of the interval. Also, $c_3 = a_2$, where $a_2 = f(x_{i+1})$, which is the value of the function on the right hand end of the interval. If you look at Figure 2.28, you will see that on the left end of the interval, all but $N^0$ are zero. At the right end of the interval, all but $N^3$ are zero. Now, when it comes to the derivatives at the left end of the interval, $N^0$ and $N^1$ have non-zero derivatives. If we differentiate equation (2.83) and set it equal to the derivative $d_1 = f'(x_i)$, we get $c_1$. In a similar fashion we get $c_2$.

Figure 2.29 shows three plots of cubic spline representations of the function $f(x) = \sin x$. The first graph is generated by using five grid points or four equal intervals with $h \approx 1.57$. The second graph is generated with five intervals, $h \approx 1.26$. Finally, the third one is with ten intervals and an interval size $h \approx 0.628$. The first and last graph in Figure 2.29 have a grid point at $x = \pi$. The second graph does not. Using our superscript notation $f^h(x)$ for the representation, we will refer to each of the above representations as $f^{1.57}$, $f^{1.26}$, and $f^{0.628}$. On paper, the three function representations may look equally good. However, that just says that our visual acuity is not good enough to make the judgement. We can estimate the error by numerically evaluating the norms $\|f - f^h\|$ for the three different $h$ values. We get

$$\|f - f^{1.57}\| = 0.01138, \tag{2.88}$$
$$\|f - f^{1.26}\| = 0.00474, \tag{2.89}$$
$$\|f - f^{0.628}\| = 0.000304. \tag{2.90}$$

For the last two graphs, the interval size was halved and you can check that the error dropped by a factor of about $15.6 \approx 16$. This seems to tell us that on the interval the error goes as $h^4$. This is an empirical estimate. We could follow the same process that we did with box functions and hat functions to derive the analytic expression. Again, it must be emphasised that the representation is accurate to a third degree polynomial and as $h \to 0$ the error goes to zero as $h^4$. We will look at another way to determine this relationship between the error and the interval size $h$ in the next section.

As was indicated earlier, there are quite a few ways by which one can fit a cubic to a set of points. At the grid points one could provide the function and the second derivative instead of the function and the first derivative. Of course, the ultimate solution would be to use only the function values at the grid points to obtain the cubics.

---

## Assignment 2.11

(1) Try out the cubic representation for various wave numbers and compare them to the examples shown using

hat functions. Compute the error in the representation. How does this error change as a function of wave number?

---

All the function bases that we have seen so far, starting at the linear hat function, ensure continuity at the nodal points. The higher order bases offer continuity of higher derivatives too. Is it possible to get a better representation for the function if we did not seek to have this continuity? For the hat functions, you can go back to equation (2.66) and see where we imposed the continuity condition. We will now relax that requirement and see what we get. Some analysis is done in section 7.1.

**Linear Interpolants on an Interval.** We want to go back and see if we can use a function that is linear on an interval as the mechanism of building a function basis. The only difference now is that we will not impose the requirement that the representation be continuous at the grid points, even if the the function being represented is continuous. We will go back to equation (2.66) which is rewritten below

$$f^h(x) = \sum_{i=1}^{n} f_i^h = \sum_{i=1}^{n} \left\{ a_i N_i^1 + b_{i+1} N_{i+1}^0 \right\}. \tag{2.91}$$

On the interval $(x_i, x_{i+1})$ we rewrite the equation as

$$f_i^h(x) = \left\{ a_i N_i^1 + b_{i+1} N_{i+1}^0 \right\}. \tag{2.92}$$

We could ask the question what are the best values of $a_i$ and $b_{i+1}$ to approximate the given curve on the interval $(x_i, x_{i+1})$? By best we mean $\|f - f_i^h\|$ in $(x_i, x_{i+1})$ is a minimum. This question is beyond our scope right now, we will answer this question in section 7.1.

We will end this section with one more question. We have suggested that there could be different linear interpolants using straight lines. We have suggested a little before that there are numerous cubics that we can use to represent our function. We have to ask: How do we know that we can always find this representation? Let us see what we have. Consider the space of functions defined on the interval $[a, b]$ spanned by a linearly independent basis $b_i(x)$. For any given function $F(x)$ defined on $[a, b]$, as long as $\langle F, b_i \rangle$ is defined for all $i$ we can find the components. To keep the discussion simple, let us assume that the basis is orthonormal. Then the components of $F$, we have seen, are $c_i$ and $F$ can be written as

**(2.93)**
$$F = \sum_i c_i b_i$$

Is the representation found by you going to be the same as the one found by me? Let us say for the sake of argument that you have gone through some process and obtained the coefficients, $C_i$. Then, the function can be written by the two representations as

**(2.94)**
$$F = \sum_i c_i b_i = \sum_i C_i b_i$$

This means that

**(2.95)**
$$\sum_i c_i b_i - \sum_i C_i b_i = \sum_i (c_i - C_i) b_i = 0$$

Since, $b_i$ are linearly independent, $c_i$ must be equal to $C_i$. So you see, we cannot get different answers unless one of us made a mistake. Okay, we see that given an $F(x)$, the representation we get is unique. How about the other way around, are there more than one function to have the same representation? Meaning, given $c_i$, is there more than one $F$. The answer is yes. We have already seen this in section 2.9.

---

**Assignment 2.12**

Using Haar functions, hat functions, and cubic splines

(1) on the interval $x \in [0, 1]$, use five nodes

$$\{0.0, 0.25, 0.5, 0.75, 1.0\}$$

and find the representations for the following functions
(a) $f(x) = x^2$,
(b) $g(x) = \sin x$,
(c) $h(x) = e^x$.
(d) $s(x) = \tan x$
Find the error in the representations.

(2) Double the number of intervals and check on the error. If you now write a program to do this, you can check for 4, 8, 16, 32 interval and see if you can discover a trend.

(3) Try $\sin nx$ for various $n$. Plot the order of the representation versus $n$ and the number of intervals.

---

## Local Error Estimates of Approximations

So far we have seen how numbers and functions can be represented / approximated on the computer. For numbers, we defined the roundoff error as the difference between a number and its representation. In the case of functions, we have defined a dot product between functions. From this, we defined a norm which we can use to measure of the difference between functions over the whole domain of definition. So, if we have a function and its representation, we can find the magnitude of the difference between them. This gives us a global measure of the difference between a function and its representation. At each point in the domain of definition of the function, we could get a local error by just subtracting the representation from the original function. Is there a way we can get an **a priori** estimate of the error? Put another way, can I say my error will be of this order if I use this kind of a representation? If I have this mechanism to estimate the error before I actually go about my business (that is the "a priori" part), I can make a choice on the basis functions, size of grid - meaning how many grid points, distribution of grids and so on.

I repeat: We have an idea of how our computations using these approximations of numbers affect our final result. We need to look at how we approximate functions more clearly. We have so far seen a few methods to approximate functions. We will now try to figure out what exactly is the approximation. That is, what is the error in the representation? We will look at this problem of approximation of functions and related entities anew.

Very often, we are interested in estimating and approximating things You say, wait a minute this is what we have been talking about so far. Not quite. Until now, we have been looking at a situation where we are given a function and we want to represent it exactly, failing which, we will seek a good approximation. Now consider a situation where we do not have the function ahead of time. "How is this possible?" you ask. Well, let's look at the following example.

> You are a student who has completed 140 credits of a 180 credit baccalaureate program. You have applied to me for a job. You plan to take up the job after the 180 credits are completed. If you provided me with your CGPA (Cumulative Grade Point Average) at the end of your 140 credits, how would I estimate your CGPA at the end of your studies?

Let us state it as a mathematics question. If I knew that a function $f(x)$ had a value $1.0$ at $x = 0.0$, then is there a way I could estimate the value at $x = 0.1$? See, it is different from having a function ahead of time and then finding an approximation to it. A year from now, I will have your actual CGPA. However, I want to have an estimate now as to what it will be and I want to get an idea of the error in that estimate.

So, if $f(0) = 1$ what is $f(0.1)$? Let us use Taylor's series and see what we get.

**(2.96)** $$f(0.1) = f(0) + \underbrace{0.1 f'(0) + \frac{0.1^2}{2!} f''(0) + \cdots}_{\textbf{truncation error}}$$

$'$ and $''$ are used to indicate derivatives. If we were to assume the value of $f(0.1)$ to also be $1.0$ then, we are essentially truncating the Taylor's series after the first term. This looks like we were using our box functions to represent the function $f$. The terms we have thrown away are identified as the truncation error. Very often, you will find that the truncation error is "declared" using the lowest "order" term in the truncation series. If we were to rewrite the above example in general terms

$$f(x+\Delta x) = f(x) + \Delta x^{\overset{\text{first order truncation}}{\text{\textcircled{1}}}} f'(x) + \frac{\Delta x^2}{2!} f''(x) + \cdots \tag{2.97}$$

The approximation that $f(x+\Delta x) = f(x)$ is said to have a truncation error that is first order, as the highest order term in the truncated part of of the series representation is $\Delta x f'(x)$. In actuality, our representation is zeroth order accurate. The representation only works accurately for a constant function, which is a zeroth degree polynomial. It is an approximation for all higher order functions. We would say we have a zeroth order representation with a first order truncation error. Clearly, for a given function, the error is small when $\Delta x$ is small. Of course, if you know the derivative $f'(x)$ then you can get a better estimate as

$$f(x+\Delta x) \approx f(x) + \Delta x f'(x). \tag{2.98}$$

The truncation error would be given by

second order truncation

$$\frac{\Delta x^{\textcircled{2}}}{2!} f''(x) + \cdots \tag{2.99}$$

and the new estimate is expected to be better than the zeroth order estimate. This can used to represent up to a first order polynomial accurately. The truncation error is second order. This is as good as representing a function using hat functions as the basis. The hat function is a first degree polynomial, a polyline to be precise. So, the representation is only first order. I repeat this for emphasis so that there is no confusion between the order of the representation and the order of the truncation term. As we go along, you will see that the order of the truncation term will be source of confusion since it is tagged by the exponent of the increment and not by the order of the derivative occurring in the truncation term.

Of course, if you want an a priori estimate of the truncation error, you need an estimate of the derivative involved in the truncation error. How do we get an estimate of the derivative? Look at equation (2.98) again. It gives us an idea of how to proceed from here.

(1) The way it is written, it tells us that if we have the function and the derivative at some point $x$, we can use that derivative to step off by $\Delta x$ to get the function at the new point $x + \Delta x$. Now all we need to do is to find a way to get the derivative at $x + \Delta x$ and we are in business. We can integrate in $x$ to construct the function. We would effectively be solving a differential equation. We will see how this works at the end of this chapter.

(2) If we have the value of the function at two points $x$ and $x + \Delta x$, we can get an estimate of the derivative.

We will start by looking at approximating the derivative. Equation (2.97) gives us a way to get an estimate for the first derivative and the associated error. First we rewrite it as

$$\textbf{(2.100)} \quad \Delta x f'(x) = f(x + \Delta x) - f(x) - \frac{\Delta x^2}{2!} f''(x) - \cdots$$

or

$$\textbf{(2.101)} \quad f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x} - \underbrace{\frac{\Delta x}{2!} f''(x) - \cdots}_{\text{trucation error}}$$

We see that an estimate with a first order truncation error for the derivative at the point $x$ is

$$\textbf{(2.102)} \quad f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x}.$$

The leading error term is in fact $\frac{\Delta x}{2} f''(x)$. Go back to the section of basis functions and you will see that we have a similar expression there (see equation (2.72)) for the estimate of the first derivative using hat function representations for $f$. As would be expected, and it can be verified from equation (2.101), the derivative of a linear function is represented exactly. The representation of the function $f(x)$ is first order. The representation of the derivative itself is constant on the interval $(x, x + \Delta x)$, and as we say "zeroth order". However, the expression for the derivative in finite difference parlance is that it is first order or that it is of the order of $\Delta x$. This means the truncation error is first order. For higher order polynomials, equation (2.102) gives an approximation to the derivative. How good the estimate is, actually, depends on the point at which this is considered the derivative. We will see this in more detail in the next section.

If you already have the derivative at $x$, then equation (2.97) again gives us a way to get an estimate for the derivative at $x + \Delta x$.

**(2.103)** $$f'(x + \Delta x) = f'(x) + \Delta x f''(x) + ...$$

Please note that if the derivative $f'(x)$ is known, then it is a first order estimate of $f'(x + \Delta x)$ from Taylor's series (2.103). This estimate is with an error $\Delta x f''(x)$, which is twice the error of (2.102).

To summarise, if we have a function at hand or know the nature of the function ( linear, quadratic, . . . , periodic, . . . ), we can decide on the representation and get an estimate of the error in our representation. On the other hand, if we are constructing the function as we go along we can still get an idea as to how good or poor is our estimate. We also see that though there are two different view points of representing the derivative of a function, they are actually related. We can tie the finite difference scheme with representing the function by, say, a hat function.

As a side effect from our look for local error estimates, we have found that we can estimate the first derivative of a function. We are interested in solving differential equations. Is there a systematic way by which we can generate approximations to derivatives? Can we again get an idea of the error involved and the sources of the error?

## Representing Derivatives - Finite Differences

In the previous section, we have seen that we can get a representation for the first derivative of a function in terms of the nodal value of the function. The representation of the function was linear and the derivative was a constant on the interval of interest. We will now start over and see if we can build up estimates of various orders for derivatives of various order. For example, we could be interested in first order, second order..., estimates of derivatives. The derivatives of interest maybe first derivatives, second derivatives and so on. Let's see how we work this. In many calculus texts, the derivative is defined as follows

$$\frac{df}{dx}(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

In order to estimate the derivative at a point $x$, one can use the values known at $x$ and $x + \Delta x$ and eliminate the limiting process. Thus, we have a finite difference approximation for the first derivative.

$$\frac{df}{dx}(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{2.104}$$

The question that remains to be answered is "how good is this estimate?".
To this end, we turn to Taylor's theorem.

**(2.105)**

$$f(x+\Delta x) = f(x) + \frac{\Delta x}{1!} f'(x) + \frac{\Delta x^2}{2!} f''(x) + \frac{\Delta x^3}{3!} f'''(x) + \frac{\Delta x^4}{4!} f''''(x) + \cdots$$

We see that by rearranging terms we can extract out the approximation (2.105) for the first derivative and write an equation instead of the approximation. We get

**(2.106)**
$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} - \frac{\Delta x}{2!} f''(x) + \cdots$$

The error therefore is of the order of $\frac{\Delta x}{2!}\frac{\mathsf{d}^2 f}{\mathsf{d}x^2}(x)$ As opposed to starting with a classical definition of the derivative, we see that using Taylor's series to expand the function at a point $(x+\Delta x)$ about the point of interest we can isolate the derivative of interest.

**(2.107)**
$$f^{h\prime}(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}.$$

The superscript $h$ is to remind us that this is an approximation with $h \equiv \Delta x$. We will drop the use of the superscript unless there is ambiguity. The expression on the right hand side of equation (2.107) is called a **forward difference** and the truncation error is referred to as first order. This approximation for the first derivative of $f$ is called a **forward difference** as the approximation is at the point $x$ and it involves the points $x$ and $x+\Delta x$. We can proceed to derive a **backward difference** formula in a similar fashion. Using the Taylor's series expansion for the function at $x - \Delta x$, that is

**(2.108)**

$$f(x - \Delta x) = f(x) - \frac{\Delta x}{1!} f'(x) + \frac{\Delta x^2}{2!} f''(x) - \frac{\Delta x^3}{3!} f'''(x) + \frac{\Delta x^4}{4!} f''''(x) + \cdots$$

we get the backward difference approximation for the first derivative at $x$ as

**(2.109)**

$$f'(x) = \frac{f(x) - f(x - \Delta x)}{\Delta x} + \frac{\Delta x}{2!} f''(x) - \frac{\Delta x^2}{3!} f'''(x) + \frac{\Delta x^3}{4!} f''''(x) + \cdots$$

Again we see that the truncation error is first order. We will now inspect the truncation error in the two approximations.

$$-\frac{\Delta x}{2} f''(x), \quad \text{and} \quad + \frac{\Delta x}{2!} f''(x)$$

We see that they have the same magnitude but are opposite in sign. Clearly, if we took an average of these approximations to the first derivative, we would expect to reduce the truncation error.

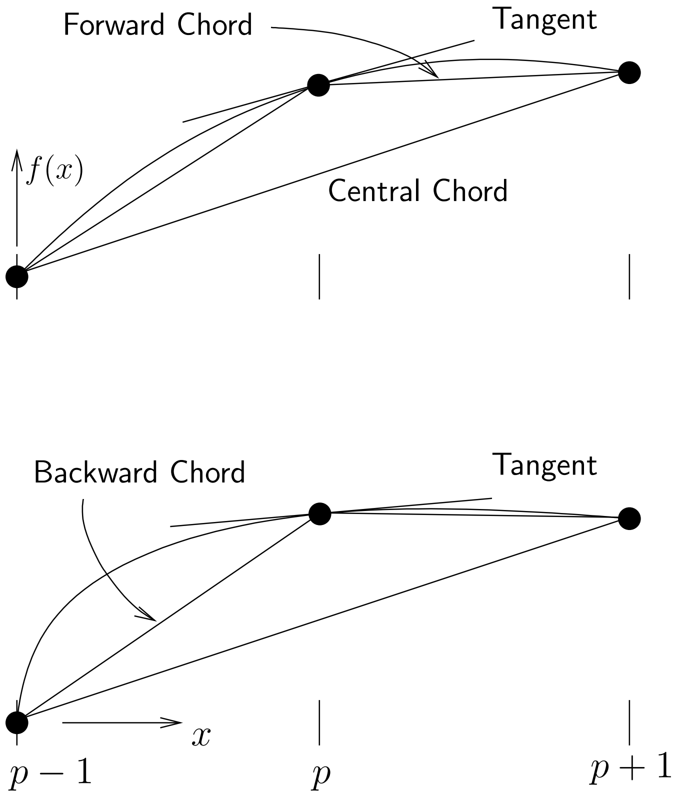

**Figure 2.30. The forward difference approximation at the point $p$ uses the slope of the forward chord as an approximation for the slope of the tangent at point $p$. Similarly, the backward difference uses the backward chord and the central difference employs the central chord. This is shown for two different types of functions**

**(2.110)** $$f'(x) \approx \frac{1}{2}\left\{\text{Forward} f'(x) + \text{Backward} f'(x)\right\}$$
$$= \frac{1}{2}\left\{\frac{f(x+\Delta x) - f(x)}{\Delta x} + \frac{f(x) - f(x-\Delta x)}{\Delta x}\right\}$$
$$= \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$$

Well, that's not bad, we have a **centred difference** approximation to the first derivative now. Can we derive it from Taylor's series? Lets take equation (2.105) and subtract equation (2.108) from it. We get

**(2.111)** $$f(x+\Delta x) - f(x-\Delta x) = 2\frac{\Delta x}{1!}f'(x) + 2\frac{\Delta x^3}{3!}f'''(x)\cdots$$

So, the first derivative can be written as
**(2.112)**

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}\boxed{-\frac{\Delta x^2}{3!}f'''(x)}\cdots$$

The truncation error in this case is second order.

We want to make the following observation: On a given interval $(a, b)$, the expression $[f(b) - f(a)]/[b - a]$ is a first order approximation of the derivatives $f'(a)$ and $f'(b)$; It is a second order estimate of the derivative $f'([a + b]/2)$. If you remember the mean value theorem, we know that

**(2.113)** $$\theta_{ab} = \frac{f(b) - f(a)}{b - a} = f'(\xi), \quad \text{for some } \xi \in [a, b]$$

This tells us that $\theta_{ab}$ is the exact derivative of $f(x)$ somewhere in the interval $[a, b]$. We just do not know where. Given no other information, our expression for the truncation error tells us that $\theta_{ab}$ has a first order truncation error as an approximation to the derivative at the points $x = a$ and $x = b$. It has a second order truncation error as an approximation at the midpoint $x = (a + b)/2$. The two examples in Figure 2.30 show the part of the hat function used to approximate the function from which the derivative is inferred. It is clear comparing the two functions that even the centred difference can be "off" quite a bit.

| $\Delta x$ | Forward Difference | Backward Difference | Central Difference |
|---|---|---|---|
| 0.1 | 3.31 | 2.71 | 3.01 |
| 0.01 | 3.0301 | 2.9701 | 3.0001 |
| 0.001 | 3.003001 | 2.997001 | 3.000001 |
| 0.0001 | 3.00030001 | 2.99970001 | 3.00000001 |

**Table 2.1. Estimation of derivative of $f(x) = x^3$ using forward differences, backward differences and central differences for a variety of parameter values**

Consider a function like $f(x) = x^3$. We can try to estimate the derivative of this function at $x = 1.$ using various values of $\Delta x$ and the three methods of estimation derived so far. A more detailed assignment is given below. The results for this function are given in the Table (2.1).

Study Table (2.1) carefully. Look at the error carefully. For convenience the error is tabulated in Table (2.2). The error term in the forward and backward differences are indicated as the sum of two terms. These can be identified as the first and second leading terms in the truncation error. The central difference approximation on the other hand has only the higher order term.

| $\Delta x$ | Forward Difference | Backward Difference | Central Difference |
|---|---|---|---|
| 0.1 | 0.3+ 0.01 | -0.3+0.01 | 0.01 |
| 0.01 | 0.03+0.0001 | -0.03+0.0001 | 0.0001 |
| 0.001 | 0.003+ $10^{-6}$ | -0.003 + $10^{-6}$ | $10^{-6}$ |
| 0.0001 | 0.0003 + $10^{-8}$ | -0.0003+$10^{-8}$ | $10^{-8}$ |

**Table 2.2. The errors in the estimation of derivative of $f(x) = x^3$ using forward differences, backward differences and central differences for a variety of parameter values. For the forward and backward differences we have indicated the truncation error as the sum of the leading term and the second term.**

One can easily derive the expressions for higher order approximations to the first derivative. These are tabulated in Table 2.3

| Order | type | Difference formula | truncation error |
|---|---|---|---|
| 1 | forward | $\frac{f_{i+1} - f_i}{\Delta x}$ | $-\frac{\Delta x}{2} f''(x)$ |
| 1 | backward | $\frac{f_i - f_{i-1}}{\Delta x}$ | $\frac{\Delta x}{2} f''(x)$ |
| 2 | central | $\frac{f_{i+1} - f_{i-1}}{2\Delta x}$ | $-\frac{\Delta x^2}{3!} f'''(x)$ |
| 2 | forward | $\frac{-f_{i+2} + 4f_{i+1} - 3f_i}{2\Delta x}$ | $\frac{\Delta x^2}{3} f'''(x)$ |
| 2 | backward | $\frac{3f_i - 4f_{i-1} + f_{i-2}}{2\Delta x}$ | $\frac{\Delta x^2}{3} f'''(x)$ |
| 3 | backward | $\frac{2f_{i+1} + 3f_i - 6f_{i-1} + f_{i-2}}{6\Delta x}$ | $-\frac{\Delta x^3}{12} f^{iv}(x)$ |
| 4 | central | $\frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12\Delta x}$ | $\frac{\Delta x^4}{30} f^{v}(x)$ |

**Table 2.3. Table of differences approximating the first derivative of a function f to different orders and the corresponding truncation errors**

We ask a simple question here. Can we make our $\Delta x$ smaller and smaller and get better and better answers? We have already seen that there is a limit of machine-$\epsilon$, below which we can't go. However, the machine-$\epsilon$ is quite small. In which case, why do I need all these different orders? The simple debate that we could have is whether a finer $\Delta x$ is better than taking a higher order approximation for the derivative. Or, more greedily, take a fine $\Delta x$ and a very high order scheme. We address the issue of taking a very fine $\Delta x$ with an assignment.

---

**Assignment 2.13**

(1) Verify the formulas in Table 2.3.

(2) Use the function $f(x) = 3x^2 + 2x + 1$ and evaluate $\theta_{ab}$, with $a = 0$ and $b = 1$. Compare $\theta_{ab}$ with $f'(0)$, $f'(1)$, and $f'(0.5)$.

(3) Here is a simple test that you can perform. Pick our favourite function: $\sin(x)$. Estimate its derivative at some value of $x$, say, $x = \pi/4$ using the different schemes shown in Table 2.3. Compute the magnitude of the relative error for different values of $\Delta x$. The **relative error** is defined as (computed derivative - actual derivative) / actual derivative. Plot the $\log|\text{error}|$ versus $\log \Delta x$. Things are not what we always expect.

(4) Verify and reproduce tables (2.1) and (2.2). Repeat the process for other functions.

---

We see from Figures 2.31 and 2.32 that the plot of the error is quite complex. Note that on the x-axis we have $\log \Delta x$. This

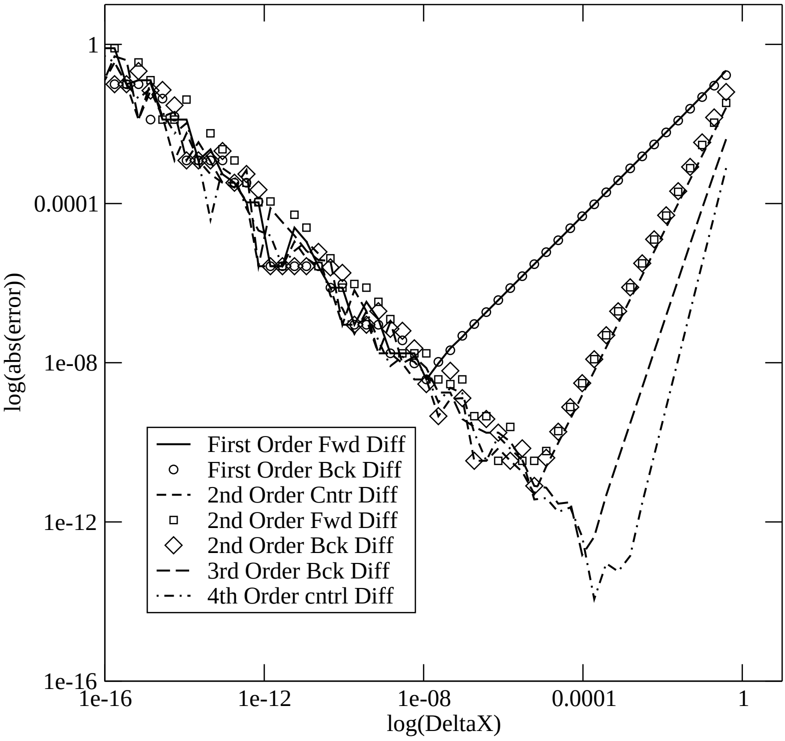

**Figure 2.31. Convergence plot: The magnitude of the relative error in the first derivative of $\sin x$ versus $\Delta x$. The derivative is evaluated at $\pi/4$ using finite differences of different orders and is plotted on a log-log plot**

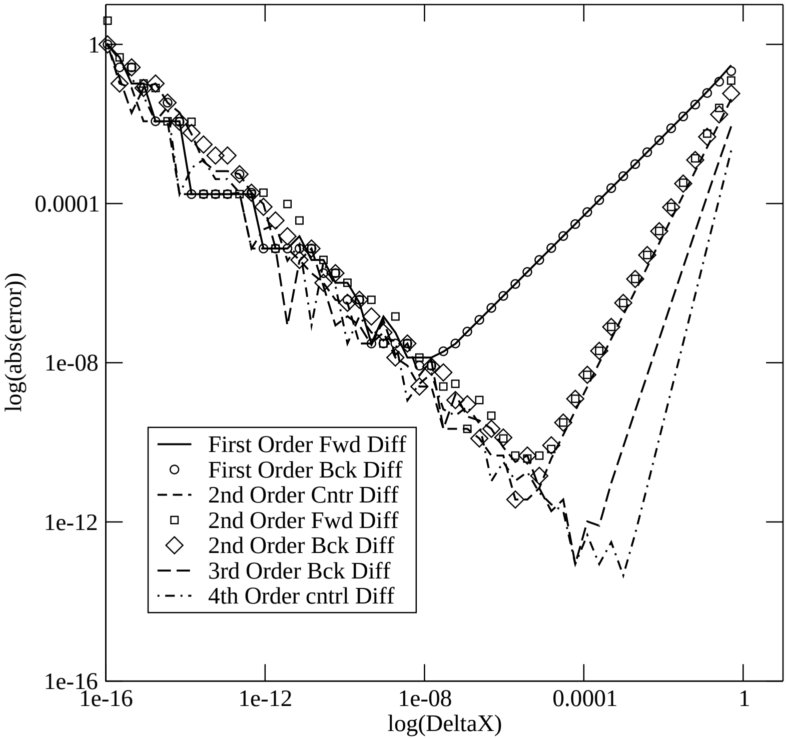

**Figure 2.32. Convergence plot: The magnitude of the relative error versus $\Delta x$ in the first derivative of $\exp(x)$ evaluated at $x = 1$ using finite differences of different orders plotted on a log-log plot**

increases left to right, indicating a coarsening of the grid as we go from left to right. Let us consider the first order error term to understand the graph better.

**(2.114)**
$$\text{error} = \underbrace{\frac{f^{h\prime} - f'}{f'}}_{\text{calculated}} = \underbrace{\frac{\Delta x}{2} \frac{f''}{f'}}_{\text{truncation error}}$$

Taking the logarithm on both sides

**(2.115)**
$$\log|\text{error}| = \log \Delta x + c$$

where $c$ is a constant. So, if we plot the absolute value of the relative error of a first order scheme versus $\log \Delta x$, we expect to get a straight line with slope one. That is, we expect to have linear convergence to the exact value as $\Delta x \to 0$. The slope of the line for the higher order schemes will be the order of the scheme. However, Figures 2.31 and 2.32 show something unexpected. Starting at the right hand side of the graphs, we see for large $\Delta x$, the truncation error is quite large. As we move to the left, the error for all the representations does indeed drop as a straight line with the appropriate slope on this log-log plot. However, there seems to be a barrier which we cannot cross. This looks like a curve with negative slope one. I say "looks like" because we also observe a certain randomness to this curve.

Look at the assignment 2.3. Remember that in the given interval the roundoff error seems to show up in a random fashion. Clearly, the roundoff error kicks in as the $\Delta x$ gets smaller and smaller. This seems to be on a curve of slope negative one. Why should it have a negative slope of magnitude one? The answer is quite simple. We are looking at a plot of the **relative** error. So, the slope is not one, it is actually the derivative of the function $f(x)$.

Now, to make sense of this, look at the numerator of the first order forward difference expression: $f(x+\Delta x) - f(x)$. In the cases that we have plotted they have the same sign. As $\Delta x$ gets smaller, they share more and more significant digits. These common digits are eliminated by the subtraction to leave the difference. The smaller the $\Delta x$, the greater the number of significant digits that are lost.

Let us consider an example here. For the sake of the example we restrict ourselves to a four digit mantissa. Say at $x$ the value of $f$ is $0.1234$ (the actual value is $0.12345678$ and the value at $x+\Delta x_1$ is $0.0$. In this contrived example, $f(x+\Delta x_1) - f(x) = 0.1234$. We tabulate values for this made up example

What is the rate at which we lose them as $\Delta x$ decreases? Can we estimate that? The answer is an emphatic "Yes!". Taylor's series again tells us that

$$f(x+\Delta x) = f(x) + \Delta x f'(x) + \cdots \tag{2.116}$$

Yes, $f(x+\Delta x)$ approaches $f(x)$ at the rate of $f'(x)$.

The truncation error decreases linearly with a slope which is the order of the scheme and when it intersects the line with negative unit slope, the roundoff error dominates the truncation error. From the graph, we observe that higher order schemes allow us to get more accurate representations of the first derivative, however, there is a tighter limit on the size of the $\Delta x$ one can use.

| $\Delta x$ | $f(x+\Delta x)$ | $f(x+\Delta x) - f(x)$ | $\Delta f$ |
|---|---|---|---|
| $\Delta x_1$ | 0.0 | -0.1234 | -0.1234 |
| $\Delta x_2$ | 0.1 | -0.0234 | $-0.0234\underline{5}$ |
| $\Delta x_3$ | 0.12 | -0.0034 | $-0.0034\underline{56}$ |
| $\Delta x_4$ | 0.123 | -0.0004 | $-0.0004\underline{567}$ |

**Table 2.4. Demonstration of increased roundoff error as $\Delta x$ gets smaller. The roundoff error would remain at the fifth significant place if $\Delta f$ were used in the derivative calculation instead of $f(x+\Delta x) - f(x)$. The underlined digits in $\Delta f$ are lost due to fixed size mantissa**

On the roundoff side, we see that once roundoff error equals or exceeds the truncation error, for every bit in the representation of $\Delta x$ that we reduce, we lose one bit in the relative error in the derivative. Which explains/is a conclusion drawn from the unit negative slope of the error curve.

One lesson that you pick up from here is that for some reason if you want to take very small increments, remember you may be just accumulating round off error instead of getting the accuracy that you wanted.

Another point that must be noted here is that if we were to use the finite difference schemes to evaluate the first derivatives of polynomials of various degrees, there is a degree of the polynomial up to which a given scheme will give the exact derivative.

A finite difference approximation for the first derivative will give the exact value for all degrees of polynomial up to a maximum $n$ for the given scheme. This scheme is called an $n^{th}$ order scheme.

Now we look at obtaining approximations for higher order derivatives. We try first to get second derivatives. We add equations (2.105) and (2.108) to eliminate the first derivative term and get an expression for the second derivative as

**(2.117)**

$$f''(x) = \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2} - \frac{\Delta x^2}{12} f''''(x) + \cdots$$

Like we did earlier, we can get different linear combinations of the function evaluated at grid points. So, we see that in general, using the Taylor's series one can approximate a derivative of order $n$, written here as $f^{\langle n \rangle}$, as

**(2.118)** $$f^{\langle n \rangle} = \sum_i \alpha_i f_i, \quad f_i = f(x_i)$$

where $\alpha_i$ are the weights[**Gea71**]. For example the second derivative at some point $x_i$ can be approximated by

**(2.119)** $$f_i^{''} = \alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3$$

where

**(2.120)** $$\alpha_1 = \frac{1}{\Delta x^2}, \quad \alpha_2 = -\frac{2}{\Delta x^2}, \text{and} \quad \alpha_3 = \frac{1}{\Delta x^2}$$

We get these weights by adding equation (2.105) to equation (2.108) You can try this out for yourself to check the derivation and also to get the truncation error.

Again one can try to derive one-sided expressions to the second derivative just as we did the first derivative. In a similar fashion, we can find expressions for the third and fourth derivatives. Here is a third derivative that is forward biased.

**(2.121)**

$$f_i''' = \frac{f_{i+2} - 3f_{i+1} + 3f_i - f_{i-1}}{\Delta x^3} - \frac{\Delta x}{2} f^{(iv)} - \frac{\Delta x^2}{4} f^{(v)} + \ldots$$

We say forward biased because it has the $i-1$ grid point included in the expression. If we derived an expression with the $f_{i+3}$ instead of $f_{i-1}$ we would had a a pure forward difference expression.

**(2.122)**
$$f_i^{\langle iv \rangle} = \frac{f_{i+2} - 4f_{i+1} + 6f_i - 4f_{i-1} + f_{i-2}}{\Delta x^4}$$

Equation (2.122) gives a centred approximation to the fourth derivative. Note that in all of these expressions, we have assumed $\Delta x$ is the same everywhere.

A final note on the expression “order”. Take equation (2.121). The truncation error is given by

Convergence is first order

Determines order of representation of $f$, in this case order 3.

**(2.123)**
$$-\frac{\boxed{\Delta x}}{2}\boxed{f^{(iv)}}$$

As $\Delta x \to 0$ this error goes to zero linearly or the representation has first order convergence. On the other hand the error is zero for polynomial up to the order three, that is, the representation is third order which we infer from the fourth derivative in the truncation error.

---

**Assignment 2.14**

Make sure you are able to derive the difference approximation to derivatives of various orders.

(1) Verify the truncation error for the third derivative given in equation (2.121).

(2) Derive the expression for the fourth derivative given in equation (2.122) and the associated truncation error in the representation.

---

## Differential Equations

Now that we know how to represent derivatives and functions on the computer, we are in a position to represent differential equations. Let us consider a simple differential equation of the form

**(2.124)**
$$\frac{du}{dt} = f(u,t), \quad u(0) = u_o$$

In order to solve for a particular problem we need the boundary conditions. In this case we may have $u(0) = u_o$.

If we were to use a first order forward difference representation for the derivative at some time, $t^q$, and evaluated the right hand side at $t^q$, we would get

**(2.125)**
$$\frac{(u^{q+1} - u^q)}{\Delta t} = f(u^q, t^q)$$

where, $u^q$ is short hand for $u(t^q)$.

So we have the "Finite Difference Representation" for equation (2.124), what do we do with it. We use it to build an automaton on our computer to solve this problem over some interval of time $(0, T)$. We can rewrite equation (2.125) as

**(2.126)**
$$u^{q+1} = u^q + \Delta t f(u^q, t^q)$$

If we take $t^0 = 0$, for a given $\Delta t$ we an find $u^1$ as

**(2.127)**
$$u^1 = u^0 + \Delta t f(u^0, 0)$$

We already know that $u^0 = u_o$. Hence, we can find $u^1$. We can repeat this process to find $u^2$, $u^3$...$u^q$...

This scheme is called the Euler's explicit scheme. Using the definition of the derivative we could also write equation (2.124)

**(2.128)**
$$du = f(u, t)dt$$

or the integral form

**(2.129)**
$$\int_{u^0}^{u(t)} du = u(t) - u^0 = \int_{t^0}^{t} f(u(\tau), \tau)d\tau$$

We could discretise the integral on the right hand side of equation (2.129) using the rectangle rule and we would get the automaton given by equations (2.125) and (2.126). The point is that the same automaton may be obtained through different paths and for historical reasons may have different names. The objective is to recognise this and not get too hung up on names. Instead of using the rectangle rule one could use the trapezoidal rule and would simultaneously get the modified Euler's scheme which is given by two steps

$$u^* = u^q + \Delta t f(u^q, t^q) \tag{2.130}$$

$$u^{q+1} = u^q + \Delta t f(u^*, t^q) \tag{2.131}$$

The second equation can of course now be iterated to get the iterated Euler's scheme and so on.

You can try to use this to solve simple differential equations for which you already know the solution. This way you can compare your computed solution to the actual solution.

---

**Assignment 2.15**

Try solving the following equations:

$$\frac{du}{dt} = t^2, \quad u(0) = 0 \tag{2.132}$$

and

$$\frac{du}{dt} = \cos t, \quad u(0) = 0 \tag{2.133}$$

Try these with different values of the boundary conditions, different values for $\Delta t$. Solve them for $t \in (0, 4\pi)$.

---

We are generating a sequence made up of $u^q$ in all our schemes. You will see through out this book that we tend to do this very often. In mathematics, we have studied the properties of sequences. Mostly, we were interested in sequences that converged. Here, we explicitly define a sequence to be **divergent** if the magnitude of its terms eventually get larger and larger. One way to guarantee that the sequence does not diverge is to make sure that the gain in magnitude across any given time step is not greater than one. This is a stronger requirement than our definition, since we do not mind something that grows and decays as long as it does not diverge. Anyway, this is conventionally what is done and we will be happy if

$$\left|\frac{u^{q+1}}{u^q}\right| < 1 \tag{2.134}$$

We are interested in sequences that do not diverge.

There are many books that present this material on finite differences in a variety of ways. An early reference is a book by Boole [**Boo60**].

Let us now pay attention to one critical component of this whole process. The intervals on which our functions are represented.

## Grid Generation I

It is clear from our attempts at representing functions, the intervals on which our basis functions are defined are very important. If you look closely at the definitions of the box function and the hat function, you will see that there is no real constraint on the intervals on which they are defined to be equal intervals. Which means that, as needed, one could take unequal intervals.

This gives rise to the question, for a given function to be represented using hat functions, is there an optimal distribution of intervals?

The intervals are easily defined by the end points. The end points of our intervals will be called “nodes”, “grids”, “grid points”. We can ask the following question. If we were approximating a function on some domain, how many intervals does it require? That simply translates to: How many grid points do we require? Where should the grid points be located? The process of answering this question by creating the set of points that we use to represent functions, derivatives and differential equations is called **grid generation**. We will take a brief peek at it now. We will take a more detailed look at grid generation later in the book.

The easiest way to generate grids is to get a uniform distribution of grid points. Say you had ten intervals with which you wanted to represent the function $\sin(x)$ on the interval $[0, \pi]$. This means you have eleven grid points. Using Haar functions what do you get?

One thing we can do is to redistribute the grid so that there are less grid points where the function does not vary much (the derivative is small) and more grid points where the function varies rapidly (the derivative is large)

We can now ask the question: Is there an optimal distribution of the grid points? What do we mean by optimal? We want to use eleven grid points and get the best representation of $\sin x$. So, if $h_i(x)$ are the Haar functions from $(x_i, x_{i+1})$ then we want to find the $\{x_i\}$ such that

**(2.135)** $$E = ||e(\{x_i\})|| = \sqrt{\int_0^\pi \left\{ \sin x - \sum_{i=0}^{10} a_i h_i(x) \right\}^2 dx}$$

is a minimum. Remembering that $x_0$ and $x_{10}$ in this example are fixed, we can differentiate the equation with respect to $x_i, i = 1, \ldots, 9$ and set it to zero. If we solve the resulting system of equations we should get a good distribution of the nine interior points. This is easy to describe in this fashion. It is quite difficult to do. Remember that $a_i = \langle h_i, \sin x \rangle$. We will see problems like this in a later section 7. The sub-discipline is called adaptive grid generation. The name is an indication that the grid adapts to the function that we are trying capture.

Can we do something with whatever we have learnt so far? We can look at equation (2.135) and see what it signifies. It is an accounting of the total error in the interval of our representation. We have derived expressions for the local error in terms of the truncation error. So, we can find the total error by adding up the magnitudes of the truncation errors. We now have an estimate of $E$. If we were using box functions $E$ can be written as

**(2.136)** $$E = \sum_i \left| \cos(x_i) \Delta x_i \right|, \quad \Delta x_i = (x_{i+1} - x_i)$$

This $E$ is the total error over the whole domain. Given that we have $N$ intervals with which we are approximating our function the average error is $E/N$.

Very often, we do not have the solution at hand to get the grid. One obvious thing to do is to get an approximate solution on an initial grid and then adapt the grid to this solution. We can then proceed to get a solution on the new grid and repeat the process. This is not always an easy proposition. If we have an idea of the solution we can cluster the grids ahead of time.

If for example we know that the function we are representing has a large derivative at $x_0$ in the interval $[x_0, x_1]$, we can generate a grid to accommodate this behaviour. The easiest thing to do is geometric clustering or stretching. If we knew a solution varied very rapidly near the origin we could take fine grids near the origin, $x_0 = 0$, and stretch them out as we move away from the origin. For example, We could take the first increment to be $\Delta x_o$. Then, if we propose to stretch the grid using a geometric scheme with a stretching factor $\alpha$ we could then take the next increment to be $\Delta x_1 = \alpha \Delta x_o$. In general we would have $\Delta x_{i+1} = \alpha \Delta x_i$

We have some idea as to how to take the description of a function $u$ given by a differential equation and obtain a representation for $u$ on the computer.

### Important ideas from this chapter

- We approximate the real line on the computer using a "finite" number of points chosen from the real line: each point represents an interval around it.
- Arrays are stored in a fashion on the computer so as to make linear sequential access to array elements fast. The way they are indexed by the programmer can have a major influence on the performance of program on that computer.
- We are hunting for functions.

- We need to organise these functions in a manner that the hunt is made easy.
- To this end we treat functions as existing in function spaces[**Moo85**].
- If the supports of functions do not overlap, they will be orthogonal to each other.
- One can find many classes of basis functions to represent the function of interest.
- The highest frequency that one can represent on a given grid is determined by the size of that grid and the polynomial order of the representation. If we know that we want to use a linear representation a function with a frequency of the order of n then our grid needs to be at least 10n and preferably 40n.
- There are many ways by which functions and derivatives can be approximated on the computer.
- On a given interval $(a, b)$, the expression $[f(b)-f(a)]/[b-a]$ is a first order approximation of the derivatives $f'(a)$ and $f'(b)$; It is a second order estimate of the derivative $f'([a+b]/2)$.
- The increments in the finite difference approximation of the derivative can't be made very small. There is a point beyond which roundoff error starts to grow.
- A scheme for representation of a function has an "order" of representation and an "order" of convergence.
- Approximation to the derivatives can be used to approximate differential equations. These representations of the differential equation can be used, very often, to get an approximate solution to the differential equation.

# Simple Problems

We have enough machinery in place to tackle a few simple and classical problems. We are going to do the following here. We will take the naive approach and try some obvious schemes for solving simple equations. We will try to develop the analysis tools that help us to ask and answer questions such as

**One:** Is the proposed technique going to produce anything at all?
**Two:** Are we generating garbage or a solution?
**Three:** How does the technique behave?
**Four:** Can we do better?

These questions help us improve on the simple schemes. Some of the improvements we will study in this chapter. Others we will study in latter chapters.

We propose to look at equations which are prototypes for a class of problems. These are: Laplace's equation which happens to the prototypical elliptic equation, the heat equation, which will represent the parabolic problems, and finally the wave equation for the hyperbolic problems. We will see more about the nature of these problems and shed some light on the classification at the end of the chapter.

## Laplace's Equation

Laplace's equation is a good place to start our discussions. It is easy to conjure simple problems that it describes. The corresponding program is easy to write and is well behaved. As we shall see, it is amenable to the simple-minded analysis that is done in this book [**ZT86**], [**Ame77**], [**Arn04**], [**Sne64**].

Let's first place Laplace's equation in a physical context. Consider the irrotational flow of a fluid. We will assume for the purpose of this discussion that the flow is two-dimensional and incompressible. The equations governing the motion of fluid are derived in section 5.2. The law of conservation of mass can be stated as

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3.1}$$

$u$ and $v$ are the velocity components along $x$ and $y$ respectively. As the flow is irrotational, we can define a potential function $\phi(x, y)$ such that

$$u = \frac{\partial \phi}{\partial x}, \quad v = \frac{\partial \phi}{\partial y} \tag{3.2}$$

Substituting back into the equation (3.1) we get the potential equation or Laplace's equation.

Laplace's equation is the prototype equation for an elliptic problem [**ZT86**]. It should be pointed out here that the equation is referred as Laplace's equation or the Laplace equation. In two dimensions, using the Cartesian coordinate system, the equation is

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \tag{3.3}$$

Since the flow is irrotational and two-dimensional, we also know from the definition of vorticity that

**(3.4)**
$$\omega_z = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} = 0$$

The stream function is defined such that

**(3.5)**
$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}$$

If this is substituted into equation (3.1), we see that the stream function generates an associated velocity field (equation (3.5)) that automatically satisfies the equation governing conservation of mass. In turn, if we were to substitute from equation (3.5) into equation (3.4), we see that the stream function also satisfies Laplace's equation.

**(3.6)**
$$\nabla^2 \psi = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0$$

In the discussions that follow, one could use either the stream function or the potential function. In using these equations to describe the solution to a problem, the boundary conditions will depend on whether the stream function is being used or the potential function is being used.

Consider the following problem.

We are interested in a unit square in the first quadrant as shown in Figure 3.1. Laplace's equation governs the solution within the unit square as indicated in the figure. It would be convenient at this point if we had a problem for which we had a closed-form solution. Then, we could act as though we did not

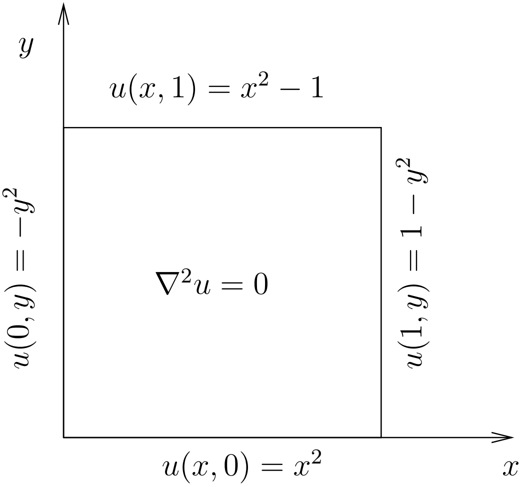

**Figure 3.1. Problem definition with Laplace's equation on a unit square**

have the solution and proceed to solve the problem and check the answer that we get. To this end, the boundary conditions are chosen so that the solution is $\phi(x,y) = x^2 - y^2$ (verify that this is a solution). So, we have the problem definition as shown in Figure 3.1. Well, what does this mean? We want to find a $\phi(x,y)$, that satisfies equation (3.3) everywhere inside the square and satisfies the boundary conditions given. Boundary conditions where the dependent variable, in this case $\phi$, is specified are called **Dirichlet boundary conditions**.

Remember that by "$\phi$ satisfies the equation" we mean that we can substitute it into the equation and find that we do indeed have an equation: the left hand side equals the right hand side. In order to substitute $\phi$ into the equation, we need to be able to evaluate the second derivatives with respect to $x$ and $y$. So, we are already searching for $\phi$ within a class of functions that have second derivatives everywhere inside the unit square. You may have studied techniques to solve this problem analytically. Here, we are going to try and get an approximation for $\phi$. This means that we will have approximations for its second derivatives. Consequently, we will have an approximation for Laplace's equation. This then is our plan of action.

Given three points that are $\Delta x$ apart, we have already seen that the second derivative of $f(x)$ can be approximated at the middle point in terms of its neighbours as

$$\frac{\partial^2 f}{\partial x^2} = \frac{f(x + \Delta x) - 2f(x) + f(x - \Delta x)}{\Delta x^2} \tag{3.7}$$

We have deliberately used partial derivatives since $f$ could be a function of $y$. Since the idea is to use this finite difference approximation for the second derivatives in both $x$ and $y$, we will consider five points as indicated in Figure 3.2 to estimate these derivatives.

We can then approximate both the $x$–derivative and the $y$–derivative at the central point $(x, y)$. So, Laplace's equation (3.3) can be rewritten at the point $(x, y)$ as

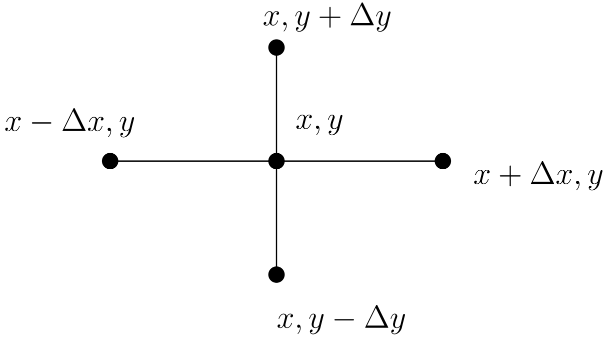

**Figure 3.2. Points used to approximate Laplace's equation in two spatial dimensions.**

**(3.8)**

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}\approx \frac{\phi(x+\Delta x,y)-2\phi(x,y)+\phi(x-\Delta x,y)}{\Delta x^2}+ \frac{\phi(x,y+\Delta y)-2\phi(x,y)+\phi(x,y-\Delta y)}{\Delta y^2}=0$$

We can solve for $\phi(x,y)$ in terms of the $\phi$ at the neighbouring points to get

**(3.9)**

$$\phi(x,y) = \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)} \left\{ \frac{\phi(x+\Delta x, y) + \phi(x-\Delta x, y)}{\Delta x^2} \right. \\ \left. + \frac{\phi(x, y+\Delta y) + \phi(x, y-\Delta y)}{\Delta y^2} \right\}$$

What are we actually doing when we solve this equation? For clarity, we have so far taken $\Delta x$ and $\Delta y$ to be constant. To get a better picture, we will set $\Delta x = \Delta y = h$ as shown in Figure 3.3. In this case, equation (3.9) reduces to

**(3.10)**

$$\phi(x,y) = \frac{\phi(x+h,y) + \phi(x-h,y) + \phi(x,y+h) + \phi(x,y-h)}{4}$$

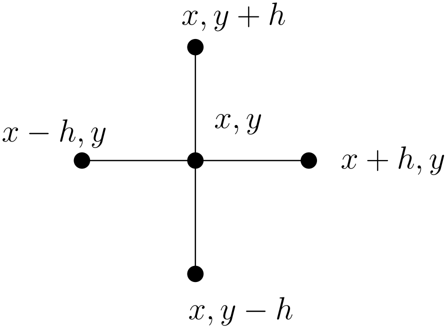

**Figure 3.3. Points employed in approximating Laplace's equation, $\Delta x = \Delta y = h$**

Referring to Figure 3.3, we see that the value of $\phi$ at the point $(x, y)$ is in fact the average of the values from the neighbouring points. We can use this to generate a solution to the problem numerically.

We know how to get an approximation of $\phi$ at a point based on four neighbours. Clearly, we cannot compute $\phi$ at every point in the unit square using equation (3.10). So, we represent the unit square with a discrete set of points. We refer to these points as **grid points**. We will now try to approximate the differential equation on these points. We can identify a grid point by its location. However, it is easier for us to index them in some fashion. Figure 3.4 shows one such arrangement. We use two indices to identify a grid point. $i$ is an index along the $x$ direction and $j$ is an index along the $y$ direction. The coordinates of the grid point $(i, j)$ in general are $(x_{ij}, y_{ij})$. Since we constrained $\Delta x = \Delta y = h$, and the mesh is Cartesian the $(i, j)$ grid point in fact has coordinates $(x_i, y_j)$. The $\phi$ approximation at that point would be $\phi_{ij}$.

If we focus again on one grid point $(i, j)$, we get

$$\phi_{ij} = \frac{\phi_{i-1j} + \phi_{i+1j} + \phi_{ij-1} + \phi_{ij+1}}{4} \tag{3.11}$$

or to put it in a programming style

$$\phi_{ij} = 0.25 * \{\phi_{i-1j} + \phi_{i+1j} + \phi_{ij-1} + \phi_{ij+1}\} \tag{3.12}$$

The $\phi_{ij}$ on the boundary grid points are determined using the given boundary conditions. At the grid points in the interior we use equation (3.11). So, we do the averaging only at the internal grid points. Taking the average at one grid point is called **relaxing** the grid point. In order to start taking the averages

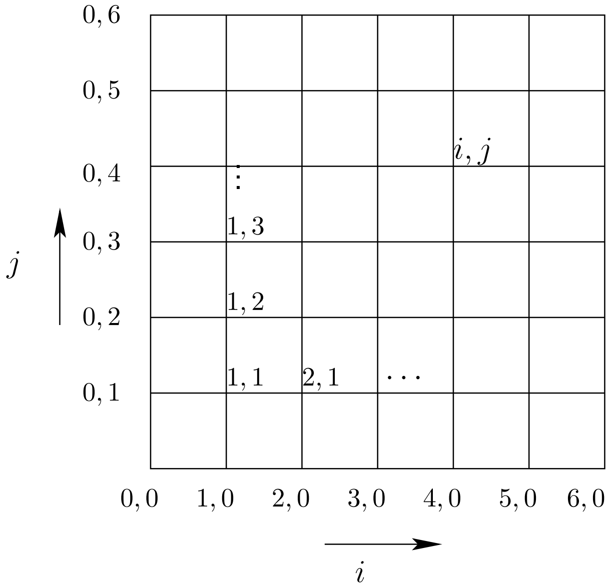

**Figure 3.4. Sample grid to represent and solve Laplace's equation**

we need to assume some initial value. We could, for example, assume all the interior $\phi$ values to be zero. That is, $\phi_{ij}^0 = 0$ for the interior values. What does the superscript $0$ mean? We propose to calculate a new set of values $\phi_{ij}^1$ by averaging the $\phi_{ij}^0$s. This is called one **iteration** or one **relaxation sweep**. Of course, we only iterate on the interior points. By this, we mean that we do not change the boundary values of $\phi_{ij}^1$. The $\phi_{ij}^1$ is, hopefully, a better approximation to $\phi$ than is $\phi_{ij}^0$. We can then iterate one more time to get $\phi_{ij}^2$. $\phi_{ij}^0$, $\phi_{ij}^1$, $\phi_{ij}^2$ ...$\phi_{ij}^q$...

are called **iterates** or **candidate solutions**. We can now write an iterative version of equation (3.10) as

$$\textbf{(3.13)} \qquad \phi_{ij}^{q+1} = 0.25 \times \left\{ \phi_{i-1j}^{q} + \phi_{i+1j}^{q} + \phi_{ij-1}^{q} + \phi_{ij+1}^{q} \right\}$$

where $q$ is the current iteration level at which we have an approximation and $q+1$ corresponds to our new and improved approximation. This is the **Jacobi iteration scheme [GL83]**. It is also called a **simultaneous relaxation** scheme, since the averaging can be done in parallel. When do we decide to quit iterating? For what value of $q$ can we stop taking averages? In order to answer this critical question, we need to take a good look at what we are doing? We seem to be generating a sequence of $\phi_{ij}$s. So we are really asking the question: when does the sequence $\phi^n$ converge? We could go with the answer: when $||\phi^{q+1} - \phi^q|| < \epsilon_c$. This is called the **convergence criterion** for the iterative scheme. How do we evaluate it? One way would be

$$\textbf{(3.14)} \qquad \|\phi^{q+1} - \phi^{q}\| = \sqrt{\sum_{i,j} \left( \phi_{ij}^{q+1} - \phi_{ij}^{q} \right)^2} \quad < \epsilon_c$$

where $\epsilon_c$ is specified by us. Let us write the steps involved in this discussion so that you can actually code it.

**One:** At any grid point $(i, j)$ we can define a $\phi_{i,j}^{q}$. The $q$ superscript tells us that it is the $q^{th}$ iterate or approximation.

**Two:** In our problem the $\phi_{i,j}^{q}$ is given on the boundaries. This is called a **Dirichlet boundary condition**.

**Three:** In order to use equation (3.13), we need $\phi^0_{i,j}$ in the interior. We will assume this value, for example, $\phi^0_{i,j} = 0$.

**Four:** We can now repeatedly apply equation (3.13) to find the "next" and hopefully better approximation to $\phi_{i,j}$.

**Five:** We stop iterating when the condition given by equation (3.14) is satisfied. When this occurs, we say our code has converged.

---

**Assignment 3.1**

(1) Write a program to solve Laplace's equation on a unit square, see Figure 3.1. Use the boundary conditions provided there.

(2) You can iterate away, till convergence.

(3) Try solving the problem with grids of various sizes: $[11 \times 11], [21 \times 21], [41 \times 41], [101 \times 101] \ldots [m \times m]$.

(4) Pick three different convergence criteria: $\epsilon_c = 10^{-2}, 10^{-4}$and Call $c = ||\phi^{q+1} - \phi^q||$ the change in solution. For a given grid $[m \times m]$, define $N(m, \epsilon_c)$ as the last value of $q$, when the code has converged. That is $c < \epsilon_c$.
   (a) Plot $c$ versus $q$.
   (b) For each $m$, plot $N$ versus $\epsilon_c$.
   (c) For each $\epsilon_c$, plot $N$ versus $m$.
   (d) For each $\epsilon_c$, plot $N$ versus $(m-2)^2$.

---

You may have already written a solver for Laplace's equation using the scheme we just developed. The hope is that you have

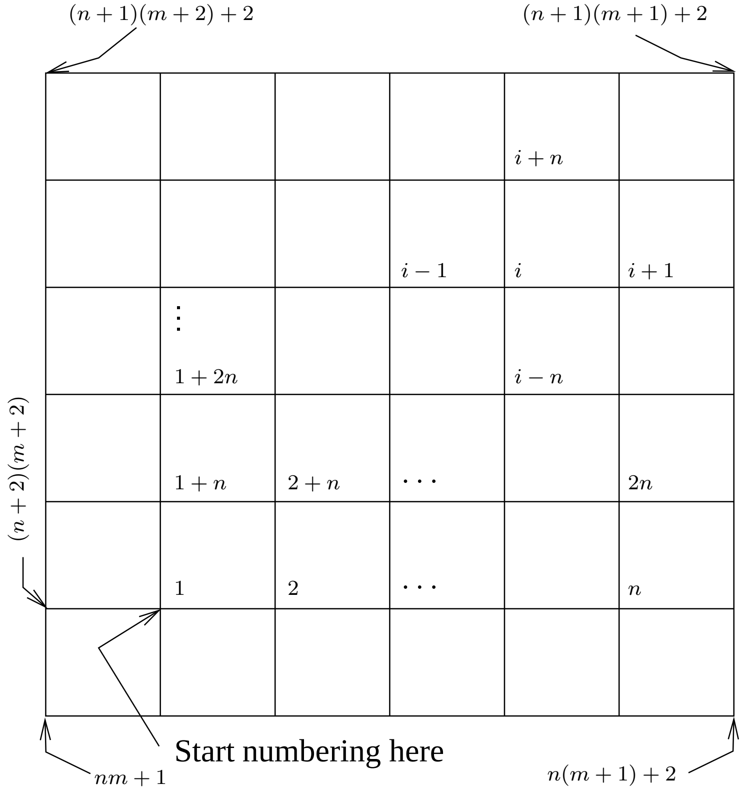

**Figure 3.5. A more convenient numbering of the grid points. It keeps the interior points together. The boundary points are kept together.**

been playing around with it trying to learn as much as possible from that code. In the assignment, I have suggested somethings that you can try out. What I mean by playing around with the code is that you try out a variety of boundary conditions, grid sizes, convergence criteria, the order in which the points are picked for iteration. How about if we pick points at random and take averages. Try out things like this and other things that may strike you as you go along. These are a few examples to start you off "playing around" with your code. I will make other suggestions as we go along.

Playing around with the code also means paying attention to what you are doing and coming up with something a little different, maybe better. If you look at the way your program proceeds, we see that we are sweeping our problem domain as given in Figure 3.5, left to right, bottom to top. Meaning, if we are relaxing any grid point $(i, j)$, the grid point to the left, $(i-1, j)$, and the one below it, $(i, j-1)$, have already been averaged in the current iteration or sweep. We have a choice of using the old values for the relaxation procedure we are about to perform or the latest value. If we use the latest values, we could rewrite our iteration equation as

$$\phi_{ij}^{q+1} = \frac{\phi_{i-1j}^{q+1} + \phi_{i+1j}^{q} + \phi_{ij-1}^{q+1} + \phi_{ij+1}^{q}}{4} \tag{3.15}$$

This is the **Gauss-Seidel iterative scheme** [**HY81**],[**GL83**]. Since the averaging can only be done one after the other, this is also called **successive relaxation**. Look closely at the two equations (3.13) and (3.15). The difference is that in equation (3.15) we are using the latest information as and when it is available. Does the code run faster? Run the two and find out.

**Assignment 3.2**

(1) Repeat assignment 3.1 using the Gauss-Seidel scheme.
(2) Plot $c$ versus $q$ for both Jacobi and Gauss-Seidel schemes on a semi-log scale. ($c$ is plotted on the log-scale). Compare the slopes of the two graphs.
(3) Find the above slopes for various grid sizes and tabulate. How does the slope depend on the size of the grid? Is the relationship between the slopes corresponding to the two schemes independent of the grid size?
(4) What about the CPU times or the run times for the two schemes. Plot time versus number of interior grid points. If you fit a straight line to this curve where does it intersect the axis for zero grids?
(5) Compute the time per iteration per grid point for the various grid sizes. Plot it versus grid size to see if goes to an asymptote with number of grid points increasing.

A suggestion that I had made as an example of playing around was to pick grid points at random and take averages. The whole point is that we normally do not pick points at random, we pick them up in a "natural" numerical sequence. Is there a more convenient way to number these grid points? It all depends on what we want. The way we have indexed grid points so far seemed pretty convenient and natural. How about the one shown in Figure 3.5? One could number all the interior points in a sequential order with one index say $i$ as shown in

Figure 3.5. The boundary points can be numbered separately after that. This has done two things for us.

(1) It has allowed us to represent $\phi$ using a one-dimensional array instead of a two-dimensional array.

(2) It has clustered all the interior points together at the beginning (or top) of the array and all the boundary conditions at the end (or bottom) of the array.

Incidentally, if you do not care to have all the interior points clustered you could number the grid points serially starting at the bottom left hand corner. However, we will stick to the version shown in Figure 3.5.

Referring still to Figure 3.5, an interior grid point with index $i$, has a left neighbour $i-1$, a right neighbour $i+1$, a bottom neighbour $i-n$ and a top neighbour $i+n$. Here $n$ is called the stride. We have to be careful now. What are the neighbours of the first interior grid point? It has neighbouring points from the boundary. At a generic interior point, the approximation for Laplace's equation becomes

**(3.16)**
$$\phi_i = \frac{\phi_{i-1} + \phi_{i+1} + \phi_{i-n} + \phi_{i+n}}{4}$$

We can then proceed to iterate again and solve the problem. Nothing should change in the solution as all we have done is change the symbols. Redo the assignment and make sure there is no change to your solution and any of the other parameters that you have checked (like convergence plots and so on).

This renumbering of the grid points gives us another way of looking at this problem and solving it. Let us first consider the way we have numbered the grid points. Take another look at this equation for a unit square at the origin. $n = m$ since the $\Delta x = \Delta y$. In reality, we have $n^2$ unknown interior points and $n^2$ equations. We are solving the equations here using either Gauss-Seidel or Jacobi schemes. So, the equation (3.16) should be written as part of a system of equations. The $i^{\text{th}}$ equation of this system of equations can be written as

**(3.17)** $$\phi_{i-n} + \phi_{i-1} - 4\phi_i + \phi_{i+1} + \phi_{i+n} = 0$$

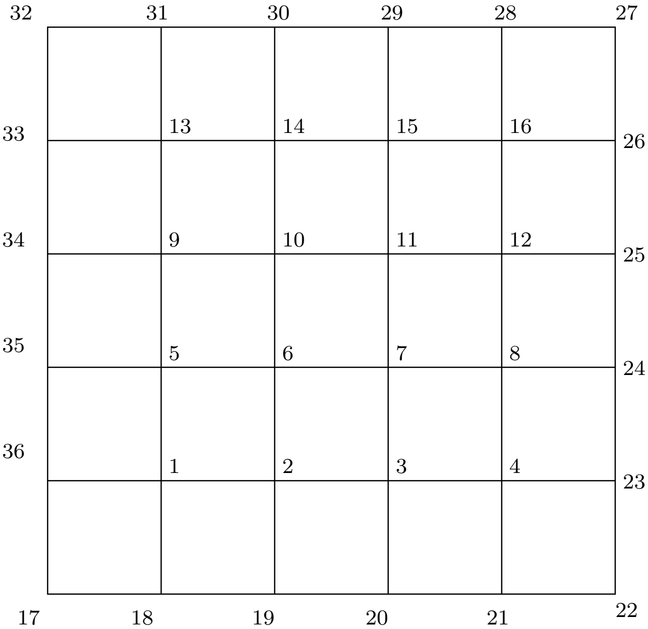

**Figure 3.6. The numbering for a $[6 \times 6]$ grid**

We will write the full system for a $[6 \times 6]$ grid with 16 interior points as shown in Figure 3.6. We have 16 unknowns and the corresponding 16 equations. These equations can be written as

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{3.18}$$

where $\boldsymbol{x}$ is vector made up of the $\phi_i$ on the interior grid points. This equation is written out in full detail in equation (3.19).

$$\textbf{(3.19)} \quad \begin{pmatrix} -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 & \mathbf{0} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & \mathbf{0} & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & \mathbf{0} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{0} & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & \mathbf{0} & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{0} & -4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 \end{pmatrix} \left\{ \begin{matrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \\ \phi_{10} \\ \phi_{11} \\ \phi_{12} \\ \phi_{13} \\ \phi_{14} \\ \phi_{15} \\ \phi_{16} \end{matrix} \right\} = \left\{ \begin{matrix} -\phi_{36} - \phi_{18} \\ -\phi_{19} \\ -\phi_{20} \\ -\phi_{21} - \phi_{23} \\ -\phi_{35} \\ 0 \\ 0 \\ -\phi_{24} \\ -\phi_{34} \\ 0 \\ 0 \\ -\phi_{25} \\ -\phi_{33} - \phi_{31} \\ -\phi_{30} \\ -\phi_{29} \\ -\phi_{28} - \phi_{26} \end{matrix} \right\}$$

From this it is clear that for an arbitrary grid with $[n \times m]$ interior grid points the stride is $n$ (or $m$). The diagonal will have a $-4$ on it. The sub-diagonal will have a $1$ on it if possible. The super-diagonal will have $1$ on it if possible. All the possible diagonals above and below will be zero excepting the $n^{th}$ above and below which will be one. Where it is not possible to place a one on any diagonal the corresponding entry will show up in $\boldsymbol{b}$ as a term subtracted from the right hand side.

This matrix is clearly a sparse matrix. Most of the entries in a **sparse matrix** are zero as is the case here. For this reason, we very rarely construct/assemble this matrix in CFD. Once we have identified the problem as that solving a system of equations, many ideas will come to mind. For small problems, you may actually consider assembling the matrix and using a direct method like Gaussian elimination to solve the problem. However, as we have done here, for bigger problems we usually use an iterative scheme like Gauss-Seidel or Jacobi schemes. There is a very simple reason for this. Direct methods like Gaussian elimination which is identical to LU decomposition involves an elimination step.

$$\boldsymbol{U}\boldsymbol{x} = \boldsymbol{L}^{-1}\boldsymbol{b} \tag{3.20}$$

This is followed by the back substitution step to get our solution.

$$\boldsymbol{x} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{b} \tag{3.21}$$

Both $U$ and $L$ are triangular matrices. The calculations, once you have the decomposition, are easy to do as at any given time you have an equation with just one unknown. For example, the first step in equation (3.20) would be to solve for the first unknown.

$$\{\boldsymbol{Ux}\}_1 = \boldsymbol{b}_1/\boldsymbol{L}_{1,1} \tag{3.22}$$

where the subscripts indicate the corresponding component of that matrix. The second equation would be in terms of the known right hand side and the **first term just calculated from the first equation**. At the end of evaluating equation (3.20), we would have all the terms that make up the vector $Ux$. The fact of the matter is that the last term would have been calculated based on all preceding terms. In an $n \times n$ system of equations, $n$ is the number of unknowns, this would have involved of the order of $n(n+1)/2$ calculations. We repeat a similar process in the backward direction to solve for $x$ using equation (3.21). So, it turns out that the very first element of the vector $x$ is a consequence of $n^2$ operations performed one after the other. The potential accumulation of cumulative roundoff error is enormous. However, the solution from this direct method may be a good guess for an iterative scheme. As a personal bias, I do not use direct methods to solve linear system of equations specified by anything more than a $400 \times 400$ matrix, though people have come up with some clever techniques to get around the size issue.

Note that the matrix $\boldsymbol{A}$ is symmetric, has negative terms on the diagonal and positive ones on the off-diagonals. It has a lot of nice properties [**Var00**] based on its structure. Anyway, now that we recognise that we are solving a system of linear equations we can bring to bear other methods from numerical linear algebra [**GL83**] to aid us in the solution of the system.

**Assignment 3.3**

(1) Rewrite a program to solve Laplace's equation on a unit square using the new numbering system.

(2) Is there any difference in the time per grid point per iteration?

(3) Is there a difference in the solution, convergence rate? The solution and convergence rate should be the same as before.

(4) What happens to the convergence rate if we iterate in a fashion alternating relaxation of the interior points between the ends of the array? That is relax the first point, then the last point, then the second point, the last but one point, and so on, till we do the middle point at the end of the iteration.

(5) If you feel up to it, you can try writing a solver using Gaussian elimination, LU decomposition or better $LDL^T$ decomposition. Refer [**GL83**] for more details.

## Convergence of Iterative Schemes

You would have solved Laplace's equation for various program parameters by now. The schemes we have looked at are called iterative schemes. An iterative scheme, as we have seen, generates a sequence of candidate solutions $\boldsymbol{x}^0$, $\boldsymbol{x}^1$, $\boldsymbol{x}^2$,..., $\boldsymbol{x}^q$,.... We want to know when this sequence converges. We asked a similar question earlier when we were writing our code. There we meant to ask, what is the terminating condition for the iterations in our program? Here we are asking: Are there conditions under which the termination condition is never met? Is it possible we run the program and it does not stop? We will take a look at this now.

An iterative scheme involves generating a sequence of solutions employing an equation of the form

$$\boldsymbol{x}^{q+1} = g(\boldsymbol{x}^{q}) \tag{3.23}$$

In particular, from Laplace's equation we have

$$\boldsymbol{x}^{q+1} = \boldsymbol{P}\boldsymbol{x}^{q} + \boldsymbol{C} \tag{3.24}$$

Where $\boldsymbol{P}$ is an iteration matrix. A closer inspection of equations (3.23) and (3.24) indicates that $g(\cdot)$ ($\boldsymbol{P}$, in our case) maps the space containing $\phi$ onto itself. Consider the system of equations with sixteen unknowns given in equations (3.18), and (3.19). The matrix $\boldsymbol{A}$ can easily be written as the sum of three matrices, $\boldsymbol{L}, \boldsymbol{D}, \boldsymbol{U}$. The entries in the sub-diagonals of the $\boldsymbol{L}$ matrix are the entries from the sub-diagonals of $\boldsymbol{A}$. The diagonal and super-diagonal entries of $\boldsymbol{L}$ are zeros. That is the entries of $\boldsymbol{L}$ are

$$l_{ij} = \begin{cases} a_{ij} & \text{for } i > j \\ 0 & \text{for } i \leq j \end{cases} \tag{3.25}$$

Similarly $\boldsymbol{U}$ has entries from the super-diagonals of $\boldsymbol{A}$ as its super-diagonals and zero everywhere else. That is the entries of $\boldsymbol{U}$ are

$$u_{ij} = \begin{cases} a_{ij} & \text{for } i < j \\ 0 & \text{for } i \geq j \end{cases} \tag{3.26}$$

Finally, as you should have guessed, the matrix $\boldsymbol{D}$ is a diagonal matrix with the entries from the diagonal of $\boldsymbol{A}$. In this particular case, $\boldsymbol{D}$ is a diagonal matrix with $-4$ on the diagonal and zeros for all other entries. With this framework in place, we can write the Jacobi scheme as

$$\boldsymbol{x}^{q+1} = \boldsymbol{D}^{-1}\{\boldsymbol{b} - \boldsymbol{L}\boldsymbol{x}^q - \boldsymbol{U}\boldsymbol{x}^q\} \tag{3.27}$$

For an interior point, with no neighbouring boundary point, for example the seventh equation from the system of equations (3.19) is

$$\phi_3 + \phi_6 - 4\phi_7 + \phi_8 + \phi_{11} = 0 \tag{3.28}$$

Rewritten in the form of equation (3.27) it is

$$\phi_7 = \underbrace{\frac{1}{-4}}_{\{\boldsymbol{D}^{-1}\}_{7,7}} \{0 \underbrace{-\phi_3 - \phi_6}_{-\{\boldsymbol{L}\boldsymbol{x}\}_7^q} \underbrace{-\phi_8 - \phi_{11}}_{-\{\boldsymbol{U}\boldsymbol{x}\}_7^q}\} \tag{3.29}$$

Clearly,

$$\boldsymbol{P}_J = \boldsymbol{D}^{-1}\{-\boldsymbol{L} - \boldsymbol{U}\}, \quad \boldsymbol{C}_J = \boldsymbol{D}^{-1}\boldsymbol{b} \tag{3.30}$$

The subscript $J$ is to remind us that it is the iteration matrix corresponding to the Jacobi iteration. In a similar fashion, Gauss-Seidel would be written as

$$\boldsymbol{x}^{q+1} = \{\boldsymbol{D} + \boldsymbol{U}\}^{-1}\{\boldsymbol{b} - \boldsymbol{L}\boldsymbol{x}^q\} \tag{3.31}$$

and the corresponding iteration matrix $\boldsymbol{P}$ is given by

$$\boldsymbol{P}_{GS} = \{\boldsymbol{D} + \boldsymbol{U}\}^{-1}\{-\boldsymbol{L}\} \tag{3.32}$$

This observation enables us to look at convergence from a different perspective. What kind of a guess should I make to start the iterations? So far, you have assumed an initial value for the interior points to be zero. Can we start with something better? Does it matter? It looks like it should matter. What if we took the solution, $\boldsymbol{x}$, as the initial guess? We expect equation (3.24) to give us the same $\boldsymbol{x}$ as the new iterate. The convergence is in one iteration. That is, any number of subsequent iterations will return the same point. Since the iterations seem to stuck on that point, which happily looks as though it is the solution that we are after, the point is called a **fixed point** of equation (3.24). So, trying to solve equation (3.24), is the same as asking for its fixed point.

---

**Assignment 3.4**

Repeat the previous assignment using different initial conditions. Use different constants, $\phi^0 = 1., -1, 2, -2$ at all of the interior points.

---

**Contraction Maps and Fixed Point Theory.** This is a grandiose sounding title. Let us look at a simple problem. Consider the iterative equation in $x$ given by

**(3.33)**
$$x^{q+1} = \alpha x^q$$

$x$ is a real number. For a given finite, real number $\alpha$, equation (3.33) is a map from the real line back onto the real line. Does this equation always have a fixed point? That is, is there a $\xi$ such that $\xi = \alpha\xi$? Is $x = 0$ a fixed point? Yes, for equation (3.33), $x = 0$ is a fixed point. If I guess any old $x^0$, will the iterations always converge to zero? How does $\alpha$ affect the process? Clearly $\alpha > 1$ is not going to get to any fixed point unless the initial guess is $x^0 = 0$. How about if $\alpha = 1$? $\alpha =$

$1$ seems to make every point on the real line a fixed point. Finally, when $\alpha < 1$, we do generate a sequence of numbers that shrink towards the origin. All of this makes sense if we notice that we are generating a sequence $x^0, x^1, \ldots, x^q, \ldots$ and that the ratio test for the convergence of a sequence tells us that we have a fixed point if $\alpha < 1$. We are now looking to get some understanding using this simple example, so that when we encounter a system of equations we are able to cope with it a little better.

Consider a region of points containing the origin $-r \leq 0 \leq r$. If this were in two dimensions we would have called it a circular region. So, we will just call it a circle of radius $r$ about the origin. What happens to the points in this circle when they are run through equation (3.33) for $\alpha < 1$? This is illustrated in Figure 3.7. Clearly, the equation maps into a circle that is smaller, in fact in this case

$$r^{q+1} = \alpha r^q \tag{3.34}$$

This map is called a contracting map for pretty obvious reasons. As $q \rightarrow \infty$ we have $r \rightarrow 0$.

Someone looking for a more precise statement of the contraction mapping can find it many advanced calculus books or in books on real analysis [**Kre89**].

There are many physical examples of contraction mappings. Consider a one-dimensional rod, say made of steel. (You maybe more comfortable with a bar of foam rubber). What happens if we apply an inward force at the two ends? The length of the rod reduces. If we had placed some markings on the rod before

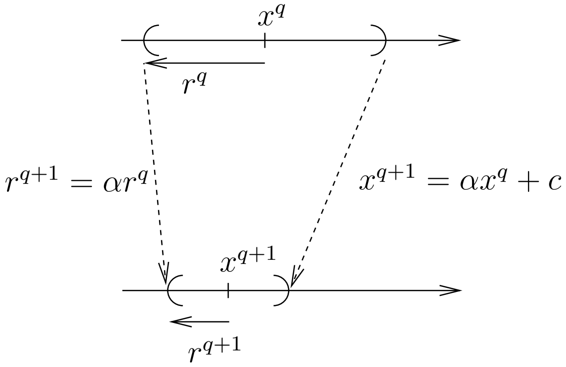

**Figure 3.7. A contraction map causes the "radius" of the interval $r$ to decrease. $\alpha < 1$.**

compression (contraction) we would have seen these markings move to a new position after compression. So we have a map from the original position of a mark to its new position. We will observe, (we are conducting a thought experiment now) that all the markings move closer to each other. That is the transformation that takes us from the initial position of the markings to the final position is a contraction mapping. We can also see that there will be one point that does not move. This is the fixed point of the map/transformation.

Meanwhile, you can ask the question: Isn't it enough if the mapping confines us to a fixed region in space? Do we really need $\alpha < 1$? Consider the map given by

$$x^{q+1} = \begin{cases} \alpha x^q & \text{for } x^q < \frac{1}{2} \\ \alpha(1 - x^q) & \text{for } x^q \geq \frac{1}{2} \end{cases} \tag{3.35}$$

Try this out for various initial conditions $x^0$ and you will see that though we are confined to the unit interval at the origin we may never converge to a fixed point.

---

**Assignment 3.5**

Run the above map for various starting points $x^0$ and $\alpha = 2.0$ and $\alpha = 1.9$

(1) Try $0$, $1$, $\frac{1}{2}$, $0.25$. Draw conclusions.
(2) Try other values of $\alpha$, $0.5$, $1.2$, $1.4$ and so on.

---

In all our endeavours here, we would like to look at spaces that are complete. What do we mean by this? Let us say that we are generating a sequence of solutions. We would like the sequence to converge to a point that is in the space in which we are looking for the solution. A policeman would like the hunt to end within his jurisdiction. If the person being pursued enters another country with which there is no extradition treaty, then the hunt does not converge. So, lets say we have a complete space $S$ and we have a map $\boldsymbol{P}$ that takes a point in $S$ and maps it back into $S$. We could restate this as, if you have some $x \in S$, then

**(3.36)**
$$y = \boldsymbol{P}x, \quad y \in S$$

In fact given an $x^0 \in S$, we can use this to generate a sequence

**(3.37)**
$$x^{q+1} = \boldsymbol{P}x^q, \quad x^q, x^{q+1} \in S$$

This map $\boldsymbol{P}$ is called a contraction mapping if for any $a, b \in S$,

$$d(\boldsymbol{P}a, \boldsymbol{P}b) < d(a, b) \tag{3.38}$$

where $d(a, b)$ is the distance between the points $a$ and $b$ and is called the metric of the space $S$. What equation (3.38) says is that the map $\boldsymbol{P}$ maps the points $a$ and $b$ into points that are closer together. It turns out that if this is the case, we can assert that there is an unique point, $\xi$, in $S$, called a fixed point, such that

$$\xi = \boldsymbol{P}\xi \tag{3.39}$$

That is, $\boldsymbol{P}$ maps this particular point into itself. I have paraphrased the Banach fixed point theorem here.

Fine, we see how this works for a scalar equation. However, we are looking at a system of equations. How does it work for a system of equations? To answer this question we will consider a simple problem using two scalar equations. We will convert this problem into a slightly more complicated problem by performing a rotation of coordinates. You can review the material on matrices given in the appendix B-1 and try out the following assignment.

**Assignment 3.6**
Consider two equations

$$x^{q+1} = \alpha_x x^q \tag{3.40}$$

$$y^{q+1} = \alpha_y y^q \tag{3.41}$$

Write a program to try this out, or better still plot a the region as it is mapped by hand on a graph sheet. Take both $\alpha_x$ and $\alpha_y$ to be in the range $(-1, 1)$. You can run each equation separately. Choose different combinations of $\alpha_x$ and $\alpha_y$.

We look at the problem in the assignment. As we now expect, if $|\alpha_x| < 1$ the $x$ sequence will converge. Similarly if $|\alpha_y| < 1$ the $y$ sequence will converge. So, in order for the the combined sequence $(x^q, y^q)$ to converge we require $\rho = \max(|\alpha_x|, |\alpha_y|) < 1$. The condition to converge seems coupled. Right now it looks, sort of, contrived. We are at times fortunate to pick the right coordinate system and get a problem that is simple. In this case, the simplicity comes from the fact that the two equations are decoupled. Since, we are not always this fortunate, we will now perform a rotation of the coordinate system to convert this simple problem into something that you would generally encounter.

First, let's rewrite the two equations as a matrix equation. Equations (3.40), (3.41) can be written in matrix form as

$$\begin{pmatrix} x^{q+1} \\ y^{q+1} \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{pmatrix} \begin{pmatrix} x^q \\ y^q \end{pmatrix} \tag{3.42}$$

This equation can be rewritten as

$$\vec{x}^{q+1} = \mathbf{\Lambda}\vec{x}^{q} \tag{3.43}$$

where

$$\vec{x} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.44}$$

and

$$\mathbf{\Lambda} = \begin{pmatrix} \alpha_x & 0 \\ 0 & \alpha_y. \end{pmatrix} \tag{3.45}$$

The iteration matrix $\mathbf{\Lambda}$ looks nothing like the matrix $\boldsymbol{P}$ that we got with Gauss-Seidel scheme for Laplace's equation. Let us rotate the coordinate system through an angle $\theta$. We can do this by pre-multiplying equation (3.42) by the matrix

$$\boldsymbol{R} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{3.46}$$

You will notice that this matrix performs a pure rotation of any vector on which it is applied. By pure rotation, we mean that it does not perform a stretch. Performing the multiplication, we will get an equation of the form

$$\boldsymbol{x}^{q+1} = \boldsymbol{P}\boldsymbol{x}^{q} \tag{3.47}$$

where $\boldsymbol{P}$ is an iteration matrix given by

$$\boldsymbol{P} = \boldsymbol{R}\mathbf{\Lambda}\boldsymbol{R}^{-1} \tag{3.48}$$

and $\boldsymbol{x} = \boldsymbol{R}\vec{x}$. $\boldsymbol{R}$ is very often called the **modal matrix**. It should be obvious that $\alpha_x$ and $\alpha_y$ are the eigenvalues of the iteration matrix $\boldsymbol{P}$. The largest of these, $\rho$, is called the **spectral**

**radius** of the iteration operator $\boldsymbol{P}$.

$$\rho(\boldsymbol{P}) = \max(|\alpha_x|, |\alpha_y|) \tag{3.49}$$

We have seen that the iteration matrix generates a convergent sequence of $\boldsymbol{x}$'s when the spectral radius of $\boldsymbol{P}$ is less then one. So, getting back to our iterative solution to Laplace's equation, it is clear that if the spectral radius of the matrix is less than one, we have a contraction mapping. Okay then, how do we find the largest eigen-value? Again, there are various techniques by which one can estimate the eigenvalues. In the appendix B-1 we stated Gershgorin's theorem. We restate it here for convenience.

If we have a matrix $\boldsymbol{A}$ and we partition the matrix into two as follows

$$\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{F}, \tag{3.50}$$

where $\boldsymbol{D}$ is a diagonal matrix made up of the diagonal of $\boldsymbol{A}$. Consequently $\boldsymbol{F}$ has a zero diagonal and the off-diagonal entries of $\boldsymbol{A}$. If $d_i$ is the $i^{\text{th}}$ entry of $\boldsymbol{D}$ and $f_{ij}$ are the entries of $\boldsymbol{F}$ then we define and $R_i$ as

$$R_i = \sum_j |f_{ij}| \tag{3.51}$$

Remember that $f_{ii} = 0$. If $z$ is a complex number then the Gershgorin's circle theorem says that the circular disc $|z - d_i| < R_i$ has an eigenvalue in it if it does not overlap other discs. If a set of $\omega$ such discs overlap, then $\omega$ eigenvalues are contained in the union of those discs. Sometimes this is easy to use, sometimes it is not. Look at the Jacobi scheme applied to the two-dimensional Laplace equation on an equi-spaced grid. The diagonal of the iteration matrix $\boldsymbol{P}$ given in equation (3.30) is clearly zero. All the circles are centred at the origin. The off-diagonal terms add up to one in most rows and to less then one when the grid point is near a boundary. We do not have the strict inequality so, in theory, the largest eigenvalue could be one. However, we do not expect the iterations to diverge. The other possibility is to use the structure of the iteration matrix to try and derive an analytical expression for the eigenvalue, which has been done for the Jacobi scheme and for the Gauss-Seidel scheme applied to the two-dimensional Laplace equation as we have studied so far.

## Properties Of Solutions To Laplace's Equation

Okay, now we know what to do and how long to do it for convergence. We have plots of the solution and other parameters related to the behaviour of our code. Can we say something about the solution so that we can be confident that it is a good approximation to the solution to the original problem?

Is it possible for two different people to solve this problem and get different answers? In this particular case, we chose the boundary conditions from a function that already satisfies our equation. How do we know for some other boundary condition whether we got the solution or not? Are there any kind of "sanity checks" that we can make to assure ourselves that we have a solution? Can we say something about the solution to a problem without actually solving it? We will take a shot at it.

Let's now take a look at what we are actually doing. Equation (3.10) says that the value of $\phi$ at a given point is the average of its neighbours. So, what is the average? The average $\phi$ cannot be larger than the largest neighbour or smaller than the smallest one. This is true of all the interior points: No point is larger than the largest neighbour or smaller than the smallest one. Therefore, we can conclude that the maximum and minimum of $\phi$ cannot occur in the interior points. The maximum or minimum will be on the boundary.

Though we have not actually done all the mathematics to make this statement, we will go ahead and extend our conclusion to the Laplace equation by saying:

> **The solution to Laplace's equation will have its maximum and minimum on the boundary**

We can pursue this line of reasoning to come up with an interesting result. If two of us write a solver to Laplace's equation, is it possible to get two different answers? Let us for the sake of argument assume this is possible. So, you get an answer $\phi_1$ and I get an answer $\phi_2$ and both of them satisfy Laplace's equation. That is

$$\nabla^2 \phi_1 = 0 \tag{3.52}$$

$$\nabla^2 \phi_2 = 0 \tag{3.53}$$

They also satisfy the same boundary conditions on $\phi$. Subtracting one from the other tells us that

$$\nabla^2 (\phi_1 - \phi_2) = 0 \tag{3.54}$$

This is possible since Laplace's equation is linear. So, $\phi_1 - \phi_2$ is also a solution to Laplace's equation. Since both $\phi_1$ and $\phi_2$

satisfy the same boundary conditions, $\phi_1 - \phi_2$ is zero on the boundaries. As $\phi_1 - \phi_2$ is a solution to Laplace's equation, its maximum and minimum occur on the boundary. We conclude that $\phi_1 = \phi_2$. Incidentally, we reasoned out the uniqueness of the solution to Laplace's equation using the maximum principle. However, you can go back and check that the same argument holds for the system of equations. This is especially true when solving the system of equations using Jacobi iterations, as we are indeed averaging in that case.

We have two results

(1) The solution to Laplace's equation has its maxima and minima on the boundary of the domain governed by that equation.
(2) The solution is unique.

Both these results are very useful for us to solve the problem. We can verify the solution with a quick check to see if the extremum is on the boundary. We also have the confidence that there is only one solution. So if we get two different solutions, one of is definitely wrong and both of us could be wrong!

## Accelerating Convergence

The assignments solving Laplace's equation should have convinced you that it can take a lot of time iterating till you get the desired convergence. This is especially true for the larger grid sizes. What can we do to get to the solution faster?

Clearly, if we started with the solution as an initial guess, we would converge in one iteration. This tells us that a better initial guess would "get us there" faster. Possible initial conditions for our problem on the unit square in increasing complexity are

- the value on one of the sides can be taken.
- linear interpolation of boundary conditions for two opposites sides of the square.
- the solution to a coarser grid say $[5 \times 5]$ can be used to determine the initial condition on a finer grid, say, $[11 \times 11]$.

You have seen that the spectral radius of the iteration operator $\boldsymbol{P}$ determines convergence. It also determines convergence rate. Let us see how this happens. The iteration that leads to the solution $\phi^h$ to Laplace's equation approximated on a grid of size $h$ is

$$\Phi^{n+1} = \boldsymbol{P}\Phi^n + \boldsymbol{C} \tag{3.55}$$

where, $\Phi^n$ is a candidate fixed point to equation (3.55) and consequently a candidate solution to Laplace's equation. On the other hand $\phi^h$ is the solution to the discrete Laplace's equation and is the fixed point of the iteration equation (3.55). Therefore,

$$\phi^h = \boldsymbol{P}\phi^h + \boldsymbol{C} \tag{3.56}$$

We will designate the difference between the actual solution and the candidate solution as $\mathfrak{e}$, that is

$$\mathfrak{e}^n = \Phi^n - \phi^h \tag{3.57}$$

If we subtract equation (3.56) from equation (3.55) we get

$$\mathfrak{e}^{n+1} = \boldsymbol{P}\mathfrak{e}^n \tag{3.58}$$

This works as both $\boldsymbol{P}$ and $\boldsymbol{C}$, in our case, do not depend upon $\phi$. Now, if we premultiply equation (3.58) by the inverse of the modal matrix $\boldsymbol{R}$, the equation becomes

$$\mathcal{E}^{n+1} = \boldsymbol{\Lambda}\mathcal{E}^{n} \tag{3.59}$$

where, $\mathcal{E}^n = \boldsymbol{R}^{-1}\mathfrak{e}^n$. Equation (3.59) is a decoupled system of equations and if we write out the equation corresponding to the largest eigenvalue

$$\mathcal{E}_\rho^{n+1} = s\rho(\boldsymbol{P})\mathcal{E}_\rho^{n} \tag{3.60}$$

where $s$ is the sign of the largest eigenvalue and $\rho$ is the spectral radius. Or,

$$\left|\frac{\mathcal{E}_\rho^{n+1}}{\mathcal{E}_\rho^{n}}\right| = \rho(\boldsymbol{P}) \tag{3.61}$$

Now, it is clear that for $\rho$ very nearly one, the number of iterations required to get $|\mathcal{E}_\rho|$ below a predetermined $\epsilon_c$ will be large. Let us see if we can come up with an estimate of $\rho$. The $\Phi^n$, for all $n$ and $\phi$ represent functions that satisfy the boundary conditions of our problem. $\mathfrak{e}^n$ and $\mathcal{E}^n$ represent the error, $e(x, y)$, which is zero on the boundaries. We will expand $e(x, y)$ in terms of the Fourier series as

$$e(x, y) = \sum_{l=1}^{N-1}\sum_{m=1}^{N-1} a_l b_m \exp\left\{i\pi\frac{lx}{L}\right\}\exp\left\{i\pi\frac{my}{L}\right\} \tag{3.62}$$

where, $N$ is the number of intervals of size $h$, $(Nh = L)$, and $L$ the side of the square on which we are solving the problem. You will notice that we have restricted the sum to the range $1$ to $N-1$. Wave number zero will not contribute anything as the boundary condition is zero (homogeneous). At the other end of the summation, we already know (see chapter 2.9) that the highest frequency that we can represent is $2\pi(N-1)/2 =$

$\pi(N-1)$. Since both Laplace's equation and our iteration equation are linear, we can check out what happens to each wave number separately.

---

**Assignment 3.7**

Verify that $e_{lm}(x,y) = \exp\left\{i\pi\frac{lx}{L}\right\}\exp\left\{i\pi\frac{my}{L}\right\}$ is an eigen-vector (or an eigen-function) of the Laplace's equation. That is, show that $\nabla^2 e_{lm} = \lambda e_{lm}$.

---

What happens to $e_{lm}$ when we crank it through one iteration of our Jacobi iteration?

$$e_{lm}^{n+1}(x_p,y_q) = 0.25\left\{e_{lm}^n(x_{p+1},y_q) + e_{lm}^n(x_{p-1},y_q) + e_{lm}^n(x_p,y_{q+1}) + e_{lm}^n(x_p,y_{q-1})\right\} \tag{3.63}$$

where, $p$ and $q$ are indices along $x$ and $y$ respectively. Since $x_{p+1}-x_p = y_{q+1}-y_q = h$, we can write this as

$$e_{lm}^{n+1}(x_p,y_q) = 0.25\left(\exp\left\{i\frac{\pi}{L}lh\right\} + \exp\left\{-i\frac{\pi}{L}lh\right\} + \exp\left\{i\frac{\pi}{L}mh\right\} + \exp\left\{-i\frac{\pi}{L}mh\right\}\right)e_{lm}^n(x_p,y_q) \tag{3.64}$$

This gives the gain $g$ in each iteration to be

**(3.65)**

$$g = \left| \frac{e_{lm}^{n+1}(x_p, y_q)}{e_{lm}^{n}(x_p, y_q)} \right| = 0.25 \left| \exp\left(i\frac{\pi}{N}l\right) + \exp\left(-i\frac{\pi}{N}l\right) \right.$$
$$\left. + \exp\left(i\frac{\pi}{N}m\right) + \exp\left(-i\frac{\pi}{N}m\right) \right|$$

Using Euler's formula, we see that

**(3.66)** $$g = 0.25 \left| 2\cos\left(\frac{\pi}{N}l\right) + 2\cos\left(\frac{\pi}{N}m\right) \right|$$

This takes it extreme values for $m = l = 1$ and $m = l = N - 1$. Both give identical values of cosine, except for the sign. So, the maximum gain is

**(3.67)** $$g_{\mathsf{max}} = \cos\left(\frac{\pi}{N}\right) \approx 1 - \frac{1}{2}\left(\frac{\pi}{N}\right)^2$$

So, the spectral radius for the Jacobi iteration, $\rho_J = |\cos(\pi/N)|$. With a hundred intervals this turns out to be approximately $0.9995$. That is really slow. Is there some way by which we can make $g_{\mathsf{max}}$ smaller? We will see one technique next.

**Successive Over Relaxation - SOR.** Since the gain is the ratio of two successive iterates, our hope lies in using the two iterates to reduce the maximum gain. This is an acceleration technique that works by taking a linear combination of the new iterate and the old iterate. Take an $\omega \in (0, 2)$. Why this restriction on $\omega$? We will come to that later. This is how the algorithm works when applied to the Gauss-Seidel scheme. Instead of calling the average of the four neighbouring points “the new iterate”, we treat it as some intermediate value in our computation. We give it a new name, $\phi^*$, so as not to confuse it with the new iterate. Our new algorithm is a two step process as follows.

$$\phi^*_{ij} = \frac{\phi^{n+1}_{i-1j} + \phi^n_{i+1j} + \phi^{n+1}_{ij-1} + \phi^n_{ij+1}}{4} \tag{3.68}$$

$$\phi^{n+1}_{ij} = \omega\phi^*_{ij} + (1-\omega)\phi^n_{ij} \tag{3.69}$$

$\omega$ is called an over-relaxation parameter. The question now is, how do we pick the over-relaxation parameter $\omega$? For the special case that we have here, that of Laplace's equation on a square with $\Delta x = \Delta y = h$, it can be shown that [**Var00**].

$$\rho_\omega = \frac{\rho_J}{1+\sqrt{1-\rho_J^2}} = \frac{\cos\left(\frac{\pi}{N}\right)}{1+\sin\left(\frac{\pi}{N}\right)} \tag{3.70}$$

$\rho_\omega$ is the spectral radius of the iteration operator corresponding to SOR.

For a general problem we may not be able to obtain an expression for the optimal $\omega$. What we usually do is perform several test runs with different values of $\omega$ so as to locate an optimal one. Clearly, there is no sense solving the problem many times since we could just take $\omega = 1$ and solve it just once. Instead, we hunt systematically for the best $\omega$. We could, for instance, iterate ten times with different values of $\omega$ and take the value that resulted in the largest drop in the residue. Different values of $\omega$? How do we pick them? Why did I say “Take an $\omega \in (0,2)$”? We will find out now.

Before we embark on a campaign to hunt down the optimal $\omega$, let us try to understand what we are doing when we use SOR.

Remember that we had shown that we were actually solving a system of linear equations, $\boldsymbol{Ax} = \boldsymbol{b}$. Let us now push this a little further. We had also pointed out earlier that $\boldsymbol{A}$ is symmetric, meaning, $\boldsymbol{A} = \boldsymbol{A}^T$. Consider the following scalar function of $\boldsymbol{x}$

**(3.71)**
$$Q(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}$$

$Q$ maps an $n$-dimensional vector into the real line. How would you find an extremum for this function? We find the gradient with respect to $\boldsymbol{x}$ and set it equal to zero and solve the resulting equation for $\boldsymbol{x}$. We take the gradient. Lo and behold, we get $\boldsymbol{Ax} = \boldsymbol{b}$. So, with symmetric $\boldsymbol{A}$, solving the linear system of equations is like finding the extremum of $Q(\boldsymbol{x})$.

To get a geometrical idea of the SOR algorithm let us see what we are doing in the process of finding this minimum. We will look at a specific simple example so that we understand SOR and get an idea as to why $\omega$ is constrained to the interval $(0, 2)$ in our case. We will look at the quadratic in one single variable. This works for us, since, when we do the operation (averaging) at a grid point, $i$, corresponding to one iteration, we are working on minimising $Q$ along that one dimension $\phi_i$. That is, $b$ has all the other terms absorbed into it. This scenario is graphed in Figure 3.8.

Consider the scenario where we have $x^n$ and would like to find $x^{n+1}$ to get at the next approximation to the minimum. In the one-dimensional case we could solve the problem directly as

**(3.72)**
$$x_{\text{min}} = \frac{b}{a}$$

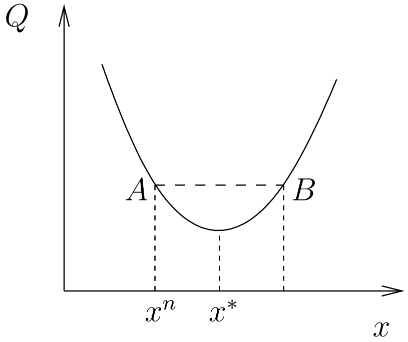

**Figure 3.8. Graph of $Q(x)$ and finding its minimum in one spatial dimension. For simplicity the axis of the parabola is parallel to the $Q$-axis**

and reach the minimum of a quadratic in one iteration. We act as though we are not aware of this and see what we get with SOR. The quadratic that we are minimising is

**(3.73)**
$$q(x) = \frac{1}{2}ax^2 - bx$$

Resulting as we know in the equation for the minimum as $ax = b$. Our iteration equation is

**(3.74)**
$$x^* = -\frac{b}{a}$$

and we use it in the SOR algorithm as follows

**(3.75)**
$$x^{n+1} = \omega x^* + (1 - \omega)x^n$$

where, $x^*$ is the solution that we get from our iteration equation (3.72) We subtract $x^n$ from both sides of this equation to get

**(3.76)** $$\Delta x^n = \omega \Delta x^*$$

Now in the one-dimensional case we would get the exact solution to the minimisation problem in one iteration. $\omega = 1$ would indeed get you the answer. However, what happens if we take $\omega = 0$? Nothing. The solution does not progress. On the other hand, what happens if we take $\omega = 2$, we end up oscillating between point $A$ and $B$ (see Figure 3.8). $\omega < 0$ and $\omega > 2$ would cause the resulting $Q(x)$ to increasing causing the iterations to diverge. We realise from this simple argument that we must seek our optimal $\omega$ in the interval $(0, 2)$.

If you are not sure if this is true for a general quadratic as opposed to the one shown in Figure 3.8, we can work it out algebraically. Equation (3.72) tells us that the minimum that we seek is at $b/a$. Let us say that our guess is off by an amount $d$. This means $\Delta x^* = d$. That is

**(3.77)** $$x^n = \frac{b}{a} + d$$

$Q(x^n)$ then works out to be

**(3.78)** $$Q(x^n) = \frac{1}{2} a \left( \frac{b}{a} + d \right)^2 - b \left( \frac{b}{a} + d \right)$$

The question is, does $Q(x^{n+1})$ take the same value? Where

**(3.79)** $$x^{n+1} = \frac{b}{a} - d$$

It does! That is, if $x^n = b/a + d$, and $x^n = b/a - d$, $Q(x^n)$ is indeed the same as $Q(x^{n+1})$. You can verify this.

To summarise

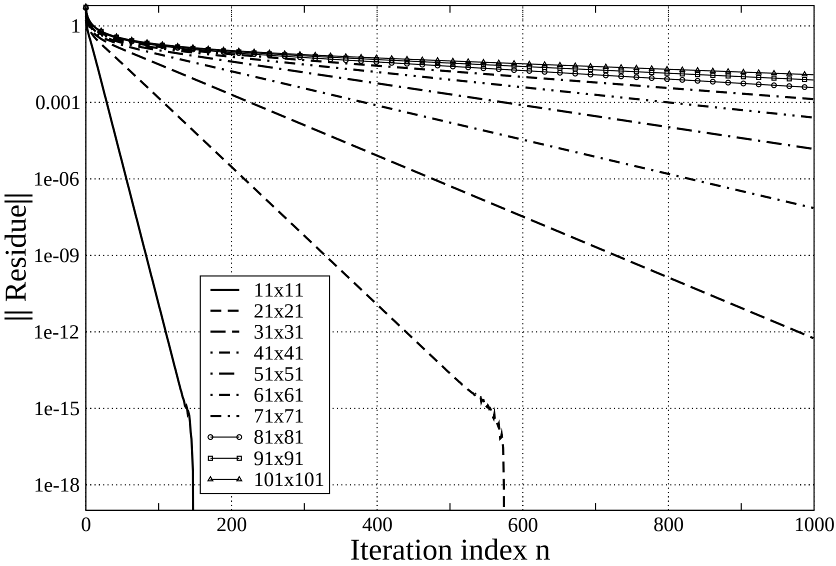

**Figure 3.9.** Plot of the norm of the residue versus iteration for the solution to Laplace's equation using the Gauss-Seidel method for various grid sizes

(1) $\omega \in (0, 2)$, the value of $Q$ decreases. That is $Q(x^{n+1}) < Q(x^n)$.
(2) $\omega = 0$ or $\omega = 2$ the value of $Q$ remains the same. $Q(x^{n+1}) = Q(x^n)$.
(3) Finally if $\omega < 0$ or $\omega > 2$, then the value of $Q$ increases leading to a situation where $Q(x^{n+1}) > Q(x^n)$.

Now that we have an idea of why $\omega$ needs to be in the range $(0, 2)$, how do we find the optimal value? A plot of the residue versus number of Gauss-Seidel iterations is shown in Figure 3.9.

It is clear that as the grid size gets finer, or the number of grids increases for the given domain, that the rate of convergence to the solution slows down. For now it should be noted that the initial drop is rapid for all of the grids. Subsequently, there is a slow down in the convergence rate.

In order to accelerate convergence, we have tried using point SOR. A plot of the residue versus iterations for various $\omega$ is shown in Figure 3.10. The plot of the terminal points is shown in Figure 3.11, where we see the residue after a hundred iterations versus $\omega$ value. Clearly, the most rapid drop occurs near $\omega = 1.8$. How does this seem from the perspective of the contraction mapping we looked at in section 3.3? Are we looking for an $\omega$ that will give us the best contraction?

There are two possible ways by which we can try to get at the optimal $\omega$. One is to run the program till the residue drops by one order in magnitude. The optimal $\omega$ is the one that takes the lowest number of iterations to cause the prescribed drop in residue. This works fine for Laplace's equation. In a general problem, however, this may result in the code running for a very long time to converge be the predetermined amount. Worse, it may never converge.

The other way is to run a set number of iterations and then decide based on the one that has the greatest residual drop. This corresponds to the plot shown in Figure 3.11. This figure illustrates how dramatic the improvement with SOR could be.

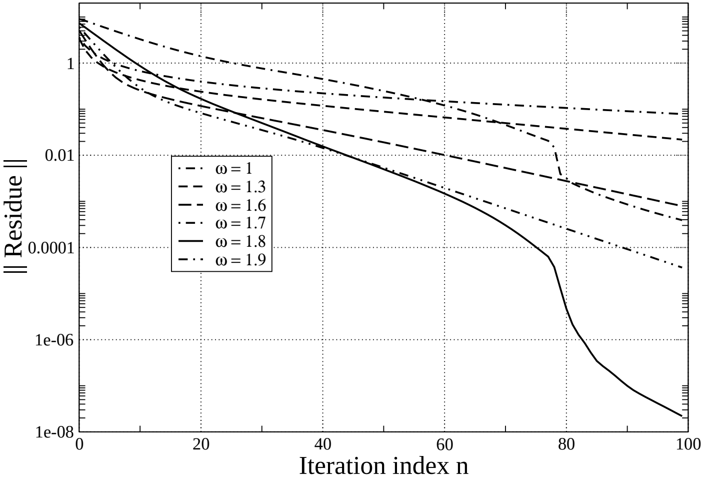

**Figure 3.10.** Plot of the norm of the residue versus iterations for the solution of Laplace's equation on a 41x41 grid using point SOR for various $\omega$ values

## Assignment 3.8

(1) Verify that the roots of the quadratic $Q(x) = c$ are symmetric about the minimum of $Q(x)$. **Hint**: The minimum of the quadratic $ax^2 + bx + c$ occurs at $x = -b/2a$. Study the expression for the two roots.

(2) Repeat the computation involved in generating Figure 3.11 for 10, 50, 100, 200, 500, 1000, iterations. For each of them, once the $\omega_{\text{opt}}$ is recovered using $\Delta\omega = 0.1$, repeat the computations for $\omega \in (\omega_{\text{opt}} - 0.1, \omega_{\text{opt}} + 0.1)$ with a new value of $\Delta\omega = 0.01$.

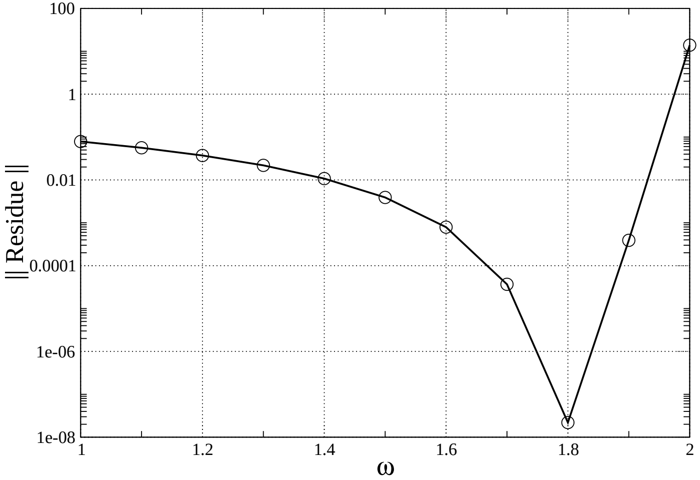

**Figure 3.11. Plot of residue after 100 iterations versus $\omega$ for the solution of Laplace's equation on a 41x41 grid using point SOR**

Did you see a drift in the $\omega_{\text{opt}}$ to the right with an increase in number of iterations?[1] This is of great concern if we do not have an expression for the optimal value and we have to hunt for the $\omega$ through numerical experiment.

## Neumann Boundary Conditions

So far, we have looked at problems where we have applied the Dirichlet type boundary conditions. That is, the function

[1] You must have learnt of uniform convergence in mathematics. Here we have a situation which, fortunately, does not have uniform convergence. However, that makes finding the $\omega_{\text{opt}}$ more difficult.

value is prescribed everywhere on the boundary. In fluid flow problems, Dirichlet boundary conditions are typical if we were solving Laplace's equation for the stream function of a two-dimensional irrotational flow. Now we will look at the same problem through the potential function, which is also determined by Laplace's equation. Look at our problem domain given in Figure 3.1. We replace the bottom boundary with a solid wall. This means that the fluid cannot penetrate the boundary described by the $x$-axis. That is the normal velocity component would be zero on that boundary. Or,

$$\frac{\partial \phi}{\partial n} = 0, \tag{3.80}$$

where, $n$ is along the normal to a no-penetration boundary. In our particular problem we take the no-penetration boundary as the unit interval $(0,1)$ on the $x$-axis. As a result, the no-penetration boundary condition turns out to be

$$\left.\frac{\partial \phi}{\partial y}\right|_{y=0} = 0, \tag{3.81}$$

How is this implemented? Most of the code that you have written does not change. After each sweep of the interior points, we need to add the boundary condition on the bottom grid points.

$$\phi_{i,0} = \phi_{i,1} \tag{3.82}$$

is the simplest way to apply this boundary condition. It uses the first order approximation of the derivative and sets it equal to zero. Subsequently, we solve for the $\phi_{i,0}$ on the boundary. For the most part this should suffice. We could also use a second order representation for the first derivative and derive an expression in a similar fashion to get

**(3.83)**
$$\phi_{i,0} = (4\phi_{i,1} - \phi_{i,2})/3$$

We will see, as we go along, that we apply these kinds of boundary conditions quite often.

---

**Assignment 3.9**

(1) Translate the equations (3.82), (3.83) to a form using one subscript and a stride.
(2) Repeat the first assignment 3.1 and implement the Neumann condition on the $x$-axis as indicated above.
(3) Does it make a difference if you first iterate in the usual fashion with Dirichlet conditions and apply the Neumann condition in later iterations. Start with the solution to the Dirichlet problem as the initial condition for the Neumann problem.
(4) Plot contours and see what changes occur due to the change in boundary conditions.

---

How does changing the boundary condition on one side affect the properties of our solution? Well, the maximum principle does not change. How about the uniqueness of the solution? It is outside the scope of our study here. We will just state that the solution in this case is indeed unique.

## First Order Wave Equation

Keep this picture in mind as you read the next few pages. There is a stream of water flowing at a speed $\lambda$ from your left to your right. You have some paper boats that you have made and are placing these boats in the stream one after another and they are carried away by the stream. Can we figure out where the boats are at any given time after they are released?

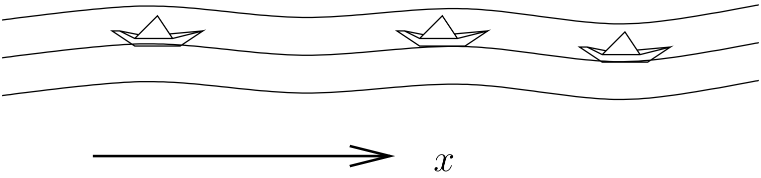

**Figure 3.12. A series of paper boats released in a stream of water flowing along the positive $x$ direction**

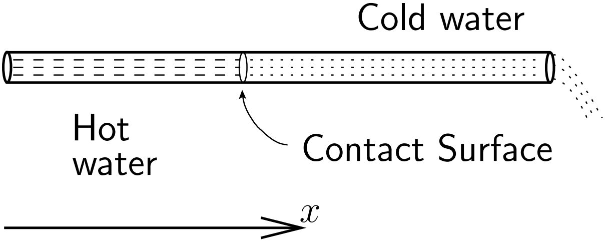

**Figure 3.13. Hot water flowing from a water heater towards a shower. The figure shows the interface between the cold water that was in the pipe all along and the hot water flowing out from the water heater at some time $t$. The interface is very often referred to as a contact surface.**

Here is another scenario. You wake up to a cold morning. You decide to turn on the water heater. You start the shower; the water is cold! Eventually and quite abruptly, the water gets hot. Can we get an idea as to when the hot water will reach us?

Try your hand at the following problem. It should help with the material that follows.

---

**Assignment 3.10**

a. Yesterday, at 9AM, a student started to **walk** down the street from his hostel. He stopped, looked at his watch and thought for awhile. He started to **hurry** to the aerospace engineering department. Near Taramani guest house, he realised it was Sunday and that the Introduction of CFD examination was on the next day. He turned around immediately and **strolled** back up the road. Which of the graphs, shown in Figure 3.14, represents his motion. Why?

b. A student started to **bike** down the street from Cauvery hostel. He stopped, at Chemistry building to chat with a friend. They started to **walk** towards GC. At HSB, which is on the way to GC, the student realised that he may need a calculator for his quiz on "Introduction to CFD". He turned around immediately and **biked** back up the road. Which of the graphs, shown in Figure 3.14, represents his motion. Why?

---

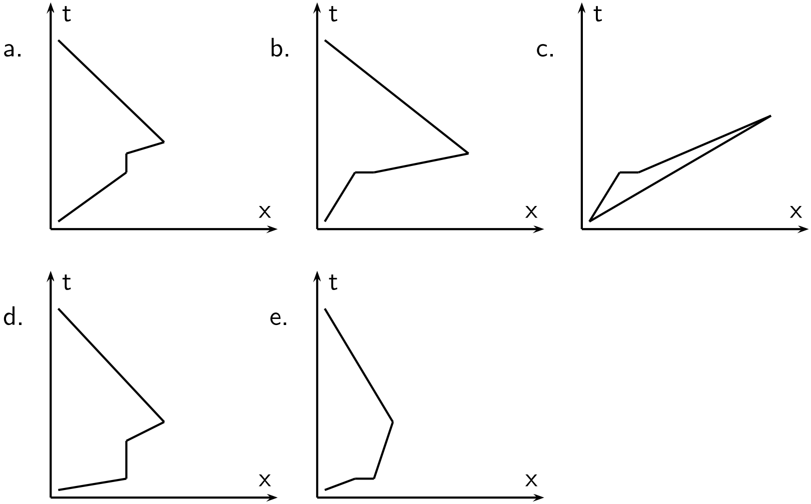

**Figure 3.14.** $x$–$t$ **plots. Try to figure out what each curve means.** $x$ **is the length measured along a road in IIT Madras.** $t$ **is time.**

With these preliminaries behind us, let us look at an equation which is as important to our study as Laplace's equation. The one-dimensional, first order, linear wave equation is

$$\frac{\partial u}{\partial t} + \lambda \frac{\partial u}{\partial x} = 0 \tag{3.84}$$

where $\lambda$ is positive. $u$ is just some property for now. We will try to understand the behaviour of this equation and then tie some physical meaning to $u$.

Let us see what this equation does for us. If the unit vector in the $t$ direction were $\hat{\jmath}$ and the unit vector in the $x$ direction were $\hat{\imath}$, then, the equation can be rewritten in terms of the gradient operator as

**(3.85)** $$(\hat{\jmath} + \lambda\hat{\imath}) \cdot \nabla u = 0$$

This is a directional derivative along the direction $\mathbf{s} = (\hat{\jmath} + \lambda\hat{\imath})$. So, the equation says that the derivative of $u$ along $\mathbf{s}$ is zero. That is

**(3.86)** $$\frac{\mathsf{d}u}{\mathsf{d}s} = 0$$

where $s$ is measured in the direction of $\mathbf{s}$.

So, $u$ is constant along $\mathbf{s}$. The direction $\mathbf{s}$ is known as a characteristic direction. The line shown in the figure is called a characteristic. The equation of that line can be written in many ways. In vector form it is

**(3.87)** $$\begin{Bmatrix} x \\ t \end{Bmatrix} = \begin{Bmatrix} x_0 \\ 0 \end{Bmatrix} + s \begin{Bmatrix} \lambda \\ 1 \end{Bmatrix}$$

where, $(x_0, 0)$ is the point $\xi$, the point where the characteristic intersects the $x$-axis. $s$ is measured along the characteristic. It takes a value zero at $x_0$. We could write out the two equations separately as

**(3.88)** $$x = x_0 + \lambda s \Rightarrow dx = \lambda ds$$

**(3.89)** $$t = s \Rightarrow dt = ds$$

These equations can then be consolidated into

**(3.90)** $$dx = \lambda dt \Rightarrow \frac{dx}{dt} = \lambda$$

This is the equation that governs the characteristic and is called the characteristic equation.

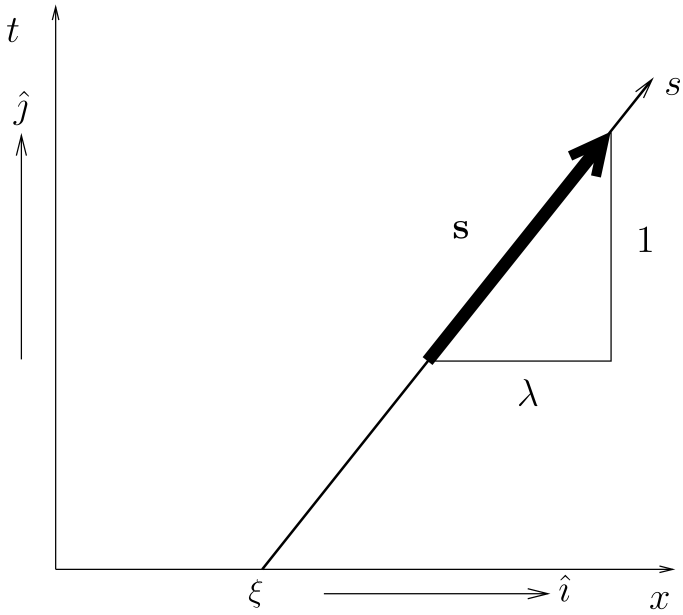

**Figure 3.15.** **Along $s$, $u$ is a constant. Since $\lambda$ is a constant, we get straight lines. $\hat{\imath}$ and $\hat{\jmath}$ are unit vectors along the $x$-axis and $t$-axis respectively**

Physically, the line shown emanating from $\xi$ in Figure 3.15, represents lines along which some property $u$ is a constant. For instance, consider the stream of water from our earlier example. It is flowing from the origin towards the right. The stream of water flows at a constant speed $\lambda$. At the point $\xi$, at time $t = 0$, one adds some dye into the stream (instead of the boats we had earlier). This dye will be carried with the stream at the

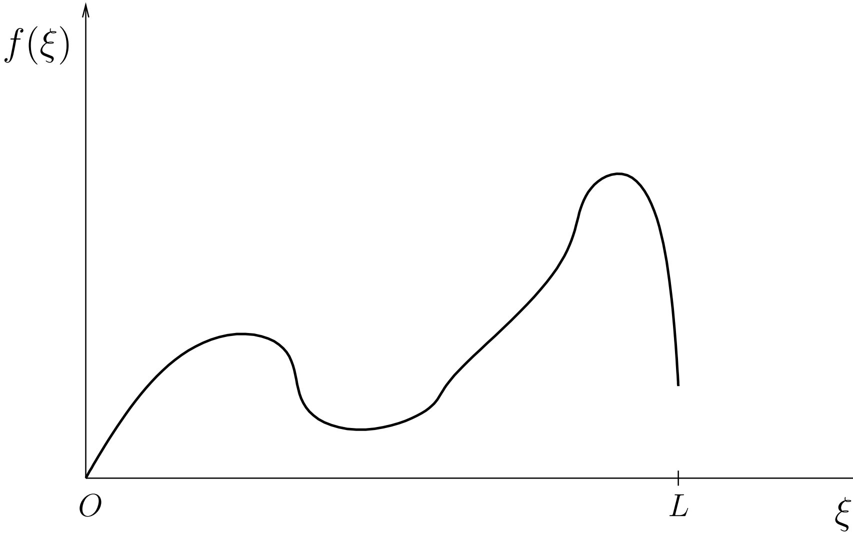

**Figure 3.16. We can take $u(x, 0) = f(x)$ as the initial condition for $u$. Note that it is defined on an interval of length $L$ at the origin**

speed $\lambda$. On the $x - t$ plane it will travel along the line drawn. From Figure 3.15, we can find out where the dye is at any time.

One interpretation of this equation is that $u$ is carried or advected with the constant speed $\lambda$. For this reason, this equation is also called an advection equation. Or, more precisely, the first order, linear, one-dimensional advection equation.

We can use the characteristics to obtain the solution to the advection equation. If at $t = 0$, we were given the value of $u(x, 0)$, we could use the characteristics to propagate this

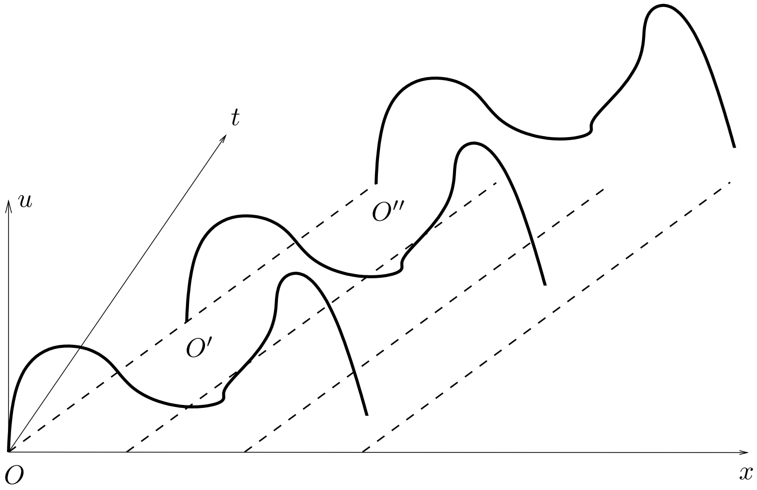

**Figure 3.17. Initial condition from 3.16 being transported by the wave equation (3.84) in the $x-t$ plane. Four characteristics are shown using dashed lines. Note that right now we are not saying anything about what happens to the the left of the line $O-O'-O$", or to the right of the characteristic emanating from $x=L$**

forward in time. The condition $u(x,0)$ would be called the initial condition. Figure 3.16 represents a possible initial condition for $u$. How can we use characteristics to determine the solution for $t>0$? Simple, the characteristic equation tells us that $u$ is constant along a characteristic. Figure 3.17 shows how the characteristics transport $u$.

We see what happens to the initial conditions prescribed. The value at the origin, point $O$, is propagated to $O'$ and $O''$ and so on. What is the value of the function to the left of $O'$

or $O''$. Well, since the information seems to be propagating left to right, it seems natural that it should come from the left boundary. So, not only do we need to provide an initial condition at $t = 0$, one also needs to provide a boundary condition at $x = 0$, say, $u(0, t) = g(t)$. These are the boundary conditions as required by the physics of the problem.

Let us consider a few cases and see what we get. Try out the following problems.

---

**Assignment 3.11**

Given the initial conditions and a boundary condition find the solution to the equation on the interval $[0, 1]$.

**(3.91)**
$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$$

(1) $u(x, 0) = 1.$, $u(0, t) = 1.0$, ... boring!
(2) $u(x, 0) = 1.$, $u(0, t) = 0.5$,
(3) $u(x, 0) = x$, $u(0, t) = 0$,
(4) $u(x, 0) = 1.$, $u(0, t) = \cos(t)$,
(5) $u(x, 0) = 1.$, $u(0, t) = \cos(2t)$, ... any difference?

---

All of these clearly indicate (except the first one) that we are just shifting the solution in time along $x - \lambda t$. In fact, given any function $f(\xi)$ as the initial condition at $t = 0$, $f(x - \lambda t)$ should be a solution. This is easily shown as follows.

**(3.92)**
$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial \xi}\frac{\partial \xi}{\partial t} = \frac{\partial f}{\partial \xi} \times (-\lambda)$$

and,

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial \xi}\frac{\partial \xi}{\partial x} = \frac{\partial f}{\partial \xi} \times (1) \tag{3.93}$$

Clearly the equation is satisfied.

$$\frac{\partial f}{\partial t} + \lambda \frac{\partial f}{\partial x} = 0 \tag{3.94}$$

As was pointed out at the end of section 1, integration is a process of guessing. Meaning, given a function $f$, we guess the integral $F$. You verify that $F$ is the integral by checking that $f = F'$, where $F'$ is the derivative of $F$. We try to use clues from the problem to make a good guess. Just now, we used the fact that the solution is constant along lines $x - \lambda t = \xi$ to guess that any differentiable function $f(\xi)$ is a solution to equation (3.84). We will weaken that statement a bit. For the sake of this discussion, assume that the initial condition is given on an interval on the $x$-axis. If we are able to expand the initial condition in terms of a Fourier series, we can then propagate the basis functions, that is $\sin(.)$ and $\cos(.)$, along the characteristics. Doing a periodic extension of the function to $\pm\infty$, $f(\xi)$ can be expanded using the Fourier series as

$$f(\xi) = \sum_n A_n e^{in2\pi\xi/L} \tag{3.95}$$

where $i = \sqrt{-1}$, $n$ is the wave number, and $L$ is the corresponding wavelength. Combining these two ideas together we guess a solution of the form

$$u(x,t) = \sum_n A_n e^{in2\pi(x-\lambda t)/L} = \sum_n u_n \tag{3.96}$$

This can be substituted back into equation (3.84) to see if it is satisfied. As the differential equation is linear, we can test each term $u_n$ individually. You can checkout "uniform convergence". I am not going to bother about it here. The two terms of equation (3.84) give

$$\begin{aligned} \frac{\partial u_n}{\partial t} &= -in\lambda\frac{2\pi}{L}A_n e^{in2\pi(x-\lambda t)/L} = -in\lambda\frac{2\pi}{L}u_n && (3.97)\\ \frac{\partial u_n}{\partial x} &= in\frac{2\pi}{L}A_n e^{in2\pi(x-\lambda t)/L} = in\frac{2\pi}{L}u_n && (3.98)\end{aligned}$$

We can clearly see that $u_n$ satisfies the wave equation. Remember, the use of Fourier series presupposes a periodic solution.

Okay. Reviewing what we have seen on the wave equation, it is clear that the discussion surrounding equations (3.84), (3.85), and (3.86) is pivotal to everything accomplished so far. We also see that the argument does not require that $\lambda$ is a constant. What do we mean by this? We have nowhere made use of the fact that $\lambda$ is a constant. Or, so it seems. One way to find out is to see what happens if $\lambda$ is not a constant. Since, we know now that $\lambda$ is a propagation speed, we are aware that a situation of varying $\lambda$ can actually occur. The speed of sound in air, for instance, depends on $\sqrt{T}$, where $T$ is the temperature expressed in Kelvin. If we have a temperature gradient, that is a spatial variation of temperature, the speed of sound would also vary appropriately. In this case the equation may look like

**(3.99)**
$$\frac{\partial u}{\partial t} + \lambda(x,t)\frac{\partial u}{\partial x} = 0$$

A particular form of equation (3.99) that is of interest to us is

**(3.100)**
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

This is the quasi-linear one-dimensional wave equation. You may also see it referred as the inviscid Burgers' equation. We can go back to our discussion on characteristics to see what it gets us. The characteristic equation (3.90) in this case becomes

**(3.101)**
$$\frac{dx}{dt} = u(x,t)$$

Let us try a new set of problems.

---

**Assignment 3.12**

Get a solution to the equation (3.100) in the first quadrant using the characteristic equation for the initial and boundary conditions given below.

(1) $u(x,0) = 1, \quad u(0,t) = 1,$

(2) $u(x,0) = \begin{cases} x & \text{for } x < 1 \\ 1 & \text{for } x \geq 1 \end{cases}, \quad u(0,t) = 0,$

(3) $u(x,0) = \begin{cases} 1-x & \text{for } x < 1 \\ 0 & \text{for } x \geq 1 \end{cases}, \quad u(0,t) = 1.$

---

Let us look at the first problem. With a constant initial condition, it is clear that the equation effectively degenerates to the linear equation. There is nothing exciting here. Let us go on to the second problem. Look at Figure 3.18. Each

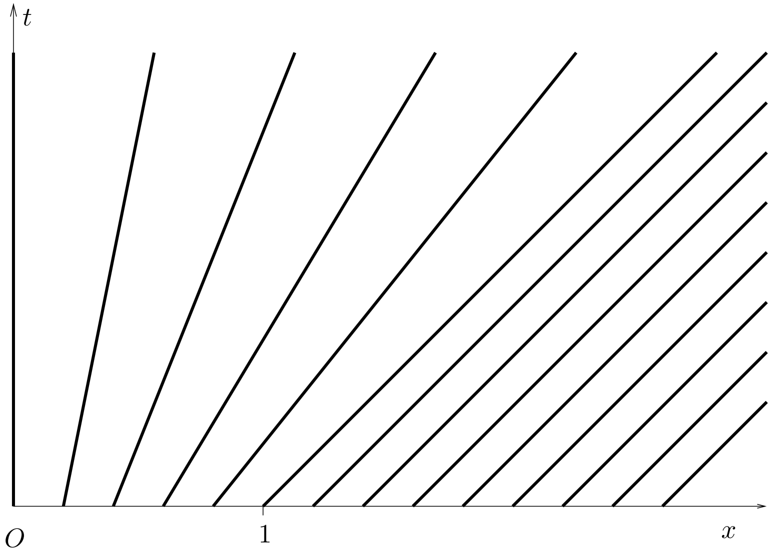

**Figure 3.18. Characteristics corresponding to problem 2 of the assignment 3.12. $\frac{dx}{dt}$ increases linearly from $0$ to $1$ and is then constant at $1$**

characteristic originates from a point taken from a set of equally spaced points on the $x$-axis. You can make out from the figure that, the characteristics emanating from the interval $[0, 1]$ of the $x$-axis are spreading out. This is called an **expansion fan**. Inspection of the expansion fan shown in figure 3.18 should tell you that the characteristics pass through $(0, 0)$, $(0.2, 0)$, $(0.4, 0)$, $(0.6, 0)$, $(0.8, 0)$, and $(1.0, 0)$. For the characteristic $i$, we know the $u_i$ and $x_i$ at time $t = 0$. Now, at $t = 0.2$, what is the corresponding $x_i$ of this characteristic? Using the slope-intercept form for a straight line we know that

**(3.102)**

$$\frac{x_i(t) - x_i(0)}{t - 0} = u(x_i(0)) = u_i = x_i(0), \quad \text{for } 0 \leq x_i(0) < 1$$

This tells us that the solution should be

**(3.103)** $$x_i(t) = x_i(0) + tx_i(0), \quad \text{for } 0 \leq x_i(0) < 1$$

Using this expression, we plot the solution for various times. Figure 3.19 shows the initial condition at $t = 0$. Check that the characteristics in Figure 3.18 match this function. Figure 3.20 shows the solution after one time unit. Let us compare the two figures to see how the solution is evolving in time. Consider the line segment $AB$ as shown in Figure 3.19. It is moving at unit speed from left to right. In Figure 3.20, we see that it has moved to the right by unit length in unit time and only point $A$ can be seen on the page. At the origin, the speed of propagation

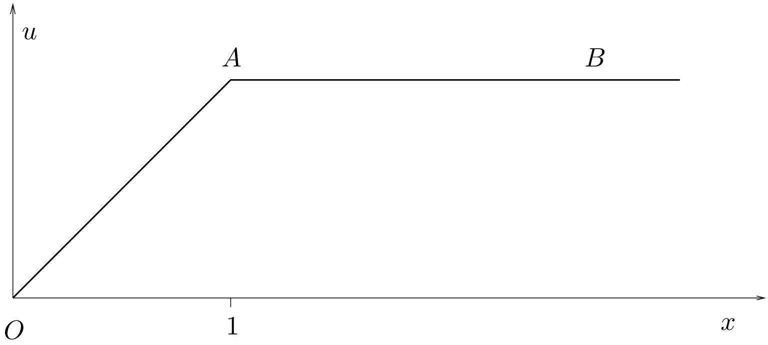

**Figure 3.19. The initial condition at $t = 0$.**

is zero and hence that point does not move at all. In between, we see that we have proportionate speed. Hence, the linearly increasing part of our function, line segment $OA$, experiences a stretch. After two time units, the ramp is much less steep.

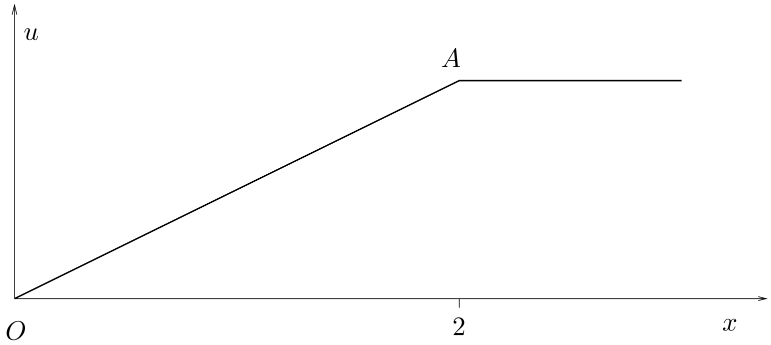

**Figure 3.20. The solution at time $t = 1$. You should recognise that the characteristic $x(0) = 1$ has unit slope.**

Since the start of the ramp on the left hand side is anchored to the origin and the end of the ramp is moving at unit speed away from the origin, the ramp angle is going to keep decreasing.

Now consider the last problem. The characteristics corresponding to the initial condition are shown in Figure 3.22. First the easy part, the characteristics coming from the $t$-axis with arrowheads correspond to the boundary condition given at $x = 0$. These are not extended all the way to the end of the figure to keep the figure from becoming unreadable. The vertical characteristics on the right correspond to the $u = 0$ part of the initial condition. Which leaves the characteristics emanating from the unit interval at the origin. They intersect each other at $x = 1$,

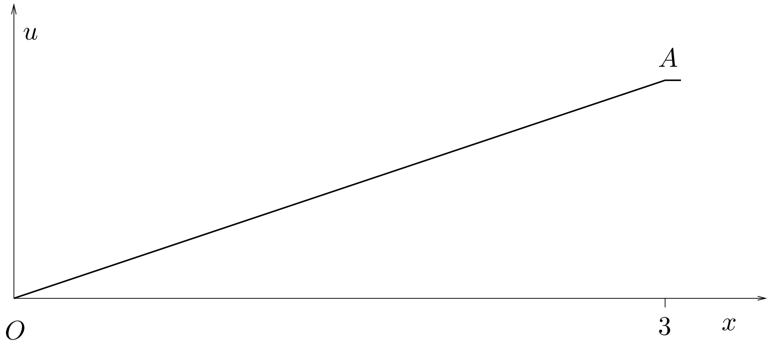

Figure 3.21. The solution at $t = 2$.

$t = 1$. We have extended the lines beyond this point to show them intersecting other characteristics. Is this a problem? Yes! This is a problem. $u_i$ is supposed to be constant on the characteristic $x_i(t)$. If they intersect, what value does $u$ take at the point $(1, 1)$?

Let us go ahead and draw the other figures and see what we get. We have the initial condition drawn in Figure 3.23. It is clear that we expect to have motion from left to right for the point at the origin. This corresponds to our first characteristic starting at the origin in Figure 3.22.

After a time of about $t = 0.25$ units we will get a graph of the new state of $u$ as shown in Figure 3.24. Here we notice, and it should not come as a surprise to you, that the ramp is getting steeper. In fact, if you look at Figure 3.25, we see that our solution continues to be continuous but the ramp is getting steeper and at $t = 1$ we end up with the function shown in Figure 3.26. All the characteristics intersect at $(1, 1)$ and this

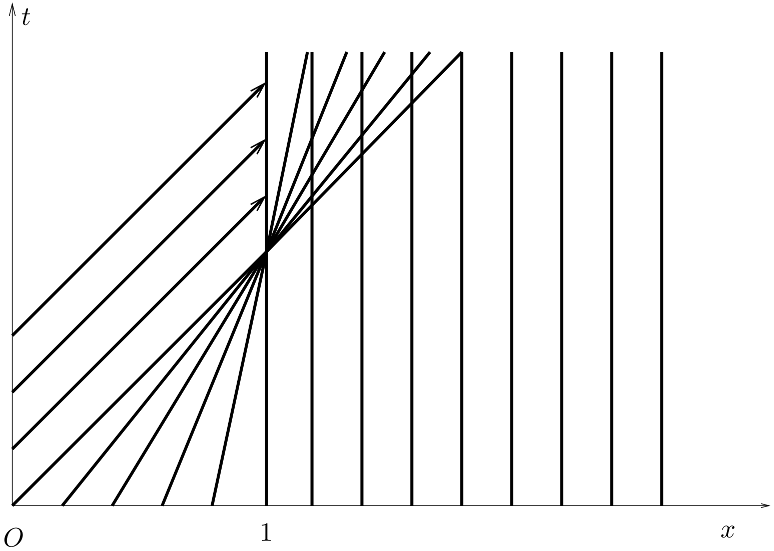

**Figure 3.22. The characteristics corresponding to the initial condition in problem 3 of the assignment. $u$ decreases from 1 to zero and is then constant at zero.**

results in this jump. How do we interpret this jump? We will draw one more figure and see what we get for a time $t > 1$. This is shown in Figure 3.27.

We will look at a discrete analogue first. A railway line is a good example of a one-dimensional space. We imagine there is a railway station at $x = 1$ and that we have a train stopped at that station. We also have at least five other trains speeding along at different speeds. The positions of the trains are indicated in Figure 3.22. Fortunately, at the station they have enough sidings and parallel through tracks that the trains

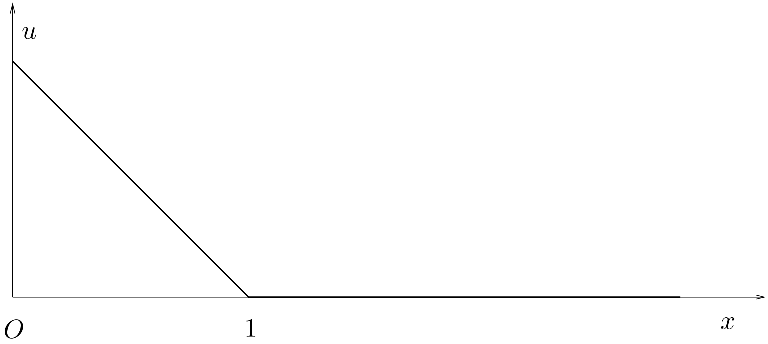

Figure 3.23. The solution at $t = 0$ which is the initial condition in problem 3 of the assignment. $u$ decreases from 1 to zero and is then constant at zero.

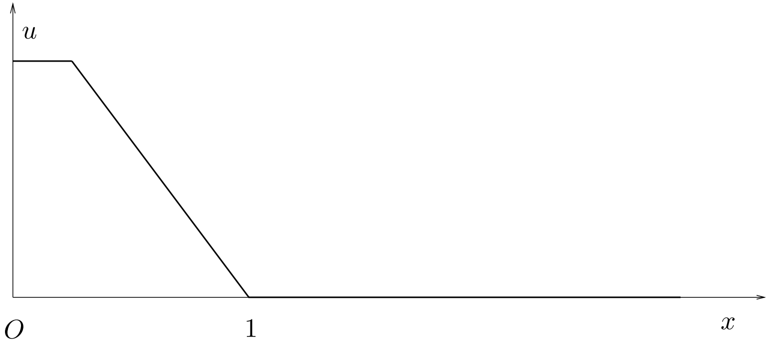

Figure 3.24. The solution at $t = 0.25$ for the initial condition shown in figure 3.23 and given in problem 3 of the assignment

are able to overtake each other. The fastest express train gets ahead of the others and yes, the function is multi-valued at the

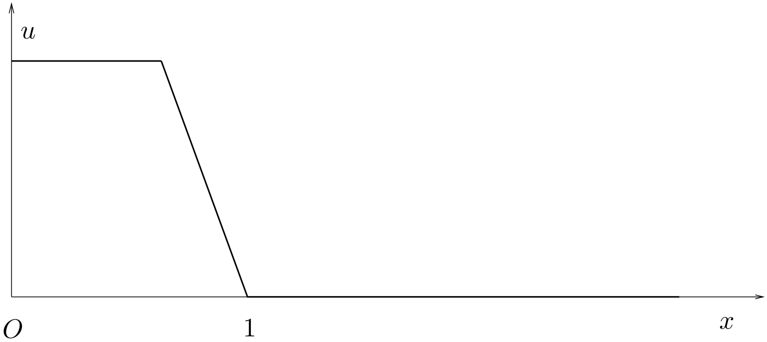

Figure 3.25. The solution at $t = 0.63$ for the initial condition shown in figure 3.23 and given in problem 3 of the assignment. It is clear that the ramp is getting steeper.

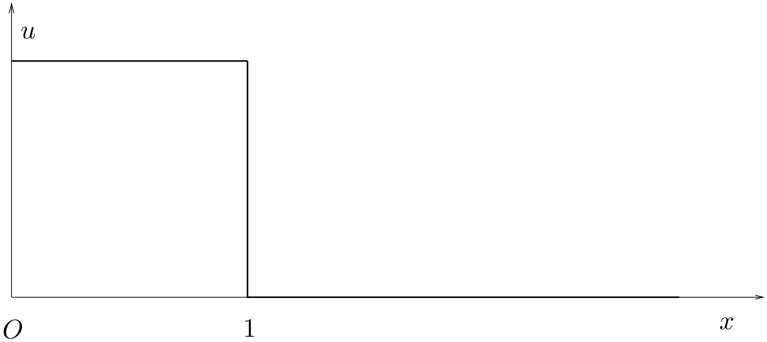

Figure 3.26. The "solution" at $t = 1$ for the initial condition shown in figure 3.23 and given in problem 3 of the assignment.

station at that time since you have more than one train going through. Now, if the trains were not allowed to get past each

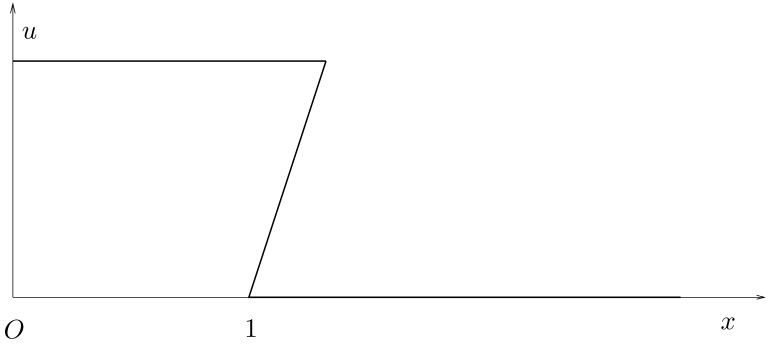

**Figure 3.27. The "solution" at $t = 1.33$ for the initial condition shown in figure 3.23 and faithfully following the characteristics drawn in figure 3.22**

other, then the solution to the differential equation stops at $(1, 1)$. I am sure you can imagine other scenarios along this line.

A continuous interpretation could be that this problem represents a wave of some kind travelling left to right. It is possible that we have land at $x = 1$. If $u$ represents the height of the water, on land, $u = 0$ The series of figures can be viewed as the building up and breaking of the wave. It is not a great model, but still it captures some features of that problem.

Another way to look at it is from a gas dynamics point of view. This is something that you will see in greater detail in the next chapter. However, for now, assume that in a pipe (I want to make sure that it is one-dimensional, see the chapter on one-dimensional problems for a more details on this) we have a gas. On the left hand end of the pipe we are capable of creating a series of disturbances. Each disturbance is in the form of a small compression, causing a small pressure change and a corresponding density change. As this wave propagates through the tube, if it results in an increase in the speed of sound behind it, the next wave that comes along is going to be travelling faster than the first and so on. The problem with this train of waves is they cannot overtake each other. They are confined to the pipe. In this context, Figure 3.27 makes no sense and does not represent the physics of the problem. Instead as shown in Figure 3.26, a discontinuity called a shock is formed. This shock continues to propagate through the pipe beyond $(1, 1)$. So we have to redraw Figure 3.22 for the gas dynamic case as shown in Figure 3.28

Is there a way to find out how fast the "shock" propagates [**Lax73**]? We will investigate this in greater detail in section 3.16. We can make the following observations.

(1) The shock seems to consume all characteristics that intersect it.
(2) We started with a continuous function and the quasi-linear wave equation generated the discontinuity. Just because our initial and boundary conditions are smooth does not mean our solution will be smooth.

We now return to our linear first order one-dimensional wave equation. How do we solve this equation numerically? Considering our success with the Laplace equation, we will just go ahead and perform the discretisation. We will do this in a section on stability. Read on and you will understand why.

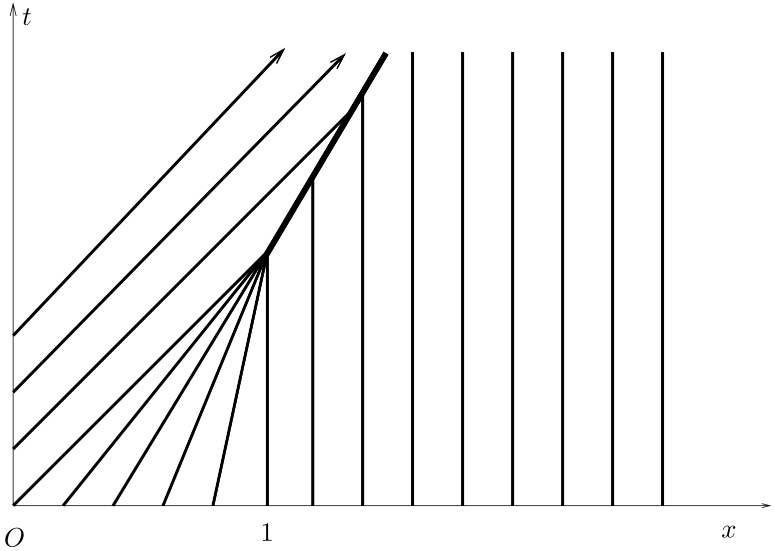

Figure 3.28. The characteristics corresponding to the initial condition in problem 3 of the assignment. $u$ decreases from 1 to zero and is then constant at zero.

## Numerical Solution to Wave Equation: Stability Analysis

This problem has a given initial value. With $\lambda$ positive, we have seen that property $u$ propagates from left to right.

Since we were successful with the Laplace equation, we repeat the same process for the wave equation. Let's consider some general grid point indexed as $p, q$. The index $p$ is in space, that is, along $x$ and the index $q$ is in time, that is, along $t$. The equation can be discretised as

$$\textbf{(3.104)} \qquad \underbrace{\frac{u_{p,q+1} - u_{p,q-1}}{2\Delta t}}_{\text{central difference in time}} + \lambda \underbrace{\frac{u_{p+1,q} - u_{p-1,q}}{2\Delta x}}_{\text{central difference in space}} = 0$$

This gives us an equation for $u$ at the next time step given that we know its values at prior time steps. This clearly will create a problem at $t = \Delta t$, since we are only given values at $t = 0$. This problem can be fixed. However, for now, we will get around this problem by using a forward difference in time. We will retain the central difference in space so as not to lose the advantage of a second order representation for the spatial derivative. This gives us

$$\textbf{(3.105)} \qquad \underbrace{\frac{u_{p,q+1} - u_{p,q}}{\Delta t}}_{\text{forward difference in time}} + \lambda \underbrace{\frac{u_{p+1,q} - u_{p-1,q}}{2\Delta x}}_{\text{central difference in space}} = 0$$

With this discretisation, we have written a finite difference approximation to the differential equation at the point $p, q$ as shown in Figure 3.29. We can solve equation (3.105) for $u_{p,q+1}$ as

$$\textbf{(3.106)} \qquad u_{p,q+1} = u_{p,q} - \lambda \Delta t \frac{u_{p+1,q} - u_{p-1,q}}{2\Delta x}$$

It looks like we have an automaton that we can use to just march in time picking up the solution as we go along. All the quantities on the right hand side are know at the current time level "$q$".

We will now look at this equation and ask the question: does the automaton generate a sequence of $u$'s that represent the solution or does the sequence diverge. Colloquially, does

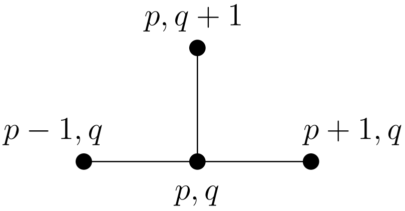

**Figure 3.29. The grid points involved in the solution to the wave equation using Forward Time–Centred Space (FTCS) scheme. The wave equation is approximated at the point $p, q$ by the finite difference equation (3.105).**

the solution "blow up"? This last question relates to stability analysis. It can be restated as: Is the scheme stable?

If you are wondering: what's the point of the discussion, let's get coding. I would suggest that you can indeed get coding and see what happens. Meanwhile, as the wave equation that we are looking at (equation 3.94) is a linear homogeneous equation, a perturbation to it would also be a linear homogeneous equation with homogeneous boundary conditions. The discrete equation would be the same as equation (3.106). We have seen earlier that a periodic solution to the equation can be written in the form given by equation (3.96) This can be rewritten as

$$u(x,t) = \sum_n A_n e^{in2\pi(x-\lambda t)/L} = \sum_n u_n \tag{3.107}$$

For convenience we can take $L = 2\pi$. You can verify that it makes no difference to the analysis that follows.

Both equation (3.84) and equation (3.106) are linear. Just to remind ourselves what we mean when we say something is linear we look at the example of a linear function $\mathcal{L}(x)$. $\mathcal{L}(x)$ is linear means that $\mathcal{L}(a_1x_1 + a_2x_2) = a_1\mathcal{L}(x_1) + a_2\mathcal{L}(x_2)$. Of course, if instead of a sum consisting of two terms, $\sum_{n=1,2} a_n x_n$ we have an infinite number of terms as in the Fourier series expansion, we do have concerns about uniform convergence of the series. This is something that you can look up in your calculus text. We will sidestep that issue here and get back to our stability analysis.

As we are dealing with a linear equation and scheme, we need to look only at one generic term, $u_n$, instead of the whole series. If the scheme is stable for any arbitrary $n$ then it is stable for all the $n$. The fact that $u_n$ does not diverge for all $n$ does not mean that the series will converge. On the other hand, if $u_n$ does diverge for some $n$, the series is likely to diverge.

To answer the question as to what happens as we advance in time using our numerical scheme / automaton, we rewrite $u_n$ as

**(3.108)**
$$u_n = a_n(t)e^{inx}$$

A blowup, as we move forward in time with our automaton, would mean the $a_n(t)$ is growing. We are going to be having lots of subscripts and superscripts going around. Since there is no chance of confusion, we know we are considering one wave number $n$ of the Fourier series, we will drop the subscript $n$ from the equation (3.108) to get

**(3.109)**
$$u = a(t)e^{inx}$$

Assuming we have a uniformly spaced grid so that $\Delta x$ is a constant, we can write at any grid point

**(3.110)** $$u_{p,q} = a_q e^{inp\Delta x}, \quad x_p = p\Delta x$$

which tells us that the gain from the time step $q$ to $q+1$ can simply be written as

**(3.111)** $$g = \frac{u_{p,q+1}}{u_{p,q}} = \frac{a_{q+1}}{a_q}$$

For stability we will require that $|g| < 1$. In equation (3.106), not only do we have $u_{p,q}$, we also have $u_{p-1,q}$ and $u_{p+1,q}$. In order to obtain a simple expression $g$, we need to rid ourselves of the $p-1$ and $p+1$ subscripts. To this end, we do the following.

**(3.112)** $$u_{p+1,q} = a_q e^{in(p+1)\Delta x} = a_q e^{inp\Delta x} e^{in\Delta x} = u_{p,q} e^{in\Delta x}$$

and

**(3.113)** $$u_{p-1,q} = a_q e^{in(p-1)\Delta x} = u_{p,q} e^{-in\Delta x}$$

We define $\theta = n\Delta x$ and substitute equations (3.112) and (3.113) into equation (3.105) to get

**(3.114)** $$u_{p,q+1} = u_{p,q} - \frac{\sigma}{2}\left\{e^{i\theta} - e^{-i\theta}\right\} u_{p,q}, \quad \sigma = \frac{\lambda \Delta t}{\Delta x}$$

Expanding the exponential using Euler's formula, $e^{i\theta} = \cos\theta + i\sin\theta$, [**Ahl79**] [**Chu77**] and dividing through by $u_{p,q}$, we get the amplification or the gain over one time step as

**(3.115)**
$$g = \frac{u_{p,q+1}}{u_{p,q}} = 1 - \frac{\sigma}{2}(\cos\theta + i\sin\theta - \cos(-\theta) - i\sin(-\theta))$$

**(3.116)** $$g = 1 - i\sigma\sin\theta$$

For stability, we do not want the $u$ to grow. So, the factor $g$ by which $u$ is multiplied each time should have a magnitude less than one. We require

$$|g|^2 = g\bar{g} = 1 + \sigma^2 \sin^2\theta < 1 \tag{3.117}$$

where $\bar{g}$ is the conjugate of $g$. We clearly cannot satisfy this requirement. This scheme is said to be **unstable**. That is, there is no condition, on say $\sigma$, for which it is stable.

We were not as lucky with the wave equation as we were with Laplace's equation. Let us have another shot at it. Let us use a forward difference in space, just as we did in time. We will use the grid points as shown in Figure 3.30. The discrete equation at the point $p, q$ now becomes

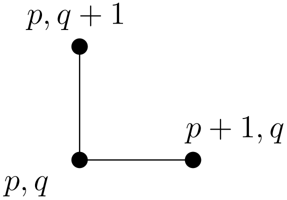

**Figure 3.30. The grid points involved in the solution to the wave equation using Forward Time–Forward Space (FTFS) scheme. The wave equation is approximated at the point $p, q$ by the finite difference equation (3.118).**

$$\frac{u_{p,q+1} - u_p}{\Delta t} + \lambda \frac{u_{p+1,q} - u_{p,q}}{\Delta x} = 0 \tag{3.118}$$

or

$$u_{p,q+1} = u_{p,q} - \lambda \Delta t \frac{u_{p+1,q} - u_{p,q}}{\Delta x} \tag{3.119}$$

Substituting from equation (3.112) into equation (3.119) we get

$$u_{p,q+1} = u_{p,q} - \sigma \left\{ e^{i\theta} - 1 \right\} u_{p,q} \tag{3.120}$$

We obtain the gain or the amplification factor $g$ as

$$g = \frac{u_{p,q+1}}{u_{p,q}} = 1 - \sigma(\cos\theta + i\sin\theta - 1) \tag{3.121}$$

and the stability requirement again is

$$\begin{aligned} |g|^2 = g\bar{g} &= (1 + \sigma - \sigma\cos\theta)^2 + \sigma^2\sin^2\theta \\ &= 1 + \sigma^2 + \sigma^2\cos^2\theta + 2\sigma - 2\sigma\cos\theta \\ &\qquad - 2\sigma^2\cos\theta + \sigma^2\sin^2\theta < 1 \end{aligned} \tag{3.122}$$

This gives

$$\begin{aligned} &1 + 2\sigma^2 - 2\sigma\cos\theta + 2\sigma - 2\sigma^2\cos\theta < 1 \\ &\Longrightarrow (\sigma^2 + \sigma)(1 - \cos\theta) < 0 \\ &\Longrightarrow \sigma(\sigma + 1) < 0 \\ &\Rightarrow \sigma < 0 \quad \text{and} \quad \sigma > -1 \end{aligned} \tag{3.123}$$

We get a condition for stability, but what does it mean? Well, let us look at the condition that $\sigma < 0$, the condition says that

$$\sigma = \lambda\frac{\Delta t}{\Delta x} < 0 \Longrightarrow \Delta t < 0 \quad \text{or} \quad \Delta x < 0 \tag{3.124}$$

We do not want to go back in time so the condition on the time step is really not of interest right now. The other condition, $\Delta x < 0$, can be interpreted as giving us a hint to fix this problem. It seems to tell us to use a backward difference in space rather than a forward difference. How do we conclude this?

Well, we are assuming $\lambda$ is positive. That is the waves are moving left to right. How about if $\lambda$ were negative. Then our current scheme would work since the wave would be moving from right to left. Our forward differencing also has points that are on the right of our current spatial location. So, to get a scheme that works for $\lambda$ positive, we use a forward difference in time and a backward difference in space. We use the grid points shown in Figure 3.31 to get the following equation.

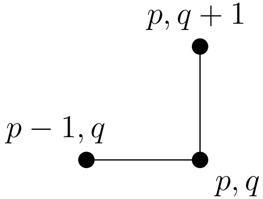

**Figure 3.31. The grid points involved in the solution to the wave equation using Forward Time–Backward Space (FTBS) scheme. The wave equation is approximated at the point $p, q$ and results in the automaton given in equation (3.125).**

**(3.125)**
$$u_{p,q+1} = u_{p,q} - \lambda \Delta t \frac{u_{p,q} - u_{p-1,q}}{\Delta x}$$

We repeat our stability analysis to get

**(3.126)**
$$u_{p,q+1} = u_{p,q} - \sigma \left\{ 1 - e^{-i\theta} \right\} u_{p,q}$$

We obtain the gain or the amplification factor $g$ as follows

**(3.127)**
$$g = \frac{u_{p,q+1}}{u_{p,q}} = 1 - \sigma(1 - \cos(-\theta) - i\sin(-\theta))$$

giving

**(3.128)**
$$g = 1 - \sigma + \sigma\cos\theta - i\sigma\sin\theta$$

Again, for stability we require that

**(3.129)**
$$|g|^2 = g\bar{g} = (1 - \sigma + \sigma\cos\theta)^2 + \sigma^2\sin^2\theta < 1$$

which expands out to

**(3.130)**
$$1 + \sigma^2 + \sigma^2\cos^2\theta - 2\sigma + 2\sigma\cos\theta - 2\sigma^2\cos\theta + \sigma^2\sin^2\theta < 1$$

This gives

**(3.131)**
$$1 + 2\sigma^2 + 2\sigma\cos\theta - 2\sigma - 2\sigma^2\cos\theta < 1$$

Some cancelling and factoring gives us

**(3.132)**
$$(\sigma^2 - \sigma)(1 - \cos\theta) < 0$$

$(1 - \cos\theta)$ is not negative and we are not concerned about $\theta = n\Delta x = 0$. So, dividing through by $(1 - \cos\theta)$ we get

**(3.133)**
$$\sigma(\sigma - 1) < 0$$

Which tells us

**(3.134)**
$$\sigma > 0 \quad \text{and} \quad \sigma < 1$$

This condition $0 < \sigma < 1$ is called the Courant-Lewy-Friedrich condition or the CFL condition [**CFL67**]. In CFD parlance now, the number $\sigma$ is referred to as the CFL number or the Courant number. In an informal discussion a colleague may ask you: "What CFL are you running your code?" By this they are requesting information on the value of $\sigma$ that you are using in your code. Coming back to the CFL condition, we see that FTBS is **conditionally stable** for this equation as it is stable if the condition $0 < \sigma < 1$ is met.

Right, we have finally come up with a scheme that worked by taking a backward difference in space. Since we are sort of groping around to construct an automaton somehow or the other, let us take one more shot at it. Backward difference in space worked. How about, if we tried a backward difference in time and went back to a centred difference in space (I really like the centred difference in space since the truncation error is better). So, we can write the Backward Time–Centred Space [BTCS] scheme. The grid points involved are shown in Figure 3.32. Note that the differential equation in this case is represented at the point $p, q+1$. We get the discrete equation

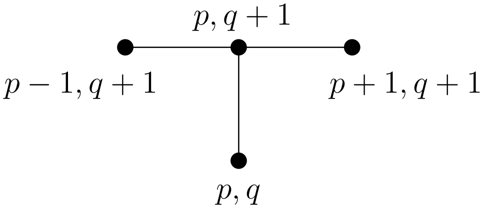

**Figure 3.32. The grid points involved in the solution to the wave equation using Backward Time-Centred Space (BTCS) scheme. Note that the wave equation is approximated at the point $p, q+1$ by the finite difference equation (3.135).**

**(3.135)**
$$u_{p,q+1} + \frac{\sigma}{2}\{u_{p+1,q+1} - u_{p-1,q+1}\} = u_{p,q}$$

We see that the gain would be given by

**(3.136)**
$$g = \frac{1}{1 + i\sigma\sin\theta}$$

Again, for stability we require that

$$g\bar{g} = \frac{1}{1+\sigma^2 \sin^2\theta} < 1 \Longrightarrow 1 < 1 + \sigma^2 \sin^2\theta \tag{3.137}$$

Which is always true for $n > 0$. Aha! BTCS is STABLE. No conditions. Remember, as the saying goes: You don't get nothin' for nothin'. The BTCS scheme on closer inspection requires that we solve a system of equations. We will look at this in greater detail later. The lesson to take away from here is

> **we cannot just discretize the equations and assume that the resulting scheme will work.**

This analysis is referred to as the von Neuman stability analysis. An important requirement is that the equation that we analyse is linear. If the equation is not linear, we will linearise it. For this reason, it is also called linearised stability analysis.

**Courant number or CFL number.** From this analysis we see the significance of the parameter $\sigma$. It is referred to as the Courant number or the CFL number. What do these stability conditions mean? What does this number mean?

A closer inspection of $\sigma$ reveals that it is non-dimensional. We have seen from our initial study of the wave equation that "$\lambda$" is a propagation speed. $\Delta x/\Delta t$ is called the **grid speed**. In a sense, the grid speed is the speed at which our program is propagating $u$. $\sigma$ is the ratio of the physical speed to the grid speed.

What does it mean that FTBS is stable if $0 < \sigma \leq 1$? This says that the physical speed needs to be less than the grid speed. Since we are choosing the grids, it tells us to choose the grids so that the grid speed is greater than the physical speed.

## Numerical Solution to Wave Equation:Consistency

So, what are we really doing when we represent the wave equation by the FTBS scheme of any other such method? We have seen that all of these representations come from truncating the Taylor's series. The discrete equations are said to approximate the original equation and we expect that by solving the discrete equation we get an approximate solution $u^h$ to the original problem. The $h$ superscript in $u^h$ just tells us that we are dealing with a solution to a discrete equation. We ask the question: $u^h$ is an approximate solution to the wave equation problem; To which problem is it an exact solution? A poorly posed question since we only have a discrete set of points and we have already seen that there are really an infinity of functions that the discrete set actually represent. See [**FH74**] and [**Cha90**] for a more detailed discussion. However, asking the question and attempting to answer it is still an useful exercise. So, let us try to answer it anyway.

Consider the FTCS scheme applied to the wave equation again.

**(3.138)**
$$u_{p,q+1} = u_{p,q} - \lambda \Delta t \frac{u_{p+1,q} - u_{p-1,q}}{2\Delta x}$$

This is supposed to approximate the differential equation at the point $(p, q)$. We will substitute for the the terms not evaluated at the current point $(p, q)$ using the Taylor's series. In this analysis, we will keep terms only up to the fourth derivative.

Equation (3.138) becomes

**(3.139)**

$$u_{p,q} + \Delta t \frac{\partial u}{\partial t} + \frac{\Delta t^2}{2!}\frac{\partial^2 u}{\partial t^2} + \frac{\Delta t^3}{3!}\frac{\partial^3 u}{\partial t^3} + \frac{\Delta t^4}{4!}\frac{\partial^4 u}{\partial t^4} + \cdots$$

$$= u_{p,q} - \frac{\lambda \Delta t}{2\Delta x}\Bigg[ u_{p,q} + \Delta x \frac{\partial u}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 u}{\partial x^2} + \frac{\Delta x^3}{3!}\frac{\partial^3 u}{\partial x^3} + \frac{\Delta x^4}{4!}\frac{\partial^4 u}{\partial x^4} + \cdots$$

$$- \left(u_{p,q} - \Delta x \frac{\partial u}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 u}{\partial x^2} - \frac{\Delta x^3}{3!}\frac{\partial^3 u}{\partial x^3} + \frac{\Delta x^4}{4!}\frac{\partial^4 u}{\partial x^4} - \cdots\right)\Bigg]$$

Simplifying and rearranging we get

**(3.140)**

$$\frac{\partial u}{\partial t} + \lambda \frac{\partial u}{\partial x} = -\lambda \frac{\Delta x^2}{3!}\frac{\partial^3 u}{\partial x^3} - \frac{\Delta t}{2!}\frac{\partial^2 u}{\partial t^2} - \frac{\Delta t^2}{3!}\frac{\partial^3 u}{\partial t^3} - \frac{\Delta t^3}{4!}\frac{\partial^4 u}{\partial t^4} + \cdots$$

We have isolated the expression for the original wave equation on the left hand side. On the right hand side, we have terms which are higher order derivatives in $x$ and in $t$. Since at any time level, we have $u$ for various $x$, we convert the time derivatives to spatial derivatives. We do this by taking appropriate derivatives of equation (3.140) and eliminating terms that have time derivatives in them. Again, remember, we will keep terms only up to the fourth derivative. A rather tedious derivation follows this paragraph. You can skip to the final result shown in equation (3.161) if you wish. I would suggest that you try to derive the equation for yourself.

We take the time derivative of equation (3.140) to get the second derivative in time.

**(3.141)**

$$\frac{\partial^2 u}{\partial t^2} + \lambda \frac{\partial^2 u}{\partial t \partial x} = -\lambda \frac{\Delta x^2}{3!}\frac{\partial^4 u}{\partial t \partial x^3} - \frac{\Delta t}{2!}\frac{\partial^3 u}{\partial t^3} - \frac{\Delta t^2}{3!}\frac{\partial^4 u}{\partial t^4} + \cdots$$

The third time derivative is

**(3.142)**
$$\frac{\partial^3 u}{\partial t^3} + \lambda \frac{\partial^3 u}{\partial t^2 \partial x} = -\frac{\Delta t}{2!}\frac{\partial^4 u}{\partial t^4} + \cdots$$

Finally, the fourth time derivative gives

**(3.143)**
$$\frac{\partial^4 u}{\partial t^4} = -\lambda \frac{\partial^4 u}{\partial t^3 \partial x} + \cdots$$

To get rid of the mixed derivative on the right hand side of equation (3.143) we differentiate equation (3.142) with respect to $x$ and multiply by $\lambda$, to get

**(3.144)**
$$\lambda \frac{\partial^4 u}{\partial x \partial t^3} = -\lambda^2 \frac{\partial^4 u}{\partial t^2 \partial x^2} + \cdots$$

Subtracting this from equation (3.143) we get

**(3.145)**
$$\frac{\partial^4 u}{\partial t^4} = \lambda^2 \frac{\partial^4 u}{\partial t^2 \partial x^2} + \cdots$$

In order to eliminate the mixed derivative on the left hand side of equation (3.141) differentiate equation (3.140) with respect to $x$ and multiply by $\lambda$ to get

**(3.146)**
$$\lambda \frac{\partial^2 u}{\partial x \partial t} + \lambda^2 \frac{\partial^2 u}{\partial x^2} = -\lambda^2 \frac{\Delta x^2}{3!}\frac{\partial^4 u}{\partial x^4} - \lambda \frac{\Delta t}{2!}\frac{\partial^3 u}{\partial x \partial t^2} - \lambda \frac{\Delta t^2}{3!}\frac{\partial^4 u}{\partial x \partial t^3} + \cdots$$

Subtracting equation (3.146) from equation (3.141) we get

**(3.147)**
$$\begin{aligned}\frac{\partial^2 u}{\partial t^2} = \lambda^2 \frac{\partial^2 u}{\partial x^2} + \lambda^2 \frac{\Delta x^2}{3!}\frac{\partial^4 u}{\partial x^4} + \lambda \frac{\Delta t}{2!}\frac{\partial^3 u}{\partial x \partial t^2} + \lambda \frac{\Delta t^2}{3!}\frac{\partial^4 u}{\partial x \partial t^3} \\ - \lambda \frac{\Delta x^2}{3!}\frac{\partial^4 u}{\partial t \partial x^3} - \frac{\Delta t}{2!}\frac{\partial^3 u}{\partial t^3} - \frac{\Delta t^2}{3!}\frac{\partial^4 u}{\partial t^4} + \cdots\end{aligned}$$

If we were to differentiate equation (3.146) twice with respect to $x$ we can eliminate the mixed derivative term in equation (3.147) which has only one time derivative in it. On performing the differentiation, we get

**(3.148)**
$$\lambda \frac{\partial^4 u}{\partial x^3 \partial t} = -\lambda^2 \frac{\partial^4 u}{\partial x^4} + \cdots$$

Substituting we get the latest expression for the second time derivative as

**(3.149)**
$$\frac{\partial^2 u}{\partial t^2} = \lambda^2 \frac{\partial^2 u}{\partial x^2} + \lambda^2 \frac{\Delta x^2}{3!} \frac{\partial^4 u}{\partial x^4} + \lambda \frac{\Delta t}{2!} \frac{\partial^3 u}{\partial x \partial t^2} - \lambda^2 \frac{\Delta t^2}{3!} \frac{\partial^4 u}{\partial x^2 \partial t^2} \\ + \lambda^2 \frac{\Delta x^2}{3!} \frac{\partial^4 u}{\partial x^4} - \frac{\Delta t}{2!} \frac{\partial^3 u}{\partial t^3} - \frac{\Delta t^2}{3!} \frac{\partial^4 u}{\partial t^4} + \cdots$$

We now differentiate this equation with respect to $x$ to get

**(3.150)**
$$\frac{\partial^3 u}{\partial x \partial t^2} = \lambda^2 \frac{\partial^3 u}{\partial x^3} + \lambda \Delta t \frac{\partial^4 u}{\partial x^2 \partial t^2} + \cdots$$

Eliminating the mixed third derivative and substituting for the fourth time derivative in equations (3.142) we get

**(3.151)**
$$\frac{\partial^3 u}{\partial t^3} = -\lambda^3 \frac{\partial^3 u}{\partial x^3} - \lambda^2 \frac{\Delta t}{2!} \frac{\partial^4 u}{\partial x^2 \partial t^2} + \lambda \frac{\Delta t}{2!} \frac{\partial^4 u}{\partial x \partial t^3} - \frac{\Delta t}{2!} \lambda^2 \frac{\partial^4 u}{\partial t^2 \partial x^2} + \cdots$$

This can be simplified to

**(3.152)**
$$\frac{\partial^3 u}{\partial t^3} = -\lambda^3 \frac{\partial^3 u}{\partial x^3} - \lambda^2 \Delta t \frac{\partial^4 u}{\partial x^2 \partial t^2} + \lambda \frac{\Delta t}{2!} \frac{\partial^4 u}{\partial x \partial t^3} + \cdots$$

We now differentiate this equation with respect to $x$

**(3.153)**
$$\frac{\partial^4 u}{\partial x \partial t^3} = -\lambda^3 \frac{\partial^4 u}{\partial x^4} + \cdots$$

We can eliminate one more mixed derivative in equation (3.152).

**(3.154)**
$$\frac{\partial^3 u}{\partial t^3} = -\lambda^3 \frac{\partial^3 u}{\partial x^3} - \lambda^2 \Delta t \frac{\partial^4 u}{\partial x^2 \partial t^2} - \lambda^4 \frac{\Delta t}{2!} \frac{\partial^4 u}{\partial x^4} + \cdots$$

We have one last mixed derivative in this equation. Actually, there are an infinity of these mixed derivatives. We have restricted ourselves to fourth derivatives. To get rid of this last mixed derivative in the third derivative term, we rewrite the equation for the second time derivative (3.149) as

**(3.155)**
$$\frac{\partial^2 u}{\partial t^2} = \lambda^2 \frac{\partial^2 u}{\partial x^2} + \lambda^2 \frac{\Delta x^2}{3} \frac{\partial^4 u}{\partial x^4} + \lambda^3 \frac{\Delta t}{2!} \frac{\partial^3 u}{\partial x^3} + \lambda^2 \frac{\Delta t^2}{3!} \frac{\partial^4 u}{\partial x^2 \partial t^2} \\ - \frac{\Delta t}{2!} \frac{\partial^3 u}{\partial t^3} + \cdots$$

If we differentiate this equation twice with respect to $x$, we get

**(3.156)**
$$\frac{\partial^4 u}{\partial x^2 \partial t^2} = \lambda^2 \frac{\partial^4 u}{\partial x^4} + \cdots$$

The third time derivative finally becomes

**(3.157)**
$$\frac{\partial^3 u}{\partial t^3} = -\lambda^3 \frac{\partial^3 u}{\partial x^3} - \lambda^4 \frac{3\Delta t}{2} \frac{\partial^4 u}{\partial x^4} + \cdots$$

**(3.158)**
$$\frac{\partial^2 u}{\partial t^2} = \lambda^2 \frac{\partial^2 u}{\partial x^2} + \lambda^3 \Delta t \frac{\partial^3 u}{\partial x^3} + \left\{ \lambda^2 \frac{\Delta x^2}{3} + \lambda^4 \frac{11 \Delta t^2}{12} \right\} \frac{\partial^4 u}{\partial x^4} + \cdots$$

and for completeness

**(3.159)**
$$\frac{\partial^4 u}{\partial t^4} = \lambda^4 \frac{\partial^4 u}{\partial x^4} + \cdots$$

**(3.160)** $$\frac{\partial u}{\partial t}+\lambda\frac{\partial u}{\partial x}=-\frac{\Delta t}{2!}\lambda^2\frac{\partial^2 u}{\partial x^2}$$
$$-\left\{\lambda\frac{\Delta x^2}{3!}+\lambda^3\frac{\Delta t^2}{3}\right\}\frac{\partial^3 u}{\partial x^3}-\frac{\Delta t}{2!}\left\{\lambda^2\frac{\Delta x^2}{3}+\lambda^4\frac{\Delta t^2}{2}\right\}\frac{\partial^4 u}{\partial x^4}+\cdots$$

We will consolidate coefficients so that they are in terms of $\Delta x$ and $\sigma$.

**(3.161)** $$\frac{\partial u}{\partial t}+\lambda\frac{\partial u}{\partial x}=-\frac{\lambda\Delta x}{2!}\sigma\frac{\partial^2 u}{\partial x^2}$$
$$-\lambda\frac{\Delta x^2}{3!}\left\{1+2\sigma^2\right\}\frac{\partial^3 u}{\partial x^3}-\sigma\lambda\frac{\Delta x^3}{12}\left\{2+3\sigma^2\right\}\frac{\partial^4 u}{\partial x^4}+\cdots$$

This equation is very often referred to as the **modified equation**. A scheme is said to be **consistent** if in the limit of $\Delta t$ and $\Delta x$ going to zero, the modified equation converges to the original equation.

For the case of the FTBS scheme, the modified equation becomes

**(3.162)**
$$\frac{\partial u}{\partial t}+\lambda\frac{\partial u}{\partial x}=\lambda\frac{\Delta x}{2}(\sigma-1)u_{xx}+\lambda\frac{\Delta x^2}{3!}(\sigma-1)(2\sigma-1)u_{xxx}$$
$$-\lambda\frac{\Delta x^3}{4!}(\sigma-1)(6\sigma^2-6\sigma+1)u_{xxxx}+\ldots$$

clearly for the first order, one-dimensional, wave equation, both FTCS and FTBS are consistent. It is interesting to note that the modified equation of FTBS is identical to the wave equation for $\sigma=1$. What does this mean? We are solving the original equation (in this case the wave equation) and not some modified equation. Does that mean we get the same answer? No! We are still representing our solution on a discrete set of points. If you tried to represent a step function on eleven grid points you would actually get a ramp. Try it. This ramp then will be propagated by the modified equation. Even though the modified equation becomes the wave equation when $\sigma = 1$, we cannot get away from the fact that we have only eleven grid points. Our representation of the differential equation is accurate in its behaviour. Our representation of the solution at any time is approximate. So, if that is the problem, what happens in the limit to the solution of the discrete problem as $\Delta t$ and $\Delta x$ going to zero? If the discrete solution goes to the solution of our original equation we are said to have a scheme which is convergent.[2]

Consistency, Stability, Convergence: There is a theorem by P. D. Lax that states that if you have the first two of these you will have the third.

---

## Assignment 3.13

(1) Verify the modified equation (3.162) for the FTBS scheme applied to the linear wave equation given in equation (3.84).

(2) Derive the modified equation for the FTFS scheme applied to the linear wave equation.

[2] If you have a serious conversation with a mathematician they may want the derivatives also to converge.

(3) Verify that the modified equation for the BTCS scheme applied to the linear wave equation is

**(3.163)**
$$\frac{\partial u}{\partial t} + \lambda \frac{\partial u}{\partial x} = \frac{\lambda \Delta x}{2} \sigma \frac{\partial^2 u}{\partial x^2} - \frac{\lambda \Delta x^2}{3!} \left\{ 2\sigma^2 + 1 \right\} \frac{\partial^3 u}{\partial x^3} + \frac{\lambda \Delta x^3}{12} \sigma \left\{ 3\sigma^2 + 2 \right\} \frac{\partial^4 u}{\partial x^4} + \cdots$$

---

## Numerical Solution to Wave Equation:Dissipation, Dispersion

What are the consequences of having these extra terms in our modified equation? Let's find out. Our original problem and a solution are

**(3.164)**
$$\frac{\partial u}{\partial t} + \lambda \frac{\partial u}{\partial x} = 0; \quad u(x,t) = a_n e^{in(x - \lambda t)}$$

Substitute the candidate solution into the equation to verify whether it is actually a solution or not; Don't take my word for it. Clearly this is a travelling wave with a wave number $n$. It never stops oscillating since it has a purely complex exponent. Now consider the equation

**(3.165)**
$$\frac{\partial u}{\partial t} + \lambda \frac{\partial u}{\partial x} = \mu_2 \frac{\partial^2 u}{\partial x^2}$$

We seek a solution in the form

**(3.166)**
$$u(x,t) = a_n e^{in(x - \lambda t)} e^{bt}$$

A nice way of saying, "I am guessing the solution looks like this". We find the derivatives

**(3.167)** $$\frac{\partial u}{\partial t} = a_n\{-in\lambda + b\}e^{in(x-\lambda t)}e^{bt} = \{-in\lambda + b\}u$$

**(3.168)** $$\lambda\frac{\partial u}{\partial x} = \lambda a_n\{in\}e^{in(x-\lambda t)}e^{bt} = in\lambda u$$

**(3.169)** $$\mu_2\frac{\partial^2 u}{\partial x^2} = -\mu_2 n^2 u$$

**(3.170)** $$\mu_3\frac{\partial^3 u}{\partial x^3} = -\mu_3 in^3 u$$

**(3.171)** $$\mu_4\frac{\partial^4 u}{\partial x^4} = \mu_4 n^4 u$$

Substituting into the equation (3.165) we get

**(3.172)** $$b = -\mu_2 n^2 \Longrightarrow u(x,t) = a_n e^{in(x-\lambda t)}e^{-\mu_2 n^2 t}$$

We observe the following. If $\mu_2 > 0$ then the solution decays. Higher wave numbers decay faster than lower wave numbers. If we look at our modified equation for the FTCS technique, equation **(3.161)**, we see that the coefficient of the second spatial derivative is $\mu_2 = -\frac{\lambda\Delta x}{2!}\sigma$. As $\mu_2$ is negative, the scheme is unstable.

Now let us see the effect of having a third derivative term by looking at the equation

**(3.173)** $$\frac{\partial u}{\partial t} + \lambda\frac{\partial u}{\partial x} = \mu_3\frac{\partial^3 u}{\partial x^3}$$

we immediately see that

**(3.174)**
$$b = -\mu_3 in^3 \Longrightarrow u(x,t) = a_n e^{in(x-\lambda t)}e^{-\mu_3 in^3 t} = a_n e^{in[x-(\lambda+\mu_3 n^2)]t}$$

The third derivative contributes to the speed of propagation of the wave. The speed depends on $n^2$, for positive $\mu_3$, higher wave numbers travel faster than lower wave numbers. This effect is known as dispersion. Now, finally, let us consider the effect of the fourth derivative in the equation.

$$\frac{\partial u}{\partial t} + \lambda \frac{\partial u}{\partial x} = \mu_4 \frac{\partial^4 u}{\partial x^4} \tag{3.175}$$

and

$$b = \mu_4 n^4 \Longrightarrow u(x,t) = a_n e^{in(x-\lambda t)} e^{\mu_4 n^4 t} \tag{3.176}$$

In this case, for stability we require $\mu_4 < 0$. This term is clearly a very strong damping mechanism. Also, as with the second derivative term, high wave numbers are damped much more than lower wave numbers.

The effect of the extra terms in the modified equation and the coefficients corresponding to FTCS and FTBS are summarised in table 3.13. We see that both FTCS and FTBS have second derivative terms. However, the coefficient $\mu_2$ for FTCS is negative and we would conclude the scheme is unstable as the solution to the modified equation is unstable. The FTFS scheme is also unstable for positive values of $\sigma$. As we have seen earlier, it is stable only for $-1 \leq \sigma < 0$. The FTBS scheme, on the other hand, is stable for $0 < \sigma \leq 1$ as $\mu_2$ is positive. It will also capture the exact solution at $\sigma = 1$. For $\sigma < 1$ it is quite dissipative. It will dissipate higher frequencies faster than lower frequencies.

Both of the schemes have third derivative terms. Since dissipation is rather rapid it may be difficult to observe the dispersion in the FTBS scheme. In the FTCS scheme we have

| Term | $\mu_2$ | $\mu_3$ | $\mu_4$ |
|---|---|---|---|
| b | $-\mu_2 n^2$ | $-i\mu_3 n^3$ | $\mu_4 n^4$ |
| FTCS | $-\lambda_1\sigma$ | $-\lambda_2(1+2\sigma^2)$ | $-2\sigma\lambda_3(2+3\sigma^2)$ |
| FTFS | $-\lambda_1(1+\sigma)$ | $-\lambda_2(1+\sigma)(2\sigma+1)$ | $-\lambda_3(1+\sigma)(6\sigma^2+6\sigma+1)$ |
| FTBS | $\lambda_1(1-\sigma)$ | $\lambda_2(1-\sigma)(2\sigma-1)$ | $\lambda_3(1-\sigma)(6\sigma^2-6\sigma+1)$ |

**Table 3.1.** These are the coefficients that occur in the modified equation of FTBS, FTFS, and FTCS. Summary of the effect of higher order terms on the solution to the one-dimensional first order linear wave equation: $\mu_2 > 0$ or $\mu_4 < 0$ is dissipative, high wave numbers decay faster than low ones, $\mu_3 \neq 0$ is dispersive, $\mu_3 > 0$, high wave numbers travel faster than low ones. Here, $\lambda_s = \lambda\Delta x^s/(s+1)!,\ s = 1, 2, 3$

dispersion, but unfortunately the scheme is not stable. However, small CFL values will slow down the growth in FTCS. High frequencies are also less unstable (meaning they do not grow as fast as low wave numbers). So we can observe the dispersion. Also, in the FTCS scheme, the coefficient of the third derivative term is negative for small $\sigma$. Therefore, low frequencies will travel faster than high frequencies. Fine. We will demonstrate dispersion in the following fashion.

(1) We will employ the FTCS scheme to demonstrate dispersion.

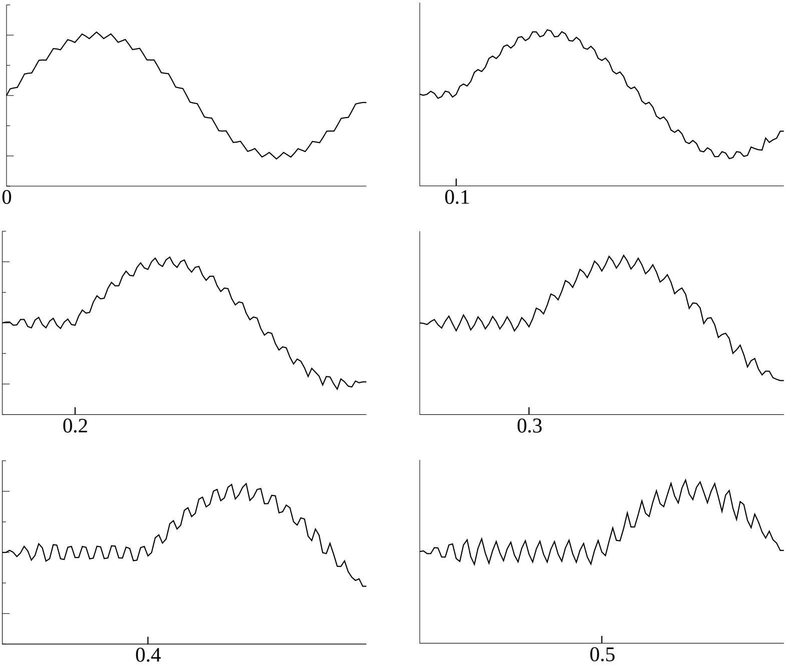

**Figure 3.33. The wave equation, with wave speed=1, is discretised using FTCS. The solutions generated using 101 grid points, with $\sigma = 0.05$ are shown at time-steps $t = 1$, $201$, $401$, $601$, $801$, and $1001$. The amplitude of the high frequency component is growing as FTCS is an unstable scheme. You can make out the low frequency component propagating from left to right faster than the high frequency wave**

(2) We will use a unit length domain divided into a 100 intervals.

(3) We will use the differential equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$$

The equation is propagating $u$ at unit speed.

(4) We start with an initial condition $u(x, 0) = \sin(2\pi x) + 0.05\sin(50\pi x)$. This initial condition corresponds to a the two wave numbers $n = 1$ and $n = 25$.

(5) We will use a CFL of 0.05.

The results of the computation for time steps $t = 1$, $201$, $401$, $601$, $801$, $1001$ are shown in Figure 3.33. With $\sigma = 0.05$, we expect that our solution should progress through the unit computational domain in 2000 time steps. This is because our equation is supposed to be propagating $u$ at unit speed. We can clearly see that the component of the solution that has wavenumber 25 seems to be stationary. The wavenumber $n = 1$, on the other hand, is being propagated by the wave equation in a routine fashion. In fact, at the times shown, the trailing edge of the low frequency part is at locations $0.0, 0.1, 0.2, 0.3, 0.4$, and $0.5$ as expected. You will also observe that the amplitude of the high frequency component is increasing in time. If we were to run this a little longer, the program would "blowup".

Let us collect our thoughts at this point and see where this leads us. We have seen that numerical schemes that are applied to the wave equation can be unconditionally unstable, conditionally stable or unconditionally stable. We have also seen that this instability, attributed earlier to the the unbounded growth in the von Neuman stability analysis can now be tied to the sign of the second order or fourth order diffusion term. Further, we can see that all frequencies may not diverge at the same rate. Schemes may introduce dissipation that did not exist in the original problem. Added to this is the fact that some schemes may introduce dispersion that does not exist in the original problem. A close inspection of these techniques indicates that the unstable schemes have the wrong kind of dissipation.

> **A thought:** We can take the FTCS scheme and add some second order dissipation to it. After all, we are actually solving the modified equation anyway. Why not modify it, carefully, to suit our stability requirement.

To justify this thought, we ask the following question:

> **Question:** What is the difference between the backward difference approximation of the first derivative and the central difference representations of the same?

First, we squint at this question to get an idea of the answer. We are asking for the difference of two first derivatives. Well, it is likely to look like a second derivative. You can make this out from Figure 2.30.

Finally, we just get in there and subtract one from the other

$$\text{diff} = \frac{u_{p,q} - u_{p-1,q}}{\Delta x} - \frac{u_{p+1,q} - u_{p-1,q}}{2\Delta x} = \frac{-u_{p+1,q} + 2u_{p,q} - u_{p-1,q}}{2\Delta x} \approx -\frac{\Delta x}{2}\frac{\partial^2 u}{\partial x^2} \tag{3.177}$$

Just out of curiosity, what **is** the difference between forward difference and the centred difference

**(3.178)**
$$\text{diff} = \frac{u_{p+1,q} - u_{p,q}}{\Delta x} - \frac{u_{p+1,q} - u_{p-1,q}}{2\Delta x}$$
$$= \frac{u_{p+1,q} - 2u_{p,q} + u_{p-1,q}}{2\Delta x} \approx \frac{\Delta x}{2}\frac{\partial^2 u}{\partial x^2}$$

Only the sign of the quantity representing the second derivative is flipped. The magnitude is the same.

Let us try to pull all of this together now. For $\lambda$ positive, waves propagating from left to right, FTBS will work. For $\lambda$ negative, wave propagating right to left, FTFS will work. So, the scheme we use should depend on the sign of $\lambda$.

We will work some magic now by defining two switches, watch.

**(3.179)**
$$s^+(\lambda) = \frac{|\lambda| + \lambda}{2}$$

**(3.180)**
$$s^-(\lambda) = \frac{|\lambda| - \lambda}{2}$$

So, a scheme which takes into account the direction of the "wind" or "stream" can be constructed as

**(3.181)**
$$u_{p,q+1} = u_{p,q} - \sigma\left\{ s^+(\lambda)\{u_{p,q} - u_{p-1,q}\} + s^-(\lambda)\{u_{p+1,q} - u_{p,q}\}\right\}$$

Such a scheme is called an **upwinding scheme** or an **upstreaming scheme**.

There is another way to achieve the same thing. How is that possible? you ask. To the FTCS scheme add the second difference term to get

**(3.182)**

$$u_{p,q+1} = u_{p,q} - \lambda \Delta t \frac{u_{p+1,q} - u_{p-1,q}}{2\Delta x} + \underbrace{|\lambda| \Delta t \frac{u_{p+1,q} - 2u_{p,q} + u_{p-1,q}}{2\Delta x}}_{\text{artificial dissipation}}$$

Lo and behold, if $\lambda$ is positive we get FTBS otherwise we get FTFS. The extra term that has been added is called **artificial dissipation**. Now, one could frown on adding this kind of artificial dissipation to stabilise the solver. After all, we are deliberately changing the governing equation. On the other hand, we are always solving something like the modified equation. In fact:

**The choice of the discretisation scheme is essentially choosing a modified equation.**

Why not engineer the modified equation? At least, then, we will get it in the form that we want.

What exactly do we have to add to eliminate the second derivative term all together from the modified equation of the FTCS scheme? By looking at the modified equation given in (3.161), you may think that adding the term

**(3.183)**
$$\sigma \frac{\lambda \Delta x}{2} \frac{\partial^2 u}{\partial x^2}$$

would do the trick. If we could add the term exactly, it would. However, we are forced to approximate the second derivative term in the expression (3.183) using a central difference approximation. The resulting modified equation still ends up having a second derivative term. It turns out what we need to add is

$$\frac{\sigma^2}{2}\left\{u_{p+1,q} - 2u_{p,q} + u_{p-1,q}\right\} \tag{3.184}$$

This eliminates the second derivative term from the modified equation.

---

**Assignment 3.14**
Verify that adding the expression (3.184) does indeed eliminate the second spatial derivative from the modified equation of FTCS applied to the first order linear one-dimensional wave equation.

---

If we were interested only in a steady state solution, one could add a mixed derivative to get rid of the dissipation term as we got to a steady state solution. For exmaple,

$$\frac{\partial u}{\partial t} + \lambda\frac{\partial u}{\partial x} = \mu_3'\frac{\partial^3 u}{\partial t \partial^2 x} \tag{3.185}$$

We will get the mixed third derivative term by taking the time derivative of the second spatial derivative in equation (3.169)

$$\mu_3'\frac{\partial}{\partial t}\frac{\partial^2 u}{\partial x^2} = -\mu_3' n^2\frac{\partial u}{\partial t} = -\mu_3' n^2\{-in\lambda + b\}u \tag{3.186}$$

Substituting back into the equation we get

$$\{-in\lambda + b\}u + in\lambda u = -\mu_3' n^2\{-in\lambda + b\}u \tag{3.187}$$

Which gives $b$ as

$$
\begin{aligned}
b &= -\mu_3' n^2 \{-in\lambda + b\} \\
&\Rightarrow \{1 + \mu_3' n^2\} b = i\mu_3' n^3 \lambda \\
&\Rightarrow b = \frac{i\mu_3' n^3 \lambda}{\{1 + \mu_3' n^2\}}
\end{aligned}
\tag{3.188}
$$

This gives a solution that looks like

$$
\begin{aligned}
u(x,t) &= a_n e^{in(x-\lambda t)} e^{i \frac{\mu_3' n^3 \lambda}{\{1+\mu_3' n^2\}} t} \\
&= a_n \exp\left\{ in \left[ x - \left( \lambda - \frac{\mu_3' n^2 \lambda}{\{1 + \mu_3' n^2\}} \right) \right] t \right\}
\end{aligned}
\tag{3.189}
$$

Again, like the earlier spatial third derivative, this mixed third derivative is also dispersive. Let us consider some of the words we have used.

- **dissipation:** This term used here in the CFD context is not to be confused with the term that appears in the energy equation of your fluid mechanics course. It is used here to indicate that the amplitude of a certain wave is decreasing.
- **decay:** Used here synonymously with dissipation. I personally prefer this term to dissipation.
- **dampen:** Dampen again means reduction, attenuation...

However, CFD literature typically refers to the process as dissipation.

Let's look at a different interpretation of the stability conditions that we have derived for the wave equation. Figure 3.34

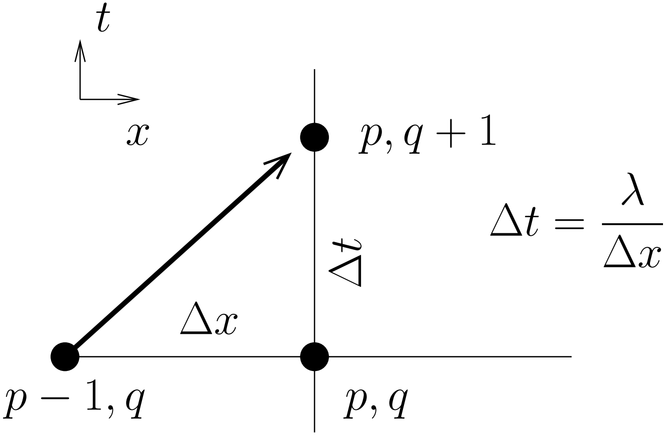

Figure 3.34. Yet another way to look at stability, FTBS has the upstream influence. The arrow indicates flow of information $(u)$. This figure represents the case of $\sigma = 1$. The arrow, in fact, is a characteristic. Since $\sigma = 1$, the zone of influence and the zone of dependence happen to be grid points

shows the three grid points that participate in the FTBS algorithm. The figure shows a setting where $\sigma = 1$. The arrow indicates the direction of propagation of information or "influence". We clearly see that the upstream influences the downstream, which is good. The point $(x_{p-1}, t_q)$ is called the **zone of dependence** of the point $(x_p, t_{q+1})$. The point $(x_p, t_{q+1})$ is called the **zone of influence** of the point $(x_{p-1}, t_q)$.

Now take a look at Figure 3.35. The physical line of influence is reproduced from Figure 3.34 for reference and is shown by the solid arrow. The grid point at $x_{p,q+1}$ is determined by $u$

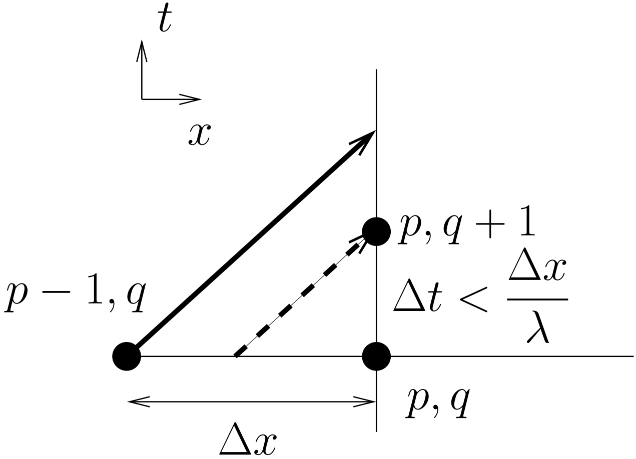

**Figure 3.35.** graphical representation of characteristics on a mesh where $\sigma < 1$ for FTBS. The solid arrow points to the zone of influence from the grid point $p-1, q$. The dashed arrow points from the the zone of dependence of the grid point $p, q+1$. The grid point $p, q+1$ is in the zone of influence of the interval $[x_{p-1}, x_p]$

that propagates along line of influence indicated by the dashed arrow. That is, the zone of dependence lies somewhere in the interval $[x_{p-1}, x_p]$. To understand this better we rewrite the discrete equation (3.125) as

$$u_{p,q+1} = u_{p,q} - \sigma \frac{u_{p,q} - u_{p-1,q}}{\Delta x} = (1-\sigma) u_{p,q} + \sigma u_{p-1,q} \tag{3.190}$$

We see that $u_{p,q+1}$ is a convex combination of $u_{p,q}$ and $u_{p-1,q}$. This should remind you of the hat functions, Laplace's equation, and SOR. First, let us look at why this is not a solution to

Laplace's equation or SOR. To make the discussion interesting look at what happens when $\sigma = 0.5$. We get the average of $u_{p,q}$ and $u_{p-1,q}$. The reason why this turns out to be the wave equation and not the Laplace equation is the fact that this average is taken to be the new value of $u$ at the point $x_p$. A clear case of propagation to the right. However, as we have seen in the modified equation for FTBS, the second derivative term is also present.

We are using hat functions to interpolate here. So, $\sigma$ values in $[0, 1]$ will give us linearly interpolated values on the interval $[x_{p-1}, x_p]$. Since we have right propagating waves, the points on the interval $\Delta x$ shown determine / influence the points on the $\Delta t = \lambda / \Delta x$ interval shown in Figure 3.34. The $\Delta t$ shown in this figure is the maximum permissible value from the stability condition. We will call it $\Delta t_m$ for this discussion. The theoretically correct value of $u$ at $t_q + \Delta t_m$ can be obtained by setting $\sigma = 1$ in the equation (3.190). This is $u(x_p, t_q + \Delta t_m) = u_{p-1,q}$. For a $\Delta t < \Delta t_m$, see Figure 3.35, that is $\sigma < 1$, a point on the interval $[x_{p-1}, x_p]$, determines the value of $u_{p,q+1}$. We can also take the equation (3.190) as saying that for $\sigma < 1$, the value $u_{p,q+1}$ is the weighted average of $u(x_p, t_q + \Delta t_m)$ and $u_{p,q}$. That is, the convex combination of a future value and a past value. This process is stable, as $u_{p,q+1}$ is bounded by $u(x_p, t_q + \Delta t_m) = u_{p-1,q}$ and $u_{p,q}$.

Now, for the case when $\sigma > 1$, see Figure 3.36, $\Delta t > \Delta t_m$. The zone of dependence of the point $(x_p, t_{q+1})$ is a point to the left of $x_{p-1}$. Our scheme only involves $u_{p,q}$ and $u_{p-1,q}$. As a consequence, we end up extrapolating to a point outside the interval $[x_{p-1}, x_p]$. This requires $\sigma > 1$. So all of this is

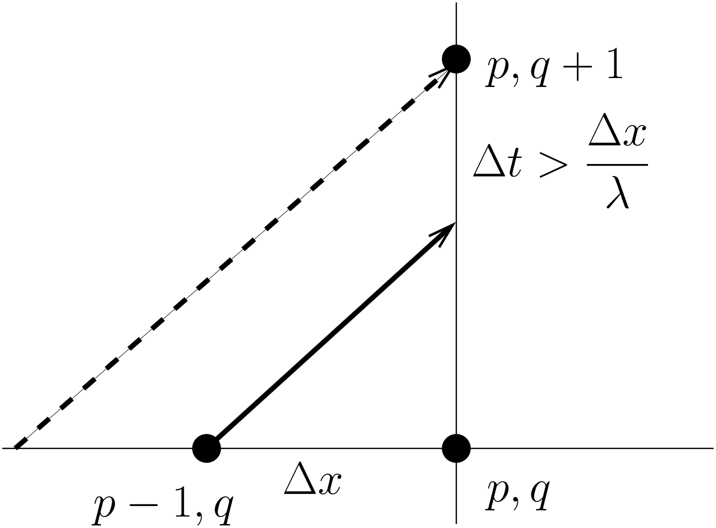

**Figure 3.36. graphical representation of characteristics on a mesh where $\sigma > 1$ for FTBS.The solid arrow points to the zone of influence from the grid point $p-1, q$. The dashed arrow points from the the zone of dependence of the grid point $p, q+1$. The grid point $p, q+1$ is outside the zone of influence of the interval $[x_{p-1}, x_p]$**

consistent, why does it not work? Remember our example at the beginning of the chapter. You may be placing boats on a stream of water, or adding dye to the stream of water. The dye is carried along with the water. We may start and stop adding dye as an when we wish. Downstream, we only have the power to observe how much dye there is in the interval $[x_{p-1}, x_p]$. Just because we observe dye in that interval now, does not mean that the interval $[x_{p-2}, x_{p-1}]$ has the same amount of dye. In fact, it may have none at all. The stream is flowing at a speed $\lambda\, ms^{-1}$ in the positive $x$ direction. Only after a time $\Delta t > \Delta t_m$ will the actual status of the dye currently in the interval $[x_{p-2}, x_{p-1}]$ be known at the point $x_p$. Fiddling around with values $u_{p,q}$ and $u_{p-1,q}$, of dye density, does not help since they have no causal relationship to the density of dye in the interval $[x_{p-2}, x_{p-1}]$. The fact that our scheme does not allow the $u$ in the interval $[x_{p-2}, x_{p-1}]$ to influence / cause the $u_{p,q+1}$ is the reason for the instability. Or simply, if $\sigma > 1$, we can no longer guarantee that the computed $u_{p,q+1}$ is bounded by $u_{p,q}$ and $u_{p-1,q}$. This means that when we view the computation along the time axis, we are extrapolating from two past data points into the future which they do not influence.

The conclusion we draw from here is that when computing a point in the future we need to ensure that we determine the future value from regions on which that future depends.

There is a final point that we will make here. This involves terms we used earlier: high wave numbers, low wave numbers. High and low are comparative terms. High compared to what? Low compared to what? These colloquially posed questions need to be answered. In fact, we have answered them in an earlier section 2.9. A uniform grid has associated with it a highest wave number that can be represented by that grid. We found that with eleven grids we could represent frequency information up to wave number four using hat functions. A wave number four would be high frequency on that grid. Wave number one would be a low frequency. On the other hand, on a grid with a hundred and one grid points, wave number four would be a low frequency and a wave number forty would be a high frequency.

## Solution to Heat Equation

Now that we have seen the wave equation with various higher order derivatives on the right hand side of the equation, let us take a look at one such equation with out the advection term. The one-dimensional heat equation is given by

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2} \tag{3.191}$$

If we employ the Euler's explicit scheme to discretise this equation we get

$$u_p^{q+1} = u_p^q + \Delta t \kappa \left\{ \frac{u_{p+1}^q - 2u_p^q + u_{p-1}^q}{\Delta x^2} \right\} \tag{3.192}$$

Performing a stability analysis as we have done before we see that the gain is given by

$$g = 1 + \lambda \left\{ e^{i\theta} + e^{-i\theta} - 2 \right\}, \quad \mathfrak{s} = \frac{\kappa \Delta t}{\Delta x^2}, \quad \theta = n\Delta x \tag{3.193}$$

This gives us a stability condition

$$0 < \frac{\kappa \Delta t}{\Delta x^2} < \frac{1}{2} \tag{3.194}$$

This is an interesting result. We saw in the earlier section that adding a second order term in the form of artificial dissipation was stabilising as seen from the perspective of the advective equation. We see now that adding a dissipation term brings with it another constraint on the time step. So, there is an upper limit on the time step. We want

**(3.195)**
$$\Delta t = \frac{\Delta x^2}{\kappa}\frac{1}{2} = \frac{\Delta x}{\lambda}$$

We basically have a limit on the amount of artificial dissipation that we can add.

We now find the modified equation for FTCS applied to the heat equation by substituting for the various terms in equation (3.192) with the Taylor's series expanded about the point $(p, q)$.

**(3.196)**
$$\begin{aligned} u_p^{q+1} &= u_p^q + \Delta t \left.\frac{\partial u}{\partial t}\right|_p^q + \frac{\Delta t^2}{2!} \left.\frac{\partial^2 u}{\partial t^2}\right|_p^q + \frac{\Delta t^3}{3!} \left.\frac{\partial^3 u}{\partial t^3}\right|_p^q + \cdots \\ &= u_p^q + \frac{\Delta t \kappa}{\Delta x^2}\left\{ u_p^q + \Delta x \left.\frac{\partial u}{\partial x}\right|_p^q + \frac{\Delta x^2}{2!} \left.\frac{\partial^2 u}{\partial x^2}\right|_p^q + \frac{\Delta x^3}{3!} \left.\frac{\partial^3 u}{\partial x^3}\right|_p^q \right. \\ &\qquad\qquad + \frac{\Delta x^4}{4!} \left.\frac{\partial^4 u}{\partial x^4}\right|_p^q + \cdots \\ &\qquad - 2u_p^q + u_p^q - \Delta x \left.\frac{\partial u}{\partial x}\right|_p^q + \frac{\Delta x^2}{2!} \left.\frac{\partial^2 u}{\partial x^2}\right|_p^q - \frac{\Delta x^3}{3!} \left.\frac{\partial^3 u}{\partial x^3}\right|_p^q \\ &\qquad\qquad \left. + \frac{\Delta x^4}{4!} \left.\frac{\partial^4 u}{\partial x^4}\right|_p^q + \cdots \right\} \end{aligned}$$

Dividing through by $\Delta t$ and cancelling terms, this simplifies to

$$(3.197) \quad \left.\frac{\partial u}{\partial t}\right|_p^q + \frac{\Delta t}{2!}\left.\frac{\partial^2 u}{\partial t^2}\right|_p^q + \frac{\Delta t^2}{3!}\left.\frac{\partial^3 u}{\partial t^3}\right|_p^q + \cdots = \kappa \left\{ \left.\frac{\partial^2 u}{\partial x^2}\right|_p^q + 2\frac{\Delta x^2}{4!}\left.\frac{\partial^4 u}{\partial x^4}\right|_p^q + 2\frac{\Delta x^4}{6!}\left.\frac{\partial^6 u}{\partial x^6}\right|_p^q + \cdots \right\}$$

which finally results in

$$(3.198) \quad \left.\frac{\partial u}{\partial t}\right|_p^q = \kappa \left.\frac{\partial^2 u}{\partial x^2}\right|_p^q + \frac{\Delta x^2}{12}\kappa \left\{ 1 - \frac{6\Delta t \kappa}{\Delta x^2} \right\} \left.\frac{\partial^4 u}{\partial x^4}\right|_p^q + \cdots$$

This modified equation does not help as much with the stability analysis as it did with the wave equation. However, it is interesting to note that the higher order term can be made zero by the proper choice of parameters $\Delta x$ and $\Delta t$.

How does this stability analysis technique work in multiple dimensions? We will do the analysis for the two-dimensional heat equation to find out. There is an ulterior motive to study this now. The two-dimensional heat equation governing the temperature distribution in a material with isotropic thermal conductivity is given by

$$(3.199) \quad \frac{\partial u}{\partial t} = \kappa \left\{ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right\}$$

If we were to apply FTCS to this equation we would get

**(3.200)**
$$u^{q+1}_{p,r} = u^q_{p,r} + \frac{\kappa\Delta t}{\Delta x^2}\left\{u^q_{p-1,r} - 2u^q_{p,r} + u^q_{p+1,r}\right\} + \frac{\kappa\Delta t}{\Delta y^2}\left\{u^q_{p,r-1} - 2u^q_{p,r} + u^q_{p,r+1}\right\}$$

where, $p$ and $r$ are grid indices along $x$ and $y$ grid lines respectively. We define

**(3.201)**
$$\mathfrak{s}_x = \frac{\kappa\Delta t}{\Delta x^2}, \text{and} \quad \mathfrak{s}_y = \frac{\kappa\Delta t}{\Delta y^2}$$

So that equation (3.200) becomes

**(3.202)**
$$u^{q+1}_{p,r} = u^q_{p,r} + \mathfrak{s}_x\left\{u^q_{p-1,r} - 2u^q_{p,r} + u^q_{p+1,r}\right\} + \mathfrak{s}_y\left\{u^q_{p,r-1} - 2u^q_{p,r} + u^q_{p,r+1}\right\}$$

If for the sake of this discussion, we take $\Delta x = \Delta y$, we have $\mathfrak{s}_x = \mathfrak{s}_y = \mathfrak{s}$, this equation becomes

**(3.203)**
$$u^{q+1}_{p,r} = (1 - 4\mathfrak{s})u^q_{p,r} + \mathfrak{s}\left\{u^q_{p-1,r} + u^q_{p+1,r} + u^q_{p,r-1} + u^q_{p,r+1}\right\}$$

Does not look familiar? Multiply and divide the second term on the left hand side by four.

**(3.204)**
$$u^{q+1}_{p,r} = (1 - 4\mathfrak{s})u^q_{p,r} + 4\mathfrak{s}\frac{u^q_{p-1,r} + u^q_{p+1,r} + u^q_{p,r-1} + u^q_{p,r+1}}{4}$$

Now for the ulterior motive. What happens if we substitute $\mathfrak{s} = 0.25$. With $\mathfrak{s} = 0.25$ we get

**(3.205)**
$$u^{q+1}_{p,r} = \frac{u^q_{p-1,r} + u^q_{p+1,r} + u^q_{p,r-1} + u^q_{p,r+1}}{4}$$

This, we recognise as the equation we got when we were trying to solve the Laplace equation using central differences (3.11) earlier in section 3. Which resembles the Jacobi iterative solution to the Laplace equation, and $q$ suddenly becomes an iteration index instead of time index. We see that marching in time is similar to sweeping in space. We may look at the iterations done in our relaxation scheme as some kind of an advancing technique in time. In fact, this suggests to us that if we only seek the steady state solution, we could either solve the steady state equations and sweep in space or solve the unsteady equations and march in time. Having two different views of the same process has the advantage that we have more opportunities to understand what we are actually doing. In a general case, if we have difficulties to analyse a scheme one way, say analysing the relaxation scheme, we can switch our view point to the unsteady problem and see if that would shed any new light on things. If $\mathfrak{s} = 0.25$ corresponds to our first attempt at solving the Laplace equation, what happens for other $\mathfrak{s}$ values? For instance, to what does $\mathfrak{s} = 0.125$ correspond?

**(3.206)**

$$u_{p,r}^{q+1} = 0.5u_{p,r}^{q} + 0.5\left\{u_{p-1,r}^{q} + u_{p+1,r}^{q} + u_{p,r-1}^{q} + u_{p,r+1}^{q}\right\}$$

In the general case

**(3.207)**

$$u_{p,r}^{q+1} = (1-4\mathfrak{s})u_{p,r}^{q} + \frac{4\mathfrak{s}}{4}\left\{u_{p-1,r}^{q} + u_{p+1,r}^{q} + u_{p,r-1}^{q} + u_{p,r+1}^{q}\right\}$$

You may think this is the the same as equation (3.75), with $\omega = 4\mathfrak{s}$. Not quite. The superscripts on the right hand side of equation (3.68) are not all the same. Why did we do over relaxation with Gauss-Seidel and not Jacobi iteration? We will answer this question now.

Let us get on with the stability analysis now. We see that we get a simple extension of the shift operator that we obtained earlier.

$$u^q_{p-1,r} = e^{-in\Delta x} u^q_{p,r} \tag{3.208}$$

$$u^q_{p+1,r} = e^{in\Delta x} u^q_{p,r} \tag{3.209}$$

$$u^q_{p,r-1} = e^{-im\Delta y} u^q_{p,r} \tag{3.210}$$

$$u^q_{p,r+1} = e^{im\Delta y} u^q_{p,r} \tag{3.211}$$

We define $\theta_x = n\Delta x$ and $\theta_y = m\Delta y$. Equation (3.200) gives us

$$g = \frac{u^{q+1}_{p,r}}{u^q_{p,r}} = 1 + 2\mathfrak{s}_x(\cos\theta_x - 1) + 2\mathfrak{s}_y(\cos\theta_y - 1) \tag{3.212}$$

Again, for the sake of this discussion, we take $\Delta x = \Delta y$, we have $\mathfrak{s}_x = \mathfrak{s}_y = \mathfrak{s}$. The expression for the gain $g$ reduces to

$$g = 1 + 4\mathfrak{s}(\cos\theta_x + \cos\theta_y - 2) \tag{3.213}$$

This shows that FTCS applied to the two dimensional heat equation leads to a conditionally stable scheme with a stability condition given by $0 < \mathfrak{s} < 0.25$. Looking at equation (3.207) as representing Jacobi iteration with a relaxation scheme, the stability analysis tells us that the $\omega$ would be limited to $(0, 1)$. $\omega = 1$ turns out to be optimal. In fact, from the heat equation point of view, we would be advancing the solution with the largest possible time step. We can use the over relaxation parameter with Gauss-Seidel, which is something to be borne in mind if we are advancing to a steady state solution using an unsteady equation.

We will use this opportunity to point out that one can use the unsteady equations to march in time towards a steady state solution. Such schemes are called **time-marching** schemes, which we will distinguish from attempts at solving the unsteady problem using the unsteady equations, in which case we are trying to perform **time-accurate** calculations. From what we have seen so far of dissipative and dispersive effects of solvers, we conclude that time-marching to a steady state solution represents an easier class of problems in comparison to time-accurate computations.

Fine, we have an idea of how this stability analysis could be extended to multiple dimensions. We are now in a position to consider the stability analysis of the FTCS scheme applied to the linear Burgers' Equation. The linear Burgers' equation can be written as

$$\frac{\partial u}{\partial t} + \lambda \frac{\partial u}{\partial x} = \kappa \frac{\partial^2 u}{\partial x^2} \tag{3.214}$$

Any of the explicit discretisation schemes that we have studied so far are likely to give a discrete equation that can be written as

$$u_p^{q+1} = A u_p^q + B u_{p+1}^q + C u_{p-1}^q \tag{3.215}$$

Now, the stability analysis we performed in the earlier sections tells us that the gain per time step can be written as

$$g = \frac{u_p^{q+1}}{u_p^q} = A + B e^{i\theta} + C e^{-i\theta}, \quad \theta = n\Delta x \tag{3.216}$$

For stability we require that

**(3.217)**
$$g\bar{g} = (A + Be^{i\theta} + Ce^{-i\theta})(A + Be^{-i\theta} + Ce^{i\theta}) \leq 1$$

This can then be combined to give

**(3.218)**
$$A^2 + B^2 + C^2 + 2A\,(B + C)\cos\theta + 2BC\cos 2\theta \leq 1$$

The left hand side of this equation is maximum when $\theta = 0$, for $A, B, C$ positive. This tells us that

**(3.219)**
$$A + B + C \leq 1$$

This stability analysis was performed with FTCS applied to Burgers' equation in mind. However, if we pay attention to equation (3.215), we see that any scheme advancing in time in an explicit fashion employing the previous grid points will have the same requirement on the stability condition. In that sense, this is a very general result.

## A Sampling of Techniques

We have seen that just using Taylor's series and generating formulas for the approximation of derivatives, we are able to convert differential equations to difference equations. These algebraic equations are then solved.

If we look back at how we manipulated the modified equations to convert temporal derivatives to spatial derivative, we see that there is an opportunity to develop a solution scheme employing Taylor's series directly. The only difference is that instead of using the modified equation to eliminate temporal differences, we use the governing equation as follows

**(3.220)**
$$\frac{\partial u}{\partial t} = -\lambda \frac{\partial u}{\partial x} \Rightarrow \frac{\partial^2 u}{\partial t^2} = -\lambda \frac{\partial^2 u}{\partial t \partial x}$$

Now, switching the mixed derivative around (assuming $\dfrac{\partial^2 u}{\partial t \partial x} = \dfrac{\partial^2 u}{\partial x \partial t}$) we get

**(3.221)**
$$\frac{\partial^2 u}{\partial t^2} = \lambda^2 \frac{\partial^2 u}{\partial x^2}$$

This process can be repeated to replace higher derivatives of $u$ with respect to $t$. We will now derive a solution technique using Taylor's series. Expanding $u_p^{q+1}$ we get about the point $(x_p, t^q)$ we get

**(3.222)**
$$u_p^{q+1} = u_p^q + \Delta t \frac{\partial u}{\partial t}\bigg|_p^q + \frac{\Delta t^2}{2} \frac{\partial^2 u}{\partial t^2}\bigg|_p^q + R(\xi)$$

Truncating the series we get

**(3.223)**
$$u_p^{q+1} = u_p^q - \lambda \Delta t \frac{\partial u}{\partial x}\bigg|_p^q + \lambda^2 \frac{\Delta t^2}{2} \frac{\partial^2 u}{\partial x^2}\bigg|_p^q$$

which can then be written as

**(3.224)**
$$u_p^{q+1} = u_p^q - \frac{\sigma}{2}\{u_{p+1}^q - u_{p-1}^q\} + \frac{\sigma^2}{2}\{u_{p+1}^q - 2u_p^q + u_{p-1}^q\}$$

Does this look familiar. Look at the expression we added earlier to the FTCS scheme given in (3.184). Clearly, we can add more terms to the series and evaluate them by converting temporal derivatives to spatial derivatives. The only problem is that every increase in order looks as though it is going to involve more grid points in space. It is also clear that though the modified wave equation that we have derived has the terms that we want to eliminate to get a (possibly) better scheme, we need to derive the terms using an "ideal modified equation" that is derived using the original equation.

All of these schemes have the time derivatives represented by a first order approximation. What if we tried to use central differences in time? A lot of what we have done in this book was with a certain naiveté. If we ran into problems, we tried to fix them and sought explanations later. We will continue to do the same. We could do this directly using Taylor's series, however, we will take the scenic route. We will approach this in two different ways. One, look at the modified equations for FTCS and BTCS given in equation (3.161) and equation (3.163), respectively. What happens if we take the average of these two equations? We get an equation that is of the order $\Delta x^2$ and is dispersive. This gives us a cue. Lets take the average of the FTCS scheme and the BTCS scheme. This gives us the Crank-Nicholson scheme. The Crank-Nicholson method applied to the wave equation gives us

$$u_p^{q+1} + \frac{\sigma}{4}\{u_{p+1}^{q+1} - u_{p-1}^{q+1}\} = u_p^q - \frac{\sigma}{4}\{u_{p+1}^q - u_{p-1}^q\} \tag{3.225}$$

Of course, this is the specific case of a general linear combination of the explicit and implicit schemes. What we have learnt from our SOR experience, is if we are going to take linear combinations, we can do it in a very generic manner.

$$u_p^{q+1} + \theta\frac{\sigma}{2}\{u_{p+1}^{q+1} - u_{p-1}^{q+1}\} = u_p^q - (1-\theta)\frac{\sigma}{2}\{u_{p+1}^q - u_{p-1}^q\} \tag{3.226}$$

The particular case of the Crank-Nicholson would would correspond to $\theta = 1/2$, which, incidentally is second order accurate in time.

The use of an intermediate point at which we represent the differential equation opens up other possibilities. We will take a second look at employing FTCS in some fashion to get a scheme. At time level $q$, consider three points $p-1$, $p$, $p+1$. We could represent the wave equation at a point in between $p-1$ and $p$ and identify that point as $p+\frac{1}{2}$. We then approximate the wave equation at the point $p+\frac{1}{2}, q$ using FTCS. We would, however, take only half a time step so as to get $u_{p+\frac{1}{2}}^{q+\frac{1}{2}}$. To be

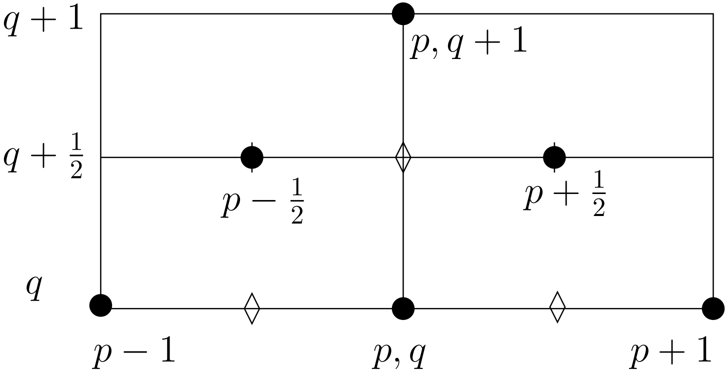

**Figure 3.37. We can use an intermediate time step to use a combination of FTCS steps to give a new possibly improved scheme. We could use the same grids and a combination of FTCS steps and CTCS steps as an alternative. The rhombus indicates points at which the differential equation is represented**

precise we can write

$$u_{p-\frac{1}{2}}^{q+\frac{1}{2}} = u_{p-\frac{1}{2}}^{q} - \frac{\sigma}{2}\left\{u_p^q - u_{p-1}^q\right\} \tag{3.227}$$

$$u_{p+\frac{1}{2}}^{q+\frac{1}{2}} = u_{p+\frac{1}{2}}^{q} - \frac{\sigma}{2}\left\{u_{p+1}^q - u_{p}^q\right\} \tag{3.228}$$

We do not have the terms $u_{p-\frac{1}{2}}^{q}$ and $u_{p+\frac{1}{2}}^{q}$ on the right hand side of each of these equations. What do we do? As always, we improvise by taking the average of the two adjacent grid points to write

$$u_{p-\frac{1}{2}}^{q+\frac{1}{2}} = \frac{u_p^q + u_{p-1}^q}{2} - \frac{\sigma}{2}\left\{u_p^q - u_{p-1}^q\right\} \tag{3.229}$$

$$u_{p+\frac{1}{2}}^{q+\frac{1}{2}} = \frac{u_{p+1}^q + u_{p}^q}{2} - \frac{\sigma}{2}\left\{u_{p+1}^q - u_{p}^q\right\} \tag{3.230}$$

Then, repeating this process to go from the intermediate time step $q+\frac{1}{2}$ to $q+1$ we can write

$$u_p^{q+1} = \frac{u_{p+\frac{1}{2}}^{q+\frac{1}{2}} + u_{p-\frac{1}{2}}^{q+\frac{1}{2}}}{2} - \frac{\sigma}{2}\left\{u_{p+\frac{1}{2}}^{q+\frac{1}{2}} - u_{p-\frac{1}{2}}^{q+\frac{1}{2}}\right\} \tag{3.231}$$

---

## Assignment 3.15

(1) Perform a stability analysis for this scheme (given by equations (3.229), (3.230), and (3.231)) and the variant of taking CTCS in the second step. Remember the definition of the gain.

(2) Find the modified equation. Make sure to include the second, third and fourth order terms. Comment on

the scheme from the point of view of dissipation and dispersion. Can you infer a stability condition?

---

How do we perform the stability analysis?

**Scheme A**: Let us do the obvious thing. We will combine the two steps into one step as we have been doing so far. Substituting for

$$u_{p+\frac{1}{2}}^{q+\frac{1}{2}}, \quad \text{and} \quad u_{p-\frac{1}{2}}^{q+\frac{1}{2}}$$

in equation (3.231), from equations (3.229) and (3.230) we get

**(3.232)**
$$u_p^{q+1} = \frac{u_{p+1}^q + 2u_p^q + u_{p-1}^q}{4} - \frac{\sigma}{2}\left\{u_{p+1}^q - u_{p-1}^q\right\} + \frac{\sigma^2}{4}\left\{u_{p+1}^q - 2u_p^q + u_{p-1}^q\right\}$$

We add and subtract $u_p^q$ to the right hand side so that we can recognise FTCS and see what extra terms have been added to it for this scheme.

**(3.233)**
$$u_p^{q+1} = \underbrace{u_p^q - \frac{\sigma}{2}\left\{u_{p+1}^q - u_{p-1}^q\right\}}_{\text{identical to FTCS}} + \frac{1+\sigma^2}{4}\left\{u_{p+1}^q - 2u_p^q + u_{p-1}^q\right\}$$

We then see that our gain across a time step is

**(3.234)**
$$g = \frac{u_p^{q+1}}{u_p^q} = 1 - i\sigma\sin\theta + \frac{1+\sigma^2}{2}\left\{\cos\theta - 1\right\}, \quad \theta = n\Delta x$$

A little algebra and trigonometry will show that it has the same stability condition $0 < \sigma < 1$.

**Scheme B**: If you struggled a little with the algebra, let us try something that may be a little easier. We assume that gain in any half step is $\gamma$. Then for the first half step

$$\gamma^- = \frac{1}{2}\left\{1 + e^{-in\Delta x} - \sigma(1 - e^{-in\Delta x})\right\} \tag{3.235}$$

$$\gamma^+ = \frac{1}{2}\left\{1 + e^{in\Delta x} - \sigma(e^{in\Delta x} - 1)\right\} \tag{3.236}$$

and consequently the gain would be

$$g = \frac{1}{2}\left\{\gamma^+ + \gamma^-\right\} - \frac{\sigma}{2}\left\{\gamma^+ - \gamma^-\right\} \tag{3.237}$$

The intermediate terms, setting $\theta = n\Delta x$, are

$$\gamma^+ + \gamma^- = 1 + \cos\theta - i\sigma\sin\theta \tag{3.238}$$

$$\gamma^+ - \gamma^- = i\sin\theta - \sigma\left\{\cos\theta - 1\right\}. \tag{3.239}$$

This gives us a gain of

$$g = \frac{1}{2}\left\{1 + \cos\theta + \sigma^2\left\{\cos\theta - 1\right\} - i2\sigma\sin\theta\right\} \tag{3.240}$$

Then, we require

$$g\bar{g} = \frac{1}{4}\left\{\sigma^2(1 - \cos\theta) + (1 + \cos\theta)\right\}^2 < 1 \tag{3.241}$$

or,

$$-2 < \sigma^2(1 - \cos\theta) + (1 + \cos\theta) < 2 \tag{3.242}$$

We see that once again we get $0 < \sigma < 1$.

Clearly instead of employing FTCS again in the second half step as shown in equation (3.231), we could use a CTCS step.

This would give us the Lax-Wendroff scheme [**LW60**]. That is take a second step as

$$u_p^{q+1} = u_p^q - \sigma \left\{ u_{p+\frac{1}{2}}^{q+\frac{1}{2}} - u_{p-\frac{1}{2}}^{q+\frac{1}{2}} \right\} \tag{3.243}$$

We observe from all the schemes that we have derived up to this point that we can choose a spatial discretisation of some kind, for example central differencing, and also pick a temporal discretisation independent of the choice of spatial discretisation chosen. In fact, if we pick a discretisation scheme in space for solving the one-dimensional first order wave equation using central differences, we could write at each point $(x_p, t)$, an ordinary differential equation given by

$$\frac{du_p}{dt} = -\lambda \frac{u_{p+1} - u_{p-1}}{2\Delta x} \tag{3.244}$$

At any $x_p$, $u_p$ is a function of time. This technique is also referred to as the method of lines[**NA63**]. In a more modern parlance and in CFD literature it is often called semi-discretisation to capture the idea of only some of the derivatives being discretised. With semi-discretisation we see that we get an ordinary differential equation in time at each grid point along the space coordinate. A quick check on solving ordinary differential equations shows us that we could indeed use a variety of schemes to integrate in time. This includes the whole family of Runge-Kutta schemes. Though we can mix any of the time derivative discretisation with the space discretisation, it is clear that one would be judicious about the combination. Some combinations may be just a waste of time. The key phrase to remember is mix and **match**. Finally, the modified equation had terms that came from both the spatial discretisation and the temporal discretisation. The combination determines the behaviour of the solver.

## Boundary Conditions

When we speak of boundary conditions we should be very clear as to why they are required. As we have done so far, we will do this in two parts. First we will look at the big picture. Then, we will look at the specific instance at hand.

(1) First the operational point of view. If solving differential equations is like integration, as we have pointed out in the first chapter, then we always have constants of integration that need to be prescribed. These are usually prescribed in terms constraints on the behaviour of the solution.

(2) Another point of view is that we have the differential equation that governs the behaviour of the system at an arbitrary point. We can build models with it. What distinguishes one model from the other? Too broad? Let us look at a more concrete question. We saw that the solution to Laplace's equation is also the steady state solution (or the equilibrium solution) to the heat equation. To restate our earlier question: How would the model differ if the physical problem were to change from the temperature distribution in a circular plate to that in a rectangular plate? We could take a square plate but change the temperature that was applied to its edges. The plate could be kept in an oven or it could be placed in a refrigerator. From this point of

view, once we have our differential equations, it is the boundary conditions that determine the problem that we are actually solving.

(3) The final view point is that our algorithm may require the function values at some point. It is possible that we are not able to use the algorithm to generate the data at the point where it is required.

The first point tells us that there is a minimum set of conditions that are required. The second point emphasises the importance of the proper application of the boundary conditions to meet our goals. After all, we have already seen from the modified equation that we are not solving the original equation, we do not want to compound it by making our problem definition wrong. The final point is a bit vague, indicating, really, that our algorithm may require more conditions than required by the constants of integration from the first point. Let us look now at the specific case of the wave equation.

Let us consider the problem of a pipe filled with cold water. The pipe is one meter long. Such a pipe is shown in Figure 3.13. The left end of the pipe is connected to the exit of a water heater. On opening a valve at the right end of the pipe, water starts to flow out of the right hand side of the pipe into the open. Hot water starts to flow in from the water heater which is maintained at a constant temperature of 65 degrees Celsius. A simple model that we can use to track the hot water front through the pipe is the first order, one-dimensional linear wave equation. In the case of the wave equation, the reason for the application of boundary conditions are quite obvious. The equation has a first derivative in space and a first derivative in time. We would expect from our experience with ordinary differential equations that we will require "one" condition corresponding to the time derivative and "one" corresponding to the spatial derivative.

The temporal derivative part is easy. We have some initial condition at time $t = 0$ (cold water in the pipe) and we would like to use the equation to find out what happens in the future ( How does the hot water-cold water interface or contact surface move?). If $\lambda$ is positive , then the wave equation is propagating left to right (the water will move from the reservoir out into the open). Information is flowing into our domain from the left and flowing out of our domain on the right. So, we need to prescribe what is happening to $u$ at the left end of the domain. If we do this, we can draw characteristics and extend the solution in time.

For the problem at hand, an initial condition needs to be prescribed. This means we need to have $u(x, 0)$. We also need for all time, knowledge of what is coming in at the left end of the problem, that is, at $x = 0, \quad t > 0$, we need to prescribe $u$. Let us say we were using FTCS to solve the problem. The grids involved are at $p - 1$, $p$, and $p + 1$. So, when we are applying FTCS at the last grid point at the rightmost edge, what is $p+1$? If we have $N$ grid points in $x$, we must necessarily stop using FTCS at the grid point $N - 1$. How do we compute the value at the $N$ grid point? We need to apply a boundary condition that our original problem did not require. That is, our algorithm seems to need a boundary condition that our differential equation did not require. We could extrapolate from the interior grid points to the boundary (effectively setting $\partial u/\partial x = 0$). There are two ways to do this. We could compute all the values we are able to calculate at time level $q+1$ and then copy $u$ from the last but one grid point to the last grid point. This is illustrated in Figure (3.38) and can be written as an equation for the last grid point as

$$u_N^{q+1} = u_{N-1}^{q+1} \tag{3.245}$$

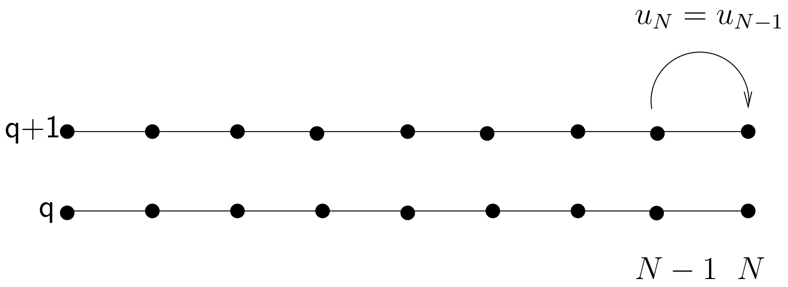

**Figure 3.38. We could compute $u$ at time level $q+1$ up to grid point $N-1$ using FTCS and copy $u_{N-1}$ to $u_N$**

To the first order, this could be interpreted as setting

$$\left.\frac{\partial u}{\partial x}\right|_N^{q+1} = 0 \tag{3.246}$$

Or, one could perform all the computations as possible with FTCS and copy the value from last-but-one grid point at the current time level to the last grid point at the new time level as show in Figure (3.39). This can be written as

$$u_N^{q+1} = u_{N-1}^{q} \tag{3.247}$$

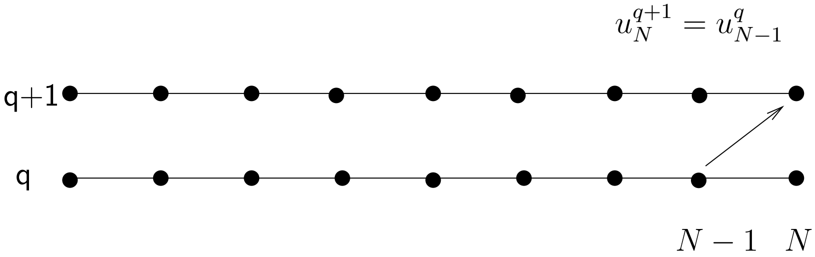

Figure 3.39. We could compute $u$ at time level $q+1$ up to grid point $N-1$ using FTCS and copy $u_{N-1}$ from time level $q$ to $u_N$ at time level $q+1$.

This effectively transports $u$ from grid point $N-1, q$ to $N, q+1$ at the grid speed. Which means that it is FTBS applied to equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \tag{3.248}$$

This gives us an idea. That is, we use FTCS for all the points

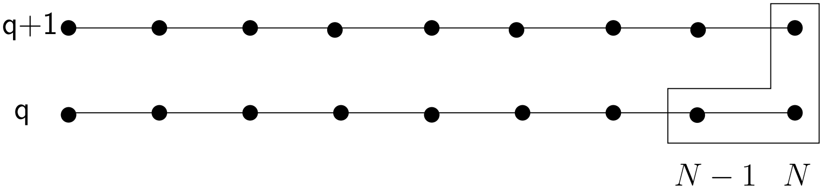

Figure 3.40. We could compute $u$ at time level $q+1$ up to grid point $N-1$ using FTCS and $u_N$ at time level $q+1$ using FTBS employing the known values at the grids $N-1, q$ and $N, q$.

that are possible to calculate using FTCS and FTBS for the last grid point. This is indicated in Figure (3.40).

> **Note:** There are boundary conditions that are prescribed as part of the problem definition and may be part of the mathematical theory of whether a problem is well posed or not. The algorithm that you choose to solve the problem may require more conditions to be prescribed. Sometimes this results in a need to manufacture more boundary conditions. At times, these conditions seem completely contrived. They are very much part of the list of assumptions made and need to be verified with as much vigour as we would our other assumptions. It cannot be emphasised enough, that the need for the extra boundary conditions may arise because of the solution technique chosen. If in the earlier discussion, we had chosen FTBS instead of FTCS there would have been no need for the generation of an extra boundary condition. You can verify what happens when FTFS is applied to this problem.

---

**Assignment 3.16**

(1) Write a program to solve the first order one-dimensional wave equation using FTCS, FTFS, FTBS and BTCS. I would suggest you develop the code using FTBS and do the BTCS code last.
(2) Verify the stability conditions that have been derived.
(3) What happens to FTBS for different values of $\sigma > 1$?
(4) Try out the different boundary conditions.

(5) Plot your results. This is a lot more fun if you make your program interactive.

---

### A Generalised First Order Wave Equation

We have already seen the quasi-linear version of the wave equation given by

**(3.249)**
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

This equation is said to be in the **non-conservative form**. The equation can also be written in what is called **the conservative form** as

**(3.250)**
$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = 0,$$

This equation is said to be in the **divergence free form** or the **conservative form**. That it called the divergence free form is obvious from the fact that the left hand side looks like the divergence of the vector $(u, f)$ and the right hand side is zero. From application of the theorem of Gauss, this equation essentially states there is no net accumulation of any property that is carried by the vector field $(u, f)$. This is a statement of a conservation law as is shown in greater detail in Chapter 5. This is the reason why equation (3.250) is said to be in conservative form. Equation (3.249) is not in conservative form and is therefore said to be in a non-conservative form. We saw earlier this particular form of equation (3.249) is like a directional derivative. In CFD parlance, this form is often called "the" non-conservative form. $f$ is called the flux term and in the case of equation (3.249), $f(u) = u^2/2$. The fact that $f$ is actually the flux can be seen, if we consider a one-dimensional control volume. The extent or the "volume" of the control volume is $x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. The amount of the property $u$ contained in the control volume is

**(3.251)**
$$U = \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u(\xi, t) d\xi$$

The rate at which this changes in time depends on the rate at which it flows in and out of the control volume.

**(3.252)**
$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial t} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u(\xi, t) d\xi = -(f(x_{i+\frac{1}{2}}) - f(x_{i-\frac{1}{2}}))$$

It is more convenient for us to define

**(3.253)**
$$\tilde{u} = \frac{U}{\Delta x_i}, \quad \Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$$

where $\tilde{u}$ represents the average state of that system in the region of interest. Equation (3.252) can consequently be written as

**(3.254)**
$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial t} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u(\xi, t) d\xi = \frac{\partial}{\partial t} (\tilde{u} \Delta x_i) = -(f(x_{i+\frac{1}{2}}) - f(x_{i-\frac{1}{2}}))$$

represents the fundamental idea behind a whole class of modern CFD techniques. We have a process by which a system evolves in time. We could integrate equation (3.252) in time to get the change in $U$ over the time period $[t_q, t_{q+1}]$ as

**(3.255)** $$U^{q+1} - U^q = \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left\{ u(\xi, t^{q+1}) - u(\xi, t^q) \right\} d\xi =$$

$$\left\{ \tilde{u}_i^{q+1} - \tilde{u}_i^q \right\} \left\{ x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}} \right\} = -\int_{t^q}^{t^{q+1}} [f(x_{i+\frac{1}{2}}, \tau) - f(x_{i-\frac{1}{2}}, \tau)] d\tau$$

Equation (3.252) gives us the process. It tell us that what is important is the boundary of the control volume. The evolutionary process of the system is at the boundary. In fact, the resounding message from that equation is

**May the flux be with you!**

From this point of view we could reduce everything down to finding the correct flux at the interface of two control volumes. The flux at the interface between volumes is the process by which our system evolves. As a consequence, the role of grid point $i$ shown in Figure 3.41 is only a place holder for $\tilde{u}$ and its location is "somewhere in the interval". We may still choose to locate it at the middle.

If equation (3.252) is integrated in time one can generate a numerical scheme to solve this equation. This class of schemes are called finite volume schemes. Note that we would be solving for $U$ in equation (3.252). We have assumed that this is some mean value $\tilde{u}$ times the length of the interval. This mean value can then be used to evaluate the fluxes. If we drop the tilde

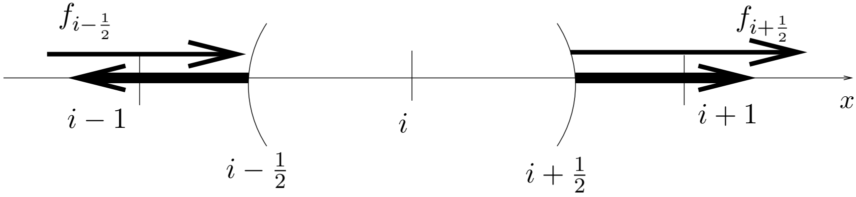

**Figure 3.41. A small control volume around the grid point $i$.**

and label the mean value at $u_i$, we can then write the spatially discretised version of equation (3.252) as

**(3.256)**
$$\frac{\partial}{\partial t}u_i\Delta\xi_i = -(f(u_{i+\frac{1}{2}}) - f(u_{i-\frac{1}{2}}))$$

where,

**(3.257)**
$$u_{i+\frac{1}{2}} = \frac{u_i + u_{i+1}}{2}$$

The algorithm then is

(1) Using equation (3.257) compute $u$ on the boundaries.
(2) Use this $u$ to find the flux on the boundaries.
(3) Employ the fluxes and equation (3.256) to advance the solution in time.
(4) Repeat this process.

If you look at equation (3.252) and equation (3.256), you will see that the argument of the flux term is different. This drives the algorithm that is generated. If we stick with equation

(3.252), we see that the flux term is evaluated at the point $x_{i+\frac{1}{2}}$. This flux can be obtained as

**(3.258)** $$f_{i+\frac{1}{2}} = \frac{f_i + f_{i+1}}{2}$$

This typically involves more calculations. Try it out and see how the two behave. Can you make out that using equation (3.258) actually is equivalent to using FTCS on the grid points. The key is the term $\frac{1}{2}f_i$ will cancel. Now it is clear in this case that if we want to add artificial dissipation of any kind we need to add it to the flux term.

We see that the solution to the equation depends rather critically on the way $f_{i+\frac{1}{2}}$ is obtained. There are a whole host of high resolution schemes which completely depend on the way in which this flux term is calculated. Clearly, if we want to ensure we are up-winding we need to know the direction of propagation at the point. The propagation vector is given by the Jacobian $\partial f/\partial u$. How do we find this at the point $i + 1/2$? Do we take the average of the derivatives at each node? If we take the derivative of the flux at the interface, which flux, meaning how do we calculate it? All of these questions lead to realm of high-resolution schemes.

We saw earlier that the generalised wave equation could result in characteristics intersecting and creating a solution which was multi valued. Let us take a closer look at that situation now. We consider a region where the solution jumps in Figure 3.28. We can pick a convenient control volume as shown in Figure 3.42 It is clear from the figure that the speed $u_s$ at which

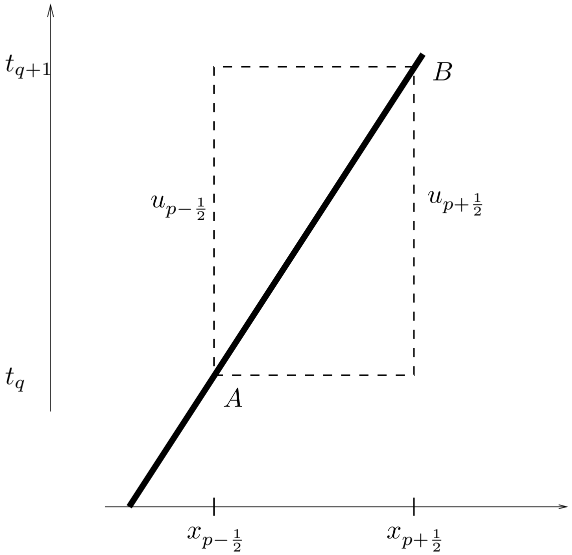

**Figure 3.42. An enlarged region from Figure 3.28 to see if we can arrive at the jump condition that will provide the propagation speed for the discontinuity**

the discontinuity propagates is

**(3.259)**
$$u_s = \lim_{B \to A} \frac{x_{p+\frac{1}{2}} - x_{p-\frac{1}{2}}}{t_{q+1} - t_q}$$

The control volume was deliberately chosen so that equation (3.259) holds. Now let us look at the generalised wave equation as applied to this control volume. We take it in the form given in equation (3.255) since it has no derivatives and we do have a discontinuity that is in the region of interest. This equation when applied to our control volume becomes

**(3.260)**

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}[u(\xi,t_{q+1})-u(\xi,t_q)]d\xi = (u_{p-\frac{1}{2}}-u_{p+\frac{1}{2}})(x_{p+\frac{1}{2}}-x_{p-\frac{1}{2}}) =$$

$$-\int_{t_q}^{t^{q+1}}[f(x_{i+\frac{1}{2}},\tau)-f(x_{i-\frac{1}{2}},\tau)]d\tau = -(f_{p+\frac{1}{2}}-f_{p-\frac{1}{2}})(t_{q+1}-t_q)$$

As indicated in the figure, the value of $u$ to the left of the discontinuity is $u_{p-\frac{1}{2}}$ and consequently, it is the value at time level $t_{q+1}$. At time level $t_q$, the same argument givens us $u_{p+\frac{1}{2}}$. Dividing through by $(t_{q+1}-t_q)$ and substituting from equation (3.259) we get

**(3.261)**
$$(u_{p+\frac{1}{2}}-u_{p-\frac{1}{2}})\frac{x_{p+\frac{1}{2}}-x_{p-\frac{1}{2}}}{t_{q+1}-t_q} = f_{p+\frac{1}{2}}-f_{p-\frac{1}{2}}$$

We take the limit as $B \to A$.

**(3.262)**
$$\lim_{B\to A} u_{p+\frac{1}{2}} = u_R$$

**(3.263)**
$$\lim_{B\to A} u_{p-\frac{1}{2}} = u_L$$

**(3.264)**
$$\lim_{B\to A} f_{p+\frac{1}{2}} = f_R$$

**(3.265)**
$$\lim_{B\to A} f_{p-\frac{1}{2}} = f_L$$

Where $u_R$ and $f_R$ correspond to the conditions on the right hand side of the discontinuity and $u_L$ and $f_L$ correspond to the conditions on the left hand side of the discontinuity. Taking the limit $B \to A$ of equation (3.260) shows us that the wave speed of the discontinuity is given by

$$u_s = \frac{f_R - f_L}{u_R - u_L} \tag{3.266}$$

This is called a jump condition in a general context. In the context of gas dynamics it is called the Rankine-Hugoniot relation.

We will now look at rewriting the discrete / semi-discrete equations in a form that gives us a different perspective on these numerical scheme. This is called the Delta form of the equations.

## The "Delta" form

So far we have looked at problems where we were simulating the evolution of a system. Quite often, we may be interested only in the steady state. That is, we are interested in the solution to Laplace's equation, we may, instead, choose to solve the two-dimensional heat equation and march to the steady state solution in time. We would like to do this efficiently. To lay the foundation for the study of these so called time-marching schemes we will look at the "Delta" form of our equations.

Consider the implicit scheme again. We apply it to the general one-dimensional wave equation given in equation (3.250), which can be discretised as follows

$$\left.\frac{\Delta u}{\Delta t}\right|^{p+1} + \left.\frac{\partial f}{\partial x}\right|^{q+1} = 0. \tag{3.267}$$

As we did before we could use Taylor's series to obtain $f^{q+1}$, [**BM80**], as

**(3.268)**
$$f^{q+1} = f^q + \Delta t \frac{\partial f}{\partial t}^q + \ldots$$

Using the fact that $f = f(u)$ and chain rule we get

**(3.269)**
$$f^{q+1} = f^q + \Delta t \frac{\partial f}{\partial u}\frac{\partial u}{\partial t} + \ldots = f^q + a^q \Delta u^q, \quad a^q = \left.\frac{\partial f}{\partial u}\right|^q$$

where $\Delta u^q = u^{q+1} - u^q$. So, equation (3.267) becomes

**(3.270)**
$$\frac{\Delta u}{\Delta t}^q + \frac{\partial}{\partial x}\left[f^q + a^q \Delta u^q\right] = 0.$$

Now, if we were only seeking the steady state solution, we are actually trying to solve the problem governed by the equation

**(3.271)**
$$\frac{\partial f}{\partial x} = 0$$

For any function $u$ other than the steady state solution this would leave a residue $R(u)$. So at time-level $q$, we have a candidate solution $u^q$ and the corresponding residue

**(3.272)**
$$\frac{\partial f(u^q)}{\partial x} = \frac{\partial f}{\partial x}^q = R(u^q) = R^q$$

We now rewrite equation (3.270) as

**(3.273)**
$$\frac{\Delta u}{\Delta t}^q + \frac{\partial a^q \Delta u^q}{\partial x} = -R^q$$

Multiplying through by $\Delta t$ and factoring out the $\Delta u^q$ gives us the operator form of the equation

**(3.274)**

$$\left\{1 + \Delta t \frac{\partial}{\partial x} a^q \right\} \Delta u^q = -\Delta t R(u(t^q)) = -\Delta t R(t^q) = -\Delta t R^q$$

It should be emphasised that the term in the braces on the left hand side is a differential operator. This equation is said to be in the delta form. Since we have taken the first terms from the Taylor's series expansion of $f^{q+1}$, it is a linearised delta form. We look at this equation carefully now to understand what it does for us. If we are looking only for the steady state solution, with a given initial guess, we can march this equation in time till we get convergence to a steady state (or so we hope).

If we designate $\left\{1 + \Delta t \frac{\partial}{\partial x} a\right\}$ by $\boldsymbol{A}$ we can write the pseudo-code to march to the steady state as follows.

```
Ai = inverse(A)
Guess u

while( R(u) > eps ):
     Du = -Dt Ai R(u)
     u = u + Du
```

Here, Ai is the inverse of the operator $\boldsymbol{A}$. This is called **time-marching**. At any given point in the time-marching scheme, the right hand side of the equation in the delta form determines whether we have the solution or not (look at the while statement in the pseudo-code). In theory, we can compute our $R$ as accurately as we desire and are able to compute. So, when we decide using this accurate $R$ that we have converged we should have an accurate solution.

Given the current approximation we compute the $R$. With these two in hand, left hand side then allows us to obtain a correction to our current approximation to the solution. The nature of this correction determines the rapidity with which we reach the steady state solution. The correction should go to zero as we approach the solution because $R \to 0$. In fact, at the solution, the correction is zero, since $R = 0$. As $R \to 0$, $\Delta u \to 0$ as long as $M$ is not singular. In that case, does the nature of the discrete version of the differential operator $M$ matter? If we are at the solution, as long as this discrete version is not singular, if $\Delta u$ is zero, does it matter what multiplies it? The answer is an emphatic: No! This leaves open the possibility that we can accelerate convergence to the steady solution by an appropriate choice of the operator $M$. What we are effectively saying is that if you are interested only in the steady state solution why not compromise on the transient to get to that solution as quickly as possible. We will see how all of this works. Let us first see how the delta form can be used to obtain a solution.

A simple minded discretisation would be to use BTCS. That is, the spatial derivative is discretised using central differences.

**(3.275)**
$$\left\{ \Delta u_p^q + \Delta t \frac{a_{p+1}^q \Delta u_{p+1}^q - a_{p-1}^q \Delta u_{p-1}^q}{2\Delta x} \right\} = -\Delta t R(u^q)$$

We immediately see that this forms a system of equations, For the typical equation not involving the boundary, we have a sub diagonal entry, a diagonal entry and a super diagonal entry. That is we have a tridiagonal system of equations which can be written as

$$\boldsymbol{A}\boldsymbol{U} = \boldsymbol{B}, \tag{3.276}$$

where $\boldsymbol{A}$ is a tridiagonal matrix (or can be made a tridiagonal matrix), $\boldsymbol{U}$ is a vector of $\Delta u$ and $\boldsymbol{B}$ is made up of $-\Delta t R$ and the boundary conditions. We can choose to solve this system of equations using LU decomposition or any type of iterative scheme. Let us write out the equation to see what it looks like. For convenience we set $\tilde{\sigma}^q = a^q \Delta t / 2\Delta x$ We will use the following problem for the illustration.

(1) We will solve the generalised wave equation on the unit interval $(0, 1]$.
(2) The initial condition is taken to be $u(x, 0) = 2. + \sin(2\pi x)$.
(3) The inlet condition is specified to be $u(0, t) = 2.0$. In this test case it does not change in time.
(4) No exit condition is specified.

We will solve this problem as follows

(1) The delta form and BTCS will be used
(2) The unit interval is split into eight equal sub-intervals using nine grid points.
(3) Our indexing starts at zero. This means our grid points are numbered $0, 1, \cdots, 7, 8$.

(4) Since the inlet condition at grid point $0$ is a constant, $\Delta u^0 = 0$. We still show it in equation (3.278) so that it is valid for the general case.

(5) Since we require the last grid point for the computation we will extrapolate from grid point $7$ to grid point $8$

Since we have seven unknowns, we get seven equations. These are written as a system as shown in equation (3.278). The last equation in this system is a simple extrapolation to the last grid point. It reads

**(3.277)**
$$-u^7 + u^8 = 0 \Rightarrow u^8 = u^7$$

**(3.278)**
$$\begin{pmatrix} 1 & \tilde{\sigma}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\tilde{\sigma}^1 & 1 & \tilde{\sigma}^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\tilde{\sigma}^2 & 1 & \tilde{\sigma}^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\tilde{\sigma}^3 & 1 & \tilde{\sigma}^5 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\tilde{\sigma}^4 & 1 & \tilde{\sigma}^6 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\tilde{\sigma}^5 & 1 & \tilde{\sigma}^7 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\tilde{\sigma}^6 & 1 & \tilde{\sigma}^8 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \left\{ \begin{matrix} \Delta u^1 \\ \Delta u^2 \\ \Delta u^3 \\ \Delta u^4 \\ \Delta u^5 \\ \Delta u^6 \\ \Delta u^7 \\ \Delta u^8 \end{matrix} \right\} = \left\{ \begin{matrix} -\Delta t R^1 - a^0 \Delta u^0 \\ -\Delta t R^2 \\ -\Delta t R^3 \\ -\Delta t R^4 \\ -\Delta t R^5 \\ -\Delta t R^6 \\ -\Delta t R^7 \\ 0 \end{matrix} \right\}$$

You will notice that the matrix $\boldsymbol{A}$ given in equation (3.278) is not symmetric. This is because of the first derivative in our equation. Laplace's equation resulted in a symmetric matrix because it involved second derivatives.

This system of equation can be solved using any number of schemes. You could use Gauss-Seidel or Jacobi iterations to solve them. Since, in this case, the system of equations is small, we could use a direct scheme like Gaussian elimination or LU decomposition.

However, we will recall again that in equation (3.276), the right hand side $\boldsymbol{B}$ determines when we have converged and the quality of the solution. At the solution $\boldsymbol{B} = 0$ as is $\boldsymbol{U} = 0$. As long as the coefficient matrix $\boldsymbol{A}$ is not singular, the result will only affect the transient and not the steady state.[3]

A simple possibility in fact is to replace $\boldsymbol{A}$ with $\boldsymbol{I}$, the identity matrix. If we discretise $R$ using central differences, can you make out that this results in FTCS? Now we will look at replacing $\boldsymbol{A}$ with something that is easy to compute and close to $\boldsymbol{A}$. Since we saw that one of the ways we could solve the system of equations was by factoring $\boldsymbol{A}$ we will see if we can factor this a little more easily. From the form of the operator in equation (3.274), we can write

$$\frac{\partial}{\partial x} = \frac{\partial^-}{\partial x} + \frac{\partial^+}{\partial x}. \tag{3.279}$$

Equation (3.274) can now be rewritten as

$$\left\{1 + \Delta t\frac{\partial^-}{\partial x}a + \Delta t\frac{\partial^+}{\partial x}a\right\}\Delta u = -\Delta t R(u(t)) \tag{3.280}$$

[3] This is not a mathematical statement. It is made here with a certain gay abandon so that we can proceed with developing schemes

This is factored approximately as

$$\left\{1+\Delta t\frac{\partial^-}{\partial x}a\right\}\left\{1+\Delta t\frac{\partial^+}{\partial x}a\right\}\Delta u=-\Delta t R(u(t)) \tag{3.281}$$

This factorisation is approximate because on expanding we find that we have an extra term

$$\begin{aligned}\left\{1+\Delta t\frac{\partial^-}{\partial x}a\right\}\left\{1+\Delta t\frac{\partial^-}{\partial x}a\right\}&=1+\Delta t\frac{\partial^-}{\partial x}a+\Delta t\frac{\partial^+}{\partial x}a\\&\qquad+\underbrace{\Delta t^2\frac{\partial^-}{\partial x}a\frac{\partial^+}{\partial x}a}_{\text{extra term}}\end{aligned} \tag{3.282}$$

If the extra term in equation (3.282) did not occur when the product on the left hand side is expanded, we would have an exact factorisation. However, we will have to live with the approximate factorisation as it stands. We notice that the extra term is of the order of $\Delta t^2$. We have already made an approximation of this order when we linearised the flux term.

An example for the kind of partitioning indicated in equation (3.279) is

$$\begin{aligned}\frac{\partial}{\partial x}a\Delta u&=\frac{a_{i+1}\Delta u_{i+1}-a_{i-1}\Delta u_{i-1}}{2\Delta x}\\&=\frac{a_i\Delta u_i-a_{i-1}\Delta u_{i-1}}{2\Delta x}+\frac{a_{i+1}\Delta u_{i+1}-a_i\Delta u_i}{2\Delta x}\end{aligned} \tag{3.283}$$

We should remember that is not quite how it is applied. In order see this can write equation (3.281) as

(3.284)
$$\left\{1 + \Delta t \frac{\partial^-}{\partial x} a\right\} \Delta u^* = -\Delta t R(u(t))$$

(3.285)
$$\left\{1 + \Delta t \frac{\partial^+}{\partial x} a\right\} \Delta u = \Delta u^*$$

(3.286)

The first equation gives us a lower triangular matrix equation. The second an upper triangular matrix. It is easy to solve for the $\Delta u^*$ and then follow up with a step of solving for $\Delta u$.

In this case, the error due to the approximate factorisation is the term

(3.287)
$$\Delta t^2 \left\{ \frac{a_{i+1}\Delta u_{i+1} - 2a_i \Delta u_i + a_{i-1}\Delta u_{i-1}}{4\Delta x^2} \right\}$$

This is obtained by the repeated application of the two operators.

### The One-Dimensional Second Order Wave Equation

When one speaks of the wave equation, normally it is understood that the topic under discussion is the one-dimensional second order wave equation. However, the first order version of the equation is of such importance to CFD that we have relegated "the" wave equation to the end of the chapter. There is more to be learnt from this equation.

(3.288)
$$\frac{\partial^2 u}{\partial t^2} - a^2 \frac{\partial^2 u}{\partial x^2} = 0$$

We are looking at this equation with out any reference to the domain or boundary conditions. In operator form this can be factored as

$$\left\{ \frac{\partial}{\partial t} + a\frac{\partial}{\partial x} \right\} \left\{ \frac{\partial}{\partial t} - a\frac{\partial}{\partial x} \right\} u = 0 \tag{3.289}$$

Inspection of the differential operator on the left hand side reveals that it has been written as a product of two one-dimensional first order wave operators. The first operator corresponds to a "right moving" wave and the second one to a "left moving" one. The corresponding characteristics are $x+at$ and $x-at$. One can solve this equation numerically by employing a CTCS scheme, that is, a central difference in time and a centred difference in space.

---

**Assignment 3.17**

(1) Solve the second order wave equation on the interval $(-1, 1)$. The initial condition is $u(x, 0) = 0$ for all $x$ except $u(0, 0) = 1$.
(2) Check the stability condition for the resulting automaton.

---

We will take a small detour into the classification of differential equations. First some observations on what we have so far.

(1) The first order linear one-dimensional wave equation had a single characteristic.

(2) The second order linear one-dimensional equation that we have seen just now has a set of two characteristics that are distinct.

The second item in the list above makes an assumption. We often tend to get prejudiced by our notation. For example, we identified $a$ as the propagation speed and that ties us down to the coordinates $x$ and $t$. To see this let us write the equation in terms the independent variable $x$ and $y$ The rewritten wave equation is

$$\frac{\partial^2 u}{\partial y^2} - a^2 \frac{\partial^2 u}{\partial x^2} = 0 \tag{3.290}$$

We now consider the case when $a = i = \sqrt{-1}$. What happens to our equation? It becomes Laplace's equation! What happens to the characteristics? Well they become $x + iy$ and $x - iy$. In complex variables parlance $z$ and $\bar{z}$, that is a complex coordinate and its conjugate[**Ahl79**][**Chu77**]. Functions of $\bar{z}$ are not analytic. So, we look for solutions of only $z$. A trip into the beautiful world of complex analysis would be more than the small detour promised here. The student is encouraged to review and checkout material on conformal mapping.

We now summarise a scheme to classify the behaviour of differential equations.

(1) If the characteristics of a system of differential equations are real and distinct we say the system is **hyperbolic** in nature.
(2) If the characteristics are complex then the system is said to be **elliptic** in nature.

(3) If they are real and identical then the system of equations is said to be **parabolic**.
(4) If we get a combination of the above then we have a mixed system and identify it as such.

Please note that this does not restrict us to second order equations.

You can ask the question: What's the big deal? Why are you bothering to define this at this point? After all we managed so far with out his classification.

This is true, we have used this classification without quite being aware of it. Why did we specify the data on all sides for the Laplace' equation? Can you solve Laplace's equation as an initial value problem with data prescribed on one side. How about the wave equation. Can you go to the beach and assert that the wave shall be a certain height or that a stream of water shall have a paper boat in it at any time that you insist, however, someone else is placing the boats somewhere upstream of you. The nature of the equations is intimately tied to the physics of the problem and to the way we apply/can apply boundary conditions.

---

**Assignment 3.18**

Classify the following equations as elliptic, parabolic, and hyperbolic.

(1) $\frac{\partial u}{\partial t} - a\frac{\partial u}{\partial x} = 0$
(2) $\frac{\partial u}{\partial t} - u\frac{\partial u}{\partial x} = 0$

(3) Discretise the second order linear wave equation given by

$$\frac{\partial^2 u}{\partial t^2} - \lambda^2 \frac{\partial^2 u}{\partial x^2} = 0. \tag{3.291}$$

Is the scheme stable? What are the characteristics associated with the equation? What is the consequence of taking $\lambda = i = \sqrt{-1}$?

---

## Important ideas from this chapter

- Laplace's equation is averaging, smoothing. The maximum and minimum of the solution occur on the boundary. The solution is unique.
- Marching heat equation in time is the same as sweeping Laplace's equation in space. This means that if the initial condition to the heat equation has any discontinuities in it, the action of the equation to smooth the discontinuity.
- The quasi-linear one-dimensional wave equation can take an initial condition which is smooth and generate a discontinuity in the solution.
- An iterative technique is basically a map of a space onto itself. If the map is a contraction mapping, the scheme will converge to a fixed point which is the solution to the equation.
- Solution techniques that are obvious don't always work.

(1) FTCS is unconditionally unstable for the first order linear wave equation. It is conditionally stable when applied to the heat equation.

(2) The scheme may not be consistent which means that the modified equation may not converge to the original equation.

- FTBS and FTFS are conditionally stable depending on the direction of propagation. This leads a class of schemes called up-winding schemes.
- Solution techniques can be dissipative and dispersive.
- Do not confuse the convergence of the scheme, that is $u^h \to u$ as $h \to 0$ with the convergence of the algorithm or the specific implementation which is your program.
- The boundary conditions provided by the physics and specified for the governing differential equations may not sufficient to solve the problem employing an algorithm. Individual algorithms may require auxiliary boundary conditions that need to be generated in some fashion.
- All the schemes that we have seen can be written in a finite volume context using the fluxes at the volume boundaries. These fluxes completely determine the evolution of the system and hence need to be computed correctly.
- Partial differential equations are classified broadly into three types: elliptic, parabolic and hyperbolic equations.

# One-Dimensional Inviscid Flow

We will look at a class of problems that are typically studied in a course on gas dynamics. I would suggest that you do at least a quick review of the material[**LR57**].

We have seen a variety of simple problems in chapter 3. The wave equation and the heat equation were one-dimensional equations for a dependent variable $u$. We say one dimension as we see only one spatial coordinate. However, mathematically they were two dimensional as the parameters of interest varied in time. The equations, therefore, had two independent variables $(x, t)$. These equations in fact dealt with the evolution of some quantity like density or some measure of internal energy (temperature is often a good measure of internal energy). Laplace's equation is a two-dimensional equation governing one dependent variable. It was essentially an equilibrium problem involving two space coordinates. Recall that it was indeed the steady state equation of the corresponding two dimensional heat equation (two spatial coordinates, one time coordinate).

To summarise, we have seen equations in one dependant variable up to this point. We will now look at a system of equations governing more than one dependent variable. Where would we run into such a situation? In a general fluid flow situation, we would have

(1) the mass distribution of the fluid, which we capture through the field property mass density or just density, $\rho$,

(2) the momentum distribution, which we capture through the momentum density, $\rho\vec{V}$,(remember that momentum is a vector, and would normally have three components)

(3) the energy distribution, through the density of total energy, $\rho E_t$.

Of course, in more complicated situations, we may want to track other parameters like fuel concentration, moisture concentration (humidity) and so on. These kinds of parameters are often referred to as species and species concentration, as they seem to be variations of mass and mass density. We will look at the derivation of the governing equations for fluid flow in chapter 5. Right now, the easiest place to start our study is the one-dimensional Euler's equation.

Having decided on the equations that we will study in this chapter, we will now address the other important component of the problem definition: the boundary conditions. We have seen this material in an earlier section 3.15, it is important that we remind ourselves.

Briefly, we know from the theory of differential equations, we may require a certain number of boundary conditions to solve the problem at hand. However, it is possible that the algorithm we generate requires more conditions in order to work. So, there are issues with respect to the boundary conditions that need to be addressed. We will look at these issues in this chapter. In this context, the one-dimensional equations are the best place to start.

This chapter will lay the foundation for working with a system of equations using the one-dimensional Euler equations.

Some of the things done here can be extended to multiple dimensions in spirit and possibly form. We will see that the process of applying boundary conditions is one of the things that will be effectively extended to multiple dimensions.

### What is one-dimensional flow?

First, one-dimensional flow should not be confused with uni-directional flow. Uni-directional flow is flow where all the velocity vectors point in the same direction, the "uni-direction" so to speak. If we consider a plane perpendicular to the uni-direction, the magnitude of the velocities at points on the plane need not be the same. Uni-directional flow with all the velocities at points on a plane perpendicular to the uni-direction is shown in Figure 4.1 On the other hand, we would have a one-dimensional flow field if the flow field was not only uni-directional, but also the velocity vectors on any plane perpendicular to the direction of flow had the same magnitude. An example of this is shown in Figure 4.2 It should noted that we will allow for a change in magnitude going from one plane perpendicular to the flow field to another parallel plane.

We do not usually deal with infinite flow fields in real life. We may use that as an effective approximation. The easiest way to generate a uni-directional flow field is to confine the fluid to a duct. Consider classical problems like Couette flow and Hagen-Poisseuille flows. They are uni-directional. One-dimensional flow fields are near impossible to generate just as it is to get a room where the velocity vectors are zero! So, bear in mind that the assumption of one-dimensional flow is a simplifying assumption.

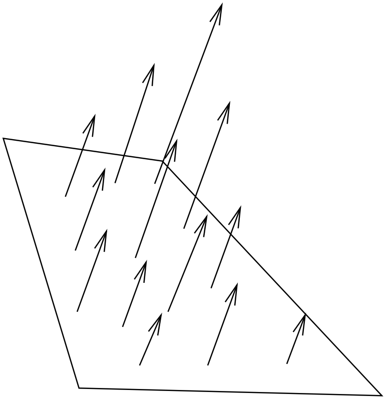

**Figure 4.1. An example of a uni-directional flow. The whole flow field is not shown. Instead, a plane perpendicular to the direction of the flow is picked and the velocity vectors at some points on the plane are shown. The vectors, as expected, are perpendicular to the plane. However, the magnitudes of these vectors need not be the same.**

We will consider a proto-typical one-dimensional flow. We have a pipe with a circular cross-section of constant area. One end of this pipe is connected to a large pressure vessel, see Figure 4.3. These large pressure vessels are also called reservoirs. The pressure vessel is full of dry air; There! we got rid of the humidity with this assumption. We will have a valve at end of the pipe which is connected to the pressure vessel. The valve can be operated to open the pipe completely and start the flow instantaneously or as slowly as one chooses. For now we will assume that we open it instantaneously. The other end of the pipe is open to the atmosphere.

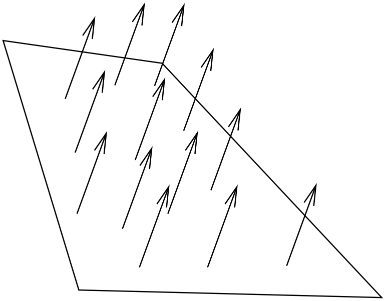

Figure 4.2. An example of a one-dimensional flow. The whole flow field is not shown. Instead a plane perpendicular to the direction of the flow is picked and the velocity vectors at some points on the plane are shown. The vectors as expected are perpendicular to the plane. In the one-dimensional case the magnitudes of these vectors are the same.

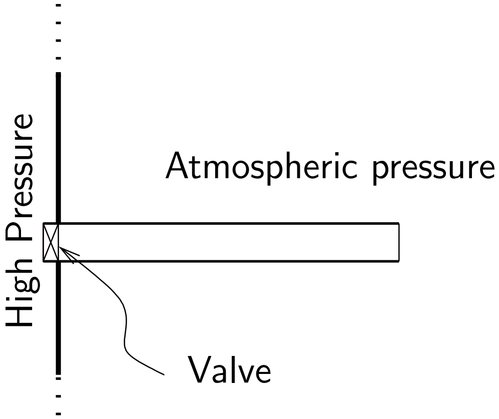

Figure 4.3. A schematic diagram of a setup that can be modelled as one-dimensional flow.

We should be clear as to what this diagram in figure 4.3 indicates. To this end we will restate salient features of the problem.

- The initial pressure on the left hand side of the pipe is higher than atmospheric pressure.
- The pressure vessel is so large that there will be no appreciable drop in pressure during the experiment.
- The valve is on the left hand side of the pipe as shown.
- The right end is open to a quiescent atmosphere.
- The initial condition in the pipe is the atmospheric condition.
- The air in the pipe is stationary to begin with.

Now that we have an idea as to the conditions with which we will operate, we can do a thought experiment to see where we go from here. If I were to suddenly open the valve, I would expect a compression wave to propagate from the left to right. This I expect from the fact that the pressure vessel is at a higher pressure than the pipe at this point in time. (An expansion wave propagates into the pressure vessel, we really won't worry about that.) Remember that a wave/surface is a compression wave if the flow passing through it experiences an increase in density. On the other hand if the density drops it is called an expansion wave. Once the compression wave comes to the end of the pipe, an expansion wave may propagate back through the pipe, bear in mind that the atmosphere now is at a lower pressure than the pipe. (As we did with the reservoir, we ignore the compression wave once it leaves the pipe.) Eventually, a flow field is setup in the pipe. In reality, this whole process can happen quite quickly.

If we ignore viscosity, we expect that a one-dimensional flow will be set up in this perfectly straight and smooth pipe. That is the hand waving physics of the problem. Let us now take a look at the equations that govern this one-dimensional, inviscid, perfect gas flow.

**The Governing Equations.** As is usual, we will write the balance laws for mass, momentum and energy for one-dimensional flow. If this does not sound familiar to you, there are three possible options at this point.

(1) You can take a quick look at chapter 5 to build up some of the background. That chapter is quite terse and general though.
(2) You could take a look at a standard text which will provide more comprehensive material than chapter 5.
(3) You can proceed with this chapter, suspending disbelief for now, and verifying for yourself that the equations are correct at a later date.

Back to the governing equations. Before we set out the equations enforcing the balance laws, we make the following assumptions to make life a little easier. We assume the flow is inviscid and adiabatic.

The equation that captures the principle of conservation of mass in one dimension is

**(4.1)** $$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0$$

and is often referred to as the one-dimensional conservation mass equation or the one-dimensional mass balance equation.

Similarly the equation governing the conservation of linear momentum can be written as

$$\textbf{(4.2)} \qquad \frac{\partial \rho u}{\partial t} + \frac{\partial(\rho u^2 + p)}{\partial x} = 0$$

and can also be called the one-dimensional balance of linear momentum equation. Note that this equation has no viscous terms as the fluid is assumed to be inviscid. Finally, the equation that captures the principle of conservation of energy is given by

$$\textbf{(4.3)} \qquad \frac{\partial \rho E_t}{\partial t} + \frac{\partial(\rho E_t + p)u}{\partial x} = 0$$

Again, it is clear that there are no terms related to viscosity or thermal conductivity in the energy equation which can be called the equation of energy balance.

Take a look at the three equations. They are very similar in form. These three equations can be consolidated into a single vector equation. The inviscid equations are called the Euler equations and the viscous equations for a Navier-Stokes viscous model are called the Navier-Stokes equations. These equations are derived in chapter 5. The one-dimensional Euler equation is

$$\textbf{(4.4)} \qquad \frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial x} = \vec{0}$$

where

$$\textbf{(4.5)} \quad \vec{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho E_t \end{bmatrix}, \qquad \vec{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (\rho E_t + p)u \end{bmatrix}, \text{ and } \quad \vec{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

where, $E_t$ is the specific total energy, and is given in terms of the specific internal energy and the speed the of the fluid as

**(4.6)**
$$E_t = e + \frac{u^2}{2}$$

For a calorically perfect gas the specific internal energy can be written in terms of the the specific heat at constant volume and the temperature

**(4.7)**
$$e = C_v T$$

We had $E_t$. We wrote that in terms of $e$, with which we added one more equation. Now, we have added yet another unknown, temperature T and we need one more corresponding equation. Fortunately for us, we can close this process by invoking the equation of state

**(4.8)**
$$p = \rho R T$$

We have added one more equation consisting only of variables defined so far and we have closure. By closure, I mean we have enough equations to determine all the parameters in the equations.

You will note that equation (4.4) is in the divergence free form or the **conservative form**. We have already seen this form in the case of the wave equation in section 3.16. In CFD there is another reason attributed to the name conservative form and that is the fact that for a steady flow across a discontinuity like a shock, see [**LR57**], the quantity $E$ is continuous across the discontinuity. For example, there may be a jump in $\rho$ and a jump in $u$, but there is no jump in $\rho u$ across the shock. The dependent variables $Q$ result in the flux term $E$. Equation (4.4) is said to be in conservative form and $Q$ given in equation (4.5) are called **conservative variables**.

---

**Assignment 4.1**

(1) Write a program that given $N$ will create an array that contains $N$ number of $Q$s. Add a function to this program that will compute a $Q$ given $p$, $T$, and $u$. Take $p = 101325\,N/m$, $T = 300\,K$ and $u = 0\,m/s$ and create and set the values in and array of size $N = 11$.

(2) Write functions that compute $p$, $\rho$, $T$ and $u$ when given a $Q$. Call these functions GetP, GetRho, GetT and GetU. Use these functions to retrieve the values set in the previous exercise.

(3) Write a function, GetE, that returns $E$ given $Q$.

(4) Test the code and set it aside. You will use it later.

---

The first order linear one-dimensional wave equation that we studied in the previous chapter is in the non-conservative form. We will try to get these equations that make up the one-dimensional Euler's equation into a similar form. We will do this in the next section.

### Analysis of the One-dimensional Equations

Unfortunately, the system of equations (4.4) do not quite resemble the wave equation that we have been studying so far. However, this is easy to fix as it does resemble the generalised advection equation which we studied in section 3.16 and in given in equation (3.250). We employ the chain rule as we did in the case of the generalised advection equation. In that case, we saw that the derivative of the flux term gave us the local propagation speed. We see that the flux term $\vec{E}$ can in fact be written as a function of $\vec{Q}$. We can then write

**(4.9)** $$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial x} = \frac{\partial \vec{Q}}{\partial t} + \nabla_Q \vec{E} \frac{\partial \vec{Q}}{\partial x} = \frac{\partial \vec{Q}}{\partial t} + \boldsymbol{A} \frac{\partial \vec{Q}}{\partial x} = \vec{0}$$

Here, $\nabla_Q$ is the gradient operator with respect to the variables $\vec{Q}$ and $\mathbf{A}$ is called a **flux Jacobian**. Rewriting the equation in the non-conservative form in terms of the conservative variables as

**(4.10)** $$\frac{\partial \vec{Q}}{\partial t} + \boldsymbol{A} \frac{\partial \vec{Q}}{\partial x} = \vec{0}$$

The equation now looks like the one dimensional linear wave equation. It still represents a coupled system which we can verify by actually figuring out the entries in the matrix representation of $\mathbf{A}$. If we were to refer to the components of $\vec{Q}$ as $q_i$, the components of $\vec{E}$ as $e_i$, and the components of $\mathbf{A}$ as $a_{ij}$, then we get

**(4.11)** $$a_{ij} = \frac{\partial e_i}{\partial q_j}$$

Now, in order to evaluate these derivatives, it is a good idea to write each of the $e_i$ in terms of the $q_j$. From equation (4.5),

**(4.12)** $$e_1 = \rho u = q_2$$

**(4.13)** $$e_2 = \rho u^2 + p = \frac{q_2^2}{q_1} + p$$

**(4.14)** $$e_3 = (\rho E_t + p)u = (q_3 + p)\frac{q_2}{q_1}$$

clearly we need to derive an expression for pressure $p$ and one for $E_t$. We have already come up with the relevant relationships to close the system of equations. From equations (4.6) and (4.7) we get

**(4.15)** $$E_t = e + \frac{u^2}{2} = C_v T + \frac{u^2}{2}$$

Substituting for the temperature from the equation of state (4.8), we get

**(4.16)** $$E_t = \frac{C_v p}{\rho R} + \frac{u^2}{2}$$

and given that $C_v = R/(\gamma - 1)$ we get

**(4.17)** $$E_t = \frac{p}{\rho(\gamma - 1)} + \frac{u^2}{2}$$

Solving for $p$ we obtain

**(4.18)** $$p = (\gamma - 1)\left\{\rho E_t - \frac{\rho u^2}{2}\right\} = (\gamma - 1)\left\{q_3 - \frac{q_2^2}{2q_1}\right\}$$

substituting back we get

**(4.19)**

$$\vec{E} = \begin{bmatrix} q_2 \\ (\gamma - 1)q_3 + \frac{1}{2}(3-\gamma)\dfrac{q_2^2}{q_1} \\ \left(\gamma q_3 - \frac{1}{2}(\gamma-1)\dfrac{q_2^2}{q_1}\right)\dfrac{q_2}{q_1} \end{bmatrix}$$

Now differentiating, we get the components of $\boldsymbol{A}$ as

**(4.20)**

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)\dfrac{q_2^2}{q_1^2} & (3-\gamma)\dfrac{q_2}{q_1} & (\gamma-1) \\ (\gamma-1)\dfrac{q_2^3}{q_1^3} - \gamma q_3\dfrac{q_2}{q_1^2} & \gamma\dfrac{q_3}{q_1} - \frac{3}{2}(\gamma-1)\dfrac{q_2^2}{q_1^2} & \gamma\dfrac{q_2}{q_1} \end{bmatrix}$$

Substituting back for the terms $q_1, q_2,$ and $q_3$ we get

**(4.21)**

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)u^2 & (3-\gamma)u & (\gamma-1) \\ (\gamma-1)u^3 - \gamma E_t u & \gamma E_t - \frac{3}{2}(\gamma-1)u^2 & \gamma u \end{bmatrix}$$

So, we now have the components of our flux Jacobian.

**Assignment 4.2**

1. Write a function GetA to get the matrix $A$ given $Q$.
2. Analytically, verify that $E = AQ$. (Obviously, this need not be true for all conservation equations)
3. Having shown that $E = AQ$, use it to test your function GetA.

We have managed to get our one-dimensional equations to "look" like the wave equation. We are disappointed, but not surprised that the matrix is not diagonal. If it were a diagonal matrix, the system of equations represented by equation (4.10) would be a system of independent equations, each one a wave equation of the kind we have studied earlier. Unfortunately, $\mathbf{A}$ is not diagonal. Maybe if we changed the dependent variables we could get a non-conservative form which is decoupled. This derivation is a standard fluid mechanics procedure, we will do it as an illustration of the process. We choose the the dependent variable $\tilde{Q}$ given by

**(4.22)**
$$\tilde{Q} = \begin{bmatrix} \rho \\ u \\ p \end{bmatrix}$$

We will now proceed to transform the equations governing balance of mass, momentum and energy from $\vec{Q}$ to $\tilde{Q}$. We start with equation of conservation of mass. Let us expand the equation (4.1) using product rule to get

$$\frac{\partial \rho}{\partial t} + \rho\frac{\partial u}{\partial x} + u\frac{\partial \rho}{\partial x} = 0 \tag{4.23}$$

We now expand conservation of linear momentum, (4.2), and simplify using balance of mass.

$$\rho\frac{\partial u}{\partial t} + \overset{\text{$u$ times conservation of mass}}{\boxed{u\frac{\partial \rho}{\partial t}}} + \rho u\frac{\partial u}{\partial x} + \boxed{u\frac{\partial \rho u}{\partial x}} + \frac{\partial p}{\partial x} = 0 \tag{4.24}$$

So, the momentum equation simplifies to

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} = 0 \tag{4.25}$$

Our intent is to finally write the equations in the form

$$\frac{\partial \tilde{Q}}{\partial t} + \tilde{\boldsymbol{A}}\frac{\partial \tilde{Q}}{\partial x} = \vec{0} \tag{4.26}$$

dividing through by $\rho$ gives

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{1}{\rho}\frac{\partial p}{\partial x} = 0 \tag{4.27}$$

Finally, we expand the equation of conservation of energy (4.3).

$E_t$ times conservation of mass

**(4.28)**

$$\rho\frac{\partial E_t}{\partial t} + \boxed{E_t\frac{\partial \rho}{\partial t}} + \rho u\frac{\partial E_t}{\partial x} + \boxed{E_t\frac{\partial \rho u}{\partial x}} + u\frac{\partial p}{\partial x} + p\frac{\partial u}{\partial x} = 0$$

Substituting for $E_t$ from equation (4.17) we get

**(4.29)** $$\rho\frac{\partial}{\partial t}\left(\frac{p}{\rho(\gamma-1)} + \frac{u^2}{2}\right) + \rho u\frac{\partial}{\partial x}\left(\frac{p}{\rho(\gamma-1)} + \frac{u^2}{2}\right) + u\frac{\partial p}{\partial x} + p\frac{\partial u}{\partial x} = 0$$

Separating out the individual terms we get

**(4.30)** $$\rho\frac{\partial}{\partial t}\left(\frac{p}{\rho(\gamma-1)}\right) + \boxed{\rho u\tfrac{\partial u}{\partial t}} + \rho u\frac{\partial}{\partial x}\left(\frac{p}{\rho(\gamma-1)}\right) + \boxed{\rho u^2\tfrac{\partial u}{\partial x}} + \boxed{u\tfrac{\partial p}{\partial x}} + p\frac{\partial u}{\partial x} = 0$$

$u$ times equation (4.25)

This gives us

**(4.31)** $$\rho\frac{\partial}{\partial t}\left(\frac{p}{\rho(\gamma-1)}\right) + \rho u\frac{\partial}{\partial x}\left(\frac{p}{\rho(\gamma-1)}\right) + p\frac{\partial u}{\partial x} = 0$$

on expanding the derivatives we get

**(4.32)**

$$\frac{1}{\gamma-1}\frac{\partial p}{\partial t} - \frac{p}{\rho(\gamma-1)}\frac{\partial \rho}{\partial t} + \frac{u}{\gamma-1}\frac{\partial p}{\partial x} - \frac{p}{\rho(\gamma-1)}u\frac{\partial \rho}{\partial x} + p\frac{\partial u}{\partial x} = 0$$

from equation (4.23) we can substitute for

**(4.33)**
$$-\frac{p}{\rho(\gamma-1)}\left(\frac{\partial \rho}{\partial t}+u\frac{\partial \rho}{\partial x}\right)=\frac{p}{\gamma-1}\frac{\partial u}{\partial x}$$

Substituting from (4.33) into (4.32) and multiplying through by $(\gamma-1)$ we get

**(4.34)**
$$\frac{\partial p}{\partial t}+u\frac{\partial p}{\partial x}+\gamma p\frac{\partial u}{\partial x}=0$$

We can write the three equations (4.23), (4.27), and (4.34) as

**(4.35)**
$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ u\\ p\end{bmatrix}+\begin{bmatrix}u & \rho & 0\\ 0 & u & \frac{1}{\rho}\\ 0 & \gamma p & u\end{bmatrix}\frac{\partial}{\partial x}\begin{bmatrix}\rho\\ u\\ p\end{bmatrix}=\begin{bmatrix}0\\ 0\\ 0\end{bmatrix}$$

This equation can be written in a compact form as

**(4.36)**
$$\frac{\partial \tilde{Q}}{\partial t}+\tilde{\boldsymbol{A}}\frac{\partial \tilde{Q}}{\partial x}=\vec{0}$$

We got a different non-conservative form. The $\tilde{\boldsymbol{A}}$ is also not a diagonal matrix. The equations are still coupled. It is clear that we cannot go about hunting at random for the set of dependent variables that will give us a decoupled system of equations. We have to use some strategy to find these variables.

We will now try to see what we can do to manipulate these equations to decouple them. For anyone familiar with matrices, similarity transformations, eigenvalues, left eigenvectors, right eigenvectors can read on. If you are not comfortable with these terms, I would suggest that you take a detour to the appendix B-1 and then come back to this point.

Back to the discussion at hand. If we want to develop a systematic process to hunt for the diagonal form of our governing equation, we can look at the relationship between the two non-conservative forms that we have derived so far and that may give us a clue. What is the relationship between $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ ? If we answer that question we can then figure out how to get the diagonal form. In order to determine the answer we need to use chain rule to relate $\vec{Q}$ and $\tilde{Q}$ as

**(4.37)**
$$d\vec{Q} = \boldsymbol{P} d\tilde{Q}$$

In terms of components this is

**(4.38)**
$$dq_i = \sum_j p_{ij} d\tilde{q}_j = \sum_j \frac{\partial q_i}{\partial \tilde{q}_j} d\tilde{q}_j$$

where $q_i$ and $\tilde{q}_j$ are components of $\vec{Q}$ and $\tilde{Q}$ respectively. If we were to multiply equation (4.9) by $\boldsymbol{P}^{-1}$, we would have

**(4.39)**
$$\boldsymbol{P}^{-1}\left(\frac{\partial \vec{Q}}{\partial t} + \boldsymbol{A}\frac{\partial \vec{Q}}{\partial x}\right) = \boldsymbol{P}^{-1}\frac{\partial \vec{Q}}{\partial t} + \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^{-1}\frac{\partial \vec{Q}}{\partial x} = \vec{0}$$

employing equation (4.37) we get

**(4.40)**
$$\frac{\partial \tilde{Q}}{\partial t} + \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}\frac{\partial \tilde{Q}}{\partial x} = \vec{0}$$

We rewrite our equation for the non-conservative form in $\tilde{Q}$

$$\frac{\partial \tilde{Q}}{\partial t} + \tilde{\boldsymbol{A}} \frac{\partial \tilde{Q}}{\partial x} = \vec{0} \tag{4.41}$$

by comparison we see that

$$\tilde{\boldsymbol{A}} = \boldsymbol{P}^{-1} \boldsymbol{A} \boldsymbol{P} \tag{4.42}$$

We say that $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ are related through a **similarity transformation**. We just want a similarity transformation that will give us a diagonal matrix.

---

**Assignment 4.3**

(1) Determine the transformation matrix $\boldsymbol{P}$ using the definition given in equation (4.37).

(2) As a continuation from assignment 4.2, evaluate $\tilde{\boldsymbol{A}}\tilde{Q}$. What do you conclude about $\vec{E}$, $\vec{Q}$, $\boldsymbol{A}$, $\tilde{\boldsymbol{A}}$, and $\tilde{Q}$?

---

Before we start this discussion, it should be mentioned that for once I am going to use prior knowledge of the problem at hand to keep the matrix algebra and the associated manipulation simple. In this case, for example, I know that we will get three distinct eigenvalues and that the similarity transformation exists to diagonalise the matrix $\boldsymbol{A}$.

If we were to find the eigenvalues of $\boldsymbol{A}$ to be $\lambda_i, i = 1, 2, 3$ and the corresponding eigenvectors $x_i$, we know that for the $k^{th}$ eigenvalue

$$\boldsymbol{A} x_k = \lambda_k x_k \tag{4.43}$$

or in terms of components

$$a_{ij} x_{jk} = \lambda_k x_{jk} \tag{4.44}$$

clearly, we can form a matrix $\boldsymbol{X}$ by placing the individual eigenvectors $x_k$ as the columns of $\boldsymbol{X}$. We rewrite equation (4.44) as

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda} \tag{4.45}$$

where, $\boldsymbol{\Lambda}$ is a diagonal matrix of the eigenvalues. If we were to pre-multiply equation (4.45) by $\boldsymbol{X}^{-1}$, we get

$$\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}^{-1}\boldsymbol{X}\boldsymbol{\Lambda} = \boldsymbol{\Lambda} \tag{4.46}$$

Yes! We have a scheme to generate the similarity transformation that we need and consequently, the new dependent variables that are governed by the corresponding equations. If we were to define $d\hat{Q} = \boldsymbol{X}^{-1} dQ$, Then we can pre-multiply the equation (4.9) to get

$$\boldsymbol{X}^{-1}\left\{\frac{\partial \vec{Q}}{\partial t} + \boldsymbol{A}\frac{\partial \vec{Q}}{\partial x}\right\} = \boldsymbol{X}^{-1}\frac{\partial \vec{Q}}{\partial t} + \boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}^{-1}\frac{\partial \vec{Q}}{\partial x} \tag{4.47}$$

Which reduces to

$$\frac{\partial \hat{Q}}{\partial t} + \boldsymbol{\Lambda}\frac{\partial \hat{Q}}{\partial x} = \vec{0} \tag{4.48}$$

If the components of $\hat{Q}$ are $\hat{q}_1, \hat{q}_2,$ and $\hat{q}_3$ then equation (4.48) in terms of components this would be

**(4.49)** $$\frac{\partial \hat{q}_1}{\partial t} + \lambda_1 \frac{\partial \hat{q}_1}{\partial x} = 0$$

**(4.50)** $$\frac{\partial \hat{q}_2}{\partial t} + \lambda_2 \frac{\partial \hat{q}_2}{\partial x} = 0$$

**(4.51)** $$\frac{\partial \hat{q}_3}{\partial t} + \lambda_3 \frac{\partial \hat{q}_3}{\partial x} = 0$$

As we can see the equations seem to be decoupled. The individual equations now resemble the first order wave equation that we have analysed earlier. There is now hope that some of that analysis will carry over to this coupled system of equations.

We have only shown that this decoupling, if possible, will be nice to do. So, how do we find the eigenvalues. You can try to find the eigenvalues of $\boldsymbol{A}$ directly. Again, prior experience with the problem shows a way to get an easier similar matrix. It turns out, it is easier find the eigenvalues of $\tilde{\boldsymbol{A}}$. $\tilde{\boldsymbol{A}}$ and $\boldsymbol{A}$ are related through a similarity transformation. A property of two matrices that are similar to each other is that their eigenvalues are the same.

The eigenvalue problem corresponding to $\tilde{\boldsymbol{A}}$ is written as

**(4.52)** $$\tilde{\boldsymbol{A}} x = \lambda x$$

The characteristic equation corresponding to $\tilde{\boldsymbol{A}}$ is given by

$$\begin{vmatrix} u-\lambda & \rho & 0 \\ 0 & u-\lambda & \frac{1}{\rho} \\ 0 & \gamma p & u-\lambda \end{vmatrix} = 0 \tag{4.53}$$

Evaluating the determinant gives us the characteristic equation

$$(u-\lambda)\left\{(u-\lambda)^2 - \frac{\gamma p}{\rho}\right\} = 0 \tag{4.54}$$

The eigenvalues are $u$, $u+c$ and $u-c$ where $c$ is the speed of sound given by $c^2 = \gamma p/\rho$. Now, some more matrix algebra machinery will allow us to determine the corresponding eigenvectors. This will give us the matrix $\boldsymbol{X}$, the matrix of eigenvectors of matrix $\boldsymbol{A}$. This matrix is often referred to as the modal matrix. As was discussed in an earlier section 3.16, we see that we have three real and distinct eigenvalues. This helps us classify this system of equations as being hyperbolic.

$$\boldsymbol{X} = \begin{bmatrix} 1 & 1 & 1 \\ u & u+c & u-c \\ \frac{u^2}{2} & \frac{c^2}{\gamma-1} + \frac{u^2}{2} + cu & \frac{c^2}{\gamma-1} + \frac{u^2}{2} - cu \end{bmatrix} \tag{4.55}$$

For a calorically perfect gas, we recognise $\frac{c^2}{\gamma-1} + \frac{u^2}{2}$ as the total enthalpy $H_t = C_p T_o$. So, $\boldsymbol{X}$ can be written more compactly as

(4.56)
$$\boldsymbol{X} = \begin{bmatrix} 1 & 1 & 1 \\ u & u+c & u-c \\ \frac{u^2}{2} & H_t + cu & H_t - cu \end{bmatrix}$$

For clarity we will indicate the eigenvalue as a subscript and write the corresponding entries of the eigenvector.

(4.57)
$$\boldsymbol{x}_u = \begin{bmatrix} 1 \\ u \\ \frac{u^2}{2} \end{bmatrix}, \quad \boldsymbol{x}_{u+c} = \begin{bmatrix} 1 \\ u+c \\ H_t + cu \end{bmatrix}, \quad \boldsymbol{x}_{u-c} = \begin{bmatrix} 1 \\ u-c \\ H_t - cu \end{bmatrix}$$

We should note that though the eigenvalues of $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ happen to be the same, the eigenvectors are not. In fact, the modal matrix $\tilde{\boldsymbol{X}}$ corresponding to $\tilde{\boldsymbol{A}}$ is

(4.58)
$$\tilde{\boldsymbol{X}} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & \frac{c}{\rho} & -\frac{c}{\rho} \\ 0 & c^2 & c^2 \end{bmatrix}$$

and the inverse is

$$\tilde{\boldsymbol{X}}^{-1} = \begin{bmatrix} 1 & 0 & -\dfrac{1}{c^2} \\ 0 & \dfrac{\rho}{2c} & \dfrac{1}{2c^2} \\ 0 & -\dfrac{\rho}{2c} & \dfrac{1}{2c^2} \end{bmatrix} \tag{4.59}$$

We go back to our original discussion. Now that we have the eigenvalues $\lambda_1 = u$, $\lambda_2 = u + c$, and $\lambda_3 = u - c$ we can rewrite equation (4.49) as

$$\frac{\partial \hat{q}_1}{\partial t} + u \frac{\partial \hat{q}_1}{\partial x} = 0 \tag{4.60}$$

$$\frac{\partial \hat{q}_2}{\partial t} + (u + c) \frac{\partial \hat{q}_2}{\partial x} = 0 \tag{4.61}$$

$$\frac{\partial \hat{q}_3}{\partial t} + (u - c) \frac{\partial \hat{q}_3}{\partial x} = 0 \tag{4.62}$$

We can relate $d\hat{Q}$ to $d\tilde{Q}$ as

$$\begin{bmatrix} d\hat{q}_1 \\ d\hat{q}_2 \\ d\hat{q}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\dfrac{1}{c^2} \\ 0 & \dfrac{\rho}{2c} & \dfrac{1}{2c^2} \\ 0 & -\dfrac{\rho}{2c} & \dfrac{1}{2c^2} \end{bmatrix} \begin{bmatrix} d\rho \\ du \\ dp \end{bmatrix} \tag{4.63}$$

Which gives us

$$d\hat{q}_1 = d\rho - \frac{dp}{c^2} \tag{4.64}$$

$$d\hat{q}_2 = \frac{\rho}{2c}du + \frac{dp}{2c^2} \tag{4.65}$$

and

$$d\hat{q}_3 = -\frac{\rho}{2c}du + \frac{dp}{2c^2} \tag{4.66}$$

Consider a point $(x_0, t_0)$. The equation of the first characteristic through this point is $x - x_0 = u(t - t_0)$. Without loss of generality, we can assume $x_0 = 0$ and $t_0 = 0$ for this discussion. Along $x = ut$, $d\hat{q}_1 = 0$ since $\hat{q}_1$ is a constant. In this fashion equations (4.64), (4.65), and (4.66) can be integrated along $x = ut$, $x = (u + c)t$, and $x = (u - c)t$ respectively.

If we just consider a point in our one-dimensional flow where the flow is from our left to right, for which we take $u$ to be positive and further if we assume the flow is supersonic, we have three equations that look very similar to the wave equation of the earlier section. We have three different characteristics. One for each equation. In the supersonic case they look as follows

Clearly the characteristics are unlikely to be identical. $\hat{q}_1$, $\hat{q}_2$, and $\hat{q}_3$ are propagated, as we know from the equations at three different speeds. In supersonic flow from the left to the right, they are propagated from left to right. What of subsonic flow? $u - c$ will be negative. Indeed, the characteristic corresponding to the $u - c$ eigenvalue, in our case $\lambda_3$, has a negative slope.

We've managed to make the system of equations that govern the one-dimensional Euler's equation look like a system

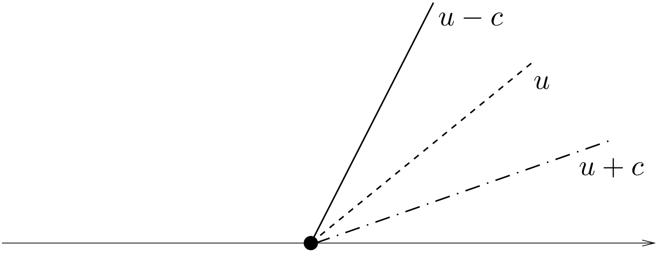

**Figure 4.4. Characteristic line segments at a point in one-dimensional supersonic flow**

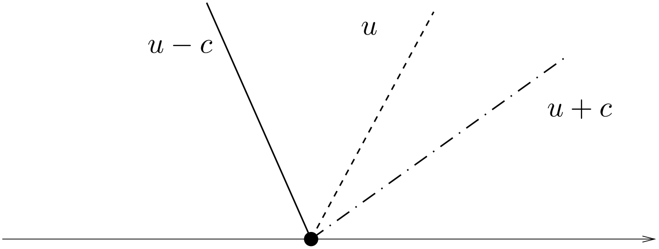

**Figure 4.5. Characteristic line segments at a point in one-dimensional subsonic flow**

made up of "wave equations". The equations are said to be in characteristic form. We will use the existence of these characteristics to analyse any numerical scheme.

At this point we pause and make sure that the following is clear. $\lambda_1 = u$, $\lambda_2 = u + c$, $\lambda_3 = u - c$ are the characteristics or eigenvalues of $\boldsymbol{A}$. Corresponding to these eigenvalues we have three eigenvectors stacked as columns in the matrix $\boldsymbol{X}$ and satisfying $\boldsymbol{AX} = \boldsymbol{X\Lambda}$. Finally, we have our dependent vector projected onto this coordinate system. These are governed by the differential equation (thanks to chain rule) as $d\hat{\boldsymbol{Q}} = \boldsymbol{X}^{-1} d\boldsymbol{Q}$. If curl$\boldsymbol{X}^{-1} = 0$, then we can integrate the differential equation and actually find a $\hat{\boldsymbol{Q}}$. However, for now, it turns out that we need only know that the differentials exist and that the differential equation can be transformed at each $\boldsymbol{Q}$.

---

**Assignment 4.4**

(1) Write a function GetC to find the speed of sound given $Q$.

(2) Rewrite the equation (4.4) in terms of the dependant variable

$$\vec{Q}' = \begin{bmatrix} \rho \\ u \\ T \end{bmatrix} \tag{4.67}$$

and show that it becomes

$$\frac{\partial}{\partial t}\begin{bmatrix} \rho \\ u \\ T \end{bmatrix} + \begin{bmatrix} u & \rho & 0 \\ \frac{RT}{\rho} & u & R \\ 0 & (\gamma-1)T & u \end{bmatrix} \frac{\partial}{\partial x}\begin{bmatrix} \rho \\ u \\ T \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.68}$$

(3) Find the eigenvalues of the for the flux Jacobian in this case. Show that they are $u$, $u + c$, $u - c$.

(4) Find the eigenvectors.

---

## A Numerical Scheme

We will go through the same process that we used for the first order linear wave equation in section 3.11. We will start by applying the Euler explicit scheme or FTCS discretisation as we have called it earlier, to the one-dimensional Euler equations used to solve the flow through the pipe given in Figure 4.3. You will notice that right now, we will only look at the discretisation of the governing equations and that the actual problem to be solved will show up in the discussion only when we want to figure out the boundary conditions that need to be applied[**RS81**].

This flow field is modelled using the one-dimensional Euler equations. FTCS applied to these equations gives us

$$\vec{Q}_p^{q+1} = \vec{Q}_p^q - \frac{1}{2}\frac{\Delta t}{\Delta x}\left\{\vec{E}_{p+1}^q - \vec{E}_{p-1}^q\right\} \tag{4.69}$$

we can employ this automaton to obtain $\vec{Q}_p^{q+1}$ given $\vec{Q}_p^q$. Bear in mind that this scheme was unconditionally unstable for the first order linear wave equation.

---

### Assignment 4.5

Write a function called FTCS that takes an array of $Q$ and a parameter which can be named DtByDx, which represents the ratio $\Delta t/\Delta x$, and takes one time step for all the interior points (that is leaving out the first and the last grid points) with the FTCS scheme as given by equation (4.69).

---

**Stability Analysis.** We now go through the process of checking out the stability analysis for FTCS applied to the one-dimensional Euler equations. Yes, we already expect that the scheme is unconditionally unstable, after all we were able to transform to the characteristic form and show that the Euler equation are the same as three one-dimensional first order linear wave equations. However, just so that the process is clear and the underlying assumptions required are transparent, we will go ahead and do the stability analysis.

We see that the FTCS scheme shown in equation (4.69) can be conveniently written as

$$\vec{Q}_p^{q+1} = \vec{Q}_p^q - \frac{1}{2}\frac{\Delta t}{\Delta x}\vec{E}_p^q\left\{e^{in\Delta x} - e^{-in\Delta x}\right\} \tag{4.70}$$

Using the result that you showed in the assignment we can write $E = AQ$

$$\vec{Q}_p^{q+1} = \vec{Q}_p^q - \frac{\Delta t}{\Delta x}\boldsymbol{A}_p^q\vec{Q}_p^q\left\{i\sin n\Delta x\right\} \tag{4.71}$$

Keep in mind that we are dealing with matrices here. We factor out operator acting on $\vec{Q}_p^q$ to the left to get

$$\vec{Q}_p^{q+1} = \left\{\boldsymbol{I} - i\frac{\Delta t}{\Delta x}\boldsymbol{A}_p^q\sin n\Delta x\right\}\vec{Q}_p^q \tag{4.72}$$

The matrices $\boldsymbol{X}$ and $\boldsymbol{X}^{-1}$ can be made normal, meaning they perform a rotation but no stretch. We define $\vec{S}_p^q = \boldsymbol{X}^{-1}\vec{Q}_p^q$. Then $\boldsymbol{X}^{-1}$ is normalised means $\|S_p^q\| = \|\boldsymbol{X}^{-1}\|\|\vec{Q}_p^q\| = \|\vec{Q}_p^q\|$.

The eigenvectors are also independent of each other. We premultiply equation (4.73) by $\boldsymbol{X}^{-1}$.

$$\textbf{(4.73)} \quad \vec{S}_p^{q+1} = \boldsymbol{X}^{-1}\left\{\boldsymbol{I} - i\frac{\Delta t}{\Delta x}\boldsymbol{A}_p^q \sin n\Delta x\right\}\vec{Q}_p^q$$

$$= \left\{\boldsymbol{I} - i\frac{\Delta t}{\Delta x}\boldsymbol{\Lambda}_p^q \sin n\Delta x\right\}\vec{S}_p^q$$

Please note, $\vec{S}$ are not characteristic variables. Taking the norm of both sides we get

$$\textbf{(4.74)} \quad \|\vec{S}_p^{q+1}\| = \|\vec{Q}_p^{q+1}\| = \left\|\boldsymbol{I} - i\frac{\Delta t}{\Delta x}\boldsymbol{\Lambda}_p^q \sin n\Delta x\right\| \|\vec{Q}_p^q\|$$

So, the relationship given by equation (4.73) and consequently the one given by equation (4.72) is a contraction mapping if $\|\vec{S}_p^{q+1}\| < \|\vec{S}_p^q\|$, that is

$$\textbf{(4.75)} \quad \left\|\boldsymbol{I} - i\frac{\Delta t}{\Delta x}\boldsymbol{\Lambda}_p^q \sin n\Delta x\right\| < 1$$

Note that $\boldsymbol{I} - i\dfrac{\Delta t}{\Delta x}\boldsymbol{\Lambda}_p^q \sin n\Delta x$ is a diagonal matrix. We just require the magnitude of the largest entry on the diagonal to be less than one. We see again that FTCS is unconditionally unstable. We would need to add artificial dissipation if we want it to work.

## Boundary Conditions

As we did in the wave equation, we will inspect the governing equations to discover how many boundary conditions are required. We see that the vector equation has one time derivative and one spatial derivative. We expect to prescribe one vector initial condition or three conditions at $t = 0$ and one vector boundary condition or actually, three boundary conditions.

This much we get from our understanding of differential equations. Let us look at it from the point of view of the physics of the problem as described at the beginning of the chapter (see Figure 4.3). The pipe is open to the atmosphere on the right hand side. When the valve is closed, the conditions in the pipe will be the ambient conditions. At the instant when the valve is opened, loosely, we have two conditions on the left in the form of $P_0$ and $T_0$. $P_0$ is the total pressure in the reservoir and $T_0$ is the total temperature in the reservoir. These are often called reservoir conditions or upstream conditions since we expect the flow from that direction. We have two conditions, we need one more. In gas dynamics this would be the back pressure or the ambient pressure $p_a$ on the right hand side of the pipe. If $p_a = P_0$, then there is no flow. If $p_a < P_0$ we will have flow. Clearly, $P_0$ and $p_a$ are boundary conditions that influence the flow. $T_0$ is a constraint on the magnitude of the flow in the form of the total energy available. The one-dimensional, steady state energy equation reduces to

$$C_p T_0 = C_p T + \frac{u^2}{2} \tag{4.76}$$

where $T$ is the static temperature measured on the Kelvin scale. This relates the total temperature at the inlet to the unknown static temperature and speed at the inlet.

Employing these boundary conditions, one can solve the one-dimensional gas dynamical problem[**Sha53**]. At this point in the wave equation, we discovered that the numerical scheme required more parameters to be specified than required by the physical problem or the differential equation. How do the boundary conditions that we have prescribed so far work with the numerical scheme? Let us see what happens when we use FTCS.

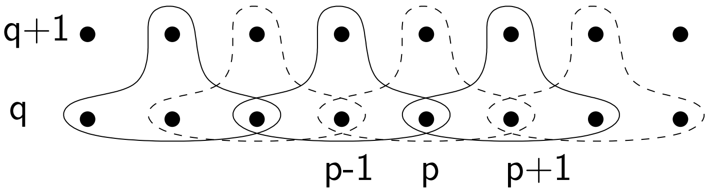

**Figure 4.6. Taking one time step with FTCS. The grid points involved in the approximating the governing equation at each of the interior grid points are shown by the bubble surrounding the grid points.**

From Figure 4.6 we can see that the $Q$ at points interior to our domain at time level $q+1$ can be found from $Q$ at all the points at time level $q$. Now, if we wish to proceed to the next time level $q+2$, we require $Q$ at all the points at time level $q+1$. From FTCS we have calculated $Q$ only at the interior points. How do we get the values of $Q$ at the boundaries? We will use the characteristics to answer this question.

For the subsonic inlet case we clearly have at the entry point characteristics as shown in Figures 4.5 and 4.7. We have something similar at the exit.

Remember, the characteristics tell us the direction in which the characteristic variables propagated. We can make the statement vague by saying that along the characteristic "information" regarding the flow is propagated. At the first grid point

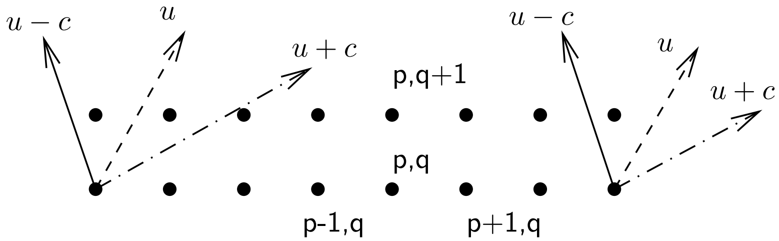

**Figure 4.7. Characteristics at a subsonic inlet and a subsonic exit at time level $q$. Grid points at time level $q+1$ are also shown.**

this indicates that two pieces of information are come from outside the domain into the domain. That is fine. In gas dynamics we would prescribe $P_0$ and $T_0$. The surprising thing is that there is one piece of information that goes from the computational domain to the boundary. We could extrapolate $u$ from within the computational domain to the boundary. So, how do we get $Q$ at the inlet? From the extrapolated $u$ and the prescribed $T_0$ we get $T$.

$$C_pT_0 = C_pT + \frac{u^2}{2} \Rightarrow T = T_0 - \frac{u^2}{2C_p} \tag{4.77}$$

From $T$, $T_0$ and $P_0$ we get $P$

$$\left\{\frac{P}{P_0}\right\} = \left\{\frac{T}{T_0}\right\}^{\displaystyle\frac{\gamma}{\gamma-1}} \Rightarrow P = P_0\left\{\frac{T}{T_0}\right\}^{\displaystyle\frac{\gamma}{\gamma-1}} \tag{4.78}$$

Then we employ equation of state to get the density, which incidentally is $q_1 = \rho = P/RT$. Then $q_2 = \rho u$. We obtain $q_3$ from equation (4.17).

Let's look at the exit condition now. Here, we see that we have one "incoming" characteristic. That is also fine. In gas dynamics one would prescribe the static pressure or the back-pressure. However, we have two "outgoing" characteristics. If we again extrapolate two quantities say, $u$ and $T_0$. We have found some way by which we can obtain all the $Q$s at the time level $q+1$ and hence can proceed to $q+2$.

I would suggest you try extrapolating different parameters and check out the behaviour of your code. At this point, no doubt, you are asking code? What code? I thought FTCS was unconditionally unstable. Right, so we add a tiny pinch of dissipation to make it work, if you are wondering why we are doing this you need to go back and review the material on the modified equation of the wave equation and dissipation in section 3.13.

So, we have done a lot of bad things. We seem to have engineered boundary conditions by extrapolation, where none existed. We have taken the equation that we are solving and deliberately modified it, just to get it working.

---

**Assignment 4.6**

(1) Write a one-dimensional Euler equation solver using FTCS. Remember, that you will need to add some artificial dissipation terms. While developing the code add something along the lines

$$\mu_2 \Delta x^2 \frac{\partial^2 Q}{\partial x^2} - \mu_4 \Delta x^4 \frac{\partial^4 Q}{\partial x^4} \tag{4.79}$$

with $\mu_2 = 0.01$ and $\mu_4 = 0.001$. While developing the solver make sure that use the following conditions as the test case.

(a) Inlet condition

**(4.80)**
$$P_o = 101325 \,\text{Pascals}, \quad T_o = 300 \,\text{Kelvin}$$

(b) Exit condition

**(4.81)**
$$P_a = 84000 \,\text{Pascals}$$

(c) Set the initial conditions to be the exit condition. You can take $T_a = 300K$.

(2) Run the above conditions for various grid sizes. Choose the time-step so that

**(4.82)**
$$\frac{\Delta t}{\Delta x} = 0.0001, 0.0005, 0.001, 0.002$$

(3) Run your code for $P_o = 121590 \,\text{Pascals}$ and $P_a = 101325 \,\text{Pascals}$

(4) Repeat the above process for higher values of $P_o$.

(5) Repeat the first problem for different values of $T_o$

---

## The Delta Form

We write the equation in what is known as the "Delta form". We start with the Flux vector $\vec{E}$. Using Taylor's series and truncating with two terms we can write

**(4.83)**
$$\vec{E}^{q+1} = \vec{E}^q + \Delta t \left[ \frac{\partial \vec{E}}{\partial t} \right]^q$$

Using chain rule we get

$$\vec{E}^{q+1} = \vec{E}^q + \Delta t \left[ \frac{\partial \vec{E}}{\partial \vec{Q}} \frac{\partial \vec{Q}}{\partial t} \right]^q = \vec{E}^q + \left[ \mathbf{A} \Delta \vec{Q} \right]^q \tag{4.84}$$

where, $\Delta Q^q = Q^{q+1} - Q^q$. BTCS applied to these equations gives us

$$\left. \frac{\partial Q}{\partial t} \right|^{q+1} + \frac{\partial}{\partial x} E^{q+1} = \left. \frac{\partial Q}{\partial t} \right|^{q+1} + \frac{\partial}{\partial x} \left( \vec{E}^q + \mathbf{A} \Delta \vec{Q} \right|^q \Big) = 0 \tag{4.85}$$

If we are interested only in the steady state solution, then that solution satisfies the equation

$$\frac{\partial E}{\partial x} = 0 \tag{4.86}$$

In equation (4.85) we can rearrange terms so that

$$\left. \frac{\partial Q}{\partial t} \right|^{q+1} + \frac{\partial}{\partial x} \left( \mathbf{A} \Delta \vec{Q} \right|^q \Big) = - \frac{\partial \vec{E}^q}{\partial x} \tag{4.87}$$

Since, we are looking at a backward time scheme

$$\frac{\Delta Q}{\Delta t}^q + \frac{\partial}{\partial x} \left( \mathbf{A} \Delta \vec{Q} \right|^q \Big) = - \frac{\partial \vec{E}^q}{\partial x} \Rightarrow \\ \left[ \mathbf{I} + \Delta t \frac{\partial \mathbf{A}}{\partial x} \right] \Delta \vec{Q} = - \Delta t \frac{\partial \vec{E}^q}{\partial x} \tag{4.88}$$

Here is a new way to build a whole class of automatons to solve our equation.

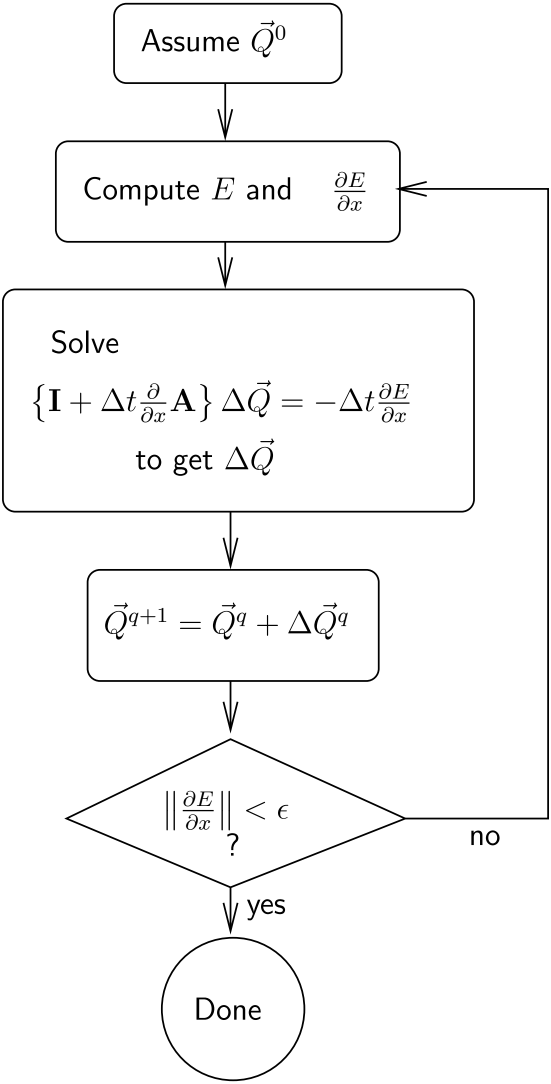
Assume $\vec{Q}^0$
Compute $E$ and $\frac{\partial E}{\partial x}$
Solve
$\left\{\mathbf{I} + \Delta t \frac{\partial}{\partial x}\mathbf{A}\right\} \Delta\vec{Q} = -\Delta t \frac{\partial E}{\partial x}$
to get $\Delta\vec{Q}$
$\vec{Q}^{q+1} = \vec{Q}^q + \Delta\vec{Q}^q$
$\left\|\frac{\partial E}{\partial x}\right\| < \epsilon$
?
no
yes
Done

**Figure 4.8. Algorithm for solving one-dimensional Euler equation**

We assume an initial $\vec{Q}$. Then use this to compute the right hand side of equation (4.88). This should be zero if we had the steady state solution. Unfortunately, we are left, typically, with a residual. This allows us to determine a correction $\Delta\vec{Q}$ using equation (4.89).

$$\left\{\mathbf{I} + \Delta t \frac{\partial \mathbf{A}}{\partial x}\right\} \Delta\vec{Q}^q = -\Delta t \frac{\partial \vec{E}^q}{\partial x} \tag{4.89}$$

We can then obtain a corrected $\vec{Q}$ using the $\Delta\vec{Q}$ that we just got using

$$\vec{Q}^{q+1} = \vec{Q}^q + \Delta\vec{Q}^q \tag{4.90}$$

This process can be repeated till $\|\Delta\vec{Q}\| < \epsilon$ or better still, till $\|\partial E/\partial x\| < \epsilon$.

We should make a few observations here. How close we are to the solution is determined by the residual on the right hand side(RHS). Anytime we stop marching in time, the quality of the approximation is clearly determined by the residual. This residual should be computed with the desired accuracy. This will then allow us to employ the left hand side (LHS) of our equation to generate a sequence of $\Delta\vec{Q}$s. The RHS is really the predicate that we use for convergence. The LHS determines how fast we get there. So, the CFD of the physics that we are trying to pick up is in the RHS. The CFD of how we are going to get there is in LHS. On the LHS we need to come up with something simple to compute which causes large decreases in the magnitude of the residue in every step.

Let us now see how we would deal with this for BTCS. The $x$-derivative in equation (4.89) is approximated using central differences. Then the full system of equations reads as shown in equation (4.91).

**(4.91)**

$$
\begin{bmatrix}
\mathbf{I} & \boldsymbol{\lambda}_2^q & 0 & 0 & \cdots & \cdots & \cdots & 0 \\
-\boldsymbol{\lambda}_1^q & \mathbf{I} & \boldsymbol{\lambda}_3^q & 0 & \cdots & \cdots & \cdots & 0 \\
0 & -\boldsymbol{\lambda}_2^q & \mathbf{I} & \boldsymbol{\lambda}_4^q & 0 & \cdots & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & -\boldsymbol{\lambda}_{p-1}^q & \mathbf{I} & \boldsymbol{\lambda}_{p+1} & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \cdots & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & \cdots & \cdots & \cdots & -\boldsymbol{\lambda}_{N-1}^q & \mathbf{I}
\end{bmatrix}
\begin{pmatrix}
\Delta\vec{Q}_1 \\ \Delta\vec{Q}_2 \\ \Delta\vec{Q}_3 \\ \vdots \\ \Delta\vec{Q}_{p-1} \\ \Delta\vec{Q}_p \\ \Delta\vec{Q}_{p+1} \\ \vdots \\ \Delta\vec{Q}_{N-1} \\ \Delta\vec{Q}_N
\end{pmatrix}
=
\begin{pmatrix}
(\mathsf{R}+\mathsf{BC})_1 \\ \mathsf{R}_2 \\ \mathsf{R}_3 \\ \vdots \\ \mathsf{R}_{p-1} \\ \mathsf{R}_p \\ \mathsf{R}_{p+1} \\ \vdots \\ \mathsf{R}_{N-1} \\ (\mathsf{R}+\mathsf{BC})_N
\end{pmatrix}
$$

where $\boldsymbol{\lambda}_p^q = \Delta t/(2\Delta x)\mathbf{A}_p^q$.

Please note that each of the entries in our $N \times N$ matrix is itself a $3\times3$ matrix. The corresponding entries in the two vectors are also vectors. These are called block structured matrices. It should be pointed out that in CFD we **NEVER** assemble this matrix as shown.

One could use a direct method like Gaussian elimination to solve the system of equations. Or an iterative techniques of

Gauss-Seidel or Gauss-Jordan. Another possibility is that we perform an approximate factorisation. How does this work? remember that we want a solution to the steady state equation. We will take the equation in operator form and factor the derivative in terms of forward differences and backward differences.

**(4.92)**
$$\left\{\mathbf{I} + \Delta t \frac{\partial^- \mathbf{A}}{\partial x}\right\}\left\{\mathbf{I} + \Delta t \frac{\partial^+ \mathbf{A}}{\partial x}\right\}\Delta \vec{Q}^q = -\Delta t \frac{\partial \vec{E}^q}{\partial x}$$

This in fact can be written as

**(4.93)**
$$\left\{\mathbf{I} + \Delta t \frac{\partial^- \mathbf{A}}{\partial x}\right\}\Delta \vec{Q}'^q = -\Delta t \frac{\partial \vec{E}^q}{\partial x}$$

**(4.94)**
$$\left\{\mathbf{I} + \Delta t \frac{\partial^+ \mathbf{A}}{\partial x}\right\}\Delta \vec{Q}^{\,q} = \Delta \vec{Q}'^q$$

If we were to expand the product in equation (4.92) we get

**(4.95)**
$$\left\{\mathbf{I} + \Delta t \frac{\partial^- \mathbf{A}}{\partial x} + \Delta t \frac{\partial^+ \mathbf{A}}{\partial x} + \Delta t^2 \frac{\partial^- \mathbf{A}}{\partial x}\frac{\partial^+ \mathbf{A}}{\partial x}\right\}\Delta \vec{Q}^q = -\Delta t \frac{\partial \vec{E}^q}{\partial x}$$

We see this factorisation when expanded gives us an extra term. How do we justify its use? Two ways to go about it.

(1) The time derivative was discretised using backward differences. The truncation error is first order. The extra term has a factor which is $\Delta t^2$. Therefore, it is subsumed by the truncation error. Note that the derivative multiplying it is still like a second derivative term.

(2) Anyway, we are seeking a steady state solution. How does it matter what the factor for $\Delta\vec{Q}$ is as long as it drives it to zero employing the residual.

The second reason is the more potent of the two. It gives us scope to explore other ways by which we can generate a sequence of $\Delta\vec{Q}$s to drive the residual to zero.

Also, please note that the FTCS scheme can be written in Delta form as

$$\{\mathbf{I}\}\,\Delta\vec{Q}^q = -\Delta t\frac{\partial\vec{E}^q}{\partial x} \tag{4.96}$$

---

**Assignment 4.7**

(1) Re-do the previous assignment 4.6, using BTCS and the delta form.
(2) Try a range of CFL values from 0.001 to 100. or more.

---

## Boundary Conditions Revisited

Well, let us at least try to clean the boundary conditions a little. We will do this in two slightly different ways. First using the delta form let us try to introduce a little rigour into the application of boundary conditions.

The plan is a standard game that we play in all of science and technology. We will transform the problem from one form {our normal coordinates} to a more convenient form {the characteristic "coordinates": $\boldsymbol{x}_u$, $\boldsymbol{x}_{u+c}$, and $\boldsymbol{x}_{u-c}$}. We then perform whatever operation is easy to perform in characteristic form and then transform back. For your reference the equation in delta form is

$$\left\{ \mathbf{I} + \Delta t \frac{\partial \mathbf{A}}{\partial x} \right\} \Delta \vec{Q}^q = -\Delta t \frac{\partial \vec{E}^q}{\partial x} \tag{4.97}$$

We will assume that $\mathbf{A}$ is constant. If we were to premultiply this equation by $(\mathbf{X}^{-1})^q_p$ we get

$$\begin{aligned} (\mathbf{X}^{-1})^q_p \left\{ \mathbf{I} + \Delta t \frac{\partial \mathbf{A}}{\partial x} \right\} \Delta \vec{Q}^q = \\ \left\{ (\mathbf{X}^{-1})^q_p + \Delta t (\mathbf{X}^{-1})^q_p \mathbf{A} \mathbf{X}^q_p (\mathbf{X}^{-1})^q_p \frac{\partial}{\partial x} \right\} \Delta \vec{Q}^q = \\ -\Delta t (\mathbf{X}^{-1})^q_p \frac{\partial \vec{E}^q}{\partial x} \end{aligned} \tag{4.98}$$

At the first grid point this becomes

$$\left\{ \mathbf{I} + \Delta t \mathbf{\Lambda}^q_1 \frac{\partial}{\partial x} \right\} \Delta \hat{Q}^q = -\Delta t (\mathbf{X}^{-1})^q_1 \left. \frac{\partial \vec{E}}{\partial x} \right|^q_1 \tag{4.99}$$

At the inlet, we can write the boundary conditions from gas dynamics as we did before. Referring to figure 4.7, for a subsonic inlet, we have two characteristics along which the characteristic variables $\hat{q}_1$ and $\hat{q}_2$ are propagated into the computational domain. Corresponding to which we prescribed two conditions: $P_o$ and $T_o$. We will need to take a closer look at this in a moment. $\hat{q}_3$ is being carried out of the domain and governed by the third component in equation (4.99). To extract the third component corresponding to the left running characteristic the matrix $\mathbb{L}$ is defined as follows

$$\mathbb{L} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.100}$$

Corresponding to the out-going characteristics at the inlet we can now use the equation

$$\mathbb{L}\left\{\mathbf{I} + \Delta t \mathbf{\Lambda}_1^q \frac{\partial}{\partial x}\right\}\Delta \hat{Q}^q = -\mathbb{L}\Delta t(\mathbf{X}^{-1})_1^q \left.\frac{\partial \vec{E}}{\partial x}\right|_1^q \tag{4.101}$$

Where do the two prescribed boundary conditions $P_o$ and $T_o$ figure? Actually, we need to transform them to the characteristic coordinates ($\boldsymbol{x}_u$, $\boldsymbol{x}_{u+c}$, and $\boldsymbol{x}_{u-c}$)and pick only those parts of them that correspond to the process of propagation from the outside of the computational domain into the domain (components along $\boldsymbol{x}_u$ and $\boldsymbol{x}_{u+c}$). We do this as follows. Define the vector $\mathbb{B}_o$ as

$$\mathbb{B}_o = \begin{bmatrix} P_o \\ T_o \\ 0 \end{bmatrix} \tag{4.102}$$

As we did when we derived the delta form, we can express $\mathbb{B}_o^{q+1}$ using Taylor's series and truncating at the linear term as

**(4.103)**
$$\mathbb{B}_o^{q+1} = \mathbb{B}_o^q + \Delta t \left.\frac{\partial \mathbb{B}_o}{\partial t}\right|^q \Rightarrow \mathbb{B}_o^{q+1} = \mathbb{B}_o^q + D_o \Delta Q$$

or

**(4.104)**
$$\Delta \mathbb{B}_o = D_o \Delta Q$$

where

**(4.105)**
$$D_o = \frac{\partial \mathbb{B}_o}{\partial Q}$$

Now, if $P_o$ and $T_o$ do not vary in time, then equation (4.103) becomes

**(4.106)**
$$D_o \Delta Q = 0$$

This can now be transformed into the characteristic coordinates by pre-multiplying by $(\mathbf{X}^{-1})_1^q$ to get

**(4.107)**
$$\mathbf{X}^{-1} D_o \Delta Q = \mathbf{X}^{-1} D_o \mathbf{X}\mathbf{X}^{-1} \Delta Q = \hat{D}_o \Delta \hat{Q} = 0$$

The subscripts indicating spatial and temporal position have been dropped. Now we use $(\mathbf{I} - \mathbb{L})$ to extract the first two components and eliminate the $\hat{q}_3$ part($\boldsymbol{x}_{u-c}$ component) of equation (4.107).

**(4.108)**
$$(\mathbf{I} - \mathbb{L})\mathbf{X}^{-1} D_o \Delta Q = (\mathbf{I} - \mathbb{L})\hat{D}_o \Delta \hat{Q} = 0$$

We combine this at the boundary with the equation governing $\hat{q}_3$ to get

**(4.109)**
$$(\mathbf{I}-\mathbb{L})\hat{D}_o \Delta \hat{Q}^q + \mathbb{L}\left\{\mathbf{I} + \Delta t \mathbf{\Lambda}_1^q \frac{\partial}{\partial x}\right\} \Delta \hat{Q}^q = -\mathbb{L}\Delta t (\mathbf{X}^{-1})_1^q \left.\frac{\partial \vec{E}}{\partial x}\right|_1^q$$

This combined equation determines the full vector $\hat{Q}_1^q$. We can pre-multiply the equation by $\mathbf{X}_1^q$ to get back to the $Q$ system.

**(4.110)**
$$\mathbf{X}_1^q \left[ (\mathbf{I} - \mathbb{L})\hat{D}_o \Delta \hat{Q}^q + \mathbb{L} \left\{ \mathbf{I} + \Delta t \mathbf{\Lambda}_1^q \frac{\partial}{\partial x} \right\} \Delta \hat{Q}^q = -\mathbb{L} \Delta t (\mathbf{X}^{-1})_1^q \left. \frac{\partial \vec{E}}{\partial x} \right|_1^q \right]$$

At the exit, we have two characteristics propagating $\hat{q}_1$ and $\hat{q}_2$ out of the domain. We have one physical condition $p_a$ prescribed. As was done at the inlet, we can define the vector $\mathbb{B}_a$.

**(4.111)**
$$\mathbb{B}_a = \begin{bmatrix} 0 \\ 0 \\ p_a \end{bmatrix}$$

Repeating the process followed at the inlet we write the condition that $\mathbb{B}_a$ as

**(4.112)**
$$D_a \Delta Q = 0$$

where,

**(4.113)**
$$D_a = \frac{\partial \mathbb{B}_a}{\partial Q}$$

We can then write for the last grid point

**(4.114)**
$$\mathbf{X}_N^q \left[ \mathbb{L}\hat{D}_a \Delta \hat{Q}^q + (\mathbf{I} - \mathbb{L}) \left\{ \mathbf{I} + \Delta t \mathbf{\Lambda}_N^q \frac{\partial}{\partial x} \right\} \Delta \hat{Q}^q = -(\mathbf{I} - \mathbb{L}) \Delta t (\mathbf{X}^{-1})_N^q \left. \frac{\partial \vec{E}}{\partial x} \right|_N^q \right]$$

How do we get $D$? We need to write $T_o$ and $P_o$ in terms of $q_1$, $q_2$, and $q_3$. One dimensional energy equation can be written as

**(4.115)**
$$C_p T_o = C_p T + \frac{u^2}{2} \Rightarrow C_v T_o = C_v T + \frac{u^2}{2\gamma} = \frac{q_3}{q_1} - \frac{\gamma - 1}{2\gamma} \frac{q_2^2}{q_1^2}$$

It looks like the expression is a little simpler if we prescribe $C_v T_o$ instead of $T_o$. Now from the isentropic relationship

**(4.116)**
$$P_o = p \left( \frac{T_o}{T} \right)^{\dfrac{\gamma}{\gamma - 1}}.$$

Using equation (4.115) we get

**(4.117)**
$$P_o = p\left(1 + \frac{u^2}{2\gamma e}\right)^{\dfrac{\gamma}{\gamma - 1}} =$$

$$(\gamma - 1)\left\{q_3 - \frac{q_2^2}{2q_1}\right\}\left(1 + \frac{q_2^2/q_1^2}{2\gamma\left[\dfrac{q_3}{q_1} - \dfrac{q_2^2}{2q_1^2}\right]}\right)^{\dfrac{\gamma}{\gamma - 1}}$$

$p$ was replaced using equation (4.18). The expression can be simplified to

**(4.118)**
$$P_o = (\gamma - 1)\left\{q_3 - \frac{q_2^2}{2q_1}\right\}\left(1 + \frac{q_2^2}{2\gamma\left[q_3 q_1 - \dfrac{q_2^2}{2}\right]}\right)^{\dfrac{\gamma}{\gamma - 1}}$$

The entries of $D_o$ are given in component form as $d_{ij}$. We know that $d_{3j} = 0$. The rest of the entries are

**(4.119)**
$$d_{11} = \frac{\gamma - 1}{\gamma}\frac{u^2}{\rho} - \frac{E_t}{\rho}$$

**(4.120)**
$$d_{12} = -\frac{\gamma - 1}{\gamma}\frac{u}{\rho}$$

**(4.121)**
$$d_{13} = \frac{1}{\rho}$$

**(4.122)**
$$d_{21} = \rho\rho_o u^2 \left[ (\gamma - 1) \left( \frac{T_o}{T} \right)^\gamma - \frac{E_t}{2e} \right]$$

**(4.123)**
$$d_{22} = \rho_o u \left[ \frac{\rho E_t}{e} - C \right]$$

**(4.124)**
$$d_{23} = \rho_o \left[ C - \frac{\rho u^2}{2e} \right]$$

where $C = \dfrac{(\gamma - 1)}{\rho} \dfrac{T_o}{T}$. So, $D_o$ is given by (Please note the matrix indicated is $D_o^T$ which is the transpose of $D_o$)

**(4.125)**
$$D_o^T = \begin{bmatrix} \frac{\gamma - 1}{\gamma} \frac{u^2}{\rho} - \frac{E_t}{\rho} & \rho\rho_o u^2 \left[ (\gamma - 1) \left( \frac{T_o}{T} \right)^\gamma - \frac{E_t}{2e} \right] & 0 \\ -\frac{\gamma - 1}{\gamma} \frac{u}{\rho} & \rho_o u \left[ \frac{\rho E_t}{e} - C \right] & 0 \\ \frac{1}{\rho} & \rho_o \left[ C - \frac{\rho u^2}{2e} \right] & 0 \end{bmatrix}$$

---

**Assignment 4.8**

Redo assignment 4.7 with the new improved boundary conditions.

---

**Boundary Conditions - a compromise.** So you think all the stuff that we just went through was a bit much. You like the extrapolation thing we did earlier as it was easy to do. If we prescribe the quantities $P_o$ and $T_o$ at the inlet, how do we compute the new $Q^{q+1}$ at the inlet? Well that is easy. At any given time step $q$, we have $Q^q$ at the inlet. From which we can compute

**(4.126)**
$$\xi_0^+ = u_0 + \frac{2c_0}{\gamma - 1}$$

and

**(4.127)**
$$\xi_0^- = u_0 - \frac{2c_0}{\gamma - 1}$$

We switched to $\xi$ as the earlier notation looks messy. We have defined $\xi^+ = \hat{q}_2$ and $\xi^- = \hat{q}_3$. The subscript "0" indicates it is at the inlet. From our previous section we can at least conclude that at a subsonic inlet one could extrapolate the quantity given in equation (4.128) from the first interior grid point to the inlet boundary.

**(4.128)**
$$\xi_1^- = u_1 - \frac{2c_1}{\gamma - 1}$$

where the subscripts indicate the grid point. We could either extrapolate $\xi_1^-$ or use the characteristic equation to extrapolate it as

**(4.129)**
$$\xi^-\Big|_0^{q+1} = \xi^-\Big|_0^q - \frac{(u-c)|_0^q\,\Delta t}{\Delta x}\left\{\xi^-\Big|_1^q - \xi^-\Big|_0^q\right\}$$

Along with the extrapolated quantity $\xi^-$ we now have the pair of characteristic variables at the inlet. We observe that this pair is from the new state and now label them $(\xi^-|^{q+1}, \xi^+|^{q+1})$. we can then compute

**(4.130)**
$$u^{q+1} = \frac{\xi^-|^{q+1} + \xi^+|^{q+1}}{2}$$

We will drop the superscript for now since we know that we are computing quantities at the new time level. The resulting expressions don't look as busy. At the inlet we have the relation between the total temperature and the static temperature as

**(4.131)**
$$C_p T_o = C_p T + \frac{u^2}{2}$$

Using equation (4.130) and equation (4.131) we can find $T^{q+1}$. From which we can find the static pressure as

**(4.132)**
$$p = P_0 \left( \frac{T}{T_0} \right)^{\frac{\gamma}{\gamma - 1}}$$

Given the static pressure and temperature we can find the density using the equation of state. We have $\tilde{Q}$ we can obtain $Q$.

What do we do at the subsonic exit? In this case, again, we prescribe the boundary condition $p_a$ and extrapolate the first two characteristic variables from the penultimate grid point to the last grid point.

---

**Assignment 4.9**

Try out the new boundary condition and see if it makes a difference.

---

Now all this talk of applying the correct boundary condition by getting the right flux at the boundary has to remind you of our earlier discussion on the finite volume method - see section 3.16. It clear again that if we breakup our problem domain into small control volumes and computed the correct fluxes at the boundaries we can determine the rate of change of our dependent variables in the interior of these small control volumes which are very often called finite volumes

**(4.133)**
$$\left(\frac{\partial Q}{\partial t}+\frac{\partial E}{\partial x}\right)_p^q=\left(\frac{\partial Q}{\partial t}+\mathbf{A}\frac{\partial Q}{\partial x}\right)_p^q=0$$

We can pre-multiply it by $(\mathbf{X}^{-1})_p^q$

**(4.134)**
$$(\mathbf{X}^{-1})_p^q\left.\frac{\partial Q}{\partial t}\right|_p^q+\mathbf{\Lambda}_p^q(\mathbf{X}^{-1})_p^q\left.\frac{\partial Q}{\partial x}\right|_p^q=\left.\frac{\partial \hat{Q}}{\partial t}\right|_p^q+\mathbf{\Lambda}_p^q\left.\frac{\partial \hat{Q}}{\partial x}\right|_p^q=0$$

For the sake of simplicity we assume $u \geq 0$ and define

**(4.135)**
$$\mathbf{\Lambda}^+=\begin{bmatrix}u & 0 & 0\\ 0 & u+c & 0\\ 0 & 0 & 0\end{bmatrix};\quad \mathbf{\Lambda}^-=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & u-c\end{bmatrix}$$

Clearly if $u$ were negative the definitions would be interchanged. The objective is to split the eigenvalues into the right running and left running ones. This objective is reached in an automatic fashion using two possible operations.

$$\Lambda^+ = \frac{\Lambda + |\Lambda|}{2}, \quad \text{or} \quad \lambda_i^+ = \max(0, \lambda_i) \tag{4.136}$$

$$\Lambda^- = \frac{\Lambda - |\Lambda|}{2}, \quad \text{or} \quad \lambda_i^- = \min(0, \lambda_i) \tag{4.137}$$

We can rewrite our governing equations as

$$\frac{\partial \hat{Q}}{\partial t} + \Lambda^+\Big|_p^q \frac{\partial \hat{Q}}{\partial x}^q_p + \Lambda^-\Big|_p^q \frac{\partial \hat{Q}}{\partial x}^q_p = 0 \tag{4.138}$$

Now we pre-multiply again by $(\mathbf{X})_p^q$ to get

$$\frac{\partial Q}{\partial t} + \mathbf{A}^+ \frac{\partial Q}{\partial x} + \mathbf{A}^- \frac{\partial Q}{\partial x} = \frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x}^+ + \frac{\partial E}{\partial x}^- = 0 \tag{4.139}$$

## Running the Code

Fragments of a one-dimensional code are provided in the segment on how these codes are written. We first look at the residue. In Figure 4.9, we have a plot of the individual residues of the equations governing conservation of mass, momentum and energy. It is also interesting to plot the residues versus each other to see if we can glean any information from them. Figure 4.10 shows the plot of the residue for the conservation of linear momentum (CLM) versus the residue for the conservation of mass equation (CM). This shows an initial smaller change in CM and then an exchange of the variation between them. Ideally, we would like to get this graph to be a straight line going to the origin. This is how it behaves from $10^{-8}$ to $10^{-15}$.

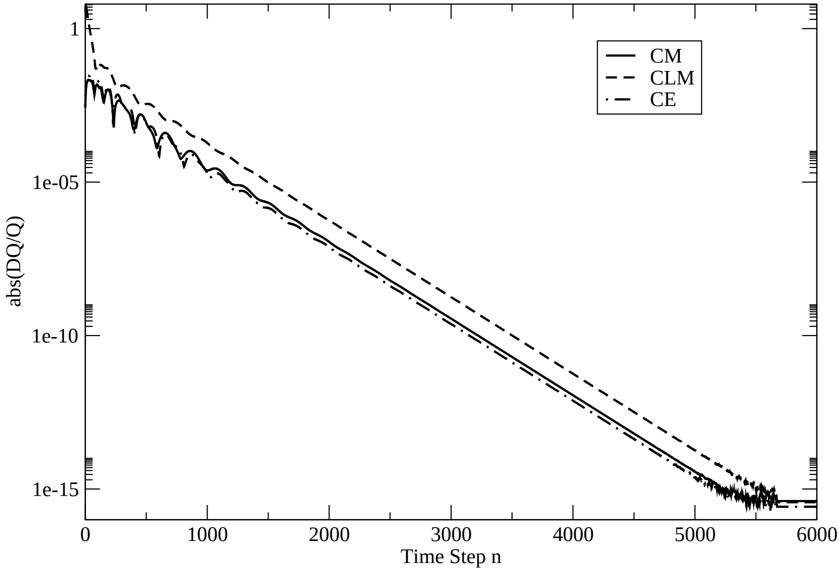

**Figure 4.9. Residue for the one-dimensional problem given in the assignment. This is plotted versus the time step index**

One observation that we can make in the earlier stages of the computations. Figure 4.11 clearly shows that conservation of mass and conservation of energy are intimately tied to each other in their behaviour. In fact you can see, in the earlier stages, that both the values increase together and decrease together.

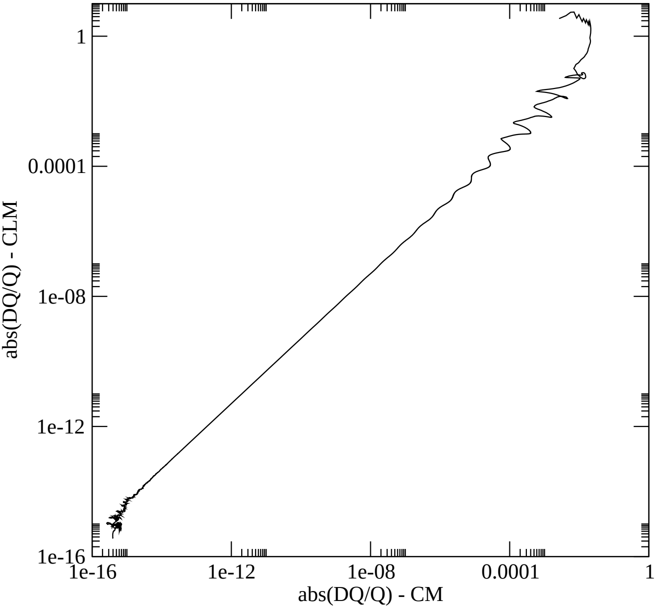

**Figure 4.10. Residues of the conservation of linear momentum (indicated as CLM) plotted versus the residue of the conservation of mass (cm) equation**

## Preconditioning

In all of these assignments, observe your solution as the program runs. This is best done by plotting one or more of the dependent variables as your code runs. For each time step, you can plot $\vec{Q}(x)$ in three separate plots. This is because of the difference in the order of magnitude of each term: $\rho \approx 1$, $\rho u \approx 100$, and $\rho E_t \approx \rho u^2 \approx 10000$. For the sake of this

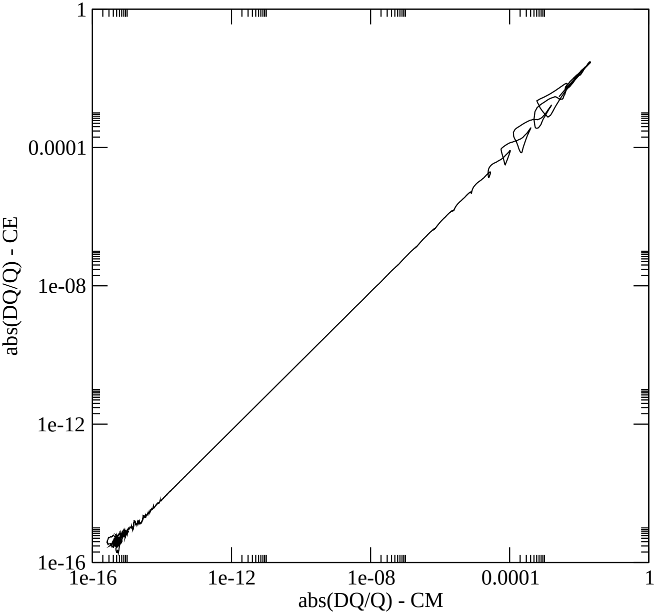

**Figure 4.11. Residue of the conservation of energy equation (CE) plotted against that of conservation of mass (cm)**

discussion you, can try running the code for this case. I have run a one-dimensional solver for the following conditions:

(1) $P_0 = 131590$ Pascals, $T_0 = 300$ Kelvin,
(2) $P_a = 101325$ Pascals,
(3) The pipe is of unit length and 1001 grid points are used to represent it. I am using FTCS with $\mu_2 = 0.01$ and $\mu_4 = 0.001$, where $\mu_2$ and $\mu_4$ are defined in equation (4.79).
(4) $\Delta t/\Delta x = 0.0001$.

Figure 4.12 shows the graphs of the density and speed at three different time steps, $n = 13000$ ( solid line ), $n = 40000$ (short dashed line), and $n = 59000$ (long dashed line). These three time indices were chosen so that the features of interest to this section can be illustrated.

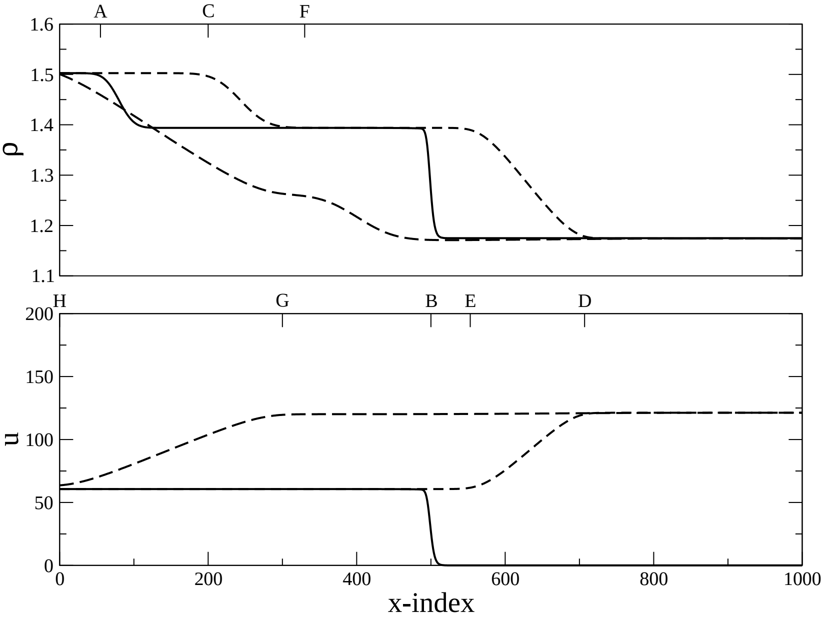

**Figure 4.12. The density $(\rho)$ and speed $(u)$ distribution at three different time steps $n = 13000$(solid line), $n = 40000$(short dashes), and $n = 59000$(long dashes).**

If you were to run this case, you could reproduce Figure 4.12 and observe the following. First, a compression wave travels from the inlet to the exit. It takes about 26000 time steps. The figure shows the compression wave at point $B$, midway along the pipe at $n = 13000$ time steps. To the left of the compression wave, the speed of the fluid is $u_L$ is of the order of 60+ m/s. To the right is the initial condition $u_R = 0$. The wave itself is formed by the intersection of the characteristics corresponding to $u + c$. For the given conditions $(u + c)_L \approx 400$ m/s and $(u + c)_R \approx 340$ m/s, the compression wave will be travelling at the average of those speeds. Now, this motion $u_L$, setup by the compression wave, carries the mass of the fluid with it. This may sound like a circular statement, the point is that we are talking about the characteristics corresponding to $u$. You see that the higher density fluid from the inlet, moving at this speed makes it to the point $A$ after $13000$ time steps. After all, it is travelling only at $u \approx 60$ m/s. This wave is a jump in density, not in pressure or in speed.[1] Both the compression wave and this density wave are smoothed out a bit because of the numerical dissipation that we have added.

Once the compression wave reaches the end of the pipe, the whole pipe is at the same pressure (it is interesting to check the relationship between $P_o$, $p$, and $\frac{1}{2}\rho u^2$ as the flow field evolves.) This is greater than the exit pressure. An expansion wave is reflected from the exit into the pipe. The location of the expansion wave after about 14000 time steps after is it is formed is shown in the figure as being between points $D$ and $E$. This expansion corresponds to the the characteristic $u - c$. Now as it happens, the wave at point $E$ is travelling at $60 - 346 \approx -286$

[1] This is a contact surface just like the one in 3.7 between the hot and cold water. If you have studied shock tubes you would have encountered this there.

m/s. On the other hand, the wave at point $D$ is travelling at $120 - 346 \approx -226$ m/s. The negative sign indicates that the wave is travelling right to left. The leading edge of the wave is travelling faster than the trailing edge. This wave is going to fan out and get broader at it travels. We have seen this earlier in Figure 3.22. This is very clear 19000 time steps later. The point corresponding to $D$ has moved to $G$ and that corresponding to $E$ has moved to $H$. Clearly, from the computations, as expected, $GH$ is longer than $DE$. On reaching the inlet, you will see a compression wave again from the inlet and so on. You will notice that as the waves move back and forth, the steady state flow field is setup.

The important point to note is that in the first pass of the waves, the compression wave took $\approx 26000$ time steps to communicate the upstream conditions to the exit. The return expansion wave took approximately 33000 time steps to communicate the downstream conditions to the inlet. In this time the wave corresponding to $u$ has not even made it to the exit! It is only at point $F$. Now, $u$ increases towards its steady state value. What happens to $u$, $u + c$, and $u - c$? You should notice that the speed at which the wave travels from the exit to the inlet decreases in each pass of the wave. This is because $u - c$ decreases with each pass of the waves. Try the following assignment and see what you can get from it.

---

### Assignment 4.10

Do the following with your favourite version of the one-dimensional code.

(1) For a given set of conditions, run you code for different number of grid points: $101$, $201$, $501$, and $1001$ and more if it makes sense to do so. Observe the plot of the residues, the wave propagation times (number of time steps) as a function of the number of grid points.

(2) The assignments so far were picked so that the Mach number for the solution is approximately 0.5. Run your code for different values of the stagnation pressure so that the Mach number takes a range of values between $(0.01, 2.0)$. Observe what happens near Mach number $0.01$ and $1.0$.

---

You should have noticed the following. You convergence plot shows a correlation with the wave motion back and forth. The larger the grid size the more obvious is this behaviour. From the second problem, it is clear that we have a severe problem near $M = 0.01$ and $M = 1$. All of these symptoms point to one problem. For a given acoustic speed $c$, if $u \approx 0$ then two of the characteristics are of the order of $c$ where one of them, $u$ is of the order of zero. The propagation speeds in our problem are very disparate. Such problems are said to be **stiff** or **ill-conditioned**. This situation also occurs when $u$ is of the order of $c$. In this case $u - c \approx 0$ while the other two are of the order of $c$. This explains the difficulty that we have at the two Mach numbers. We conclude

> **Disparate propagation speeds has a strong effect on the convergence to a steady state solution**

Naturally we ask: if we are only interested in the steady state solution can we do something to make the wave speeds the same order of magnitude and not contaminate the steady state solution? Restated, if I do not care for the transient, can I do something to get to the solution faster? Or a blunt statement of fact: I am willing to live with a non-physical transient if I can get to the correct steady state solution rapidly.

We have a clue as to how this can be done from our experience with the Delta form. If some term in our equation is going to zero, we can multiply it with a non-singular expression and not affect our solution. in the case of our governing equations, when we get to the steady state, the time derivatives should be zero. We can pre-multiply the time derivative term with a matrix $\mathbf{\Gamma}$ to get an equation

$$\mathbf{\Gamma}\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} = 0 \tag{4.140}$$

So, what is $\mathbf{\Gamma}$? That is for us to decide. Let us now multiply equation (4.140) through by $\mathbf{\Gamma}^{-1}$. We get

$$\frac{\partial Q}{\partial t} + \mathbf{\Gamma}^{-1}\boldsymbol{A}\frac{\partial Q}{\partial x} = 0 \tag{4.141}$$

This is the equation written in non-conservative form. We see that we have indeed got an opportunity to modify the physics of the problem. We need to choose $\mathbf{\Gamma}$ in a manner that $\mathbf{\Gamma}^{-1}\boldsymbol{A}$ has eigenvalues that are almost equal to each other. This process of changing the eigenvalues of a problem of interest to make it more amenable to solution is called **preconditioning**. In this context, since we achieve this end by multiply only the transient term we call it preconditioning the unsteady term.

We want to keep it simple for the sake of this discussion. We will seek a $\mathbf{\Gamma}$ so that the eigenvalues of $\mathbf{\Gamma}^{-1}\boldsymbol{A}$ are $(1, 1, -1)$ and the eigenvectors are the same. We are propagating the same "physical" quantities. The only difference is that the propagation speeds are equal. This tells that

**(4.142)**
$$\mathbf{\Gamma}^{-1}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\mathbf{\Lambda}_1, \quad \mathbf{\Lambda}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Since the eigenvectors are the same, we have $\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\mathbf{\Lambda}$. Substituting, we get

**(4.143)**
$$\mathbf{\Gamma}^{-1}\boldsymbol{X}\mathbf{\Lambda} = \boldsymbol{X}\mathbf{\Lambda}_1$$

If we post-multiply through by $\mathbf{\Lambda}_1^{-1}$, pre-multiply by $\mathbf{\Gamma}$, and rearrange the equation, we get

**(4.144)**
$$\mathbf{\Gamma}\boldsymbol{X} = \boldsymbol{X}\mathbf{\Lambda}\mathbf{\Lambda}_1^{-1} = \boldsymbol{X}|\mathbf{\Lambda}|, \quad |\mathbf{\Lambda}| = \begin{bmatrix} |u| & 0 & 0 \\ 0 & |u+c| & 0 \\ 0 & 0 & |u-c| \end{bmatrix}$$

We choose the $\mathbf{\Gamma}$

**(4.145)**
$$\mathbf{\Gamma} = \boldsymbol{X}|\mathbf{\Lambda}|\boldsymbol{X}^{-1}$$

## Finite Volume Method

We will look at the finite volume method in greater detail in Chapter 6. We will continue the discussion that we started in Section 3.16. I have reproduced the Figure 3.41 here for convenience. The "volume" indexed by $i$ extends from $x_{i-\frac{1}{2}}$ to $x_{i+\frac{1}{2}}$. The mean value of $\vec{Q}$ in the volume is labelled notionally

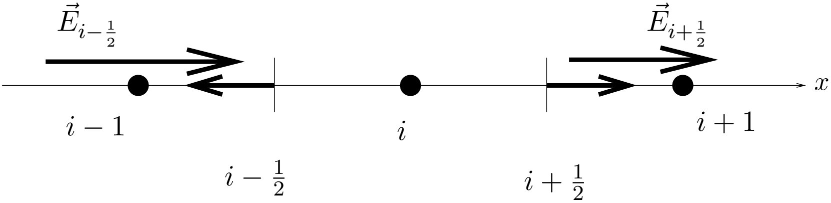

Figure 4.13. A small control volume about the grid point i. The boundaries of the volume are shown at $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$. The short vectors are outward unit vectors "normal" to the boundary. The other two vectors are the flux terms $\vec{E}_{i-\frac{1}{2}}$ and $\vec{E}_{i+\frac{1}{2}}$

as $\vec{Q}_i$. Integrating equation (4.4) with respect to $x$ from $x_{i-\frac{1}{2}}$ to $x_{i+\frac{1}{2}}$ gives us

$$\textbf{(4.146)}\quad \frac{\mathrm{d}}{\mathrm{d}t}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \vec{Q}(\xi,t)d\xi = \frac{\mathrm{d}}{\mathrm{d}t}\left(\vec{Q}_i\Delta x_i\right) = -\left(\vec{E}(x_{i+\frac{1}{2}},t) - \vec{E}(x_{i-\frac{1}{2}},t)\right)$$

where $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. In order the get the time evolution of the parameter $\vec{Q}_i$, we need to integrate this equation in time. Unfortunately, this means that we need to determine $\vec{E}(x_{i+\frac{1}{2}},t)$ and $\vec{E}(x_{i-\frac{1}{2}},t)$. We know that $\vec{E}_{i\pm\frac{1}{2}} = E(\vec{Q}_{i\pm\frac{1}{2}})$. How do we determine $\vec{Q}_{i\pm\frac{1}{2}}$? Further, it is clear that we should use $\vec{E}^{\pm}_{i\mp\frac{1}{2}}$ and not just $\vec{E}$. At a given point to determine $\vec{E}^{\pm}$ one also needs to determine $u$ and $u \pm c$. Again, we need the state at the boundaries of the control volume.

---

**Assignment 4.11**

Write a finite volume solver using equation (4.146).

---

**Roe's Averaging.** We will determine $\vec{Q}_{i\pm\frac{1}{2}}$ using a process called Roe's averaging [**Roe81**]. We have done a similar (though not identical) derivation to determine the wave speed in section 3.16 equation (3.266).

Consider the interface between two cells $i$ and $i+1$. This is indicated as the point $i+\frac{1}{2}$ in Figure 4.13. To the left of this interface we have the state $\vec{Q}_i$ and to the right of it we have $\vec{Q}_{i+1}$. We need to use these two states to determine $\vec{E}_{i+\frac{1}{2}}$.

If the system of equations were such that the flux Jacobian $A$ was a constant $\mathbb{A}$ in the interval $(x_i, x_{i+1})$ over the time period $\Delta t$, we could use our characteristic equations to propagate the characteristic variables in time over the period $\Delta t$. The flux Jacobian $A$ is not likely to be a constant. We will assume it is. What should we take as the that constant value $\mathbb{A}$? We want an average value of $A$. Again, we would like to determine $\mathbb{A}$ using the two states $\vec{Q}_i$ and $\vec{Q}_{i+1}$.

From the two paragraphs can we summarise a question as follows: Can we find a $\vec{Q}_{i+\frac{1}{2}}$ such that

$$\mathbb{A} = A(\vec{Q}_{i+\frac{1}{2}}) = \left.\frac{\partial \vec{E}}{\partial \vec{Q}}\right|_{i+\frac{1}{2}} ? \tag{4.147}$$

Now, we know that $\mathbb{A}$ is related to $\vec{E}$ and $\vec{Q}$ through their respective changes over the interval. That is

**(4.148)** $$\vec{E}_{i+1} - \vec{E}_i = \Delta\vec{E}_{i+\frac{1}{2}} = \mathbb{A}\cdot\Delta\vec{Q} = \mathbb{A}\cdot\left(\vec{Q}_{i+1} - \vec{Q}_i\right)$$

We want the entries of $\mathbb{A}$ that are given by equation (4.21) and repeated here for convenience.

**(4.149)**

$$\mathbb{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)u_h^2 & (3-\gamma)u_h & (\gamma-1) \\ (\gamma-1)u_h^3 - \gamma E_h u_h & \gamma E_h - \frac{3}{2}(\gamma-1)u_h^2 & \gamma u_h \end{bmatrix}$$

where $u_h = u^2_{i+\frac{1}{2}}$ and $E_h = E_{t i+\frac{1}{2}}$. Also,

**(4.150)** $$\Delta\vec{Q} = \begin{bmatrix} \Delta\rho \\ \Delta(\rho u) \\ \Delta(\rho E_t) \end{bmatrix}, \qquad \Delta\vec{E} = \begin{bmatrix} \Delta(\rho u) \\ \Delta(\rho u^2 + p) \\ \Delta\{(\rho E_t + p)u\} \end{bmatrix}$$

We can use equation (4.148) to determine the entries in $\mathbb{A}$. We see from equation (4.149), that this should allow us to determine $u_{i+\frac{1}{2}}$ and $E_{t i+\frac{1}{2}}$. The student is encouraged to continue with this derivation to see where the difficulties arise. We will follow standard procedure here and convert the various expressions containing $E_t$ and $p$ into terms using $H_t$. $H_t$ is the total enthalpy.

**(4.151)** $$\rho H_t = \rho E_t + p$$

First we eliminate $p$.

We have already seen in equation (4.18) that

**(4.152)**
$$p = (\gamma - 1)\left\{\rho E_t - \frac{\rho u^2}{2}\right\}$$

We use this expression to eliminate $p$ from equation (4.151) to get

**(4.153)**
$$\rho H_t = \gamma \rho E_t - \frac{\gamma - 1}{2}\rho u^2 \Rightarrow E_t = \frac{1}{\gamma}\left(H_t + \frac{\gamma - 1}{2}u^2\right)$$

Consequently, we can verify that

**(4.154)**
$$p = \frac{\gamma - 1}{\gamma}\left\{\rho H_t - \frac{\rho u^2}{2}\right\}$$

To keep the expressions looking simple we will drop the $i + \frac{1}{2}$ subscript in $\mathbb{A}$. We can now write

**(4.155)**
$$\begin{bmatrix} \Delta(\rho u) \\ \Delta(\rho u^2 + p) \\ \Delta\{(\rho H_t u\} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma - 3)u^2 & (3 - \gamma)u & (\gamma - 1) \\ Cu & -C & \gamma u \end{bmatrix} \begin{bmatrix} \Delta\rho \\ \Delta(\rho u) \\ \Delta(\rho E_t) \end{bmatrix}$$

where $C = \frac{\gamma - 1}{2}u^2 - H_t$

The second equation still has $p$ on the left hand side and the last equation still has a $\rho E_t$ on the right hand side. We will substitute for them from equations (4.153) and (4.154) as we multiply the matrix equation out to get the individual equations.

The first equation gives us nothing; just $\Delta(\rho u) = \Delta(\rho u)$. The second equation gives us

**(4.156)**

$$\frac{\gamma-3}{2}u^2\Delta\rho+(3-\gamma)u\Delta(\rho u)+(\gamma-1)\Delta\left(\frac{\rho H_t}{\gamma}+\frac{\gamma-1}{2\gamma}\rho u^2\right)= \Delta\left(\frac{\gamma+1}{2\gamma}\rho u^2+\frac{\gamma-1}{\gamma}\rho H_t\right)$$

The $\rho H_t$ terms cancel, giving

**(4.157)**

$$\frac{1}{2}(\gamma-3)u^2\Delta\rho+(3-\gamma)u\Delta(\rho u)+\frac{(\gamma-1)^2}{2\gamma}\Delta(\rho u^2) = \frac{\gamma+1}{2\gamma}\Delta(\rho u^2)$$

We remind ourselves that $u$ is actually $u_{i+\frac{1}{2}}$ for which we have a quadratic

**(4.158)** $$au^2_{i+\frac{1}{2}} + bu_{i+\frac{1}{2}} + c = 0$$

where

**(4.159)** $$a = \Delta\rho$$

**(4.160)** $$b = -2\Delta(\rho u)$$

**(4.161)** $$c = \Delta(\rho u^2)$$

The two solutions can be written as

**(4.162)** $$u_{i+\frac{1}{2}} = \frac{\Delta(\rho u) \pm \sqrt{\Delta(\rho u)^2 - \Delta\rho\Delta(\rho u^2)}}{\Delta\rho}$$

We cannot simplify this any further without substituting for the terms on the right hand side as

**(4.163)** $$\Delta\rho = \rho_{i+1} - \rho_i = \rho_{_R} - \rho_{_L}$$

**(4.164)** $$\Delta(\rho u) = \rho_{i+1}u_{i+1} - \rho_i u_i = \rho_{_R}u_{_R} - \rho_{_L}u_{_L}$$

**(4.165)** $$\Delta(\rho u^2) = \rho_{i+1}u_{i+1}^2 - \rho_i u_i^2 = \rho_{_R}u_{_R}^2 - \rho_{_L}u_{_L}^2$$

As in section 3.16, we have use the subscripts $R$ and $L$ to indicate states to the right and left of the interface at $x_{i+\frac{1}{2}}$. When we substitute and simplify terms under the square root we get

**(4.166)** $$u_{i+\frac{1}{2}} = \frac{\rho_{_R}u_{_R} - \rho_{_L}u_{_L} \pm \sqrt{\rho_{_R}\rho_{_L}}(u_{_R} - u_{_L})}{\rho_{_R} - \rho_{_L}}$$

By combining the two terms containing $\sqrt{\rho_{_R}}u_{_R}$ and $\sqrt{\rho_{_L}}u_{_L}$ and taking the negative sign from the ±, we get

**(4.167)**
$$u_{i+\frac{1}{2}} = \frac{\sqrt{\rho_{_R}}u_{_R}(\sqrt{\rho_{_R}} - \sqrt{\rho_{_L}}) + \sqrt{\rho_{_L}}u_{_L}(\sqrt{\rho_{_R}} - \sqrt{\rho_{_L}})}{\left(\sqrt{\rho_{_R}}\right)^2 - \left(\sqrt{\rho_{_L}}\right)^2}$$

So,

**(4.168)** $$u_{i+\frac{1}{2}} = \frac{\sqrt{\rho_{_R}}u_{_R} + \sqrt{\rho_{_L}}u_{_L}}{\sqrt{\rho_{_R}} + \sqrt{\rho_{_L}}}$$

A peculiar looking expression. However, the fact that it can be written as

**(4.169)** $$u_{i+\frac{1}{2}} = \alpha u_{_R} + (1-\alpha)u_{_L}, \quad \alpha = \frac{\sqrt{\rho_{_R}}}{\sqrt{\rho_{_R}} + \sqrt{\rho_{_L}}}$$

gives us confidence. Why did we not take the other root? You can followup on that and figure it out. Now, repeat this derivation for $H_{ti+\frac{1}{2}}$ to get

**(4.170)** $$H_{ti+\frac{1}{2}} = \alpha H_{t_R} + (1-\alpha) H_{t_L}, \quad \alpha = \frac{\sqrt{\rho_R}}{\sqrt{\rho_R} + \sqrt{\rho_L}}$$

---

**Assignment 4.12**

(1) Take the positive sign in equation (4.166) and see what is the consequence.
(2) Use the Roe average in your finite volume solver.

---

## Quasi-One-Dimensional Flow

In the last few chapters we have built up quite a bit of CFD machinery. How is all of this applicable to a problem like the nozzle we looked at in the first chapter. We want to keep the equations simple and account for the area variation. We will use the quasi-one-dimensional form of the governing equations to solve the problem. The governing equations now look like

**(4.171)** $$\frac{\partial Q\mathcal{A}}{\partial t} + \frac{\partial E\mathcal{A}}{\partial x} = H$$

The term $\mathcal{A}$ is the area of cross-section of the duct and in fact varies with $x$. For a converging-diverging nozzle (CD nozzle), we would have an $\mathcal{A}(x)$ which first decreases and then increases. The other new term in the equation is the $H$ on the right hand side of the equation. These terms are usually referred to as source term. We have used the term $E$ for the $x$ component of the flux, we expect to use $F$ and $G$ for the $y$ and $z$ components of the flux. As a consequence we are left with $H$ as the source term. In this case the source term is a consequence of writing the quasi-one-dimensional equations in the conservative form and has components

$$H = \begin{pmatrix} 0 \\ \\ p\frac{d\mathcal{A}}{dx} \\ \\ 0 \end{pmatrix} \tag{4.172}$$

For moderate variations in $\mathcal{A}$ we do not expect this change to make much of a difference to our programs. However, our experience to date with our expectations and how things have turned out makes it clear that we need to check this out.

---

**Assignment 4.13**

(1) Write a program to compute flow through a converging duct.
(2) Diverging duct.
(3) C-D duct.

---

## Important ideas from this chapter

- Equations can be written in conservative form or non-conservative form. They are typically used in conservative form.

- The stability analysis from the one-dimensional linear wave equation can be extended to the one-dimensional Euler equations, after decoupling the system of equations.
- One can employ characteristics to decide on the boundary conditions to be applied.
- The characteristics velocities determine the convergence of the scheme. Very disparate characteristics result in a problem that is said to be ill-conditioned. One way to solve this problem is to employ preconditioning.
- The crux of a typical finite volume scheme is that the state is not known where the flux is required. One interpolates and manages.
- Hopefully from the assignments: Indiscriminate use of artificial dissipation can lead programs to converge, but to what?

# Tensors and the Equations of Fluid Motion

We have seen that there are a whole range of things that we can represent on the computer. Our objective was to dive into the process of representing and solving partial differential equations on the computer. We have solved some simple problems such as Laplace's equation on a unit square at the origin in the first quadrant. From the description of the problem, you can see that it was really a very specific problem. It was simple enough that we could analyse the resulting discrete equations.

Similarly, we have studied the first order one-dimensional linear wave equation. We have also seen the heat equation and the quasi linear version of the wave equation. We have some basic understanding of the issues involved in representing and solving these problems on the computer.

Finally, in the last chapter, we looked at the one-dimensional Euler's equation though the formal derivation of the equations was put off till this chapter. We will look at solution in multiple dimensions in chapter 6.

We are now in a position to solve a larger class of problems. We will expand on the simple problems we studied in the preceding chapters as a means to motivate this chapter. We will then do the essentials of tensor calculus required to derive the equations of fluid motion. The equations of motion will be derived in vector form so as to be independent of the particular coordinate system. Taken along with the tensor calculus you should be able to specialise these equations to any particular coordinate system.

## Laplace Equation Revisited

We solved the Laplace equation on a unit square. How would we handle the problem domain shown in Figure 5.1? Again, we are given boundary conditions on the four sides of the quadrilateral. If we tried to solve it using a Cartesian mesh as we did before, we would get something that looks like the domain and mesh shown in Figure 5.2. Everything looks fine till you look at the top edge of the trapezium. Excepting two points, there are

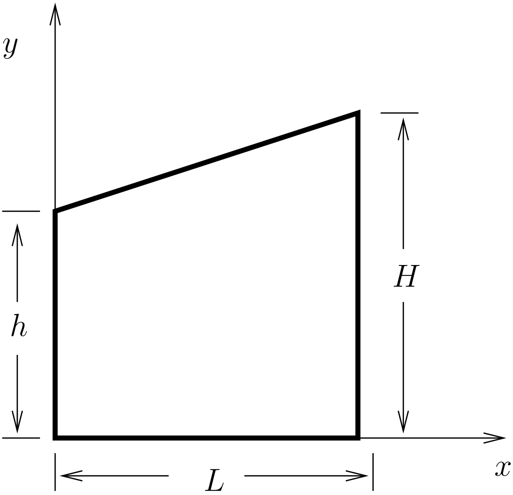

**Figure 5.1. A trapezoidal domain on which Laplace equation is to be solved**

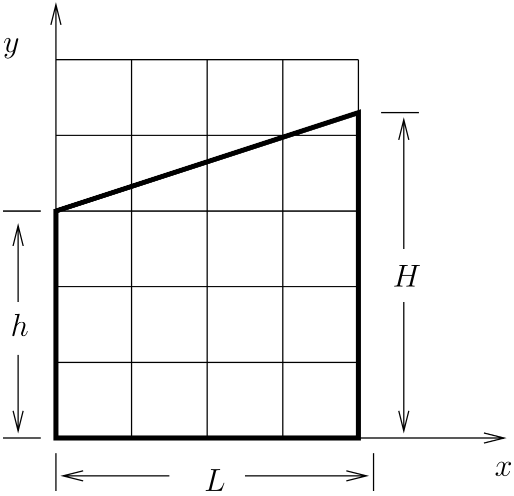

**Figure 5.2. Trapezoidal domain with an underlying Cartesian mesh.**

no other grid points on the boundary. How then can we apply the boundary condition? We could forcibly insert mesh points at the intersection of the grid lines the boundary. This would lead to unequal distances between the mesh points in our discretisation. Those points then have to be dealt with as a special case. The other possibility is to consider every grid point as being separated from neighbouring grid points by unequal distances as shown in Figure 5.3. It is likely that we will have to treat some points as special points. We have come to expect that we will treat the boundary points differently from the rest of the grid points anyway. After all, look at what we did in the other problems that we have seen, especially the one-dimensional Euler equations. Since we are trying to motivate tensor calculus,

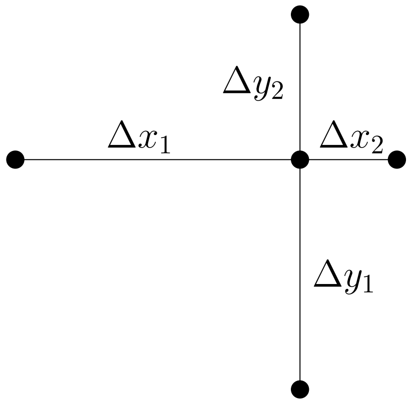

**Figure 5.3. A grid point with neighbouring points placed at uneven distances.**

our interest lies in a third possibility. That is to generate a non-Cartesian mesh. One such mesh is shown in Figure 5.4. If we study the mesh shown in the figure, we see that the mesh lines conform to the boundary of the domain. Imagine in your mind that you have a rubber sheet the shape of this trapezium. You could stretch the sheet so that the stretched sheet looked like a square. The mesh lines would coincide with the Cartesian grid lines. This tells us that we need to perform a transformation of our coordinates so that we are back to solving our problem on a Cartesian mesh.

If our coordinates in the stretched sheet are $(\xi, \eta)$, the mesh lines seen in Figure 5.4 would be constant $\xi$-lines and constant $\eta$-lines. The Figure 5.4 is drawn in the $x - y$ plane. Clearly, what we need is a transformation going from one coordinate

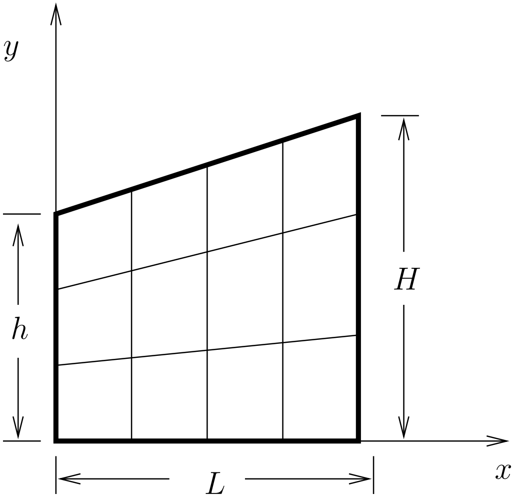

**Figure 5.4.** **A non-Cartesian mesh in a Trapezoidal domain. The sides of the quadrilateral are mesh lines.**

system to another. Say,

$$\xi = \xi(x, y), \tag{5.1}$$

$$\eta = \eta(x, y), \tag{5.2}$$

and the corresponding reverse relationship

$$x = x(\xi, \eta), \tag{5.3}$$

$$y = y(\xi, \eta). \tag{5.4}$$

We are in a position where, given the solution at a point in one coordinate system, we can provide the solution at the corresponding point in the other coordinate system. Let us step back for a minute to see where we are.

We have Laplace equation given to us in a trapezoidal domain in the $x-y$ coordinate system. A little stretch will give us a square in the $\xi-\eta$ coordinate system, but what happens to Laplace's equation in the $\xi-\eta$ plane? We use the coordinate transformation given by equations (5.1) to (5.3) along with chain rule to transform the derivatives. For example, here are the first derivatives in $x$ and $y$.

**(5.5)**
$$\frac{\partial}{\partial x} = \underbrace{\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi}}_{A} + \underbrace{\frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}}_{B}$$

**(5.6)**
$$\frac{\partial}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}$$

Since, the transformation is known, we can determine the partial derivative on the right hand side of equation (5.5). How do we use the expression given by equation (5.5)? We can take the corresponding partial derivative of $\phi$. On doing this, we get

**(5.7)**
$$\frac{\partial \phi}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial \phi}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial \phi}{\partial \eta}$$

**(5.8)**
$$\frac{\partial \phi}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial \phi}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial \phi}{\partial \eta}$$

So far it looks manageable. Since we want to solve Laplace's equation, we now look at the second derivatives. The second

$x$–derivative is

$$\textbf{(5.9)}\quad \begin{aligned}\frac{\partial^2}{\partial x^2} &= \frac{\partial}{\partial x}\left\{\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right\} \\ &= \underbrace{\frac{\partial^2 \xi}{\partial x^2}\frac{\partial}{\partial \xi}}_{A_1} + \underbrace{\left(\frac{\partial \xi}{\partial x}\right)^2 \frac{\partial^2}{\partial \xi^2} + \frac{\partial \xi}{\partial x}\frac{\partial \eta}{\partial x}\frac{\partial^2}{\partial \xi \partial \eta}}_{A_2} \\ &\qquad + \underbrace{\frac{\partial^2 \eta}{\partial x^2}\frac{\partial}{\partial \eta}}_{B_1} + \underbrace{\frac{\partial \eta}{\partial x}\frac{\partial \xi}{\partial x}\frac{\partial^2}{\partial \xi \partial \eta} + \left(\frac{\partial \eta}{\partial x}\right)^2 \frac{\partial^2}{\partial \eta^2}}_{B_2}\end{aligned}$$

This is a little messy. To make sure we understand this clearly, the term $A$ in equation (5.5) results in the terms identified as $A_1$ and $A_2$ in equation (5.9). The same is true of the terms marked $B$ in the two equations. $A_1$ and $A_2$ are a consequence of applying product rule. The two terms in $A_2$ emerge from applying equation (5.5) to obtain the derivative of the $\partial/\partial\xi$ term with respect to $x$. In a similar fashion we can write the second derivative with respect $y$ as

$$\textbf{(5.10)}\quad \begin{aligned}\frac{\partial^2}{\partial y^2} &= \frac{\partial^2 \xi}{\partial y^2}\frac{\partial}{\partial \xi} + \left(\frac{\partial \xi}{\partial y}\right)^2 \frac{\partial^2}{\partial \xi^2} + \frac{\partial \xi}{\partial y}\frac{\partial \eta}{\partial y}\frac{\partial^2}{\partial \xi \partial \eta} \\ &\quad + \frac{\partial^2 \eta}{\partial y^2}\frac{\partial}{\partial \eta} + \frac{\partial \eta}{\partial y}\frac{\partial \xi}{\partial y}\frac{\partial^2}{\partial \xi \partial \eta} + \left(\frac{\partial \eta}{\partial y}\right)^2 \frac{\partial^2}{\partial \eta^2}\end{aligned}$$

Then the transformed Laplace equation can be written as

$$\textbf{(5.11)} \quad \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \\ \left(\xi_x^2 + \xi_y^2\right) \frac{\partial^2 \phi}{\partial \xi^2} + 2\left(\xi_x \eta_x + \xi_y \eta_y\right) \frac{\partial^2 \phi}{\partial \xi \partial \eta} + \left(\eta_x^2 + \eta_y^2\right) \frac{\partial^2 \phi}{\partial \eta^2} \\ + \left(\xi_{xx} + \xi_{yy}\right) \frac{\partial \phi}{\partial \xi} + \left(\eta_{xx} + \eta_{yy}\right) \frac{\partial \phi}{\partial \eta} = 0$$

To keep things more compact, we decide to use the notation that the subscript indicates differentiation with respect to that parameter. So,

$$\textbf{(5.12)} \qquad \xi_x = \frac{\partial \xi}{\partial x}$$

Using this notation uniformly, the Laplace equation in the $\xi - \eta$ plane is given by

$$\textbf{(5.13)} \quad \left(\xi_x^2 + \xi_y^2\right) \phi_{\xi\xi} + 2\left(\xi_x \eta_x + \xi_y \eta_y\right) \phi_{\xi\eta} + \left(\eta_x^2 + \eta_y^2\right) \phi_{\eta\eta} \\ + \left(\xi_{xx} + \xi_{yy}\right) \phi_\xi + \left(\eta_{xx} + \eta_{yy}\right) \phi_\eta = 0$$

The domain for the problem has become easier, the equation does not quite fit in one line! Also, it is not in quite the right form. The coefficients are still expressed in the $x, y$ coordinate system. We make the following observations and see if we can clear the air a bit.

- We want to solve problems that involve complicated domains. There may be many methods to handle complicated problems, performing transformation of coordinates is definitely one way to do it.

- We do not want to have to re-derive our governing equation in every new coordinate system that we encounter. We need a general frame work in which we can derive our equations.
- The introduction of the subscript notation gave some relief in handling the equation. So, the proper choice of notation is going to make life easier for us. Further, we can do work on more difficult / complex problems with the effort that we are currently expending.
- We observe that the only difference between equation (5.9) and (5.10) is the replacement of $x$ with $y$. Again, we need the notation that will help us to abstract these kinds of patterns out, so that we do not have to repeat the derivation for each coordinate.
- We want to solve problems in three dimensions and not just one and two dimensions. If we are going to perform transformations in three dimensions, we need to have some minimal understanding of geometry in three dimensions.

We will address the last point here by looking at a little differential geometry. Coordinate lines in three dimensions are curves in three dimensions and we will try to get a handle on them. A region of interest in three dimensions will be a volume and it is defined using surfaces. We will take a brief look at surfaces. Tensor calculus is a tool to address the rest of the issues raised in our list of observations. We will do a little tensor calculus and some geometry.

As further motivation as to why one needs tensor calculus, consider the following conundrum. If you have learnt only "calculus", the sequence of courses typically taught in an undergraduate curriculum, this is for you to puzzle over; to show you there must be life beyond "calculus". Consider a potential flow in two dimensions. The velocity can be written in component form as $(u, v)$ in Cartesian coordinates. If we were to transform the velocity to some other coordinates $(\xi, \eta)$ we get

$$u = \frac{dx}{dt} = x_t = x_\xi \xi_t + x_\eta \eta_t = x_\xi U + x_\eta V \tag{5.14}$$

$$v = \frac{dy}{dt} = y_t = y_\xi \xi_t + y_\eta \eta_t = y_\xi U + y_\eta V \tag{5.15}$$

Where $(U, V)$ are the velocities in the $\xi - \eta$ coordinates. The matrix representation of this transformation equation is

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{pmatrix} U \\ V \end{pmatrix} \tag{5.16}$$

We also have from the definition of the potential

$$u = \frac{\partial \phi}{\partial x} = \phi_x = \phi_\xi \xi_x + \phi_\eta \eta_x = \xi_x U + \eta_x V \tag{5.17}$$

$$v = \frac{\partial \phi}{\partial y} = \phi_y = \phi_\xi \xi_y + \phi_\eta \eta_y = \xi_y U + \eta_y V \tag{5.18}$$

Which has a representation

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} \xi_x & \eta_x \\ \xi_y & \eta_y \end{bmatrix} \begin{pmatrix} U \\ V \end{pmatrix} \tag{5.19}$$

They contradict each other and are **wrong**

Why are these equations, (5.16) and (5.19), different? How can the $u$ and $v$ transform in two different ways? One immediate conclusion that the equations are wrong. We should be able to figure out what is wrong with these equations since there are basically three terms involved. The left hand side of these two equations are clearly fine since they are the quantities with which we start and are a given. The chain rule part follows from calculus. That procedure looked right. That leaves the $U$ and $V$ and of course, the $=$ symbol. We want the equation relating velocities in the two coordinate systems. That means there is a problem with the assumption that the $U$ and $V$ in equation (5.16) are the same as the $U$ and $V$ in equation (5.19). So, there may be two different kinds of $U$ and $V$. To clear up these issues **study tensor calculus.**[**You93**][**Ari89**][**SS82**]! We will do a very quick overview here.

## Tensor Calculus

Very often, we assume that a vector $\vec{V}$ can be written in terms of a global basis vectors $\hat{e}_1, \hat{e}_2, \hat{e}_3$ as follows

**(5.20)**
$$\vec{V} = v^1\hat{e}_1 + v^2\hat{e}_2 + v^3\hat{e}_3 = \sum_{i=0}^{3} v^i\hat{e}_i$$

We will see what we mean by a global basis as we go along. For now, do not confuse the superscript on $v$ with exponentiation. We deliberately chose superscripts and subscripts since we anticipate that we are going to encounter two different kinds of entities. We will see that superscripted entities are said to be contravariant and subscripted entities are covariant. So, $v^1$

may be different from $v_1$. We will see what this means as we go along. If we agree that any time the index is repeated it implies a summation, we can simply write

$$\vec{V} = v^i \hat{e}_i \tag{5.21}$$

Now, THAT is compact. It is called Einstein's summation convention. It only gets better. By itself, the equation does not even restrict us to three dimensions. It is our assumption that we use three dimensions. In this book, we will restrict ourselves to two / three dimensions. You should note that

$$\vec{V} = v^i \hat{e}_i = v^k \hat{e}_k \tag{5.22}$$

Since there is a summation over the index, the index itself does not survive the summation operation. The choice of the index is left to us. It is called a **dummy index**.

We now define the notation with respect to coordinate systems. Consider Figure 5.5. It indicates a differential line element with points $\boldsymbol{P}$ and $\boldsymbol{Q}$ at each end of the element. We define $\vec{x}(.)$ as a coordinate function which returns the coordinates of a point in the Cartesian coordinate system. That is, for a point $\boldsymbol{P}$, $\vec{x}(\boldsymbol{P})$ gives us the corresponding coordinates. If we had another coordinate system overlayed on the same region, the point $\boldsymbol{P}$ will have the corresponding coordinates $\vec{\xi}(\boldsymbol{P})$ in that coordinate system. The coordinate function is simple to imagine if we look at it component-wise.

$$\vec{x}(\boldsymbol{P}) = x^1(\boldsymbol{P})\hat{e}_1 + x^2(\boldsymbol{P})\hat{e}_2 + x^3(\boldsymbol{P})\hat{e}_3 \tag{5.23}$$

Since we are dealing with Cartesian coordinates, $x^i$ and $x_i$ are the same. we have already seen that if $\vec{P}$ is the position vector

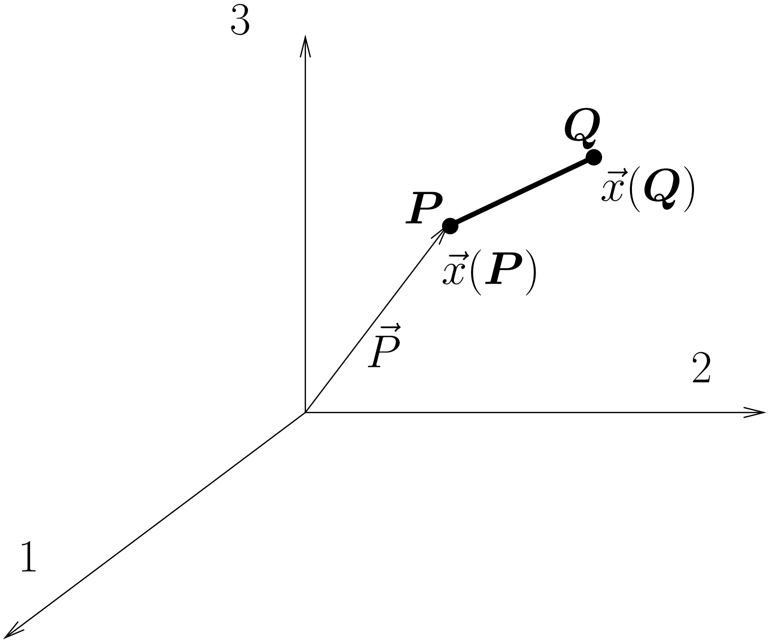

**Figure 5.5.** A Cartesian coordinate system used to locate the point $\boldsymbol{P}$ and $\boldsymbol{Q}$. $\vec{x}(\boldsymbol{P})$ gives the position vector of $\boldsymbol{P}$ in the Cartesian coordinate system. That is, $\vec{P} = \vec{x}(\boldsymbol{P})$. $\boldsymbol{PQ}$ forms a differential element.

for $\boldsymbol{P}$ then

**(5.24)**
$$x_i(\boldsymbol{P}) = \vec{P} \cdot \hat{e}_i$$

Consider the problem of coordinate transformations in two dimensions. Let us restrict ourselves for the sake of this discussion to rotations. We take our standard $x - y$ coordinate and rotate through an angle $\theta$ to get the $\xi - \eta$ coordinates. The basis vectors in $x - y$ are $\vec{e}_1$ and $\vec{e}_2$. The basis vectors in the $\xi - \eta$ coordinates are $\vec{\epsilon}_1$ and $\vec{\epsilon}_2$. You can check that the basis vectors are related as follows:

**(5.25)**
$$\begin{pmatrix} \vec{\epsilon}_1 \\ \vec{\epsilon}_2 \end{pmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} \vec{e}_1 \\ \vec{e}_2 \end{pmatrix}$$

We see that by using indices we can simply represent this as

**(5.26)**
$$\vec{\epsilon}_i = A_i^j \vec{e}_j$$

Now, a vector $\vec{s}$ can be represented in the $x - y$ and the $\xi - \eta$ coordinate systems as

**(5.27)**
$$\vec{s} = s^i \vec{e}_i = \psi^i \vec{\epsilon}_i$$

Substituting for $\vec{\epsilon}_i$ from equation (5.26) we get

**(5.28)**
$$\vec{s} = s^i \vec{e}_i = s^j \vec{e}_j = \psi^i \vec{\epsilon}_i = \psi^i A_i^j \vec{e}_j$$

where $i$ and $j$ are dummy indices. Even though they are dummy indices, by the proper choice of these dummy indices here we can conclude that

**(5.29)**
$$s^j = \psi^i A_i^j = A_i^j \psi^i$$

Compare equations (5.26) and (5.29). The unit vectors transform one way, the components transform the opposite [ or contra ] way. We see that they too show the same behaviour we saw with the velocity potential. Vectors that transform like each other are covariant with each other. Vectors that transform the opposite way are contravariant to each other. This is too broad a scenario for us. We will stick with something simpler. **Covariant** entities will be subscripted. **Contravariant** entities will be superscripted.

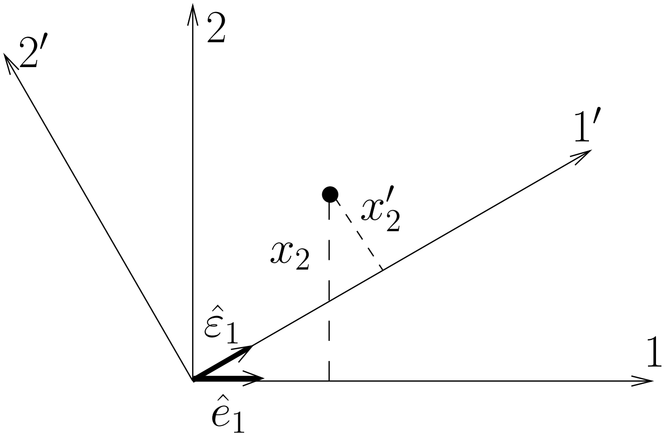

**Figure 5.6. The basis vectors rotate with the coordinate axes (only $\hat{e}_1$ and $\vec{\varepsilon}_1$ are shown). The coordinates of the point in the new system are as though the point had moved clockwise and the coordinate system was fixed. That is $x_2' < x_2$ in this particular case.**

An example where this will be obvious to you is the case of the rotation of the Cartesian coordinate system. Again, we restrict ourselves to two dimensions. If you rotate the standard Cartesian coordinate system counter-clockwise, you see that the coordinate lines and the unit vectors ( as expected ) rotate in the same direction. They are covariant. The actual coordinate values do not change in the same fashion. In fact, the new values corresponding to a position vector look as though the coordinate system was fixed and that the position vector was rotated in a clockwise sense ( contra or opposite to the original coordinate rotation ). These two rotations are in fact of equal magnitude and opposite in sense. They are, indeed, inverses of each other. We will investigate covariant and contravariant quantities more as we go along. Right now, we have assumed that we have a position vector. Let us take a closer look at this.

We have made one assumption so far that the basis vector is global. We used the term global basis in the beginning of this section. What do we mean by a **global basis**? We want the basis to be the same, that is constant, at every point. Such a set of basis vectors is also said to be **homogeneous**. For example, the basis vectors in the standard Cartesian coordinate system do not depend on the $(x, y)$ coordinates of a point. Consider the trapezium in Figure (5.7) We see that the tangent to the $\eta =$ constant coordinate lines change with $\eta$. In general, the basis vectors change from point to point. We do not have a global basis. Also, consider the standard polar coordinate system (see Figure 5.8). The usual symbols for the basis vectors are $\hat{e}_\theta$ and $\hat{e}_r$. Both of these vectors depend on $\theta$. Again, for the familiar and useful polar coordinate system, we do not have a global basis. That is the basis vectors are not constant. They are not homogeneous. In fact, in the case of polar coordinates we have as the position vector at any point $\vec{P} = r\hat{e}_r$. Does the position vector not depend of $\theta$ at all? The fact of the matter is that the $\hat{e}_r$ depends on $\theta$, as the basis is not homogeneous. Fortunately, $\hat{e}_r$ and $\hat{e}_\theta$ depend only on $\theta$. So, we are still able to write $\vec{P} = r\hat{e}_r$.

Another example of a coordinate system with which you are familiar and is used for doing log-log plots is shown in figure 5.9. In this case, the basis vectors seem to be oriented in the same fashion. However, the length of the vector seems to change. It

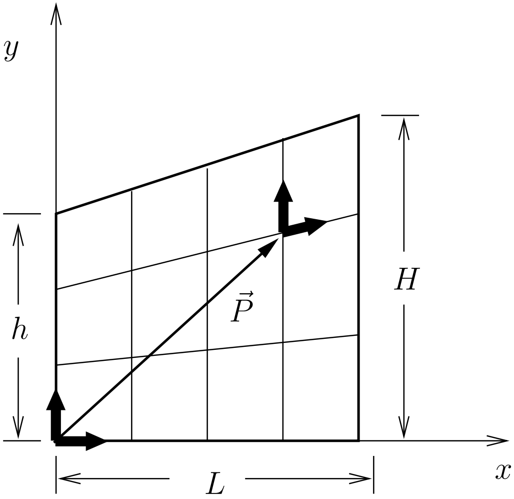

**Figure 5.7. The basis vectors at the origin and the basis vectors at some other point are clearly not the same. The position vector, $\vec{P}$ is geometrically very easy to draw. It cannot be written simply as a linear combination of some basis vector in the coordinate system shown here.**

is clear that the notion of distance between points is an issue here.

Looking at these examples, we realise that we need to spend a little time trying to understand generalised coordinate systems. Let's just consider one coordinate line in some generalised coordinate system. We will see that in three dimensions, it is a space curve. Let us first look at space curves and some of their properties. We are especially interested in the tangent to these

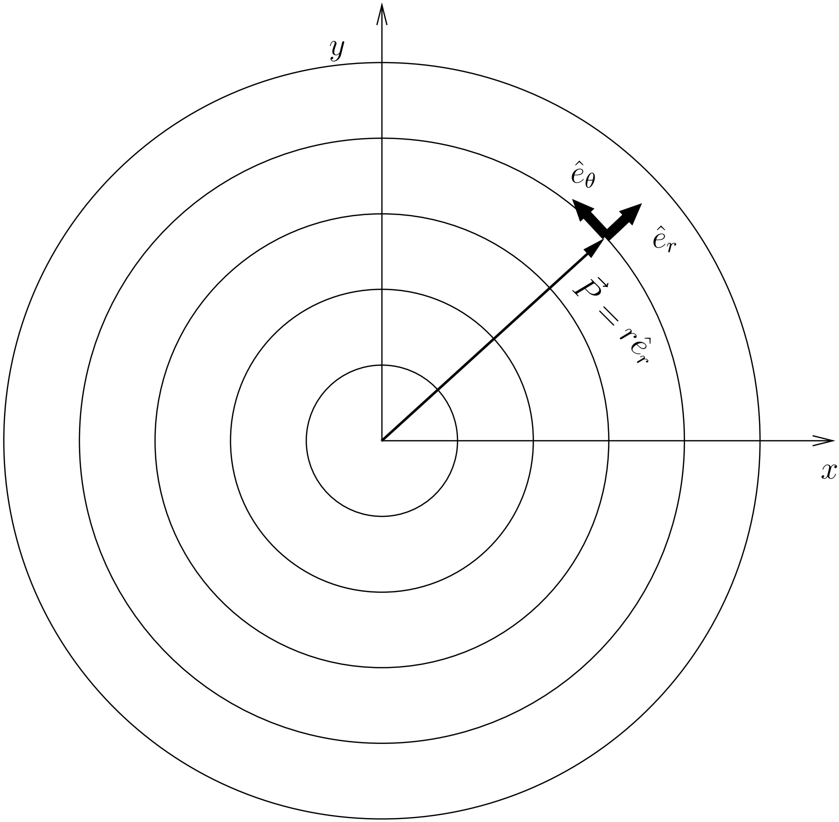

Figure 5.8. The position vector in polar coordinates is given by $\vec{P} = r\hat{e}_r$. At first glance it seems as though there is no $\theta$ dependence. However $\hat{e}_r$ is a function of $\theta$ as is $\hat{e}_\theta$. Only the constant $r$ coordinate lines are shown.

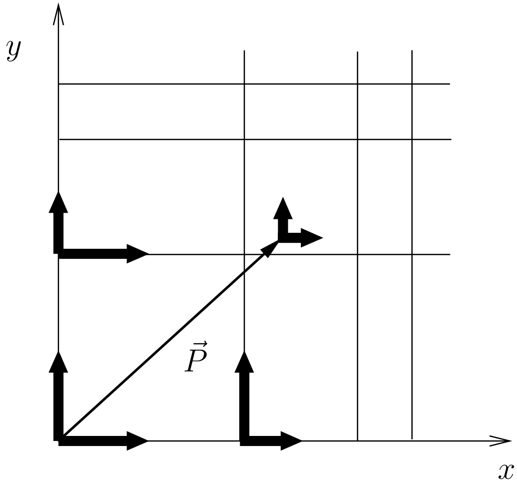

**Figure 5.9. A "log-log" coordinate system. We have a rectangular coordinate system, however the unit vectors are still a function of position making it impossible to write the position vector drawn $\vec{P}$ as a linear combination of the basis vectors.**

curves since the tangent to the coordinate line is a part of our basis.

Consider the coordinate line shown in Figure 5.10. The curve is determined by a function $\vec{\alpha}(\xi^1)$. It is a coordinate line in a coordinate system labelled $\vec{\xi}$ which has three components $(\xi^1, \xi^2, \xi^3)$. The figure shows the coordinate line corresponding to $\xi^2$ =constant, and $\xi^3$ =constant. To belabour the point, it is the $\xi^1$ coordinate line since it is parametrised on $\xi^1$. The tangent to this line is given by

**(5.30)**
$$\vec{\varepsilon}_1 = \frac{d\vec{\alpha}(\xi^1)}{d\xi^1}$$

In fact, for any of the coordinate lines of $\xi^i$ we have for the corresponding $\vec{\alpha}_i$

**(5.31)**
$$\vec{\varepsilon}_i = \frac{d\vec{\alpha}_i}{d\xi^i}, \quad \text{no summation on } i$$

This basis vector is called the covariant basis vector. We note the following

- In equation (5.31), though the subscripts are repeated, there is no summation implied over $i$. The subscript on the $\vec{\alpha}$ is there to conveniently indicate three generic coordinate functions.
- Some new tensor notation convention: ( see equation (5.31) ) the superscript of the derivative on the on the right hand side becomes a subscript on the left.

We consider an example to understand this process better. Let us take a look at polar coordinates in two dimensions $(\xi^1, \xi^2)$. A $\xi^2 = \theta$ coordinate line corresponds to a $\xi^1 = r =$constant line. For the given $r$, the curve is parametrised as

**(5.32)**
$$\vec{\alpha}(\theta) = \vec{\alpha}(\xi^2) = r\cos(\theta)\hat{\imath} + r\sin(\theta)\hat{\jmath} = \xi^1\cos(\xi^2)\hat{e}_1 + \xi^1\sin(\xi^2)\hat{e}_2$$

As was seen earlier, $\hat{e}_1$ and $\hat{e}_2$ are the standard basis vectors in the Cartesian coordinate system. You may be used to calling them $\hat{\imath}$ and $\hat{\jmath}$. The tangent to this curve is $\vec{\varepsilon}_2$.

**(5.33)**
$$\vec{\varepsilon}_2 = -r\sin(\theta)\hat{e}_1 + r\cos(\theta)\hat{e}_2 = -\xi^1\sin(\xi^2)\hat{e}_1 + \xi^1\cos(\xi^2)\hat{e}_2$$

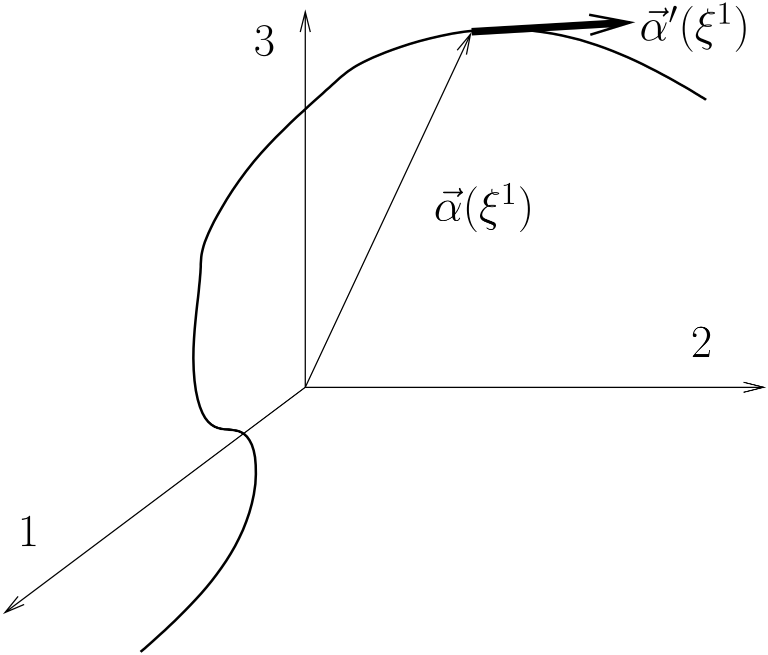

**Figure 5.10. A coordinate line belonging to a three dimensional generalised coordinate system. This line is shown embedded in our usual Cartesian coordinate system. $\vec{\alpha}$ is shown as a function of $\xi^1$ alone as the other two, $\xi^2$ and $\xi^3$ are held constant to obtain this line. The local tangent vector is one of the basis vectors for the generalised coordinates**

We note two points here.

- $\vec{\varepsilon}_2$ is not a unit vector. If you normalise it, you get the "physical" basis vector $\vec{\varepsilon}_\theta$.

- $\hat{e}_1$ and $\hat{e}_2$ are not functions of $\xi^2$. That is the reason why we only have two terms on the right hand side of equation 5.33. Otherwise we would have had more derivative terms due to the application of product rule.

How about the other coordinate line corresponding to $\theta = \xi^2 =$constant? The equation of such a line is given by

**(5.34)**

$$\vec{\alpha}(r) = \vec{\alpha}(\xi^1) = \xi^1 \cos(\xi^2)\hat{e}_1 + \xi^1 \sin(\xi^2)\hat{e}_2, \quad \xi^2 = \text{constant}$$

For constant $\xi^2 = \theta$, this will correspond to a radial line. The tangent vector to this line is given by

$$\vec{\varepsilon}_1 = \frac{\partial \vec{\alpha}}{\partial \xi^1} = \cos(\xi^2)\hat{e}_1 + \sin(\xi^2)\hat{e}_2 \tag{5.35}$$

This in fact turns out to be a unit vector and is the same as $\vec{\varepsilon}_r$.

We can learn something from the study of the polar coordinate system. Why does $\vec{\varepsilon}_2$ depend on $\xi^1$? $\xi^2$ is the angle measured from the $x$-axis. The angle $\xi^2$ is measured in radians which is the arc length at some radius that subtends $\xi^2$ at the centre nondimensionalised by that radius. Naturally, when using $\xi^2$ as a coordinate, the corresponding arc length depends on $\xi^1$.

Let's pause and take stock of what we have and where we are. We have seen that there are coordinate systems where the basis vectors are not homogeneous. So, just writing a relation like equation (5.21), $\vec{V} = v^i \hat{e}_i$, for a position vector $\vec{V}$ may not be possible. We will start dealing only with differentials. A differential element $\boldsymbol{PQ}$ is shown in the Figure 5.11. It is represented in the $X$ coordinate system as $d\vec{x} = dx^i \hat{e}_i$. The $\hat{e}_i$ are the basis vectors in this coordinate system. We can transform from the $X$ coordinates to the $\Xi$ coordinates where the

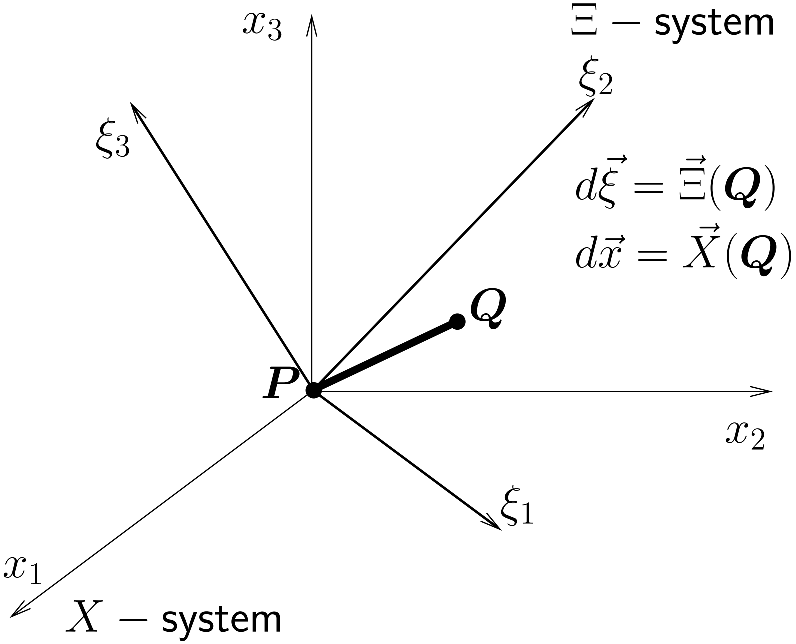

**Figure 5.11. Figure 5.5 is redrawn and zoomed. The origin of our Cartesian coordinate system is translated to the point $\boldsymbol{P}$. The differential element $\boldsymbol{PQ}$ is now represented in terms of the translated coordinate system and a similar system of the generalised coordinates.**

basis vectors are $\vec{\varepsilon}_i$. The differential $\boldsymbol{PQ}$ can be written in the $\Xi$ coordinates system as $d\vec{\xi} = d\xi^i\vec{\varepsilon}_i$. How are the two representations for the given differential element at a given point related? Clearly, the length of the element should not depend on our choice of the coordinate system. Or, put another way, if two people choose two different coordinate systems, the length of this particular element should work out to be the same. As we had done earlier, here are the two equations that relate the Cartesian coordinates $x^i$ to the generalised coordinates $\xi^i$.

**(5.36)** $$x^i = x^i(\xi^1, \xi^2, \xi^3)$$

and

**(5.37)** $$\xi^i = \xi^i(x^1, x^2, x^3)$$

In the Cartesian coordinate system the length $ds$ is given by

**(5.38)** $$(ds)^2 = d\vec{x} \cdot d\vec{x} = dx^i \hat{e}_i \cdot dx^j \hat{e}_j = dx^i dx^j \hat{e}_i \cdot \hat{e}_j$$

Remember that $\hat{e}_i$ are the basis vectors of a Cartesian coordinate system and are orthogonal to each other. Consequently, we can define a useful entity called the Kronecker delta as

**(5.39)** $$\delta_{ij} = \hat{e}_i \cdot \hat{e}_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

With this new notation we can write

**(5.40)** $$(ds)^2 = d\vec{x} \cdot d\vec{x} = dx^i dx^j \delta_{ij} = dx^i dx_i = \sum_i (dx^i)^2$$

Following the convention we have used so far (without actually mentioning it) we see that

**(5.41)** $$dx^j \delta_{ij} = dx_i$$

That is, $j$ is a dummy index and disappears leaving $i$ which is a subscript. For the first time we have seen a contravariant quantity converted to a covariant quantity. If you think of matrix algebra for a minute, you will see that $\delta_{ij}$ is like an identity matrix. The components $dx^i$ are the same as the components $dx_i$ in a Cartesian coordinate system. Hence, equation (5.40) can be written as

**(5.42)**
$$(ds)^2 = dx^i dx_i = \sum_i (dx_i)^2$$

The length of the element is invariant with transformation meaning the choice of our coordinates should not change the length of the element. A change to the $\Xi$ coordinates should give us the same length for the differential element $\boldsymbol{PQ}$. The length in the $\Xi$ coordinates is given by

**(5.43)**
$$(ds)^2 = d\vec{\xi}\cdot d\vec{\xi} = d\xi^i \vec{\varepsilon}_i \cdot d\xi^j \vec{\varepsilon}_j = d\xi^i d\xi^j \vec{\varepsilon}_i \cdot \vec{\varepsilon}_j = d\xi^i d\xi^j g_{ij} = d\xi^i d\xi_i$$

$g_{ij}$ is called the metric. Following equation (5.41), we have defined $d\xi_i = g_{ij} d\xi^j$. Why did we get $g_{ij}$ instead of $\delta_{ij}$? We have seen in the case of the trapezium that the basis vectors need not be orthogonal to each other since the coordinate lines are not orthogonal to each other. So, the dot product of the basis vectors $\vec{\varepsilon}_i$ and $\vec{\varepsilon}_j$ gives us a $g_{ij}$ with non-zero off-diagonal terms. It is still symmetric, though. In this case, unlike the Cartesian situation, $d\xi^i$ is different from $d\xi_i$.

We can define another set of basis vectors which are orthogonal to the covariant set as follows

**(5.44)**
$$\vec{\varepsilon}_i \cdot \vec{\varepsilon}^j = \delta_i^j$$

where,

**(5.45)**
$$\delta_i^j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

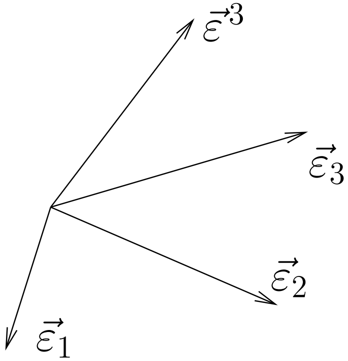

**Figure 5.12. The covariant basis vectors $\vec{\varepsilon}_1$, $\vec{\varepsilon}_2$, and $\vec{\varepsilon}_3$ are shown. In general they may not be orthogonal to each other. $\vec{\varepsilon}^3$ is also shown. It is orthogonal to $\vec{\varepsilon}_1$ and $\vec{\varepsilon}_2$ and $\vec{\varepsilon}_3 \cdot \vec{\varepsilon}^3 = 1$**

This new basis, $\vec{\varepsilon}^i$, is called the contravariant basis or a dual basis. This is demonstrated graphically in figure 5.12. This basis can be used to define a metric

**(5.46)** $$g^{ij} = \vec{\varepsilon}^i \cdot \vec{\varepsilon}^j$$

Now, is the definition given for $d\xi_i$ consistent with this definition of the contravariant basis? Is $d\vec{\xi} = d\xi_i \vec{\varepsilon}^i$? That is, if we take the dot product of a vector with a basis vector, do we get the corresponding component? We have,

**(5.47)** $$d\vec{\xi} = d\xi_i \vec{\varepsilon}^i \Rightarrow \quad d\vec{\xi} \cdot \vec{\varepsilon}^j = d\xi_i \underbrace{\vec{\varepsilon}^i \cdot \vec{\varepsilon}^j}_{g^{ij}} = d\xi^j,$$

and

**(5.48)** $$d\vec{\xi} = d\xi_i \vec{\varepsilon}^i \Rightarrow \quad d\vec{\xi} \cdot \vec{\varepsilon}_j = d\xi_i \vec{\varepsilon}^i \cdot \vec{\varepsilon}_j = d\xi_j,$$

and

**(5.49)**
$$d\vec{\xi} = d\xi^i \vec{\varepsilon}_i \Rightarrow \quad d\vec{\xi} \cdot \vec{\varepsilon}_j = d\xi^i \vec{\varepsilon}_i \cdot \vec{\varepsilon}_j = d\xi_j,$$

and finally,

**(5.50)**
$$d\vec{\xi} = d\xi^i \vec{\varepsilon}_i \Rightarrow \quad d\vec{\xi} \cdot \vec{\varepsilon}^j = d\xi^i \vec{\varepsilon}_i \cdot \vec{\varepsilon}^j = d\xi^j,$$

So, to get the contravariant components of a tensor, dot it with the contravariant basis vectors. Likewise, to get the covariant components of a tensor, dot it with the covariant basis vectors. The effect of $g_{ij}$ on a contravariant term is to lower the index or convert it to a covariant term. Similarly, the effect of $g^{ij}$ on a covariant term is to raise the index or convert it to a contravariant term. So, what is $g_{ij}g^{jk}$?

**(5.51)**
$$g_{ij}g^{jk} = g_i^k = \vec{\varepsilon}_i \cdot \vec{\varepsilon}^k = \delta_i^k$$

The metric tensors are inverses of each other.

At this point you really can protest: "Wait a minute, where is this going? "Fascinating" as it is, how is it relevant to CFD?" Look at the trapezium in Figure 5.4. Imagine that this trapezium represents a channel through which some fluid, like water, can flow. The top and bottom of the trapezium are solid walls. If we were solving for the potential flow through a channel with the top and bottom of the trapezium being solid walls, this tells us, we need to apply the boundary condition $\partial\phi/\partial n = 0$, where $n$ is measured along a line that is perpendicular to the surface. Look at the top of the trapezium. A zoomed view is shown in Figure 5.13. Your coordinate line is not normal to the top surface. How do we get the derivative along the normal. You can find the derivatives along $\vec{\varepsilon}_1$ and $\vec{\varepsilon}_2$ and use Taylor's series in two dimensions to get the normal derivative. You will find that

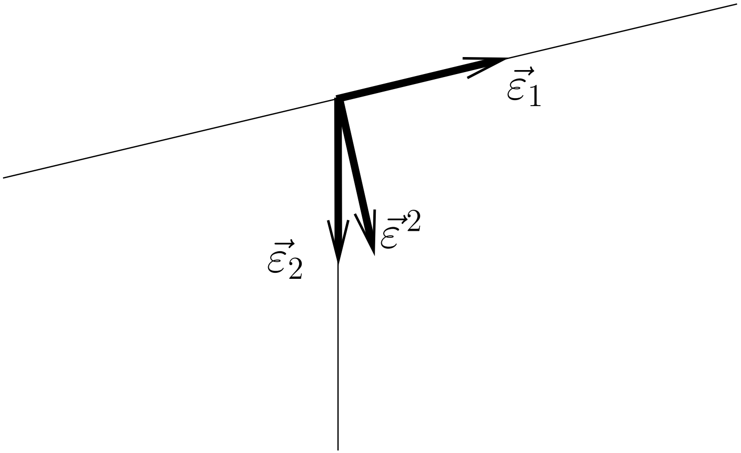

**Figure 5.13. A zoomed view of the non-Cartesian mesh in a Trapezoidal domain shown in Figure 5.4. The two covariant basis vectors and one contravariant basis vector are shown.**

you are just reinventing everything we have done so far. What you want is the contravariant basis vector and not the covariant basis vector. Why? This is because the covariant basis vector is along the coordinate line and the contravariant one is perpendicular to it. The top of the trapezium is a coordinate line. The contravariant basis vector is perpendicular to it, which is what we want. We do need this stuff, so let's soldier on. First, an assignment.

**Assignment 5.1**

(1) Expand the following using the summation convention assuming that we are working in three dimensions.
(a) $a^i b^j \delta_{ij}$, (b) $\delta^j_j$, (c) $\delta^j_i \delta^i_j$, (d) $\delta^i_i \delta^j_j$

(2) Repeat the above problem assuming we are dealing with tensors in two space dimensions.

(3) Find the covariant and contravariant bases vectors and the corresponding metric tensors for the following coordinate systems. $x^i$ are the Cartesian coordinates.

(a) Cylindrical coordinates. $\xi^1 = r$, $\xi^2 = \theta$, and $\xi^3 = z$ in conventional notation.
$x^1 = \xi^1 \cos \xi^2$, $x^2 = \xi^1 \sin \xi^2$, and $x^3 = \xi^3$.

(b) Spherical coordinates. $\xi^1 = R$, $\xi^2 = \theta$, and $\xi^3 = \phi$ in conventional notation.
$x^1 = \xi^1 \sin \xi^2 \cos \xi^3$, $x^2 = \xi^1 \sin \xi^2 \sin \xi^3$, and $x^3 = \xi^1 \cos \xi^2$

(c) Parabolic cylindrical coordinates.
$x^1 = \frac{1}{2}\left\{(\xi^1)^2 - (\xi^2)^2\right\}$, $x^2 = \xi^1 \xi^2$, and $x^3 = \xi^3$.

(4) Compute the covariant and contravariant velocity components in the above coordinate systems.

You have seen in multivariate calculus that given a smooth function $\phi$, in a region of interest, we can find the differential $d\phi$ as

**(5.52)**
$$d\phi = \frac{\partial \phi}{\partial \xi^i} d\xi^i$$

Now, we also know that this is a directional derivative and can be written as

**(5.53)**
$$d\phi = \nabla\phi \cdot d\vec{\xi} = \frac{\partial \phi}{\partial \xi^i} d\xi^i$$

where,

**(5.54)**
$$\nabla = \vec{\varepsilon}^{\,j} \frac{\partial}{\partial \xi^j}, \quad d\vec{\xi} = \vec{\varepsilon}_i d\xi^i$$

We managed to define the gradient operator $\nabla$. What happens when we take the gradient of a vector? How about the divergence? We first write the gradients of a scalar function and a vector function as

**(5.55)**
$$\nabla\phi \;=\; \vec{\varepsilon}^{\,j} \frac{\partial \phi}{\partial \xi^j}$$

**(5.56)**
$$\nabla\vec{V} \;=\; \vec{\varepsilon}^{\,j} \frac{\partial \vec{V}}{\partial \xi^j}$$

If we look carefully at the two equation above, we see that equation (5.56) is different. It involves, due to the use of product rule, the derivatives of the basis vectors. In fact, equation (5.56) can written as

**(5.57)**
$$\nabla\vec{V} = \vec{\varepsilon}^{\,j} \frac{\partial \vec{V}}{\partial \xi^j} = \vec{\varepsilon}^{\,j} \left\{ \frac{\partial v^i}{\partial \xi^j} \vec{\varepsilon}_i + v^i \frac{\partial \vec{\varepsilon}_i}{\partial \xi^j} \right\}$$

So, what is the nature of the derivative of the basis vector? For one thing, from the definition of the covariant basis in equation (5.31) we have

**(5.58)**
$$\frac{\partial \vec{\varepsilon}_i}{\partial \xi^{\,j}} = \frac{\partial^2 \vec{\alpha}}{\partial \xi^j \partial \xi^i} = \frac{\partial \vec{\varepsilon}_j}{\partial \xi^i}$$

We have dispensed with the subscript on $\vec{\alpha}$ so as not to create more confusion. We will use the correct $\vec{\alpha}$ corresponding to the coordinate line. We can see from equation (5.58) that its component representation is going to be symmetric in the two indices $i$ and $j$. As we have already seen in equation (5.47), to find the contravariant components of this entity we can dot it with $\vec{\varepsilon}^k$ to get

**(5.59)**
$$\left\{ \begin{matrix} k \\ ij \end{matrix} \right\} = \vec{\varepsilon}^k \cdot \frac{\partial \vec{\varepsilon}_i}{\partial \xi^j}$$

The set of $\left\{ \begin{smallmatrix} k \\ ij \end{smallmatrix} \right\}$ are called a Christoffel symbols of the second kind. We took the dot product with $\vec{\varepsilon}^k$ so that equation (5.57) can be rewritten as

**(5.60)**
$$\nabla \vec{V} = \vec{\varepsilon}^j \left\{ \frac{\partial v^i}{\partial \xi^j} \vec{\varepsilon}_i + v^i \left\{ \begin{matrix} k \\ ij \end{matrix} \right\} \vec{\varepsilon}_k \right\}$$

Since $i$ and $k$ are dummy indices (meaning we are going to sum over their values) we swap them for a more convenient expression

**(5.61)**
$$\nabla \vec{V} = \vec{\varepsilon}^j \left\{ \frac{\partial v^i}{\partial \xi^j} \vec{\varepsilon}_i + v^k \left\{ \begin{matrix} i \\ kj \end{matrix} \right\} \vec{\varepsilon}_i \right\}$$

This allows us to write

**(5.62)**
$$\frac{\partial \vec{V}}{\partial \xi^j} = \left\{ \frac{\partial v^i}{\partial \xi^j} + v^k \left\{ \begin{matrix} i \\ kj \end{matrix} \right\} \right\} \vec{\varepsilon}_i$$

In pure component form this is written as

**(5.63)**
$$v^i_{;j} = \frac{\partial v^i}{\partial \xi^j} + v^k \left\{ \begin{matrix} i \\ kj \end{matrix} \right\}$$

This is called the covariant derivative of the contravariant vector $v^i$. Staying with our compact notation, the covariant derivative is indicated by the semi-colon in the subscript. This is so that we do not confuse it with the plain derivative $\partial v^i/\partial\xi^j$.

So, if we have Christoffel symbols of the second kind do we have any other kind? Yes, there are Christoffel symbols of the first kind. They are written in a compact form as $[ij,k]$ and it are given by

**(5.64)**
$$[ij,k] = \begin{Bmatrix} l \\ ij \end{Bmatrix} g_{lk} = \frac{\partial\vec{\varepsilon}_i}{\partial\xi^j}\cdot\vec{\varepsilon}^{\,l} g_{lk} = \frac{\partial\vec{\varepsilon}_i}{\partial\xi^j}\cdot\vec{\varepsilon}_k$$

The Christoffel symbols of the first kind can be directly obtained as

**(5.65)**
$$[ij,k] = \frac{1}{2}\left(\frac{\partial g_{jk}}{\partial\xi^i} + \frac{\partial g_{ki}}{\partial\xi^j} - \frac{\partial g_{ij}}{\partial\xi^k}\right)$$

This can be verified by substituting for the definition of the metric tensor. The peculiar notation with brackets and braces is used for the Christoffel symbols (and they are called symbols) because, it turns out that they are not tensors. That is, though they have indices, they do not transform the way tensors do when going from one coordinate system to another. We are not going to show this here. However, we should not be surprised that they are not tensors as the Christoffel symbols encapsulate the relationship of the two coordinate systems and would necessarily depend on the coordinates.

The divergence of $\vec{V}$ is defined as the trace of the gradient of $\vec{V}$. That is

**(5.66)**
$$\text{div}\vec{V} = \vec{\varepsilon}^{\,j}\cdot\left\{\frac{\partial v^i}{\partial\xi^j}\vec{\varepsilon}_i + v^k \begin{Bmatrix} i \\ kj \end{Bmatrix}\vec{\varepsilon}_i\right\}$$

**Assignment 5.2**

For the coordinate systems given in assignment 5.1,

(1) Find the Christoffel symbols of the first and second kind.
(2) Find the expression for the gradient of a scalar potential.
(3) Find the gradient of the velocity vector.
(4) Find the divergence of the velocity vector.
(5) Find the divergence of the gradient of the scalar potential that you just found.

In the case of the velocity potential $\vec{V} = \nabla\phi$ we get,

$$\vec{V} = \vec{\varepsilon}^{\,j}\frac{\partial\phi}{\partial\xi^j} = \vec{\varepsilon}_k g^{kj}\frac{\partial\phi}{\partial\xi^j} = \vec{\varepsilon}_k g^{kj} v_j = v^k\vec{\varepsilon}_k \tag{5.67}$$

If we now take the divergence of this vector using equation (5.66) we get

$$\begin{aligned}\nabla^2\phi &= \vec{\varepsilon}^{\,j}\cdot\left\{\frac{\partial v^i}{\partial\xi^j}\vec{\varepsilon}_i + v^k\left\{\begin{matrix} i \\ kj\end{matrix}\right\}\vec{\varepsilon}_i\right\} \\ &= \vec{\varepsilon}^{\,j}\cdot\left\{\frac{\partial}{\partial\xi^j}\left(g^{il}\frac{\partial\phi}{\partial\xi^l}\right)\vec{\varepsilon}_i + v^k\left\{\begin{matrix} i \\ kj\end{matrix}\right\}\vec{\varepsilon}_i\right\}\end{aligned} \tag{5.68}$$

Completing the dot product we get

$$\nabla^2\phi = \left\{\frac{\partial}{\partial\xi^i}\left(g^{il}\frac{\partial\phi}{\partial\xi^l}\right) + v^k\left\{\begin{matrix} i \\ ki\end{matrix}\right\}\right\} \tag{5.69}$$

Substituting for $v^k$ from equation (5.67) we get

$$\textbf{(5.70)} \qquad \nabla^2\phi = \left\{ \frac{\partial}{\partial\xi^i}\left(g^{il}\frac{\partial\phi}{\partial\xi^l}\right) + g^{kl}\frac{\partial\phi}{\partial\xi^l}\left\{ \begin{matrix} i \\ ki \end{matrix} \right\} \right\}$$

This much tensor calculus will suffice. A more in depth study can be made using the numerous books that are available on the topic [**You93**], [**SS82**].

## Equations of Fluid Motion

We have seen enough tensor calculus so that if we derive the governing equations in some generic coordinate system, we can always transform the resulting equations into any other coordinate system. In fact, as far as possible, we will derive the equations in vector form so that we can pick the component form that we feel is appropriate for us. We can conveniently use the Cartesian coordinate system for the derivation with out loss of generality.

We will first derive the equations of motion in integral form. We will do this in a general setting. Let us consider some fluid property $\mathcal{Q}$, whose property density is given by $Q$. In terms of thermodynamics, $\mathcal{Q}$ would be an extensive property and $Q$ would be the associated intensive property. For example, consider a situation in which we have added some ink to flowing water. At any given time, the mass of ink in a small elemental region of interest may be $dm_{\mathsf{ink}}$. If the volume of the elemental region is $d\sigma$, then these two measures defined on that region are related through the ink density as

$$\textbf{(5.71)} \qquad dm_{\mathsf{ink}} = \frac{dm_{\mathsf{ink}}}{d\sigma}d\sigma = \rho_{\mathsf{ink}}d\sigma$$

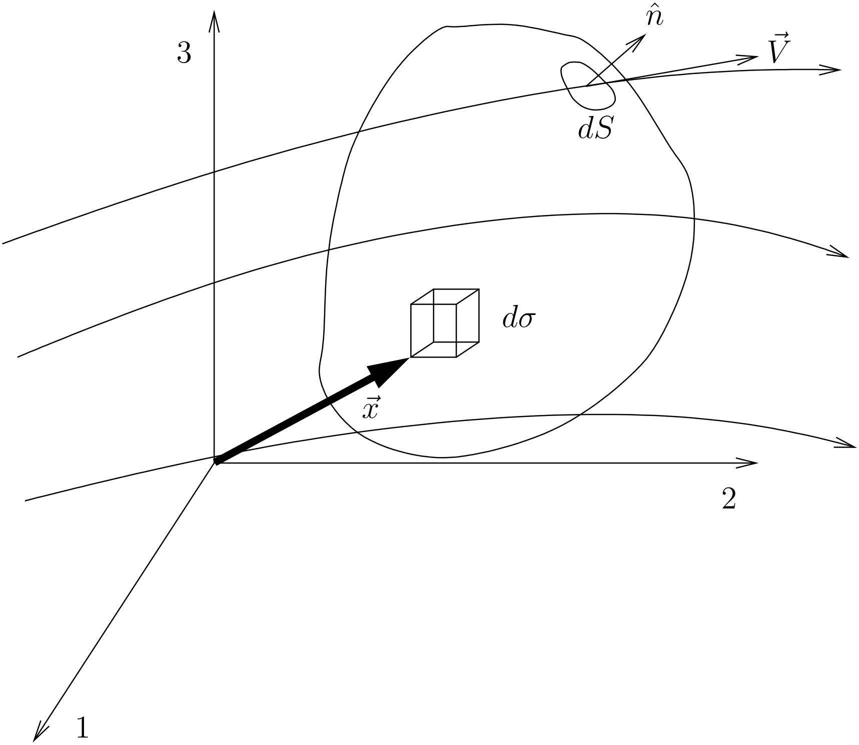

**Figure 5.14.** An arbitrary control volume chosen in some fluid flow. An elemental area on the controls surface $dS$ and and elemental volume $d\sigma$ within the control volume are also shown. Note that in most coordinate systems we may not be able to indicate a position vector $\vec{x}$.

We would like to write out the balance laws for a general property, $\mathcal{Q}$. We **arbitrarily** pick a control volume. One such volume is indicated in the Figure 5.14. For the sake of simplicity, we pick a control volume that does not change in time. This control volume occupies a region of volume $\sigma$. This control volume has a surface area $S$. It is located as shown in the figure and is immersed in a flow field. Within this control volume, at an arbitrary point $\vec{x}$, we pick a small elemental region with volume $d\sigma$. From equation (5.71), the amount of the property of interest at time $t$, $d\mathcal{Q}(\vec{x},t)$, in the elemental control volume is $Q(\vec{x},t)d\sigma$. Then the total quantity contained in our control volume at any instant is

**(5.72)**
$$\mathcal{Q}_\sigma(t) = \int_\sigma Q(\vec{x},t)d\sigma$$

The time rate of change of this quantity is

**(5.73)**
$$\frac{d\mathcal{Q}_\sigma}{dt} = \frac{d}{dt}\int_\sigma Q(\vec{x},t)d\sigma$$

Then we ask ourselves the question, why is there a rate of change? There is change because the property $\mathcal{Q}$ is carried / transported in and out of the control volume by the fluid. It is also possible, based on the nature of $\mathcal{Q}$, that it is somehow created or destroyed in the control volume. There may be many mechanisms by which $\mathcal{Q}$ can be changed into some other property. Let us now look at the transport of $\mathcal{Q}$ by the flow.

At any arbitrary point on the surface of the control volume that we have shown in Figure 5.14, we can determine the unit surface normal vector. We can pick a small elemental area $dS$ at that point. The surface normal is perpendicular to this element. By convention, we choose to pick a surface normal that points out of the control volume. The rate at which our property $\mathcal{Q}$ flows out through this elemental area is given by $Q\vec{V} \cdot \hat{n} dS$. The total efflux (outflow) from the control volume is

**(5.74)**
$$\int_S Q\vec{V} \cdot \hat{n} dS$$

Since this is a net efflux, it would cause a decrease in the amount of $\mathcal{Q}$ contained in the control volume. So, our balance law can be written as

**(5.75)**
$$\frac{d}{dt}\int_\sigma Q d\sigma = -\int_S Q\vec{V} \cdot \hat{n} dS + \text{ any other mechanism to produce } \mathcal{Q}$$

Before going on we will make the following observation. Though the control volume can be picked arbitrarily, we will make sure that it is smooth enough to have surface normals almost everywhere. Almost everywhere? If you think of a cube, we cannot define surface normals at the edges and corners. We can break up the surface integral in equation (5.75) into the sum of six integrals, one for each face of the cube.

**Conservation of Mass.** Let us look at an example. If the property we were considering was mass, $\mathcal{Q}_\sigma(t)$ would be the mass of fluid in our control volume at any given time. The corresponding $Q$ would be mass density which we routinely refer to as the density, $\rho$. Ignoring mechanisms to create and destroy or otherwise modify mass, we see that the production terms disappear, leaving only the first term on the right hand side of equation (5.75). This gives us the equation for **balance of mass** as

$$\frac{d}{dt}\int_{\sigma}\rho d\sigma = -\int_{S}\rho\vec{V}\cdot\hat{n}dS \tag{5.76}$$

This equation is also called the **conservation of mass** equation.

**Conservation of Linear Momentum.** On the other hand, if we consider the property $\mathcal{Q}$ to be momentum, the property density $Q$ turns out to be $\rho\vec{V}$, which is the momentum density. In this case, we know that the total momentum in the control volume can also be changed by applying forces. For the sake of this discussion, forces come in two flavours. There are those that correspond to action across a distance, these forces are often called body forces. The others that depend on proximity are called surface forces[1]. We can write our equation of **balance of linear momentum** as

$$\frac{d}{dt}\int_{\sigma}\rho\vec{V}d\sigma = -\int_{S}\rho\vec{V}\vec{V}\cdot\hat{n}dS + \int_{\sigma}\vec{f}d\sigma + \int_{S}\vec{T}dS \tag{5.77}$$

Here, $\vec{f}(\vec{x})$ is the body force per unit volume at the point $\vec{x}$ within the control volume. $\vec{T}(\vec{x})$ is the traction force per unit area (often called traction force or just traction) acting at some point $\vec{x}$ on the control surface. If we are willing or able to ignore the body force, we are left with the traction force to be handled. From fluid mechanics, you would have seen that we can associate at a point, a linear transformation called the

[1] As with everything that we do in physics, what we mean by this really depends on length scales. We have assumed that we are dealing with a continuum and that implicitly has a bifurcation of the length scales built into it.

stress tensor, which relates the normal to a surface element to the traction force on that element. That is

$$\vec{T} = \boldsymbol{\tau} \cdot \hat{n} \tag{5.78}$$

where, $\vec{T} = T_i \vec{\varepsilon}^{\,i}$, $\boldsymbol{\tau} = \tau_{ij} \vec{\varepsilon}^{\,i} \vec{\varepsilon}^{\,j}$, and $\hat{n} = n_k \vec{\varepsilon}^{\,k}$. This gives us the Cauchy equation in component form as

$$T_i = \tau_{ij} n^j \tag{5.79}$$

The **momentum balance** equation can be written as

$$\frac{d}{dt} \int_\sigma \rho \vec{V} d\sigma = - \int_S \rho \vec{V} \vec{V} \cdot \hat{n} dS + \int_S \boldsymbol{\tau} \cdot \hat{n} dS \tag{5.80}$$

Combining terms we get

$$\frac{d}{dt} \int_\sigma \rho \vec{V} d\sigma = - \int_S \left\{ \rho \vec{V} \vec{V} - \boldsymbol{\tau} \right\} \cdot \hat{n} dS \tag{5.81}$$

**Conservation of Energy.** Finally, if we consider the total energy as the property of interest so that we write out the balance law for energy. Considering the form of the first two equations, we will define the total energy density as $\rho E_t$, where $E_t$ is the specific total energy defined as

$$E_t = e + \frac{1}{2} \vec{V} \cdot \vec{V}, \tag{5.82}$$

Where $e$ is the specific internal energy defined for a perfect gas as $e = C_v T$. $C_v$ is the specific heat at constant volume and $T$ is the temperature measured on the Kelvin scale. We need to look at the production terms again in equation (5.75). The total energy in our control volume can be changed by

(1) the forces from the earlier discussion doing work on the control volume,

(2) the transfer of energy by the process of heat through radiation and conduction,

(3) the apparent creation of energy through exo-thermic or endo-thermic chemical reactions,

(4) and finally, of course, the transportation of energy across the control surface by the fluid.

We will ignore radiation and chemical reactions here. This results in an equation for the balance of energy as

**(5.83)**

$$\frac{d}{dt}\int_\sigma \rho E_t d\sigma = -\int_S \rho E_t \vec{V}\cdot\hat{n}dS + \int_\sigma \vec{f}\cdot\vec{V}d\sigma + \int_S \vec{T}\cdot\vec{V}dS - \int_S \vec{q}\cdot\hat{n}dS$$

Here, $\vec{q}$ is the term quantifying heat. Again, if we are in a position to ignore body forces we get

**(5.84)**

$$\frac{d}{dt}\int_\sigma \rho E_t d\sigma = -\int_S \rho E_t \vec{V}\cdot\hat{n}dS + \int_S \vec{V}\cdot\boldsymbol{\tau}\cdot\hat{n}dS - \int_S \vec{q}\cdot\hat{n}dS$$

which we conveniently rewrite incorporating the other balance laws as

**(5.85)**
$$\frac{d}{dt}\int_\sigma Q d\sigma = -\int_S \vec{\mathcal{F}}\cdot\hat{n}dS$$

where we have

**(5.86)**
$$Q = \begin{Bmatrix} \rho \\ \rho\vec{V} \\ \rho E_t \end{Bmatrix}, \quad \vec{\mathcal{F}} = \begin{Bmatrix} \rho\vec{V} \\ \rho\vec{V}\vec{V} - \boldsymbol{\tau} \\ (\rho E_t)\vec{V} - \boldsymbol{\tau}\cdot\vec{V} + \vec{q} \end{Bmatrix}$$

where, $\boldsymbol{\tau} \cdot \vec{V}$ is the rate at which the traction force does work on the control volume. This, gives us a consolidated statement for the balance (conservation) of mass, linear momentum, and energy. The great thing about this equation is that it can be cast in any three dimensional coordinate system to get the component form. It is written in a coordinate free fashion. Though, it is good to admire, we finally need to solve a specific problem, so we pick a coordinate system convenient for the solution of our problem and express these equations in that coordinate system. There is another problem. As things stand, there is some element of ambiguity in the dot products of the form $(\boldsymbol{\tau} \cdot \vec{V}) \cdot \hat{n}$. These ambiguities are best resolved by writing the expression in terms of components.

**(5.87)** $\vec{T} \cdot \vec{V} = T_i \vec{\varepsilon}^{\,i} \cdot \vec{\varepsilon}_l V^l = \tau_{ij} \vec{\varepsilon}^{\,i} (\vec{\varepsilon}^{\,j} \cdot \vec{\varepsilon}^{\,k}) n_k \cdot \vec{\varepsilon}_l V^l = \tau_{ij} n^j V^i$

The differential form of equation (5.85) can be obtained by applying the theorem of Gauss to the right hand side of the equation and converting the surface integral to a volume integral.

**(5.88)**
$$\int_\sigma \left\{ \frac{\partial Q}{\partial t} + \text{div}\vec{\mathcal{F}} \right\} d\sigma = 0$$

The control volume is chosen arbitrarily. As a consequence, the integral needs to be zero for any $\sigma$ over which we integrate. This is possible only if

**(5.89)**
$$\frac{\partial Q}{\partial t} + \text{div}\vec{\mathcal{F}} = 0$$

The form of equation (5.85) is quite general. We could add, as required, more terms to the $\vec{\mathcal{F}}$ on the right hand side. We could also add as many equations as required. If you have other properties that need to be tracked, the corresponding equations can be incorporated. However, for our purpose, these equations are quite general. We will start with a little specialisation and simplification.

We now decompose the stress tensor $\boldsymbol{\tau}$ into a spherical part and a deviatoric part. The spherical part we will assume is the same as the pressure we have in the equation of state. The deviatoric part (or the deviation from the sphere) will show up due to viscous effects. So, $\boldsymbol{\tau}$ can be written as

$$\boldsymbol{\tau} = -p\mathbf{1} + \boldsymbol{\sigma} \tag{5.90}$$

$\mathbf{1}$ is the unit tensor and $\boldsymbol{\sigma}$ is the deviatoric part. Do not confuse $\boldsymbol{\sigma}$ a tensor with the control volume $\sigma$. Through thermodynamics, we have an equation of state / constitutive model for $p$. Typically, we use something like $p = \rho RT$, where $T$ is the temperature in Kelvin and $R$ is the gas constant. We need to get a similar equation of state / constitutive model for $\boldsymbol{\sigma}$. Assuming the fluid is a Navier-Stokes fluid, that is the fluid is Newtonian, isotropic and Stokes hypothesis holds we get

$$\boldsymbol{\sigma} = -\frac{2}{3}\mu\,\mathrm{tr}\boldsymbol{D} + 2\mu\boldsymbol{D}, \quad \text{where} \tag{5.91}$$

$$\boldsymbol{D} = \frac{1}{2}(\boldsymbol{L} + \boldsymbol{L}^T), \quad \text{and} \tag{5.92}$$

$$\boldsymbol{L} = \nabla\vec{V} \tag{5.93}$$

where $\mu$ is the coefficient of viscosity and $\mathrm{tr}\boldsymbol{D}$ is the trace of $\boldsymbol{D}$, which is the sum of the diagonals of the matrix representation of the tensor. $\boldsymbol{L}^T$ is the transpose of $\boldsymbol{L}$. Really, $\boldsymbol{D}$ is the symmetric part of the the gradient of $\vec{V}$ and is called the deformation rate. Equation (5.89) with $\boldsymbol{\tau}$ written in this fashion is called the Navier-Stokes equation. Since, we are right now looking at inviscid flow, we can ignore the viscous terms. So, for the Euler's equation we have

$$\vec{T} = -p\mathbf{1} \cdot \hat{n} \tag{5.94}$$

where, $\mathbf{1}$ is the unit tensor. The Euler's momentum conservation equation can be written as

$$\frac{d}{dt}\int_\sigma \rho\vec{V}d\sigma = -\int_S \rho\vec{V}\vec{V} \cdot \hat{n}dS - \int_S p\mathbf{1} \cdot \hat{n}dS \tag{5.95}$$

Combining terms we get

$$\frac{d}{dt}\int_\sigma \rho\vec{V}d\sigma = -\int_S \left\{\rho\vec{V}\vec{V} + p\mathbf{1}\right\} \cdot \hat{n}dS \tag{5.96}$$

which we conveniently rewrite as

$$\frac{d}{dt}\int_\sigma Qd\sigma = -\int_S \vec{\mathcal{F}} \cdot \hat{n}dS \tag{5.97}$$

where we have

$$Q = \left\{\begin{matrix} \rho \\ \rho\vec{V} \\ \rho E_t \end{matrix}\right\}, \quad \vec{\mathcal{F}} = \left\{\begin{matrix} \rho\vec{V} \\ \rho\vec{V}\vec{V} + p\mathbf{1} \\ (\rho E_t + p)\vec{V} \end{matrix}\right\} \tag{5.98}$$

giving us a consolidated statement for the conservation (or balance) of mass, linear momentum, and energy. These equations are collectively referred to as the Euler's equation. There are, as is usual, a set of auxiliary equations to complement these equations. The constitutive model given by the equation of state is

**(5.99)**
$$p = \rho R T$$

and

**(5.100)**
$$E_t = e + \frac{\vec{V} \cdot \vec{V}}{2}$$

**(5.101)**
$$e = C_v T$$

With these equations included, we have a closed set of equations that we should be able to solve. The equations are in integral form. We can employ the theorem of Gauss on the surface integral in equation (5.97) and convert it to a volume integral like so

**(5.102)**
$$\frac{d}{dt}\int_\sigma Q d\sigma = -\int_S \vec{\mathcal{F}} \cdot \hat{n} dS = -\int_\sigma \mathsf{div}\vec{\mathcal{F}} d\sigma$$

This gives us the following equation

**(5.103)**
$$\int_\sigma \left( \frac{\partial Q}{\partial t} + \mathsf{div}\vec{\mathcal{F}} \right) d\sigma = 0$$

which is valid for all possible control volumes on which we have surface normals and can perform the necessary integration. Remember, this "particular" $\sigma$ was chosen arbitrarily. We conclude that the integral can be zero for any $\sigma$ only if the integrand is zero. The differential form of the Euler's equation can be written as

**(5.104)**
$$\frac{\partial Q}{\partial t} + \mathsf{div}\vec{\mathcal{F}} = 0$$

If we use normal convention to write $\vec{\mathcal{F}}$ in Cartesian coordinates as

**(5.105)**
$$\vec{\mathcal{F}} = E\hat{\imath} + F\hat{\jmath} + G\hat{k}$$

our governing equation in Cartesian coordinates then becomes

**(5.106)**
$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = 0$$

Clearly, given any other basis vector, metrics, Christoffel symbols, we can write the governing equations in the corresponding coordinate system.

---

**Assignment 5.3**

(1) Given the concentration of ink at any point in a flow field is given by $c_i$, derive the conservation equation in integral form for ink. The diffusivity of ink is $D_i$.
(2) From the integral from in the first problem, derive the differential form
(3) Specialise the equation for a two-dimensional problem.
(4) Derive the equation in polar coordinates.

---

**Non-dimensional Form of Equations.** So far, in this book, we have not talked of the physical units used. How do the equations depend on physical units that we use. Does the solution depend on the fact that we use millimetres instead of metres? We would like to solve the non-dimensional form of these equations. We will demonstrate the process of obtain the non-dimensional form of the equation using the two-dimensional Euler's equation written in Cartesian coordinates.

**(5.107)**
$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = 0$$

To this end, we define the following reference parameters and relationships. It should be noted that the whole aim of this choice is to retain the form of the equations.

We have a characteristic length $L$ in the problem that we will use to scale lengths and coordinates. For example

**(5.108)**
$$x^* = \frac{x}{L}, \qquad \text{and} \qquad y^* = \frac{y}{L}$$

We employ reference density $\rho_r$ and pressure $p_r$ to non-dimensionlise the density and the pressure respectively. As a result we get the non-dimensionalisation for the temperature through the equation of state.

**(5.109)**
$$\rho^* = \frac{\rho}{\rho_r}, \text{ and } p^* = \frac{p}{p_r}, \text{ along with } p = \rho R T \text{ gives } T^* = \frac{T}{T_r},$$

where,

**(5.110)**
$$T_r = \frac{p_r}{\rho_r R},$$

and the equation of state reduces to

**(5.111)**
$$p^* = \rho^* T^*$$

Consider the one-dimensional energy equation from gas dynamics. This relation tells us that

**(5.112)**
$$C_p T_o = C_p T + \frac{V^2}{2}$$

If we divide this equation through by $T_r$ and nondimensionalise speed with a reference speed $u_r$ we get

**(5.113)**
$$C_p T_o^* = C_P T^* + \frac{V^{*2} u_r^2}{2T_r}$$

Now we see that if we define

**(5.114)**
$$u_r = \sqrt{RT_r}$$

equation (5.113) reduces to

**(5.115)**
$$\frac{\gamma}{\gamma - 1} T_o^* = \frac{\gamma}{\gamma - 1} T^* + \frac{V^{*2}}{2}$$

Now, consider the first equation, conservation of mass, from equations (5.107). This becomes

**(5.116)**
$$\frac{\rho_r}{\tau} \frac{\partial \rho^*}{\partial t^*} + \frac{\rho_r u_r}{L} \frac{\partial \rho^* u^*}{\partial x^*} + \frac{\rho_r u_r}{L} \frac{\partial \rho^* v^*}{\partial y^*} = 0$$

where $\tau$ is some characteristic time scale to be defined here. Dividing through by $\rho_r u_r$ and multiplying through by $L$, we get

**(5.117)**
$$\frac{L}{u_r \tau} \frac{\partial \rho^*}{\partial t^*} + \frac{\partial \rho^* u^*}{\partial x^*} + \frac{\partial \rho^* v^*}{\partial y^*} = 0$$

Clearly, if we define the time scale $\tau = L/u_r$ we get back our original equation. I will leave it as an exercise in calculus for the student to show that given the following summary

**(5.118)**
$$x^* = \frac{x}{L}, \qquad \text{and} \qquad y^* = \frac{y}{L}$$

**(5.119)**
$$\rho^* = \frac{\rho}{\rho_r}, \qquad \text{and} \qquad p^* = \frac{p}{p_r},$$

**(5.120)**
$$T_r = \frac{p_r}{\rho_r R}, \qquad \text{and} \qquad u_r = \sqrt{RT_r}$$

equation (5.107) reduces to

**(5.121)**
$$\frac{\partial Q^*}{\partial t^*} + \frac{\partial E^*}{\partial x^*} + \frac{\partial F^*}{\partial y^*} = 0$$

where

**(5.122)**
$$Q^* = \begin{bmatrix} \rho^* \\ \rho^* u^* \\ \rho^* v^* \\ \rho^* E_t^* \end{bmatrix}, E^* = \begin{bmatrix} \rho^* u^* \\ \rho^* u^{*2} + p^* \\ \rho^* u^* v^* \\ [\rho^* E_t^* + p^*] u^* \end{bmatrix}, F^* = \begin{bmatrix} \rho^* v^* \\ \rho^* u^* v^* \\ \rho^* v^{*2} + p^* \\ [\rho^* E_t^* + p^*] v^* \end{bmatrix}$$

Very often for the sake of convenience the “stars” are dropped. One has to remember that though these basic equations have not changed form. Others have changed form. The equation of state becomes $p^* = \rho^* T^*$ and the one-dimensional energy equation changes form. Any other auxiliary equation that you may use has to be non-dimensionalised using the same reference quantities.

A careful study will show you that if $L$, $p_r$ and $\rho_r$ are specified then all the other reference quantities can be derived. In fact, we typically need to fix two reference quantities along with a length scale and the others can be determined. The other point to note is that we typically pick reference quantities based on the problem at hand. A review of dimensional analysis at this point would be helpful.

---

**Assignment 5.4**

(1) Non-dimensionalise the Euler's equation in the differential form for three-dimensional flows.

(2) Try to non-dimensionalise the Burgers' equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{5.123}$$

$u$ does not have the units of speed.

---

## Important ideas from this chapter

- Tensor calculus is a necessity.
- Vector form is big picture, component form is nitty-gritty detail.
- For balance laws, if you know the physics and chemistry you can write the mathematics.

# Multi-dimensional flows and Grid Generation

In the last chapter, we have derived the equations of motion in a very general setting. Earlier, we have seen how one-dimensional flow problems can be handled. We can now move on to modelling flows in two and three spatial dimensions. Some things will extend directly from the chapter (chapter 4) on one-dimensional flows. It is amazing as to how many things do not naturally extend to multiple dimensions. Of course, if the theory does not extend and the algorithm can be extended, someone is going to try it out to see if it works. After that, many schemes get a life of their own. On the other hand, the approximate factorisation schemes will extend to multiple dimensions and will be more useful here.

We will look at a class of finite volume methods and finite difference methods. We will also pay attention to the application of the associated boundary conditions.

## Finite Volume Method

Look at the process that we used to derive the governing equations in the previous chapter. You will see that we started off with a volume $\sigma$. We derived an expression for the rate of change of the amount of a conserved property, $\mathcal{Q}$, contained in that volume. We based this on $\boldsymbol{\mathcal{F}}$, the fluxes through surfaces bounding that volume. The fact that we have the time rate of change of a property of interest gives us an idea for a scheme. If we take the volume small enough, we could assume the property density $Q$ to be constant through the volume. Or, we can use a representative mean value $\bar{Q}$, that is constant through the volume. Then, the time derivative of this single representative value can be determined from the fluxes. Since the volumes that we consider here are not infinitesimal volumes, they are called **finite volumes** or simply **cells**. Such techniques are called finite volume methods.

We will start by rewriting equation (5.85) here.

$$\textbf{(6.1)} \qquad \frac{d}{dt}\int_{\sigma} Q d\sigma = -\int_{S} \boldsymbol{\mathcal{F}} \cdot \hat{n} dS$$

This is an equation for flow in three dimensions. Of course, with the appropriate interpretation of $Q$, $\sigma$, $\boldsymbol{\mathcal{F}}$, and $S$, it could also represent the flow in two spatial dimensions. The basic idea behind the finite volume method is pretty straight forward. We just repeatedly apply our conservation laws given in equation (6.1) to a bunch of control volumes. We do this in a systematic fashion. We fill the region of interest with polyhedra. We ensure that there are no gaps and no overlaps. No gaps and no overlaps means that the sum of the volumes of the polyhedra is the volume of our problem domain. This is called a tessellation. So we would say: We tessellate the region of interest using polyhedra.

Consider two such volumes as shown in Figure 6.1. The volumes share a face $ABCD$. Whatever flows through this face, flows **from** one of these volumes **to** the other volume. We define a $\bar{Q}$ which is the mean value of $Q$ in a given volume

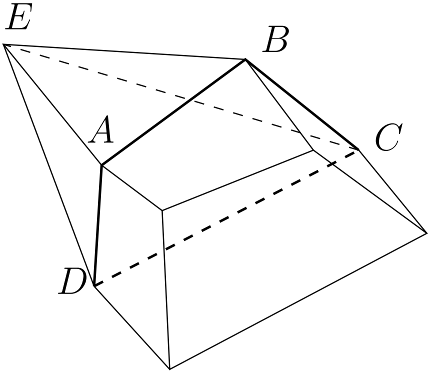

Figure 6.1. Two polyhedra adjacent to each other. They share a face $ABCD$.

as a representative for the flow properties in that cell. This is given by

$$\bar{Q} = \frac{1}{\Delta\sigma}\int_{\Delta\sigma} Q d\sigma \tag{6.2}$$

where $\Delta\sigma$ is the volume of the cell. (Now you understand why we needed to introduce the term cell. "Volume of the volume" does not sound great and if we were discussing a two-dimensional problem, "area of the volume" sounds worse.) We decide, for the sake of this discussion, to store the volume averaged properties $\bar{Q}$ at the centre of a given cell, For this reason, the scheme will be called a **cell centred scheme**. The flux term at the face needs to be estimated in terms of these cell centred $\bar{Q}$ values. In a similar fashion for the volume $ABCDE$, one can compute the total flux through all of the faces $ABCD$, $BCE$, $ADE$, and $DCE$. For this volume, we can now compute the right hand side of equation (6.1). Including the left hand side, this equation is in fact

**(6.3)**

$$\frac{d}{dt}\bar{Q}\Delta\sigma = - \text{ total flux through faces } ABCD, BCE, ADE, \text{ and } L$$

In general, for a polyhedra with four or more faces, we can write

**(6.4)** $$\frac{d}{dt}\bar{Q}\Delta\sigma = -\sum_k \int_{S_k} \boldsymbol{\mathcal{F}}_k \cdot \hat{n}_k dS_k$$

For all of the volumes, this gives us a time rate of change of $\bar{Q}$. For each volume, these equations can be integrated in time to get the time evolution of $\bar{Q}$. This is called the **finite volume method**.

**Computing Fluxes.** It should be obvious from equation (6.4) that for a given cell, the fluxes are very important to the determination of the time evolution of $\bar{Q}$. Just as we had done earlier, we will look at two simple minded methods to get the $\boldsymbol{\mathcal{F}}$ on a face given the $\bar{Q}$ at the cell centres. We see from Figure 6.2 that for the face between two cells indexed by $i$ and $j$, one needs to find the flux at the interface $ABCD$ using $\bar{Q}_i$ and $\bar{Q}_j$. One possibility is that, we compute the flux term $\boldsymbol{\mathcal{F}}$ at each cell and find the flux on the face $ABCD$ by taking the average of these fluxes. That is

**(6.5)** $$\boldsymbol{\mathcal{F}}_{ij} = \frac{1}{2}\left(\boldsymbol{\mathcal{F}}_i + \boldsymbol{\mathcal{F}}_j\right), \quad \boldsymbol{\mathcal{F}}_k = \boldsymbol{\mathcal{F}}(\bar{Q}_k)$$

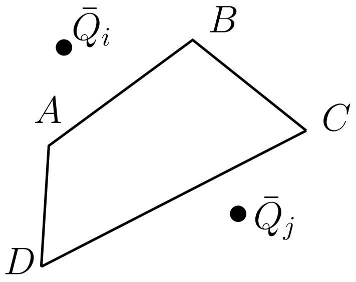

**Figure 6.2. The shared face $ABCD$ has a flux $\mathcal{F}$ through it which may be determined from $\bar{Q}_i$ and $\bar{Q}_j$**

The subscripts $ij$ indicate that we are talking about the face shared by cells $i$ and $j$. The other possibility is to compute an average $\bar{Q}_{ij}$ at the face and find the corresponding $\boldsymbol{\mathcal{F}}$.

---

**Assignment 6.1**

(1) Write a function Get3dFlux( Q ), which will compute the inviscid flux given the variable $\bar{Q}$. Use this function to evaluate fluxes as necessary in the next two functions.
(2) Write a function GetAvgFaceFlux( Qi, Qj ), which will compute the fluxes and return the average given in equation (6.5).
(3) Write a function GetAvgQ( Qi, Qj ) which will return the averaged value $Q_{ij}$.

---

Let us look at the second problem in the context of equation (6.3). Since the integral of the sum is the sum of the integrals, we can split the integral implied by the right hand side of equation (6.3) into two parts. That is

**(6.6)** $$\frac{1}{2}\sum_k \int_{S_k} \boldsymbol{\mathcal{F}}_i \cdot \hat{n}_k dS_k + \frac{1}{2}\sum_k \int_{S_k} \boldsymbol{\mathcal{F}}_k \cdot \hat{n}_k dS_k$$

$k$ is an index that runs over all cells neighbouring the cell $i$. Since, $\boldsymbol{\mathcal{F}}_i$ in the first sum does not depend on $k$, it can be taken out and the resulting sum is zero. Only the neighbours contribute to the net flux in this scheme. The resulting scheme will look like FTCS and likely to require that we add some viscous dissipation. Worse, the current cell seems to make no contribution to the net flux. You can try using the fluxes calculated in both fashions as indicated in problem 2 and problem 3. However, we suspect that using average $Q$ and then computing the fluxes may be better.

How do we calculate the derivatives required in the Navier-Stokes equations and in the artificial dissipation terms that we may want to add?

**Computing Derivatives in Finite Volume Method.** How does one compute a derivative at a point in the finite volume approach? We know one equation that relates the integral of a function to something containing derivative terms: Theorem of Gauss. Theorem of Gauss gives us for a vector function $\vec{A}$

**(6.7)** $$\int_\sigma \text{div}\vec{A} d\sigma = \int_S \vec{A} \cdot \hat{n} dS$$

Let us see what happens if we take $\vec{A} = u\hat{\imath}$, where $\hat{\imath}$ is the standard unit vector in the $x$ direction. The theorem reduces to

**(6.8)** $$\int_\sigma \frac{\partial u}{\partial x} d\sigma = \int_S u\hat{\imath} \cdot \hat{n} dS$$

It should be borne in mind that if we had wanted the $x$-derivative of $v$ we would have set $\vec{A} = v\hat{\imath}$.

In a similar fashion the $y$-derivative of $u$ can be obtained from

$$\textbf{(6.9)} \qquad \int_{\sigma} \frac{\partial u}{\partial y} d\sigma = \int_{S} u\hat{\jmath} \cdot \hat{n} dS$$

where $\hat{\jmath}$ is the standard unit vector in the $y$ direction. Again, if we want the second derivative we would have

$$\textbf{(6.10)} \qquad \int_{\sigma} \frac{\partial^2 u}{\partial x^2} d\sigma = \int_{S} \frac{\partial u}{\partial x} \hat{\imath} \cdot \hat{n} dS$$

It is left as an exercise to the reader to figure out all of the combinations of derivatives that are required. This need occurs usually in the computation of the viscous terms in the Navier-Stokes equations or in the computation of any artificial dissipation terms that the reader may choose (or be forced) to add to the code being developed. Either way, it is a good thing to know.

Are you wondering how to extract the derivative from equation (6.8)? Compare that equation to equation (6.3). They are similar except for the time-derivative. You can discretise the left hand side of equation (6.8) as

$$\textbf{(6.11)} \qquad \int_{\Delta\sigma} \frac{\partial u}{\partial x} d\sigma = \frac{\partial u}{\partial x} \Delta\sigma$$

The right hand side of equation (6.8) looks like flux term and can be evaluated in a similar fashion.

Let us consider an example at this point. We will consider the equation of conservation of mass alone. The integral form given in equation (5.76) is rewritten here for convenience. It reads

$$\frac{d}{dt}\int_\sigma \rho d\sigma = -\int_S \rho\vec{V}\cdot\hat{n}dS \tag{6.12}$$

If we assume that the flow is incompressible, meaning the density $\rho$ is a constant, this equation becomes

$$\int_S \vec{V}\cdot\hat{n}dS = 0 \tag{6.13}$$

Now consider a situation where the flow field is irrotational and we can actually find a $\phi$ so that, $\vec{V} = \nabla\phi$. This assumption gives us

$$\int_S \frac{\partial\phi}{\partial n}dS = 0 \tag{6.14}$$

where $n$ is along the normal $\hat{n}$. We have the integral form of Laplace's equation. We can use the finite volume method to solve it. However, we are not ready to start writing a program.

We are aware that all cells are not created equal. Some of them are at the boundary. Which means one or more faces of the cell are not shared by another cell. How do we compute the net flux for the cells that have a boundary, a boundary of our problem domain? Or, how do we apply boundary conditions in this scheme of things? We will look at one way of doing this.

**Applying Boundary Conditions.** We will take a look at a scheme to apply boundary conditions in finite volume techniques. We need to do this as we need the fluxes on all the faces of a volume to advance its state in time. A volume, $i$, which is at the boundary will have a face which is not shared by any other volumes. One choice is to prescribe the required fluxes as boundary conditions.

**Determining Fluxes at the Boundary Faces.** Consider the equation (6.4). For the faces corresponding to the boundary we need to evaluate $\boldsymbol{\mathcal{F}} \cdot \hat{n}$. At an inlet and exit boundary, the flux can be determined if $Q$ is known on the face. The $Q$ can be obtained on these boundaries as indicated in the chapter 4. We just apply the one-dimensional equation perpendicular to the exit or inlet face.

One difference in multidimensional flows is that we may have walls. We will consider the solid wall boundary condition in an inviscid flow. How do we determine $\boldsymbol{\mathcal{F}} \cdot \hat{n}$? For the Euler's equation, we have written $\boldsymbol{\mathcal{F}}$ as three terms.

**(6.15)**
$$\boldsymbol{\mathcal{F}} = \left\{ \begin{matrix} \rho\vec{V} \\ \rho\vec{V}\vec{V} + p\mathbf{1} \\ (\rho E_t + p)\vec{V} \end{matrix} \right\}$$

This is reproduced here from equation (5.98). The first term $\rho\vec{V}$, for a solid wall gives

**(6.16)**
$$\rho\vec{V} \cdot \hat{n} = 0$$

The first entry in $\boldsymbol{\mathcal{F}} \cdot \hat{n}$ is zero. The second term $\rho\vec{V}\vec{V} + p\mathbf{1}$ consists of two terms. Again, due the solid wall $\rho\vec{V}\vec{V} \cdot \hat{n} = 0$, leaving $p\mathbf{1} \cdot \hat{n} = p\hat{n}$. The final entry in $\boldsymbol{\mathcal{F}} \cdot \hat{n}$ can be verified to be zero. This gives the flux through the face of a cell corresponding to a solid wall as

**(6.17)**
$$\boldsymbol{\mathcal{F}} \cdot \hat{n} = \left\{ \begin{matrix} 0 \\ p\hat{n} \\ 0 \end{matrix} \right\}$$

How do we find $p$ on the wall? We will take a local coordinate system on the cell face that corresponds to the solid wall. We take the $x$-coordinate direction perpendicular to this wall. Let's look at the $x$-momentum equation in the differential form. This turns out to be

$$\frac{\partial \rho u}{\partial t} + \frac{\partial}{\partial x}\left(\rho u^2 + p\right) + \frac{\partial \rho u v}{\partial y} + \frac{\partial \rho u w}{\partial z} = 0 \tag{6.18}$$

If we take find the limit of this equation as we approach the wall, that is as $x \to 0$. We know that $u \to 0$ as $x \to 0$. As a consequence, at the wall

$$\frac{\partial p}{\partial x} = 0 \tag{6.19}$$

Since we are talking only about the normal, it is usual to use a coordinate $n$ instead of $x$. We write

$$\frac{\partial p}{\partial n} = 0 \tag{6.20}$$

The easiest way to impose this condition is to copy the cell centre value to the face.

**Using Pseudo Volumes.** We will take a different tack here. We will generate a volume outside our problem domain which shares the boundary face with cell $i$. Since it is not part of our original problem we will call it a **pseudo volume**, or a pseudo cell. Clearly, we need to determine the $Q$ in this pseudo volume. This is done so as to satisfy our boundary conditions. Again at the inlet and exit, we employ techniques developed in chapter 4. Here is how we apply the wall boundary conditions for an inviscid flow. We will consider the mass flux term first.

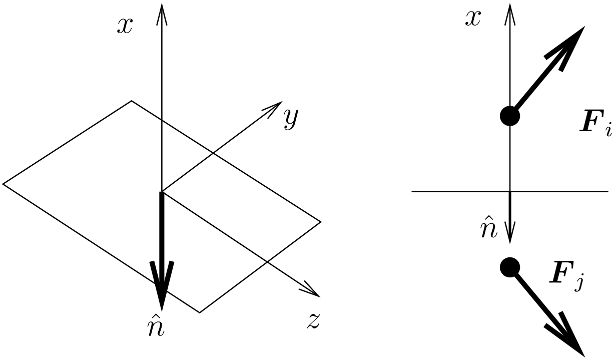

**Figure 6.3.** A surface element, a unit vector $\hat{n}$ normal to that element and pointed out of the cell, and points on either side of the element are shown. A "local" coordinate system is also indicated. If the element represents a solid wall, the mass flux through the surface element is zero

Referring to Figure 6.3, we see that the solid wall condition requires that the mass flux normal to the surface is zero. So, if the mass flux vector is $\boldsymbol{F}_i$ in the problem domain and $\boldsymbol{F}_j$ in the pseudo volume they should be related by the expression

$$\boldsymbol{F}_j = \boldsymbol{F}_i - 2(\boldsymbol{F}_i \cdot \hat{n})\hat{n} \tag{6.21}$$

We see immediately that taking the average flux across the face will result in zero normal flux (check this for your self). This is a vector condition. It corresponds to prescribing three quantities, $\rho u$, $\rho v$, and $\rho w$. We need two more conditions, after all, we require $Q$ in the pseudo volume. We set

$$\frac{\partial p}{\partial n} = 0, \quad \text{and } \frac{\partial T}{\partial n} = 0 \tag{6.22}$$

We have already derived the first condition The second boundary condition in equation (6.22) is the **adiabatic wall** condition. We are asserting that there is no Fourier heat conduction through that boundary. These conditions in turn can be written as

**(6.23)**
$$\frac{\partial \rho}{\partial n} = 0 \quad \text{and,} \quad \frac{\partial E_t}{\partial n} = 0$$

The first condition comes from taking the derivative of the equation of state with respect to $n$. The second follows from the definition of $E_t$. These conditions can be imposed by just copying the $\rho$ and $\rho E_t$ from the interior volume to the pseudo volume.

What happens in the viscous case? The first condition on mass flux changes. The fluid adheres to solid walls and at these walls one needs to set the velocity to zero.

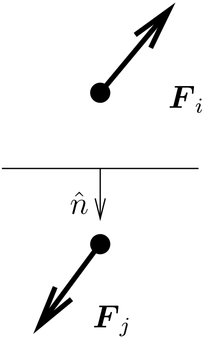

Figure 6.4. $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$ are mass fluxes in the interior and pseudo cells. $\hat{n}$ is a unit normal pointing out of the domain. For a viscous flow, not only is the mass flux through the surface element zero, the velocity at the boundary is zero

This is done by just setting the mass flux term in the pseudo volume to the opposite of that in the interior cell. This is shown in Figure 6.4 That is

**(6.24)**
$$\boldsymbol{F}_j = -\boldsymbol{F}_i$$

Again, we have prescribed three conditions on the boundary. The other two conditions on the pressure and temperature remain the same. In the case of pressure, the boundary condition is one of the consequences of the viscous boundary layer on the wall. The condition, however, is valid only within the boundary layer. This requires that the first cell be completely immersed in the boundary layer. In fact, it will turn out that we need at least three cells in the boundary layer as an absolute minimum. I would recommend five to ten cells. We have not talked about grids yet. We see that there is a constraint on the grids for viscous flow.

We will now look at an important assignment. The first two problems use the heat equation written in the integral form. Use this as an exercise to understand the structure of the program. Once these practice problems are coded, you can move onto the next two problems which involve solutions to the Euler's equations in two and three dimensions.

---

**Assignment 6.2**

The heat equation can be written in integral form as

**(6.25)**
$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\sigma} \phi d\sigma = \int_{S} \nabla\phi \cdot \hat{n} dS$$

Using the finite volume method to solve equation (6.25).

(1) Write a program two solve the two-dimensional problem given in Figure 6.5. Use $10 \times 10$ set of volumes to approximate the square domain. Note that the boundary volumes have pseudo volumes as neighbours. Draw Iso-therms and convergence plots ( norm of the error versus time step ).

(2) Use the unit cube shown in Figure 6.6 as the computational domain. Use pseudo volumes to apply boundary conditions. Tessellate the volume using $10 \times 10 \times 10$ cells. Start with an initial condition of 300 K. Draw Isotherms ( In most graphics packages you would use the iso-surface feature ) and convergence plots

(3) Use the inlet and exit conditions given in assignment 4.6 to compute the flow in a straight two-dimensional channel as shown in Figure 6.5. Instead of the temperature conditions given there, use the flow conditions given in the assignment 4.6. The side boundaries, in addition to being adiabatic, are also solid walls. Draw convergence plots, density contours, velocity vector plots, pressure contours, and temperature contours.

(4) Repeat the previous problem using the three dimensional domain shown in Figure 6.6.

---

## Finite Difference Methods

Instead of using the equations in integral form (in fact an integro-differential form) we employ the Euler equations in the differential form. We have seen finite difference schemes in

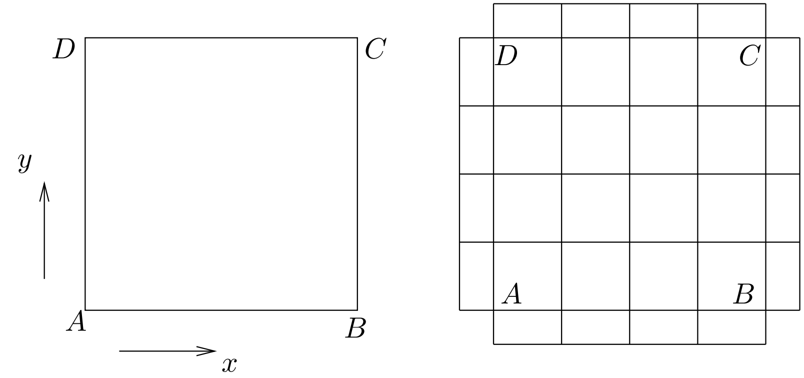

**Figure 6.5. A unit square. The governing equation is given by equation (6.25). $AB$ is held at 400 K and $CD$ is held at 300 K. The $BC$ and $AD$ are adiabatic. A $4 \times 4$ cell discretisation is shown. Note the pseudo volumes indicated on the sides**

earlier chapters. We have used them to solve Laplace's equation, wave equation and heat equation in chapter 3. We have also used them to solve the one-dimensional Euler equations in chapter 4. We will now see how these techniques extend to multi-dimensional flows. Let's start with two-dimensional problems and then extend to three dimensions.

**Two Dimensional Euler Equations.** The equations governing two dimensional inviscid flow can be written in differential form as

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = 0 \tag{6.26}$$

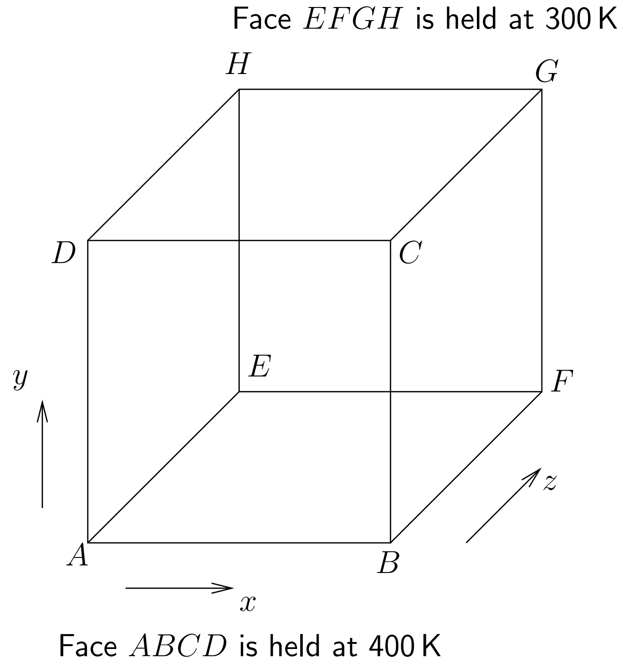

**Figure 6.6. A unit cube. The governing equation is given by equation (6.25). The front and the rear faces are held at 400 K and 300 K respectively. The rest of the faces are adiabatic**

Where

**(6.27)**

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E_t \end{bmatrix}, \quad E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ (\rho E_t + p)u \end{bmatrix}, \quad \text{and} \quad F = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ (\rho E_t + p)v \end{bmatrix}$$

As we had done earlier, we now have a choice of explicit and implicit schemes to solve these problems. The simplest explicit scheme that we can apply is the FTCS scheme. In the two dimensional case we use central differences along each coordinate line. That is

**(6.28)**

$$Q_{p,r}^{q+1} = Q_{p,r}^{q} - \Delta t \left( \frac{E_{p+1,r}^{q} - E_{p-1,r}^{q}}{2\Delta x} + \frac{F_{p,r+1}^{q} - F_{p,r-1}^{q}}{2\Delta y} \right)$$

We may find that we have to add some second order and fourth order dissipation terms to stabilise the scheme. There is a huge class of multi step methods to march these equations in time. Then, again there are many implicit schemes that one can employ too.

Implicit schemes typically involve solving a large system of equations. We can use a whole plethora of numerical algorithms to solve this system of equations [**GL83**]. This makes implicit schemes computational expensive. We now endeavour to use of an idea called approximate factorisation to simplify the numerical solution of these equations.

The basic idea of the **approximate factorisation schemes** is to factor the implicit operator into two or more factors that are easier to invert. In order to ensure that the factorisation cost is not too large, the factors or chosen with a structure that is predetermined and the product of the factors results in an approximation to the original operator.

**Alternating Direction Implicit scheme.** One of the earliest factorisation schemes is the ADI scheme [**Jr55**], [**PJ55**]. This is an acronym for **Alternating Direction Implicit scheme**.

The implicit operator is factored as operators along the various coordinate directions. In the two dimensional case the linearised block implicit scheme would give the equation in the delta form as

$$\left\{ I + \Delta t \frac{\partial}{\partial x} A + \Delta t \frac{\partial}{\partial y} B \right\} \Delta Q = -\Delta t R(Q) \tag{6.29}$$

This can be written as the product of two factors

$$\left\{ I + \Delta t \frac{\partial}{\partial x} A \right\} \left\{ I + \Delta t \frac{\partial}{\partial y} B \right\} \Delta Q = -\Delta t R(Q) \tag{6.30}$$

Each of the factors is an operator in one of the coordinate direction. In a sense we are solving two one-dimensional problems.

$$\left\{ I + \Delta t \frac{\partial}{\partial x} A \right\} \Delta Q^* = -\Delta t R(Q) \tag{6.31}$$

$$\left\{ I + \Delta t \frac{\partial}{\partial y} B \right\} \Delta Q \ = \Delta Q^* \tag{6.32}$$

We solve the first equation for $\Delta Q^*$ and then the second equation for $\Delta Q$. The individual factors result in an equation that can be solved using central differences. This would then give as two separate tridiagonal systems to solve. The advantage is that we have reduced the bandwidth of the system of equations. This is a tremendous savings in computational effort. The natural question to ask is what was the price.

Well, we can multiply the two factors and see the approximation involved. Please remember that the individual factors are actually operators.

**(6.33)**

$$\left\{ I + \Delta t \frac{\partial}{\partial x} A + \Delta t \frac{\partial}{\partial y} B + \Delta t^2 \frac{\partial}{\partial x} A \frac{\partial}{\partial y} B \right\} \Delta Q = -\Delta t R(Q)$$

We have this extra term

**(6.34)**
$$\Delta t^2 \frac{\partial}{\partial x} \left\{ A \frac{\partial}{\partial y} (B \Delta Q) \right\}$$

This term is a second order in $\Delta t$. As we have already neglected terms with $\Delta t^2$, we are willing to live with this. We ignore the fact that the rest of the term consists of a derivative in $x$ and a derivative in $y$.

**LU - approximate factorisation.** We could go further and ask ourselves the question: Do we really want to solve tridiagonal systems? After all, how are we going to solve the tridiagonal system of equations? We may end up using some scheme like LU decomposition to solve the tridiagonal system. Why not just pick the approximate factors as lower and upper triangular matrices. We can achieve this by writing the derivative operator as the sum of a forward difference operator and a backward difference operator. For example,

**(6.35)**
$$\frac{\partial f}{\partial x} \approx \frac{f_{p+1} - f_{p-1}}{2 \Delta x} = \frac{1}{2} \frac{f_{p+1} - f_p}{\Delta x} + \frac{1}{2} \frac{f_p - f_{p-1}}{\Delta x}$$

or

$$\frac{\partial}{\partial x} = \frac{\partial^+}{\partial x} + \frac{\partial^-}{\partial x} \tag{6.36}$$

We can then write equation (6.29) as

$$\left\{ I + \Delta t \frac{\partial^-}{\partial x} A + \Delta t \frac{\partial^-}{\partial y} B + \Delta t \frac{\partial^+}{\partial x} A + \Delta t \frac{\partial^+}{\partial y} B \right\} \Delta Q = -\Delta t R(Q) \tag{6.37}$$

This can now be factored approximately as

$$\left\{ I + \Delta t \frac{\partial^-}{\partial x} A + \Delta t \frac{\partial^-}{\partial y} B \right\} \left\{ I + \Delta t \frac{\partial^+}{\partial x} A + \Delta t \frac{\partial^+}{\partial y} B \right\} \Delta Q = -\Delta t R(Q) \tag{6.38}$$

Which we can then write as

$$\left\{ I + \Delta t \frac{\partial^-}{\partial x} A + \Delta t \frac{\partial^-}{\partial y} B \right\} \Delta Q^* = -\Delta t R(Q) \tag{6.39}$$

$$\left\{ I + \Delta t \frac{\partial^+}{\partial x} A + \Delta t \frac{\partial^+}{\partial y} B \right\} \Delta Q \ = \Delta Q^* \tag{6.40}$$

The first equation is a lower triangular system. The second equation corresponds to an upper triangular system. They can be solved through a sequence of forward substitution and back substitution respectively.

To see how this actually works, we write out the left hand side of the unfactored equation in discretised form at the grid point $(i, j)$. To keep the equations compact we define

**(6.41)**
$$\alpha_i = \frac{\Delta t}{2\Delta x} A_i, \quad \text{and} \quad \beta_i = \frac{\Delta t}{2\Delta y} B_i$$

**(6.42)**
$$\Delta Q_i + \underbrace{\alpha_{i+1}\Delta Q_{i+1} - \alpha_i \Delta Q_i}_{\Delta t \frac{\partial^+}{\partial x} A\Delta Q} + \underbrace{\beta_{i+N}\Delta Q_{i+N} - \beta_i \Delta Q_i}_{\Delta t \frac{\partial^+}{\partial y} B\Delta Q}$$
$$+ \underbrace{\alpha_i \Delta Q_i - \alpha_{i-1}\Delta Q_{i-1}}_{\Delta t \frac{\partial^-}{\partial x} A\Delta Q} + \underbrace{\beta_i \Delta Q_i - \beta_{i-N}\Delta Q_{i-N}}_{\Delta t \frac{\partial^-}{\partial y} B\Delta Q} = -\Delta t R(Q_i)$$

We can write this in the factored form given in equation (6.39) as

**(6.43)**
$$\Delta Q_i^* + \alpha_i \Delta Q_i^* - \alpha_{i-1}\Delta Q_{i-1}^* + \beta_i \Delta Q_i^* - \beta_{i-N}\Delta Q_{i-N}^* = -\Delta t R(Q_i)$$

and

**(6.44)**
$$\Delta Q_i + \alpha_{i+1}\Delta Q_{i+1} - \alpha_i \Delta Q_i + \beta_{i+N}\Delta Q_{i+N} - \beta_i \Delta Q_i = \Delta Q_i^*$$

These equations are used to sweep through the domain. Equation (6.43) is used to go from grid point indexed by $i = N+1$ to $i = M$ ( see Figure 6.7). This is called the L-sweep. Equation (6.44) is used to sweep from $i = M - N - 1 = p - 1$ back to $i = 0$. This is called the U-sweep.

Let us consider the L-sweep first. That is we will look at equation (6.43). It is clear that at grid point $N + 1$

**(6.45)**
$$\Delta Q_{N+1}^* + \alpha_{N+1}\Delta Q_{N+1}^* - \alpha_N \Delta Q_N^*$$
$$+ \beta_{N+1}\Delta Q_{N+1}^* - \beta_1 \Delta Q_1^* = -\Delta t R(Q_{N+1})$$

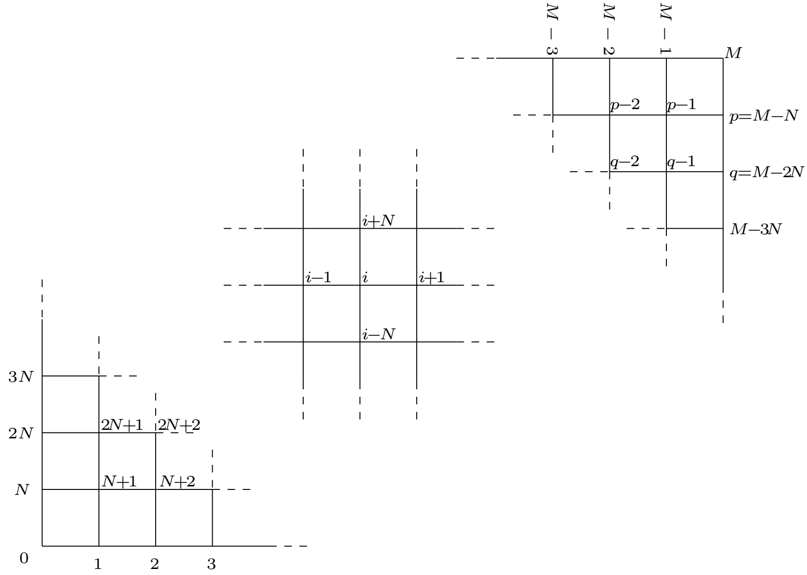

**Figure 6.7. Application of LU approximate factorisation in two dimensions. A typical L-sweep starts at the lower left hand corners and proceeds to the top right hand corner. A U-sweep does the reverse. $p$ and $q$ are defined only to keep the figure uncluttered**

Both grid points indexed as $N$ and $1$ are on the boundary. As a consequence these, pending figuring out applications of boundary conditions, need to be known. They can be taken over to the right hand side of the equation leaving

**(6.46)**

$$[I+\alpha_{N+1}+\beta_{N+1}]\Delta Q^*_{N+1} = -\Delta t R(Q_{N+1}) + \alpha_N \Delta Q^*_N + \beta_1 \Delta Q^*_1$$

Which can be solved for $\Delta Q^*_{N+1}$. In fact, at an arbitrary grid point $i$, (here $R_i = R(Q_i)$)

**(6.47)**

$$\Delta Q^*_i = [I + \alpha_i + \beta_i]^{-1} \left( -\Delta t R_i + \alpha_{i-1} \Delta Q^*_{i-1} + \beta_{i-N} \Delta Q^*_{i-N} \right)$$

we can solve for the $\Delta Q^*_i$ as the $\Delta Q^*_{i-1}$ and $\Delta Q^*_{i-N}$ would have already been calculated. In this manner the $\Delta Q^*$ for all the $i$'s can be found. Now that $\Delta Q^*$ is known, we can sweep equation (6.44). This U-sweep goes from grid index $i = M - N - 1 = p - 1$ back to $i = 0$.

**(6.48)**

$$\Delta Q_i = [I - \alpha_i - \beta_i]^{-1} \left( \Delta Q^*_i - \alpha_{i+1} \Delta Q_{i+1} - \beta_{i+N} \Delta Q_{i+N} \right)$$

We need to address the issue of applying boundary conditions. Like we did with the finite volume method, we could create some pseudo grid points as the neighbours of the points on the boundary and use them to impose the boundary conditions. The other possibility is to apply boundary conditions in the same manner as the one dimensional case. In either case, we will have quantities that are prescribed - like the back pressure $p$ at the exit of a pipe, and we will have quantities that are taken from the domain to the boundary, again like the pressure towards a solid wall. In the L-sweep, conditions that are prescribed at the lower and the left boundary are applied. Conditions that are propagated from the domain to the boundary are imposed on the top and right boundaries. Vice-versa for the U-sweep. This should be apparent after inspecting equations (6.47) and (6.48).

**Three-Dimensional Euler Equations.** These equations look very similar to the two dimensional counterpart. In fact the same nondimensionalisation can be used to get

**(6.49)**

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = 0$$

Where

**(6.50)**

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E_t \end{bmatrix}$$

and
**(6.51)**

$$E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ [\rho E_t + p]u \end{bmatrix}, \; F = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ [\rho E_t + p]v \end{bmatrix}, \text{ and } G = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ [\rho E_t + p]w \end{bmatrix}$$

We can apply ADI to this system of equations written in Delta form. The point to be noted is that we will get three factors instead of two. One for each spatial coordinate direction. Though this is a very large saving in computational effort from the original equation, we will end up solving three tridiagonal systems of equations.

In the three dimensional case the linearised block implicit scheme would give the equation in the delta form as

**(6.52)**

$$\left\{ I + \Delta t \frac{\partial}{\partial x} A + \Delta t \frac{\partial}{\partial y} B + \Delta t \frac{\partial}{\partial z} C \right\} \Delta Q = -\Delta t R(Q)$$

This can be written as the product of three factors

**(6.53)** $$\left\{ I + \Delta t \frac{\partial}{\partial x} A \right\} \left\{ I + \Delta t \frac{\partial}{\partial y} B \right\} \left\{ I + \Delta t \frac{\partial}{\partial z} C \right\} \Delta Q = -\Delta t R(Q)$$

Each of the factors is an operator in one of the coordinate direction. In a sense we are solving three one-dimensional problems. The three one-dimensional problems can each be solved using various methods including LU decomposition. The question remains: why not just go to an LU approximate factorisation? It just produces two factors.

**(6.54)** $$\left\{ I + \Delta t \frac{\partial^-}{\partial x} A + \Delta t \frac{\partial^-}{\partial y} B + \Delta t \frac{\partial^-}{\partial z} C + \Delta t \frac{\partial^+}{\partial x} A + \Delta t \frac{\partial^+}{\partial y} B + \Delta t \frac{\partial^+}{\partial z} C \right\} \Delta Q = -\Delta t R(Q)$$

This can now be factored approximately as

**(6.55)** $$\left\{I+\Delta t\frac{\partial^-}{\partial x}A+\Delta t\frac{\partial^-}{\partial y}B+\Delta t\frac{\partial^-}{\partial z}C\right\}\left\{I+\Delta t\frac{\partial^+}{\partial x}A+\Delta t\frac{\partial^+}{\partial y}B+\Delta t\frac{\partial^+}{\partial z}C\right\}\Delta Q=-\Delta tR(Q)$$

Which we can then write as

**(6.56)**
$$\left\{I+\Delta t\frac{\partial^-}{\partial x}A+\Delta t\frac{\partial^-}{\partial y}B+\Delta t\frac{\partial^-}{\partial z}C\right\}\Delta Q^*=-\Delta tR(Q)$$

**(6.57)**
$$\left\{I+\Delta t\frac{\partial^+}{\partial x}A+\Delta t\frac{\partial^+}{\partial y}B+\Delta t\frac{\partial^+}{\partial z}C\right\}\Delta Q\ \ =\Delta Q^*$$

The first equation is a lower triangular system. The second equation corresponds to an upper triangular system. They can be solved through a sequence of forward substitution and back substitution respectively.

We need to address the issue of applying boundary conditions. At the boundary, we consider the one-dimensional equations as applicable perpendicular to the boundary. That is, we describe our equations normal to the boundary and tangential to the boundary and ignore the tangential equations. This would be akin to taking only the first factor from the ADI scheme at a boundary parallel to the YZ plane. Since this is a one-dimensional problem at this point, our procedure for applying boundary conditions can be used here. The only difference is the we also need to propagate $\rho v$ and $\rho w$.

Again attention needs to be paid to how and when the boundary conditions are applied. In the ADI case, the boundary condition is applied during the appropriate spatial sweep. In the case of approximate LU decomposition, the boundary conditions also need to be factored into a lower component and an upper component so that they can be applied at the appropriate time in the sweep. That is the lower triangular part during the L-sweep and the upper triangular part during the U-sweep .

**Addition of Artificial Dissipation.** As in the one-dimensional Euler equations, it is possible, as need, to add both second and fourth order terms in order to attenuate high frequency oscillations. We have a choice of adding these terms either to the right hand side of equation (6.52) or to the left hand side. If we add it to the right hand side it appears in terms of $Q$ which is at the current time step. Hence the term becomes and "explicit" term. On the other hand if we add it to the left hand side it would be written in the delta form and would appear in the system of equations to be solved. Consequently, the term would be implicit in nature.

**Assignment 6.3**

(1) Use the differential form of the equations and redo assignment 6.2.

## Grid Generation

We have seen numerical techniques to solve problems in two and three spatial dimensions using finite volume methods or finite difference methods. So far, for convenience, all the assignment have been restricted to domains in which the Cartesian coordinate system worked well. Now, we will try to lay the foundation for solving problems in more general regions.

This part of the chapter will address the issue of where things are located. We have so far talked about points at which a function has been expanded using Taylor's series, or points at which the function or its derivatives have been somehow "approximated". Where are these points? How do we determine their location? Which are the neighbouring points? How are these points distributed? Is there an optimal location for the points? Is there an optimal distribution of the points? Is there an optimal number of points?

We have a lot of questions regarding these points that we have rather casually introduced into our study. Now we will set about answering a few of these questions.

The study of these questions and their answers falls in the realm of grid generation[**ea99**]. We have already seen some grids in the earlier chapters. After a study of this chapter the reader should be able to generate grids in a given computational domain and solve a problem. This two part statement is very important. We often ask, I have generated these two grids, which is better? If we are asking this question without reference to a problem, we might need to pick up some generic criterion to get a figure of merit for a given grid. On the other hand, if there is a definite problem at hand to be solved, that problem and its solution then determine the answer to this question.

The question of where things are located was addressed systematically by René Descartes. He brought the power of abstraction of algebra to bear on geometry, resulting in analytic geometry. The realisation that one could assign numbers in the form of coordinates to points in space was a revolution. Cartesian coordinates are named so in his honour.

The fundamental job of a coordinate system is to help us locate points. The fundamental job of a grid is to help us locate points, determine neighbours and distances to neighbours. Let us, therefore, start with Cartesian coordinates and ask the question:

### Why Grid Generation?

We will use a two-dimensional problem to motivate the study of grid generation. We solved the Laplace equation on a unit square in section 3. After that, we have used a unit square and a unit cube to solve the Euler's equation in many forms in the current chapter. We were able to use Cartesian grids for the solution of this problem. This was possible as the domain on which the problem was defined, a unit square at the origin, conforms to, or is amenable to a Cartesian mesh. That is the boundary of our problem domain is made up of coordinate lines. What do we do if the problem domain were not this convenient? What if it was changed to one with top boundary slanting? We have encountered this question in chapter 5. This is shown in Figure 6.8. We considered a Cartesian mesh in that chapter and abandoned it. Here we will postpone the discussion on that decision. We will revisit Cartesian meshes again later. We even saw that we could generating a boundary conforming mesh. This is shown in Figure 5.4 is reproduced here for convenience in Figure 6.9. To get a clear idea as to where to go from here,

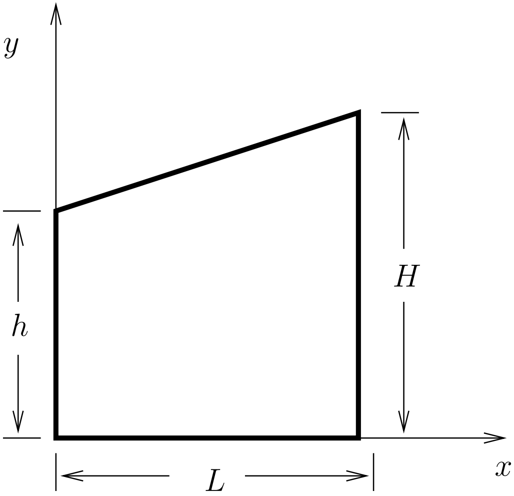

**Figure 6.8. Trapezoidal domain on which Laplace equation is to be solved**

we pay attention to simple geometries and find out why they are simple. Here are two examples:

(1) If we want to solve problems on a square or a rectangle we can use Cartesian coordinates.

(2) If we want to solve a problem on a circle we could use polar coordinates.

Notice that by simple geometry, we mean one which is made up of the coordinate lines of, for example, the Cartesian mesh. In fact, we want to generate a coordinate system where the boundaries are coordinate lines. Since our focus is on the problem domain at hand, we say such a coordinate system is boundary conforming. We will try to perform a transformation of coordinates of some sort, to be boundary conforming. The point now is to find such a coordinate transformation.

We observe that, as we go from the origin to the other end of the trapezium along the $x$-axis, the height of the trapezium increases in a linear fashion. Actually, what is important is that it does so in a known fashion. Let us take $M$ grid points in the $y$ direction. At any $x$ location, we divide the local height of the trapezium into $M-1$ equal intervals. We see that we can obtain a mesh, that looks ordered, and the grid points are on the boundary. We now have grids on our boundary and will be

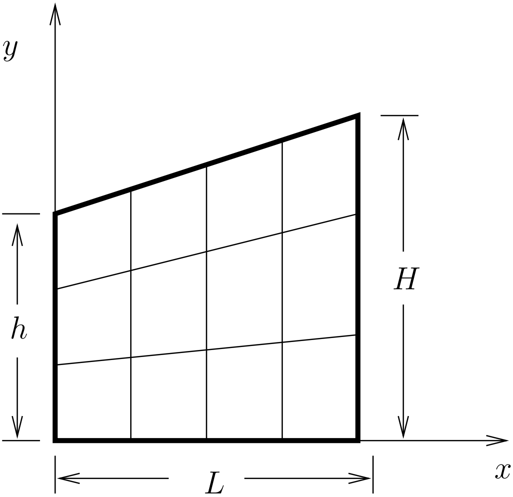

**Figure 6.9. Grid generated by spacing grid points proportionately in the $y$-coordinate direction**

able to approximate our equations on this grid. The details of this process will be discussed in the section on algebraic grids.

Another possibility is that we give up the structure that we get from using a coordinate system, retain a Cartesian mesh and decompose our domain into smaller domains whose geometry is known in the Cartesian mesh. This mesh or tessellation as it is sometimes called, can then be used to solve our differential equation. Figure 6.10 shows a triangulation of the given domain. We now solve our problem on these triangles.

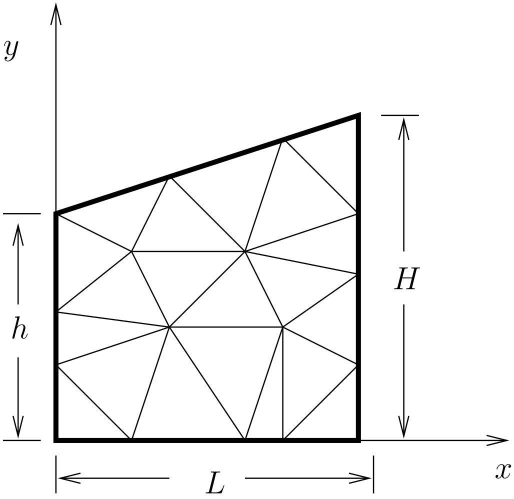

**Figure 6.10. Trapezoidal domain on which Laplace equation is to be solved**

## A Brief Introduction to Geometry

We have already seen that to obtain a coordinate transform is to have knowledge of the coordinate lines. We plan to do a quick review of the geometry of space curves. Along the way, we will also do a review of the geometry of surfaces. Before that, let us look at some more motivation to look at the space curve, since one-dimensional problems could be of interest in themselves.

An electrical engineer is designing a circuit board. The engineer finds that there is one particular component that is going to result in a lot of heat. The engineer could put a heat sink on it. Taking the surrounding components and the overall thermal environment in the enclosure into account the engineer decides to transfer the energy out somehow. The solution proposed is an insulated silver wire that is attached to the component on one side and snakes it way to a heat sink outside the box.

What is the temperature distribution in the wire? There are lots of ways to solve this problem. For our purpose, we will consider the one-dimensional heat equation as governing the problem and we will discretise this equation to get the solution. In order to do this we need to discretise the wire.

This problem, to some, may look like a contrived problem. In reality, we would pump some liquid around, we propose this solution to keep things simple for our discussion here. However, the ability to distribute grid points along a space curve can be viewed as one of the fundamental activities of some grid generators.

Now, we will do a quick review of the geometry of a space curve. To get an idea of a space curve as a one-dimensional entity, consider the following example. Chennai central is the main railway station in Chennai. You get onto the train there and you could wind your way through India and find yourself getting off at New Delhi. As you go along, you look out of the window and see stones with markings on them; they tell you how far you are from Chennai. (The stones are very often called milestones, though these days they specify the distance in kilometres.) The track which started in Chennai central has a single unique number associated with it along the track, which is the distance from Chennai. When you are at point 2177 km you have reached Hazarat Nizamuddin, New Delhi. The turns that the train takes should convince you that the track is not a “straight run”. On the other hand, with a distance of 2177 km covered, the curvature of the earth cannot be ignored either. So, the track forms a space curve in a coordinate system at the centre of the earth and fixed to the earth. We can measure distance along the track to identify a point on the track or we could use time, especially if you are on a train without any intermediate stops. Since, a single parameter is enough to locate ourselves on the track, it is one-dimensional in nature.

A space curve can be parametrised by one parameter, say $t$. We have,

**(6.58)** $$\boldsymbol{\alpha}(t) = x(t)\boldsymbol{e}_1 + y(t)\boldsymbol{e}_2 + z(t)\boldsymbol{e}_3$$

where, $\boldsymbol{e}_1$,$\boldsymbol{e}_2$, and $\boldsymbol{e}_3$ are the standard basis. We restrict ourselves to curves in a two dimensions / three dimensions. So a curve in two dimensions, $\mathbb{R}^2$, would be a map from the real line into a plane. Formally,

**(6.59)** $$\boldsymbol{\alpha}(t) : \{U \subset \mathbb{R} \rightarrow \mathbb{R}^2\}$$

As an example, consider the mapping $\boldsymbol{\alpha}(t)$ given by the components $(\cos t, \sin t)$. What does the graph of this map look like? If $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are the basis along the $1$ and $2$ coordinate directions, then,

**(6.60)**
$$\boldsymbol{\alpha}(t) = \boldsymbol{e}_1 \cos t + \boldsymbol{e}_2 \sin t$$

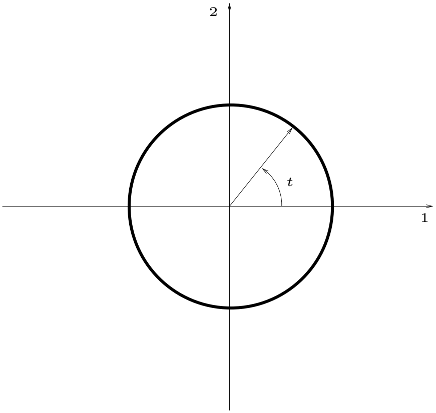

**Figure 6.11. Curve parametrised by $\boldsymbol{\alpha}(t) = \boldsymbol{e}_1 \cos t + \boldsymbol{e}_2 \sin t$**

## Assignment 6.4

(1) How would the graph of the curve given by the following equation look?

**(6.61)**
$$\boldsymbol{\alpha}(t) = \boldsymbol{e}_1 \cos 2t + \boldsymbol{e}_2 \sin 2t$$

What changes?

(2) How about

**(6.62)** $$\boldsymbol{\alpha}(t) = \boldsymbol{e}_1 \cos t^2 + \boldsymbol{e}_2 \sin t^2$$

---

For both the curves given in the two problems, find the tangent vector at any point. What happens to the magnitude of the tangent vector? What happens to the unit vector along the tangent vector? You should find that the magnitude or the speed of the curves changes, the unit vector does not.

In general, the space curve in three dimensions may be as shown in Figure 6.12. Normally, the **tangent** to the curve or the velocity as it is often called in differential geometry, is given by the derivative of $\boldsymbol{\alpha}(t)$ as

**(6.63)** $$\boldsymbol{\alpha}_t = x_t \boldsymbol{e}_1 + y_t \boldsymbol{e}_2 + z_t \boldsymbol{e}_3$$

where the subscript indicates differentiation with respect to $t$.

Right, we have a parametrised version of our wire. How do we generate a grid on the wire? Well, we have the parameter that goes from $a$ to $b$. We could just divide (partition) the interval $[a, b]$ into equal intervals and from there find the corresponding $(x, y, z)$ coordinates from the specified $\boldsymbol{\alpha}(t)$. This may not give you a grid that is uniformly distributed along the wire. How do we generate a uniform grid on this wire? That is, we want grid points that are equispaced along the length of the wire. To do this, we need the length of the wire.

The length of the curve from $t = a$ to $t = b$ is given by

**(6.64)** $$s = \int_a^b \left|\boldsymbol{\alpha}_t\right| dt$$

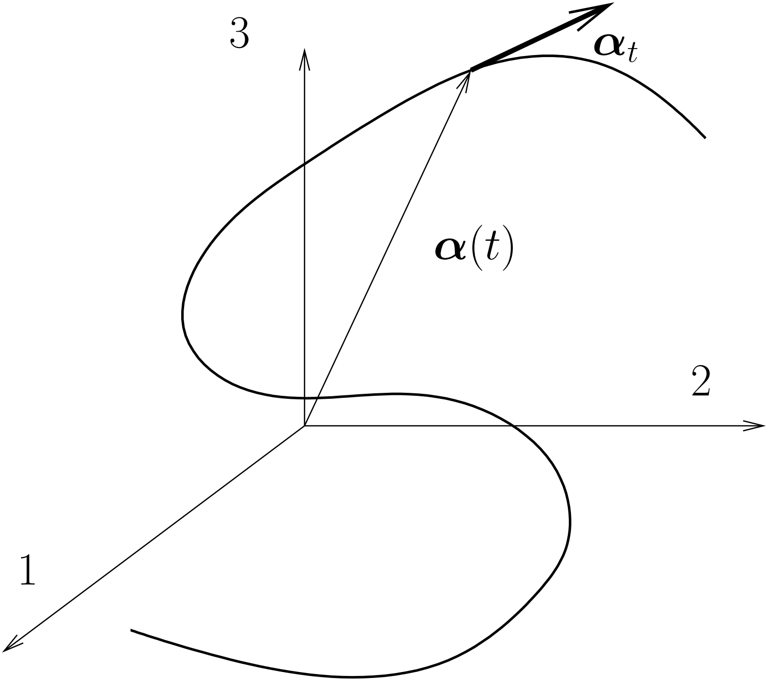

**Figure 6.12. Curve parametrised by** $\boldsymbol{\alpha}(t) = x(t)\boldsymbol{e}_1 + y(t)\boldsymbol{e}_2 + z(t)\boldsymbol{e}_3$

We can get this from our tensor calculus. Given the map $\boldsymbol{\alpha}$ from the Cartesian coordinate system to $t$ we can define a differential length for a line segment in $t$ as

**(6.65)**
$$ds^2 = \boldsymbol{\alpha}_t \cdot \boldsymbol{\alpha}_t dt^2$$

Since we are mapping to one dimension the metric tensor has only one component, $g_{11}$.

Sometimes, we are lucky and we have a parametrisation $s$, where $s$ is the length along the space curve. For example, this occurs in our railroad track. We could use the milestones or the kilometre markers as a measure along the length of the rails to indicate position. That is, we are given a situation where

**(6.66)**
$$\boldsymbol{\alpha}(s) = x(s)\boldsymbol{e}_1 + y(s)\boldsymbol{e}_2 + z(s)\boldsymbol{e}_3$$

where,

**(6.67)**
$$s = \int_a^b \left|\boldsymbol{\alpha}_s\right| ds,$$

and

**(6.68)**
$$\left|\boldsymbol{\alpha}_s\right| = 1$$

Any curve that is parametrised so that equation (6.68) is true, is said to have a unit speed parametrisation or the curve is just called a **unit speed curve**.

If we were to generate a grid on our railroad line at uniform intervals along the length, we could identify these by an index. If we put a grid point at every railway station on the route, we could number them starting with zero for Chennai. We know then that the fifth railway station comes on the railroad after the fourth and before the sixth. Given a railway station $i$, we know it is preceded by $i-1$ (excepting Chennai) and succeeded by $i+1$ (except at Delhi). Thanks to this underlying structure, such a grid is called a **structured grid**.

On the other hand, someone can decide to sort the railway station according to alphabetical order and number in that order. In that case, we could not be sure that of the neighbour of $i$ without looking at the sorted list. Since this does not have an immediate structure from which the predecessor and successor can be found, this is called an **unstructured grid**.

It may seem that some rearrangement will allow a structured grid to be converted to an unstructured grid. This is so in one-dimensional problems. However, if one looks at Figure 6.10, it is clear that this is not possible in this case. In the case of an unstructured grid, we are forced to retain information on neighbours.

---

**Assignment 6.5**

(1) Find the curvature and torsion of the curves given in assignment 6.4.

(2) Find the tangent $\boldsymbol{t}$, normal $\boldsymbol{n}$, and binormal $\boldsymbol{b}$ for the curve given by $(a\cos t, b\sin t, ct)$. Specialise the problem to $a = b$. With $a = b$, what are $\boldsymbol{t}$, $\boldsymbol{n}$, and $\boldsymbol{b}$ if we change the parametrisation to $t = s/\sqrt{a^2 + c^2}$?

---

**Properties of Space Curves.** We will now look a little more closely at properties of space curves. We call the tangent vector to a curve with unit speed parametrisation as $\boldsymbol{t}$. Now, the question is what is the derivative of $\boldsymbol{t}$? We will answer this in two parts. Clearly, as the curve is a unit speed curve, there is, by definition, no change in the magnitude or the speed. The only other property of a vector that can change is its "direction". First, for any unit vector

**(6.69)** $$\boldsymbol{t}\cdot\boldsymbol{t} = 1 \Rightarrow \frac{d}{ds}\{\boldsymbol{t}\cdot\boldsymbol{t}\} = \boldsymbol{t}_s\cdot\boldsymbol{t} + \boldsymbol{t}\cdot\boldsymbol{t}_s = 2\boldsymbol{t}_s\cdot\boldsymbol{t} = 0$$

We will assume that $\boldsymbol{t}_s$ is not the zero vector. This means that the derivative is orthogonal to the original vector. For the unit speed curve, $\boldsymbol{t}_s$ is also the rate at which the curve moves away from the tangent and hence represents the curvature of the curve. We can see this from the example shown in Figure 6.13. For the sake of simplicity, a planar curve is chosen. As was pointed out in the assignment, the direction of the tangent to a curve at a point is a property of that curve at that point and does not really depend on the parametrisation. The tangents are shown in this figure as position vectors of points on a unit circle. Since the curve is smooth, the tangent varies in a smooth

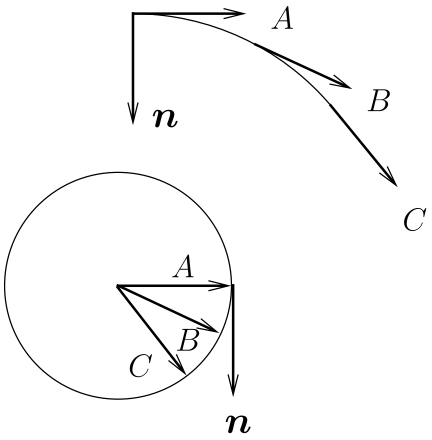

**Figure 6.13. A planar curve is shown with tangents at three different points. These tangent vectors are shown located at a single point. The arrow heads lie on a unit circle since the tangent vectors in this case are unit vectors**

fashion. It is clear then that the derivative of the tangent vector $\boldsymbol{t}_A$ at the point $A$ is perpendicular to $\boldsymbol{t}_A$.

So, if $\boldsymbol{t}_s$ is along a direction called $\boldsymbol{n}$, then

**(6.70)** $$\boldsymbol{t}_s = \kappa \boldsymbol{n}$$

where, $\kappa$ is the **curvature** at that point. We get back to our space curve now. We have the unit vector $\boldsymbol{t}$ which is the tangent vector. We also have the vector $\boldsymbol{n}$ which is normal to the tangent vector and we know that $\boldsymbol{t}_s$ is proportional to $\boldsymbol{n}$. We have two orthogonal vectors in three dimensions at a given point on the curve. We could define a third vector $\boldsymbol{b}$, orthogonal to $\boldsymbol{t}$ and $\boldsymbol{n}$, to complete the triad.

**(6.71)** $$\boldsymbol{b} = \boldsymbol{t} \times \boldsymbol{n}$$

It would be interesting, and useful, to see how this triad evolves along the curve. We already have $\boldsymbol{t}_s$ we can obtain equation for the other two rates. Since $\boldsymbol{b} = \boldsymbol{t} \times \boldsymbol{n}$, we can take a derivative to find

**(6.72)** $$\boldsymbol{b}_s = \boldsymbol{t}_s \times \boldsymbol{n} + \boldsymbol{t} \times \boldsymbol{n}_s = \boldsymbol{t} \times \boldsymbol{n}_s = -\tau \boldsymbol{n}$$

By convention the sign on the right hand side of equation (6.72) is taken as negative. Since $\boldsymbol{b}_s$ is orthogonal to $\boldsymbol{b}$ and $\boldsymbol{t}$, it must be along $\boldsymbol{n}$. $\boldsymbol{b}_s$ is the rate at which $\boldsymbol{b}$ is twisting away from the plane formed by $\boldsymbol{t}$ and $\boldsymbol{n}$. It represents the **torsion** of the curve.

In a similar fashion, we find $\boldsymbol{n}_s$ using $\boldsymbol{n} = \boldsymbol{b} \times \boldsymbol{t}$ to be $\tau \boldsymbol{b} - \kappa \boldsymbol{t}$. We can write these three equations as one matrix equation for the triad $\boldsymbol{t}$, $\boldsymbol{n}$, and $\boldsymbol{b}$ as

$$\textbf{(6.73)} \qquad \frac{\partial}{\partial s}\begin{Bmatrix}\boldsymbol{t}\\ \boldsymbol{n}\\ \boldsymbol{b}\end{Bmatrix} = \begin{pmatrix} 0 & \kappa & 0\\ -\kappa & 0 & \tau\\ 0 & -\tau & 0\end{pmatrix}\begin{Bmatrix}\boldsymbol{t}\\ \boldsymbol{n}\\ \boldsymbol{b}\end{Bmatrix}$$

which gives us the evolution of the triad with the parameter $s$. It looks like this system of equations tells us that if we are given the curvature and torsion of a curve and a point on that curve we can reconstruct the curve.

**Surfaces and Manifolds.** Analogous to the one-dimensional space represented by the space curve, a surface in three dimensions turns out be a "two-dimensional" world. Look at the surface of the earth. Most everyone is thinking in terms of "floor space". Our thinking, though we are embedded in a three-dimensional space is very often two-dimensional[1]. These surfaces are two dimensional as we require two parameters to index them. A possible representation would be

$$\textbf{(6.74)} \qquad \boldsymbol{\sigma}(u,v) = f(u,v)\boldsymbol{e}_1 + g(u,v)\boldsymbol{e}_2 + h(u,v)\boldsymbol{e}_3$$

Of course, $\boldsymbol{\sigma}(a,v)$ is a coordinate line on this surface as is $\boldsymbol{\sigma}(u,b)$. These two lines pass through the point $\boldsymbol{\sigma}(a,b)$. Clearly $\boldsymbol{\sigma}(a,v)$ is a space curve and its tangent at the point $\boldsymbol{\sigma}(a,b)$ is given by $\boldsymbol{\sigma}_v(a,v)$. The subscript $v$ indicates differentiation with respect to $v$. Likewise, we have $\boldsymbol{\sigma}_u(u,b)$ a tangent to $\boldsymbol{\sigma}(u,b)$. Here, the subscript $u$ indicates differentiation with respect to $u$.

[1] **He is intelligent, but not experienced. His pattern indicates two-dimensional thinking.–Spock, Star Trek: Wrath of Khan**

For a "regular surface", $|\boldsymbol{\sigma}_u \times \boldsymbol{\sigma}_v| \neq 0$. In fact, the unit normal to the surface is defined as

**(6.75)**
$$\boldsymbol{N} = \frac{\boldsymbol{\sigma}_u \times \boldsymbol{\sigma}_v}{|\boldsymbol{\sigma}_u \times \boldsymbol{\sigma}_v|}$$

If $\boldsymbol{\gamma}(t)$ is a curve on the surface, its tangent at any point, $\boldsymbol{\gamma}_t$, lies in the plane spanned by $\boldsymbol{\sigma}_u$ and $\boldsymbol{\sigma}_v$ and it is normal to $\boldsymbol{N}$. The length along this curve is given by equation (6.64). Using chain rule we can write the expression for $|\boldsymbol{\gamma}_t|$ as

**(6.76)**
$$|\boldsymbol{\gamma}_t|^2 = \boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_u u_t^2 + 2\boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_v u_t v_t + \boldsymbol{\sigma}_v \cdot \boldsymbol{\sigma}_v v_t^2$$

Recognising that

**(6.77)**
$$ds^2 = |\boldsymbol{\gamma}_t|^2 dt^2 = (E u_t^2 + 2F u_t v_t + G v_t^2) dt^2$$

where,

**(6.78)**
$$E = \boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_u, \quad F = \boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_v, \text{and} \quad G = \boldsymbol{\sigma}_v \cdot \boldsymbol{\sigma}_v$$

We then identify the following expression as the First Fundamental Form.

**(6.79)**
$$Edu^2 + 2Fdudv + Gdv^2$$

We could ask the question as to how much the surface curves away from the tangent plane. This question is critical for us since it will determine, ultimately, the size of our grid spacing. From the point $\boldsymbol{\sigma}(u, v)$, how much does $\boldsymbol{\sigma}(u + \Delta u, v + \Delta v)$ deviate? This answered simply by looking at

**(6.80)** $$(\boldsymbol{\sigma}(u+\Delta u, v+\Delta v) - \boldsymbol{\sigma}(u,v)) \cdot \boldsymbol{N}$$

Expanding using Taylor's series and cancelling $\boldsymbol{\sigma}(u,v)$ gives us

**(6.81)**
$$\left\{ (\underbrace{\boldsymbol{\sigma}_u \Delta u + \boldsymbol{\sigma}_v \Delta v}_{\text{in the tangent plane}}) + \frac{\boldsymbol{\sigma}_{uu}\Delta u^2 + 2\boldsymbol{\sigma}_{uv}\Delta u \Delta v + \boldsymbol{\sigma}_{vv}\Delta v^2}{2} + \ldots \right\} \cdot \boldsymbol{N}$$

Since the $\boldsymbol{N}$ is orthogonal to the tangent plane the first two terms do not contribute to the curvature. In fact, we have taken the dot product to identify and eliminate these terms. This leaves an expression that we identify as the Second Fundamental Form.

**(6.82)** $$Ldu^2 + 2Mdudv + Ndv^2$$

where,

**(6.83)** $$L = \boldsymbol{\sigma}_{uu} \cdot \boldsymbol{N}, \quad M = \boldsymbol{\sigma}_{uv} \cdot \boldsymbol{N}, \text{and} \quad N = \boldsymbol{\sigma}_{vv} \cdot \boldsymbol{N}$$

Try out the following problems:

---

**Assignment 6.6**

(1) Compute the expression for the unit surface normals to the following surfaces:
(a) $\boldsymbol{\sigma}(u,v) = (a\cos u, a\sin u, bv)$,
(b) $\boldsymbol{\sigma}(u,v) = (u, v, u^2 - v^2)$,

(c) $\boldsymbol{\sigma}(u, v) = (u, v, 2uv)$.

(2) Compute the first and second fundamental forms for b) and c)

---

Now that we have some idea of the geometry of curves and surfaces, we can start looking at the grid generation. We have already seen that we can have structured and unstructured grids. Let us start by looking at ways to generate structured grids.

## Structured Grids

What are structured grids? We discretise our domain using grid points and associated volumes. If we are able to index our grid points using some scheme such that the neighbours of a given grid point can be inferred from its index, we have a structured grid.

**Algebraic grids.** The easiest way to generate structured grids is by using some algebraic expression to relate the physical coordinates to the computational coordinates. In the case of the problem presented by 6.8, we can generate an algebraic grid as shown in Figure 6.9. Here, we have chosen our grid points so that they are "proportionately" spaced in the $y$ direction. Let us work out the algebraic expression for this grid.

The top of the domain is a straight line that starts at the point $(0, h)$ and ends at $(L, H)$. The equation of this line is

$Y(x) = h + x(H - h)/L$. Now for a given $(x, y)$ the corresponding $(\xi, \eta)$ are obtained as

**(6.84)** $$\xi(x, y) = \frac{x}{L}$$

**(6.85)** $$\eta(x, y) = \frac{y}{Y} = \frac{y}{h + x(H - h)/L}$$

We can always get back from $(\xi, \eta)$ to $(x, y)$

**(6.86)** $$x(\xi, \eta) = \xi L$$

**(6.87)** $$y(\xi, \eta) = \eta(h + \xi(H - h))$$

Since we are able to go back and forth from the two coordinate systems, we will now transform our governing equation, say Laplace's equation to the new coordinate system. Employing chain rule we get for the first derivative

**(6.88)** $$\frac{\partial \phi}{\partial x} = \frac{\partial \phi}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial \phi}{\partial \eta}\frac{\partial \eta}{\partial x}$$

**(6.89)** $$\frac{\partial \phi}{\partial y} = \frac{\partial \phi}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial \phi}{\partial \eta}\frac{\partial \eta}{\partial y}$$

**(6.90)**

This will be written in a compact notation as

**(6.91)** $$\frac{\partial \phi}{\partial x} = \phi_x = \phi_\xi \xi_x + \phi_\eta \eta_x$$

**(6.92)** $$\frac{\partial \phi}{\partial y} = \phi_y = \phi_\xi \xi_y + \phi_\eta \eta_y$$

**(6.93)**

Then for the second derivative we have

**(6.94)**
$$\begin{aligned}\frac{\partial^2\phi}{\partial x^2} = \phi_{xx} = \frac{\partial}{\partial x}\left(\phi_x\right) = \frac{\partial}{\partial x}\left(\phi_\xi\xi_x + \phi_\eta\eta_x\right) \\ = \frac{\partial}{\partial x}\left(\phi_\xi\right)\xi_x + \phi_\xi\xi_{xx} + \frac{\partial}{\partial x}\left(\phi_\eta\right)\eta_x + \phi_\eta\eta_{xx} \\ = \phi_{\xi\xi}\left(\xi_x\right)^2 + 2\phi_{\xi\eta}\xi_x\eta_x + \phi_\xi\xi_{xx} + \phi_{\eta\eta}\left(\eta_x\right)^2 + \phi_\eta\eta_{xx}\end{aligned}$$

**(6.95)**
$$\begin{aligned}\frac{\partial^2\phi}{\partial y^2} = \phi_{yy} = \frac{\partial}{\partial y}\left(\phi_y\right) = \frac{\partial}{\partial y}\left(\phi_\xi\xi_y + \phi_\eta\eta_y\right) \\ = \frac{\partial}{\partial y}\left(\phi_\xi\right)\xi_y + \phi_\xi\xi_{yy} + \frac{\partial}{\partial y}\left(\phi_\eta\right)\eta_y + \phi_\eta\eta_{yy} \\ = \phi_{\xi\xi}\left(\xi_y\right)^2 + 2\phi_{\xi\eta}\xi_y\eta_y + \phi_\xi\xi_{yy} + \phi_{\eta\eta}\left(\eta_y\right)^2 + \phi_\eta\eta_{yy}\end{aligned}$$

So, Laplace's equation can be rewritten as

**(6.96)**
$$\begin{aligned}\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = \phi_{xx} + \phi_{yy} = \phi_{\xi\xi}\left(\xi_x^2 + \xi_y^2\right) + \phi_{\eta\eta}\left(\eta_x^2 + \eta_y^2\right) \\ + 2\phi_{\xi\eta}\left(\xi_x\eta_x + \xi_y\eta_y\right) + \phi_\xi\left(\xi_{xx} + \xi_{yy}\right) + \phi_\eta\left(\eta_{xx} + \eta_{yy}\right) = 0\end{aligned}$$

Clearly, we have a grid that conforms to our domain, however, the governing equations, even if it is just Laplace's equation can become quite complex.

**Elliptic Grids.** A grid generation scheme which employs elliptic differential equations to determine the transformation is called an elliptic grid generation scheme.

If you look back at our solution to differential equation, you will see we have reduced the differential equations to algebraic equations. Recall that in the potential flow problem, stream lines and potential lines are orthogonal to each other. They in fact form a mesh. They also satisfy Laplace equation [**Win67**]. We have already seen that we can solve Laplace equation on a square. Here is our plan of action:

(1) Assume $(\xi, \eta)$ satisfy Laplace's equation
(2) Transform these equations into the $(\xi, \eta)$ coordinates.
(3) solve for $(x, y)$ in the $(\xi, \eta)$ coordinates.

We already have the transformed equations (6.96). Since the coordinates satisfy Laplace's equation we can simplify to get

$$\phi_{\xi\xi}\left(\xi_x^2 + \xi_y^2\right) + 2\phi_{\xi\eta}\left(\xi_x\eta_x + \xi_y\eta_y\right) + \phi_{\eta\eta}\left(\eta_x^2 + \eta_y^2\right) = 0 \tag{6.97}$$

Let us recall how the chain rule works for the transformation of differentials from one coordinate system to another.

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix}. \tag{6.98}$$

Alternately, we can write

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix}. \tag{6.99}$$

We can see that

$$\begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix} = \begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix}^{-1} \tag{6.100}$$

This equation will allow us to replace the derivatives of $(\xi, \eta)$ with respect to $(x, y)$ in equation (6.97) to get the equations in terms of the derivatives of $(x, y)$ with respect to $(\xi, \eta)$.

We define the Jacobians of the transformations which are the determinants of the $2 \times 2$ matrices in equations (6.98) and (6.99) as follows

**(6.101)**
$$J = x_\xi y_\eta - y_\xi x_\eta$$

**(6.102)**
$$\Gamma = \xi_x \eta_y - \eta_x \xi_y$$

We then get the relationships

**(6.103)**
$$\xi_x = \frac{y_\eta}{J} \qquad \xi_y = -\frac{x_\eta}{J}$$
$$\eta_x = -\frac{y_\xi}{J} \qquad \eta_y = \frac{x_\xi}{J}$$

and

**(6.104)**
$$x_\xi = \frac{\eta_y}{\Gamma} \qquad x_\eta = -\frac{\xi_y}{\Gamma}$$
$$y_\xi = -\frac{\eta_x}{\Gamma} \qquad y_\eta = \frac{\xi_x}{\Gamma}$$

Why would we do this? Well, we know that we want the transformed coordinate system to be a unit square, say in a Cartesian coordinate system. Which means, we actually know the $(\xi, \eta)$. In order to have a transformation, we need to find for a given $(\xi, \eta)$ the corresponding $(x, y)$. So, we really need differential equations for $(x, y)$ in terms of $(\xi, \eta)$. To this end we divide equation (6.97) by $\Gamma^2$ and substitute from equations (6.104) to get

**(6.105)**
$$\left(x_\eta^2 + y_\eta^2\right) \phi_{\xi\xi} - 2\left(x_\xi x_\eta + y_\xi y_\eta\right) \phi_{\xi\eta} + \left(x_\xi^2 + y_\xi^2\right) \phi_{\eta\eta} = 0$$

We have the equations that need to be solved in the rectangular domain in the $(\xi, \eta)$ plane. We now need the boundary conditions to actually solve the equation. The rectangle in the $(\xi, \eta)$ plane has four sides. To each of these sides we need to map the sides of our original problem domain from the physical domain. For instance if the original problem were a unit circle (a circle with radius one) centred at the origin as shown in the Figure 6.14.

Let us look at the equations that we are about to solve.

$$ax_{\xi\xi} - 2bx_{\xi\eta} + cx_{\eta\eta} = 0 \tag{6.106}$$

$$ay_{\xi\xi} - 2by_{\xi\eta} + cy_{\eta\eta} = 0 \tag{6.107}$$

where

$$a = x_\eta^2 + y_\eta^2, \tag{6.108}$$

$$b = x_\xi x_\eta + y_\xi y_\eta \quad \text{and,} \tag{6.109}$$

$$c = x_\xi^2 + y_\xi^2 \tag{6.110}$$

We see that if we were to discretise these equations we would be dividing by $\Delta\xi$ and $\Delta\eta$. To eliminate this division we can take $\Delta\xi = \Delta\eta = 1$. The simplified equation with the Gauss-Seidel iterative scheme gives

$$\begin{aligned} x_{i,j}^{n+1} = d^n \big\{ & a^n (x_{i-1,j}^{n+1} + x_{i+1,j}^{n}) \\ & - 0.5 b^n (x_{i-1,j-1}^{n+1} - x_{i-1,j+1}^{n+1} - x_{i+1,j-1}^{n+1} + x_{i+1,j+1}^{n}) \\ & + c^n (x_{i,j-1}^{n+1} + x_{i,j+1}^{n}) \big\} \end{aligned} \tag{6.111}$$

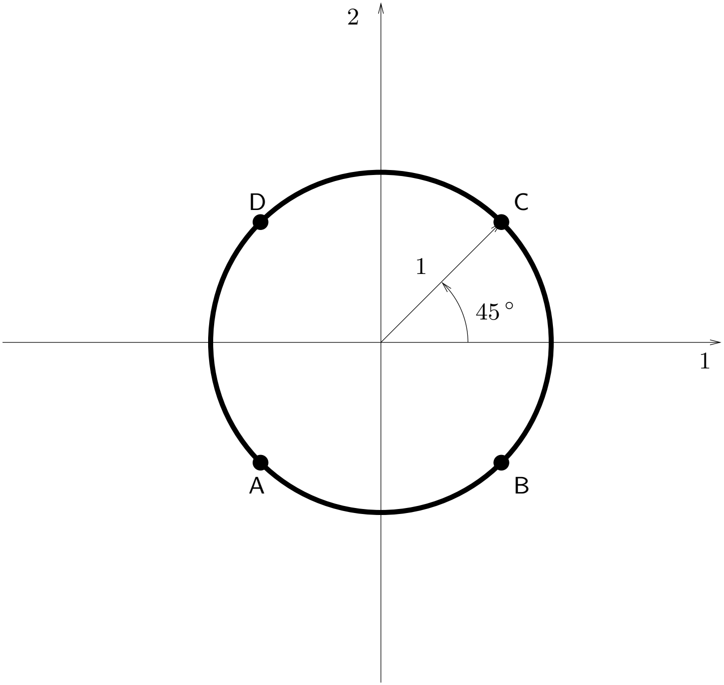

**Figure 6.14. Unit circle centred at the origin to be mapped into a square in the computational domain**

**(6.112)**
$$y_{i,j}^{n+1} = d^n \left\{ a^n(y_{i-1,j}^{n+1} + y_{i+1,j}^{n}) \right. \\ - 0.5b^n(y_{i-1,j-1}^{n+1} - y_{i-1,j+1}^{n+1} - y_{i+1,j-1}^{n+1} + y_{i+1,j+1}^{n}) \\ \left. + c^n(y_{i,j-1}^{n+1} + y_{i,j+1}^{n}) \right\}$$

where

$$a^n = 0.25 * \left[(\Delta_j x^n_{i,j})^2 + (\Delta_j y^n_{i,j})^2\right] \tag{6.113}$$

$$b^n = 0.25 * \left[(\Delta_i x^n_{i,j})(\Delta_j x^n_{i,j}) + (\Delta_i y^n_{i,j})(\Delta_j y^n_{i,j})\right] \tag{6.114}$$

$$c^n = 0.25 * \left[(\Delta_i x^n_{i,j} + (\Delta_i y^n_{i,j})^2\right] \tag{6.115}$$

$$d^n = \frac{1}{2[a^n + c^n]} \tag{6.116}$$

and $\Delta$ is defined such that

$$\Delta_\alpha S_\alpha = S_{\alpha+1} - S_{\alpha-1}, \quad S = x, y, \quad \alpha = i, j. \tag{6.117}$$

Here, $n$ is the iteration level. The coefficients are not changed during the current iteration.

As we had indicated at the beginning of this section, we decided to use Laplace's equation to determine $(\xi, \eta)$ so that the resulting coordinate lines are orthogonal to each other. This we obtained from our analogy with potential lines and stream lines. Now we know that streamlines are drawn towards sources and pushed away from sinks. So, if we want our grid lines to be clustered in a region due to the solution having large gradients in that region, then one can add a sink appropriately and sources appropriately so that such a clustering occurs. Unlike in case of the unstructured grids, it is difficult to perform very localised grid refinement without affecting a reasonably large region around the area of interest.

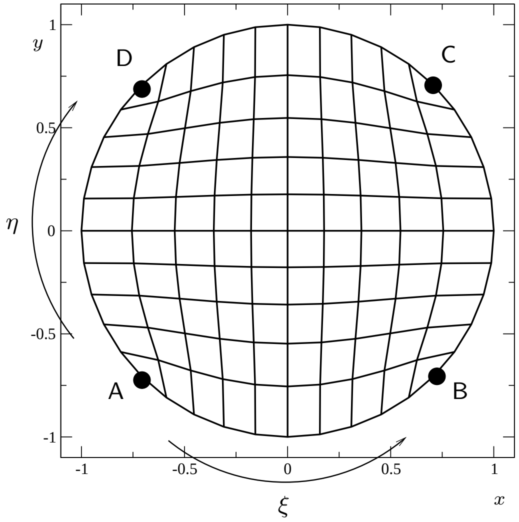

**Figure 6.15. Constant $\xi$ lines and constant $\eta$ lines generated in the unit circle using elliptic grid generation. This is a $11 \times 11$ grid.**

**Parabolic and Hyperbolic Grids.** Similar to the last section if the nature of the differential equations employed to generate the grid are hyperbolic or parabolic then the grid generation scheme is called a hyperbolic grid generation scheme or a parabolic grid generation scheme.

We will look at hyperbolic grid generators first. These schemes can be used generate grid by themselves. However, they are increasingly used to generate a hybrid grid where hyperbolic grids are used near solid boundaries and and any other kind of grid generation scheme including unstructured grids are used away from the boundary.

One simple way to generate a hyperbolic grid is to start at the surface where the grid is required and to travel along a perpendicular to the surface by a set distance, $\Delta\zeta$. The end points of all of these normals now defines a constant $\zeta$ plane. This should remind the reader of two things. First the wave equation and its properties we studies earlier. The second is algebraic grid generation. Once we have offset by $\Delta\zeta$ we are now in a position to determine the new normal directions and then take another $\Delta\zeta$ step. In this fashion, one can generate $\zeta$ grid lines and the corresponding $\xi - \eta$ planes. As in the wave equation of course, one has to be careful that the $\zeta$-grid lines do not intersect. One solution is to add some dissipation terms which will convert the equations to parabolic equations, resulting in parabolic grid generation.

## Generating Two Dimensional Unstructured Grids

Why would we want to generate unstructured grids. The common reason is that for complex geometries it is easier to generate unstructured grids rather than structured grids. At the end of the chapter we will summarise the relative advantages and disadvantages for the various grid generation techniques.

As opposed to structured grids, unstructured grids will require us to store data on neighbours and other such information that can be inferred in structured grid.

Unstructured grids come in many flavours. We will restrict ourselves to two popular ones: unstructured triangular meshes and unstructured Cartesian meshes. The latter is not to be confused with the regular Cartesian coordinate system, though it is fashionable to refer to an unstructured Cartesian mesh as just a Cartesian mesh.

**Triangulation.** For the domain given by Figure 6.8 an unstructured triangulation is given in Figure 6.10. How do we automate the generation of these triangles?

Before we start developing an algorithm to solve the triangulation problem, let us define a few terms.

**Node:** It is a point the domain. The vertices of various polygons will typically be nodes. Many of them together may be used to define the boundary of the region of interest.

**Edge:** It is an oriented line segment connecting two nodes: the StartNode and the EndNode. A triangle, for example is three nodes connected by three edges.

In order to start the grid generation, we need to specify the boundary of the region to be triangulated. This can be done by prescribing a sequence of nodes or edges. Let us say we prescribe a sequence of nodes. In two dimensions, a region can be described by a loop. The nodes of a loop are prescribed in order. This means that an edge connects to consecutive nodes in a given loop. The orientation of the edges is such that as we look from the StartNode towards the Endnode, the domain of interest is to our left. This is done by prescribing our outermost loop in a counter-clockwise direction. This is very critical, as it will help us figure out whether we have a point in the region of interest or not. Put another way, if you are going to give me an enclosure, you have to give me a method by which I can decide whether a point is in the enclosure or not. We come to the following understanding:

> The domain in two dimensions is defined employing loops. The loops are oriented in such a fashion as to have the domain always to the left of any segment of the loop[see Figure 6.16.

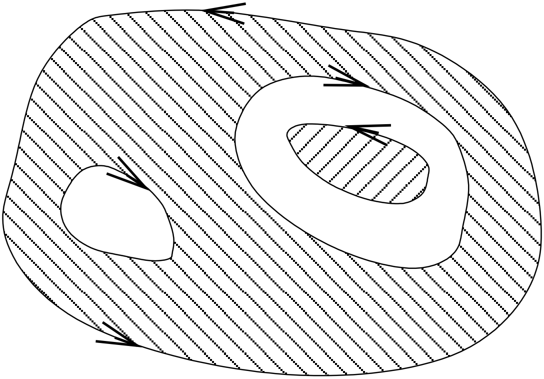

**Figure 6.16. The region to be triangulated is shown hatched. The loop is oriented such that the domain is always to the left**

We will look at an algorithm called the boundary triangulation algorithm. The first step in our grid generation is

**Step One:** Given the loops forming the domain by appropriately ordered nodes, create the edges from consecutive nodes of a loop. The set of edges form the boundary of the approximation to the domain.

We now have a set of edges, $\mathcal{E}$, and a set of nodes, $\mathcal{N}$, associated with those edges. We are ready to work out a triangulation algorithm.

> **Step Two:** While the set of edges, $\mathcal{E}$, is not empty, pick an edge $e$.

We will now employ this edge to generate a triangle.

> **Step Three:** Find the set, $\mathcal{L}$, of all nodes that are to be triangulated and the to left our edge $e$.

Given the edge $e$, we need to find a node to form a triangle. Clearly any node that is to the right of the edge will form a triangle that is outside the domain. So, we form $\mathcal{L}$ from which to pick a node. We now proceed to find a candidate node from the set $\mathcal{L}$ to form a triangle.

> **Step Four:** Find the nearest node from the set $\mathcal{L}$ and check to see if a triangle can be formed
>
> **Check One:** Is the area of the triangle too small?
>
> **Check Two:** Do the new Edges formed intersect any of the existing edges?

The first check to to make sure that we do not pick three collinear points. The second check is to make sure that the triangles do not overlap. Remember, in our chapter on representations, we saw that non-overlapping domains gave us functions that are independent of each other. So from the list $\mathcal{L}$ of the nodes we will find a node, $n$, that allows us to form a triangle.

> **Step Five:** Form the triangle with the node $n$. Check whether new edges have to be added. If a new edge is added, its orientation is such that the interior of the

triangle is to the left of the edge. This can be achieved by ensuring that the StartNode of the new edge is the EndNode of the existing edge or vice-versa.

**Step Six:** We remove edge $e$ from the set $\mathcal{E}$. We check to see if any of the edges formed employing the end points of $e$ and the node $n$ already exist in the set $\mathcal{E}$. Any such edge is removed from the set $\mathcal{E}$.

**Step Seven:** The loop was opened out when edges were dropped. The new edges that were created to form the triangle are reversed in orientation and added to $\mathcal{E}$. This results in the loop being closed again.

**Step Eight:** Go back to **Step Two** and form another triangle

This is a simple method to generate unstructured triangular meshes. Let us look at an example as given in Figure 6.17. I would suggest that you use a circle to test your code when you are first developing it. The circle is easier.

For the sake of this discussion (i) will be node "i". So for example, (3) is node "3". i-j will be the edge connecting (i) to (j). That is, $1-2$ will be an edge going from (1) to (2) which has the opposite orientation of $2-1$ which is an edge going from (2) to (1).

$\mathcal{N} = \{(1), (2), (3), (4), (5), (6), (7), (8)\}$ is the set of nodes from which we get the set of edges to be triangulated. This set is $\mathcal{E} = \{1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 8-1\}$.

**Pick an edge:** The set $\mathcal{E}$ is not empty, we pick the first edge from it: $e = 1-2$.

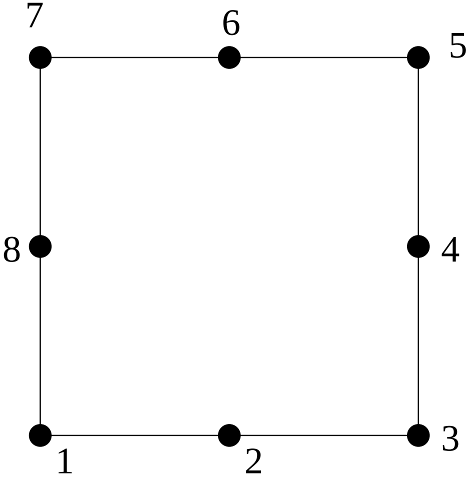

Figure 6.17. This is the initial domain given as a counter-clockwise loop of nodes labeled 1, 2, 3, 4, 5, 6, 7, 8, 1

**Form the left list:** The list of nodes, $\mathcal{L}$, which are to the left of $e$ is formed: $\mathcal{L} = \{(4), (5), (6), (7), (8)\}$

**Pick nearest node:** The node closest to $e$ is (8).

- **Triangle? :** With this it is possible for us to form the triangle made up of the edges $1-2$, $2-8$, $8-1$.
- **Old Edges? :** of these besides $e$, 8-1 is an existing edge from $\mathcal{E}$.
- **Intersect:** The new edge $2-8$ does not intersect any existing edges.
- **Triangle Size:** Triangle is not approaching zero area.

**Form Triangle:** We can form the triangle $\{1-2, 2-8, 8-1\}$.

**Housekeeping:** –

**first:** We will remove edges $1-2$ and $8-1$ from the list of edges $\mathcal{E}$.

**second:** The new edge in the triangle is $2-8$. We add the opposite of this edge $8-2$ to $\mathcal{E}$.

**third:** We remove the node (1) from $\mathcal{N}$ as it has no edge connected to it.

**State 1:** $\mathcal{E} = \{2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 8-2\}$, $\mathcal{N} = \{(2), (3), (4), (5), (6), (7), (8)\}$, and $\mathcal{T} = \{(1-2, 2-8, 8-1)\}$

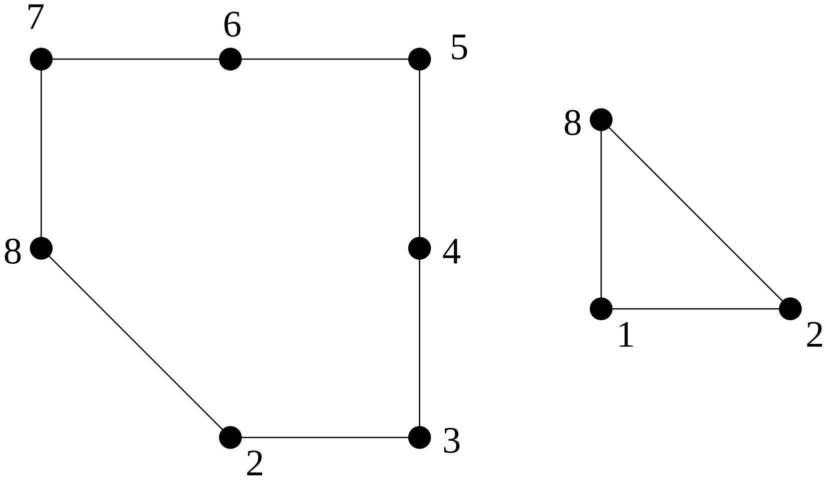

Figure 6.18. The domain remaining to be triangulated and the new triangle formed, Note that the edge $8-2$ is new

We can see what we have at the end of this effort in Figure 6.18. We proceed along the same lines.

**Pick an edge:** The set $\mathcal{E}$ is not empty, we pick the first edge from it: $e = 2 - 3$.

**Form the left list:** The list of nodes, $\mathcal{L}$, which are to the left of $e$ is formed: $\mathcal{L} = \{(4), (5), (6), (7), (8)\}$

**Pick nearest node:** The node closest to $e$ is (4).

**Triangle? :** With this it is possible for us to form the triangle made up of the edges $2 - 3$, $3 - 4$, $4 - 2$.

**Old Edges? :** of these besides $e$, $3 - 4$ is an existing edge from $\mathcal{E}$.

**Intersect:** The new edge $4 - 2$ does not intersect any existing edges.

**Triangle Size:** Triangle is not approaching zero area.

**Form Triangle:** We can form the triangle $\{2 - 3, 3 - 4, 4 - 2\}$.

**Housekeeping:** –

**first:** We will remove edges $2 - 3$ and $3 - 4$ from the list of edges $\mathcal{E}$.

**second:** The new edge in the triangle is $4 - 2$. We add the opposite of this edge $2 - 4$ to $\mathcal{E}$.

**third:** We remove the node (3) from $\mathcal{N}$ as it has no edge connected to it.

**State 2:** $\mathcal{E} = \{4 - 5, 5 - 6, 6 - 7, 7 - 8, 8 - 2, 2 - 4\}$, $\mathcal{N} = \{(2), (4), (5), (6), (7), (8)\}$, and $\mathcal{T} = \{(1 - 2, 2 - 8, 8 - 1), (2 - 3, 3 - 4, 4 - 2)\}$

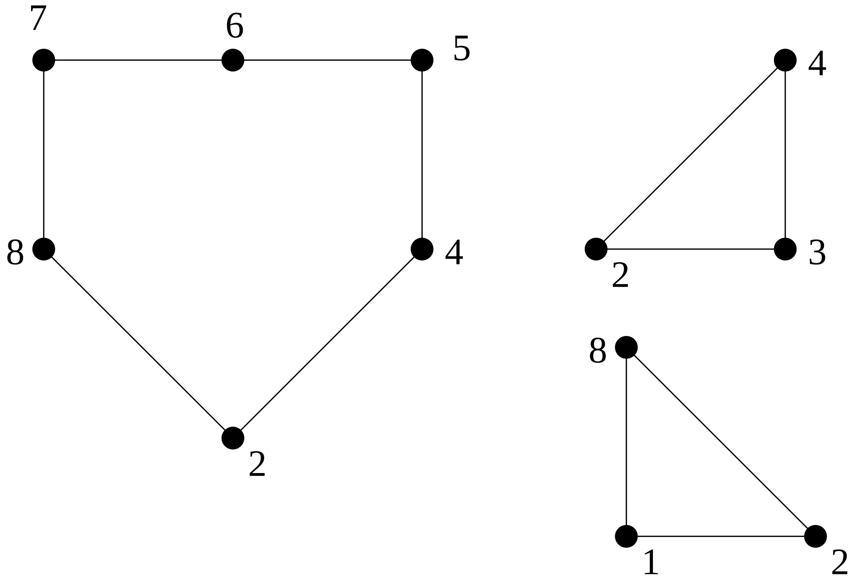

Figure 6.19. The remaining domain to be triangulated and the two triangles formed so far. Note the new edge $2-4$

We can see what we have at the end of this effort in Figure 6.19. Again, we proceed along the same lines.

**Pick an edge:** The set $\mathcal{E}$ is not empty, we pick the first edge from it: $e = 4-5$.

**Form the left list:** The list of nodes, $\mathcal{L}$, which are to the left of $e$ is formed: $\mathcal{L} = \{(2), (6), (7), (8)\}$

**Pick nearest node:** The node closest to $e$ is (6).

**Triangle? :** With this it is possible for us to form the triangle made up of the edges $4-5$, $5-6$, $6-4$.

**Old Edges? :** of these besides $e$, $5-6$ is an existing edge from $\mathcal{E}$.

**Intersect:** The new edge $6-4$ does not intersect any existing edges.

**Triangle Size:** Triangle is not approaching zero area.

**Form Triangle:** We can form the triangle $\{4-5, 5-6, 6-4\}$.

**Housekeeping:** –

**first:** We will remove edges $4-5$ and $5-6$ from the list of edges $\mathcal{E}$.

**second:** The new edge in the triangle is $6-4$. We add the opposite of this edge $4-6$ to $\mathcal{E}$.

**third:** We remove the node (5) from $\mathcal{N}$ as it has no edge connected to it.

**State 3:** $\mathcal{E} = \{6-7, 7-8, 8-2, 2-4, 4-6\}$, $\mathcal{N} = \{(2), (4), (6), (7), (8)\}$, and $\mathcal{T} = \{(1-2, 2-8, 8-1), (2-3, 3-4, 4-2), (4-5, 5-6, 6-4)\}$

We can see what we have at the end of this effort in Figure 6.20. Again, we proceed along the same lines.

**Pick an edge:** The set $\mathcal{E}$ is not empty, we pick the first edge from it: $e = 6-7$.

**Form the left list:** The list of nodes, $\mathcal{L}$, which are to the left of $e$ is formed: $\mathcal{L} = \{(2), (4), (8)\}$

**Pick nearest node:** The node closest to $e$ is (8).

**Triangle? :** With this it is possible for us to form the triangle made up of the edges $6-7$, $7-8$, $8-6$.

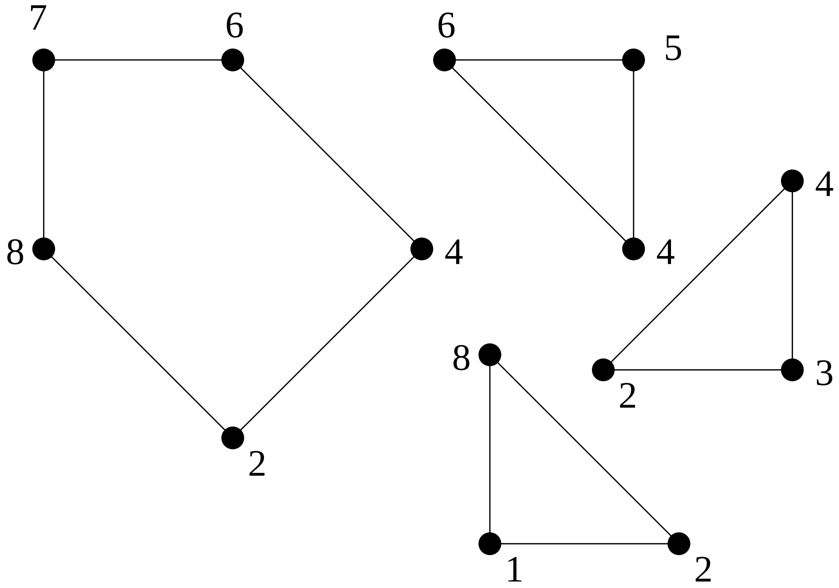

**Figure 6.20. The remains of the domain to be triangulated and the three triangles formed thus far. A new edge $4-6$ has been added**

- **Old Edges? :** of these besides $e$, $7-8$ is an existing edge from $\mathcal{E}$.
- **Intersect:** The new edge $8-6$ does not intersect any existing edges.
- **Triangle Size:** Triangle is not approaching zero area.

**Form Triangle:** We can form the triangle $\{6-7, 7-8, 8-6\}$.

**Housekeeping:** –

**first:** We will remove edges $6-7$ and $7-8$ from the list of edges $\mathcal{E}$.

**second:** The new edge in the triangle is $8-6$. We add the opposite of this edge $6-8$ to $\mathcal{E}$.

**third:** We remove the node (7) from $\mathcal{N}$ as it has no edge connected to it.

**State 4:** $\mathcal{E} = \{8–2, 2–4, 4–6, 6–8\}$, $\mathcal{N} = \{(2), (4), (6), (8$ and $\mathcal{T} = \{(1-2, 2-8, 8-1), (2-3, 3-4, 4-2), (4-5, 5-6, 6-4), (6-7, 7-8, 8-6)\}$

We can see what we have at the end of this effort in Figure 6.21. Again, we proceed along the same lines.

**Pick an edge:** The set $\mathcal{E}$ is not empty, we pick the first edge from it: $e = 8-2$.

**Form the left list:** The list of nodes, $\mathcal{L}$, which are to the left of $e$ is formed: $\mathcal{L} = \{(4), (6)\}$

**Pick nearest node:** The node closest to $e$ is (4).

**Triangle? :** With this it is possible for us to form the triangle made up of the edges $8-2$, $2-4$, $4-8$.

**Old Edges? :** of these besides $e$, $2-4$ is an existing edge from $\mathcal{E}$.

**Intersect:** The new edge $4-8$ does not intersect any existing edges.

**Triangle Size:** Triangle is not approaching zero area.

**Form Triangle:** We can form the triangle $\{8-2, 2-4, 4-8\}$.

**Housekeeping:** –

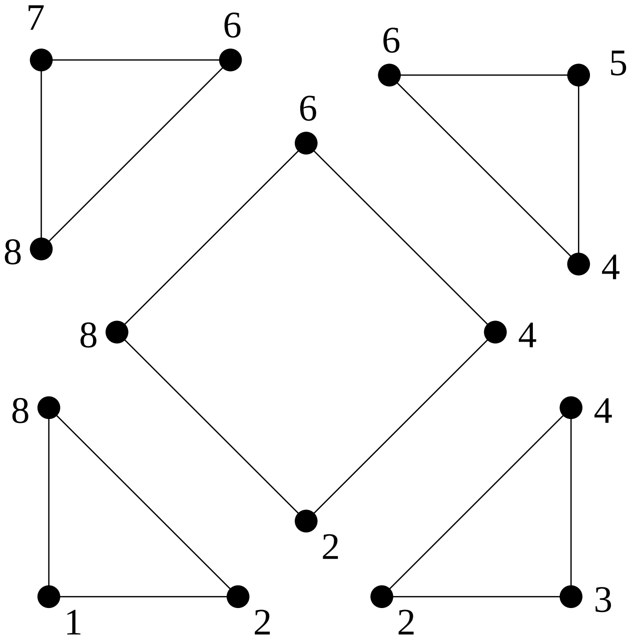

**Figure 6.21. The domain remaining to be triangulated and the four triangles formed. The edge $6-8$ is new**

**first:** We will remove edges $8-2$ and $2-4$ from the list of edges $\mathcal{E}$.

**second:** The new edge in the triangle is $4-8$. We add the opposite of this edge $8-4$ to $\mathcal{E}$.

**third:** We remove the node (2) from $\mathcal{N}$ as it has no edge connected to it.

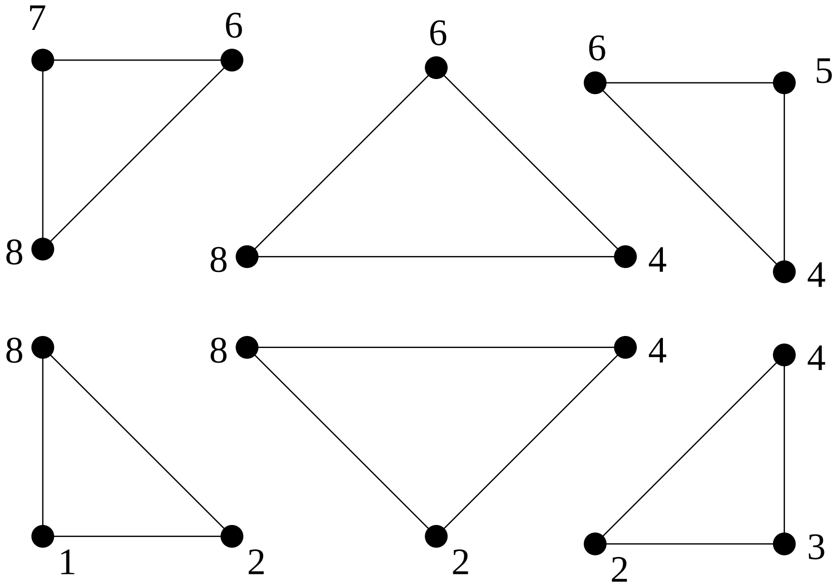

**Figure 6.22. All the triangles are formed. $8-4$ was the last new edge to be added**

**State 5:** $\mathcal{E} = \{4-6, 6-8, 8-4\}$, $\mathcal{N} = \{(4), (6), (8)\}$, and $\mathcal{T} = \{(1-2, 2-8, 8-1), (2-3, 3-4, 4-2), (4-5, 5-6, 6-4), (6-7, 7-8, 8-6), (8-2.2-4, 4-8)\}$

Again, we proceed along the same lines and the final triangle is formed. We can see what we have at the end of this effort in Figure 6.22.

**Assignment 6.7** Generate a triangular mesh on a circular region. The input file should consist of

(1) Number of loops in the problem domain.
(2) Number of points/nodes in the zeroth loop.
(3) $x_0, y_0$
(4) $x_1, y_1$
(5) $\vdots$
(6) $x_n, y_n$
(7) Number of points/nodes in the first loop.
(8) $\vdots$
(9) Number of points in the last loop.
(10) data for last loop.

Once you have it working for a circle, try it out for a square, rectangle, and shapes with holes in them.

---

A close inspection of the algorithm shows that we essentially deal with edges and nodes to form a triangle. In fact, once we have picked an edge, we are hunting for nodes in the left list. This gives us a cue that we can specify nodes other than the ones (along with the corresponding edges) that we use to prescribe the domain to be triangulated. These nodes then will participate in the griding though they may not be associated initially with any edge. Clearly the nodes have to be in the interior of the domain, otherwise, they will not figure in the formation of a viable edge.

We now have a method by which we can generate a triangulation based on the boundaries specified along with the nodes provided. How do we ensure that the triangles are of the quality that we want? Which brings us to the questions: What is a quality triangle? How do we measure this quality? If we are given no conditions, what is an ideal triangle in two dimensions. Most people would zero in on an equilateral triangle.

An equilateral triangle has

- equal angles,
- equal sides,
- ...
- largest enclosed area by a triangle for a given perimeter.

We could use the last point mentioned in the list as a point of reference. The area of the equilateral triangle of side "a" is given by

$$A = \frac{1}{2} a \frac{\sqrt{3}}{2} a = \frac{\sqrt{3}}{4} a^2. \tag{6.118}$$

The perimeter is $S = 3a$ for the equilateral triangle. We can define a non-dimensional parameter "Q" representing the quality of the triangle as

$$Q = 12\sqrt{3} \frac{A}{S^2} \tag{6.119}$$

$Q$ would be 1 for an equilateral triangle and less then 1 for other triangles. For three collinear points $Q = 0$.

Now that we have figured out a way to generate grids employing the boundary nodes and we are able to assess the quality of a triangle, we are in a position to see if it is possible to improve the overall quality of the triangulation.

With out disturbing the nodes if we want to try to improve the triangulation, we can only move the edges around using the same nodes.

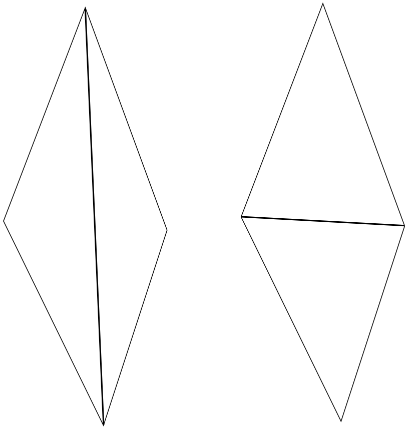

**Figure 6.23. Two Possible Triangulations with the Four Nodes**

Figure 6.23 shows two possible triangulations given four nodes. The question is which of these two triangles is better. The "max-min" criterion tells us that the preferable triangulations of the two is the one which has the larger smallest angle. Once we have the mechanism to check for the better triangulation we can now generate an algorithm to improve the triangulation.

(1) For a given node "A" find all the triangles that have that node as a vertex.
(2) There may be two triangles on either side of any triangle that share the node "A".

(3) Consider a triangle "T". Pick the triangle opposite to the given triangle.

(4) Apply the swap criterion and consider whether the edge should be swapped. Repeat this for all the triangles shared by the node.

(5) Repeat this for all the nodes.

(6) Repeat this whole process till no swap occurs.

We will now look at a method to insert nodes into our triangulation to increase the "richness" of the triangulation.

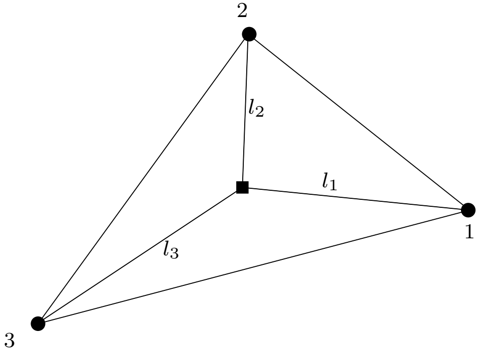

**Figure 6.24. Inserting a node at the centroid of a triangle, the existing nodes are shown using filled circles, the new node is shown with a filled square**

Consider the Figure 6.24. We want to decide whether to insert a node at the centroid and form the three triangles shown.

(1) At each of the nodes one can define a node spacing function (nsf).

(2) One can look at the centroid of a triangle and define the node spacing function at the centroid in terms of the node spacing function at vertices of the triangle.
(3) Let $l_i$ be the distance between the centroid and the $i^{th}$ vertex of the triangle.
(4) If the min$(l_i)$ is greater than the nsf at the centroid, insert the centroid as a new node in the triangulation.
(5) Form three triangles by joining the vertices of the triangle to the centroid and forming new edges and hence the new triangles.

Having inserted the nodes one can redo the swapping algorithm to improve the new triangulation. An added mechanism to improve the triangulation at the end of the swap sweep with a given node is to move that node to the centroid of the polygon formed by the triangles that share that node. This should be done only if that polygon is convex. Otherwise, the centrod may lie outside of the polygon. This step can be incorporated into the swapping algorithm. Note that the boundary nodes and the user specified nodes should not be moved. Only the nodes inserted by the insertion algorithm can really be moved.

There are other ways by which one can refine the given grid. Referring to Figure 6.25, we can insert nodes on the three edges making up the triangle and form a triangle using the three new nodes. As can be seen from the figure, this leaves the original triangle split into four smaller and similar triangles. They are congruent to each other and similar to the parent triangle. The quality of the triangulation does not drop drastically due to this process. However, the swapping and smoothing process can be attempted after the refinement.

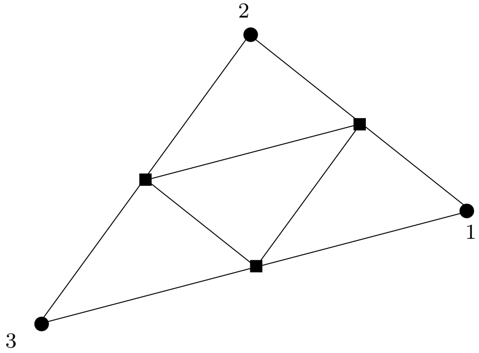

**Figure 6.25. Replacing a given triangle with four smaller similar triangles. The new nodes are indicated using filled squares.**

Since this kind of node insertion results in nodes on the edge, the neighbouring triangle needs to be modified to conform to the new set of triangles.

This can be done quite easily as shown in Figure 6.26. Due to the refinement process if the adjacent triangle has a node on its edge, it is split into two by joining the node opposite to that edge to the new node as shown by the thicker line in the figure. If two of the edges of the triangle happen to have nodes on them due to two neighbours being refined then that triangle is also refined by splitting into four.

It is clear that if we have an initial coarse mesh generated using boundary triangulation that one can generate a hierarchical mesh by refining each triangle into four as required. This process can be represented by a quad tree since each triangle typically has four children. This can in fact be used for adaptive

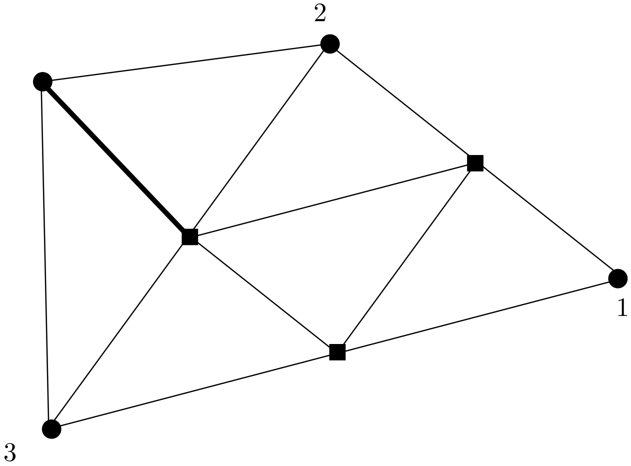

**Figure 6.26. Triangle 1-2-3 is refined as shown in Figure 6.25. An existing triangle on edge 2-3 becomes a quadrilateral. This is split into two triangles by adding an edge from the new node. The objective is conform without the introduction of any new nodes**

griding, where the decision to refine may be based on a computed solution. The advantage with this unstructured grid is that the refinement can be performed at the place where it is required instead of having to refine the whole grid.

It should be clear that this kind of refinement of breaking the geometrical entity into four children is also possible for quadrilaterals. In particular, quadrilaterals that are part of a Cartesian mesh.

**Cartesian Meshes.** Figure 6.27 shows an unstructured Cartesian mesh on the trapezoidal domain. Figure 6.28 shows the zoomed-in part of the mesh. The strategy here is quite clear. Any cell that is not completely inside our domain, in this case the trapezium, is subdivided. In the two dimensional case, it is subdivided into four parts. The same test is applied to the new cells that are formed. This process is continued till the geometry is captured to "our satisfaction". Clearly, we will have to place a lower limit on the size of any cell. Of course, we have a limit on how many cells we are able to handle with the computational resources available.

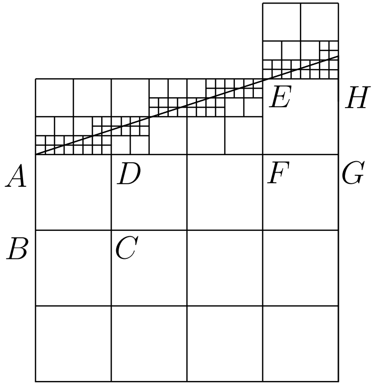

**Figure 6.27. Trapezoidal domain on which Laplace equation is to be solved - with a Unstructured Cartesian mesh**

The fundamental technology required for the Cartesian meshes is that of a tree based data structure. We form what are called spatial data structures[**Sam03**]. In the two dimensional case, the parent cell is broken into four sub-cells. This process can clearly be captured by a quad tree. The issue simply is: how

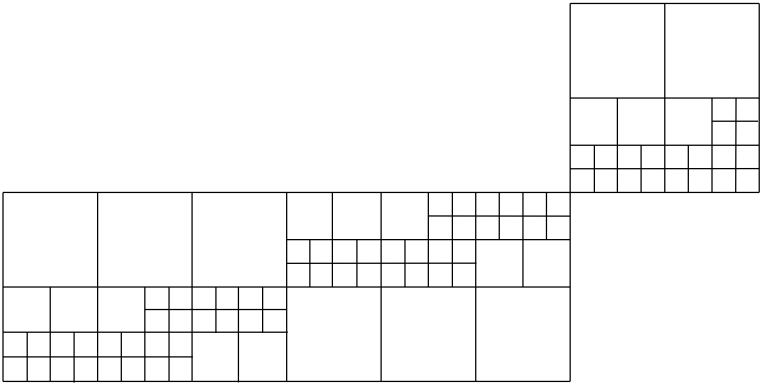

**Figure 6.28. Zoomed part of the Unstructured Cartesian mesh from Figure 6.27**

do we decide when a cell needs to be split. There can be three possible reasons.

(1) We need a finer grid because the problem being solved requires a richer mesh - refined representation of the domain to resolve the physics better.

(2) We need to resolve the boundary to the necessary degree - refine the representation of the boundary to capture the domain better and consequently resolve the physics better.

(3) Finally, we may need to split a cell simply because its neighbour has been split many levels down. For example, consider the squares $ABCD$ and $EFGH$ in Figure 6.27. The square above each of these has been divided to three levels in order to capture the boundary. From the point of implementing a finite volume solver on this

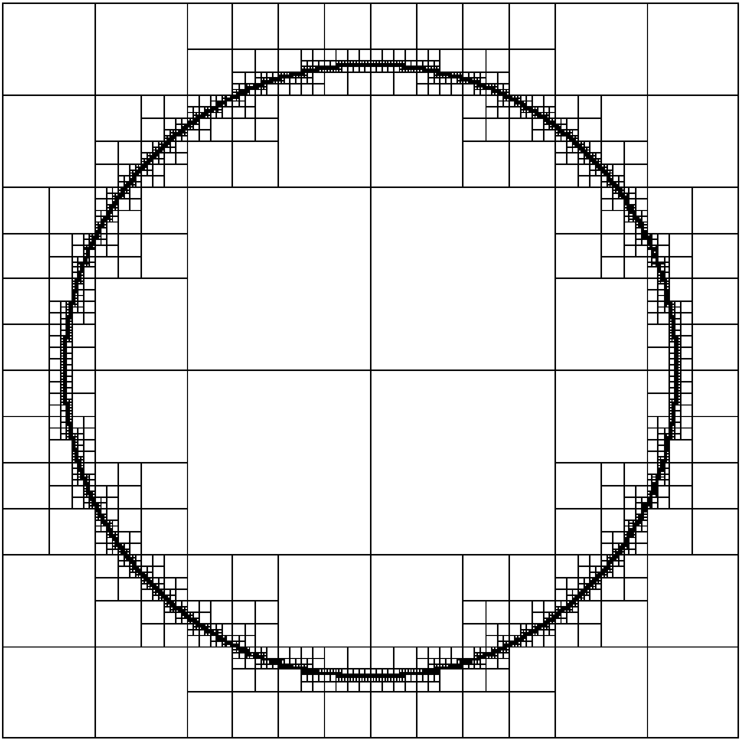

Figure 6.29. An unstructured Cartesian mesh generated to capture the circle $x^2 + y^2 = 1$, the mesh goes down ten levels

mesh, the disparity in cell size is large. So, $ABCD$ and $EFGH$ are candidates to be split.

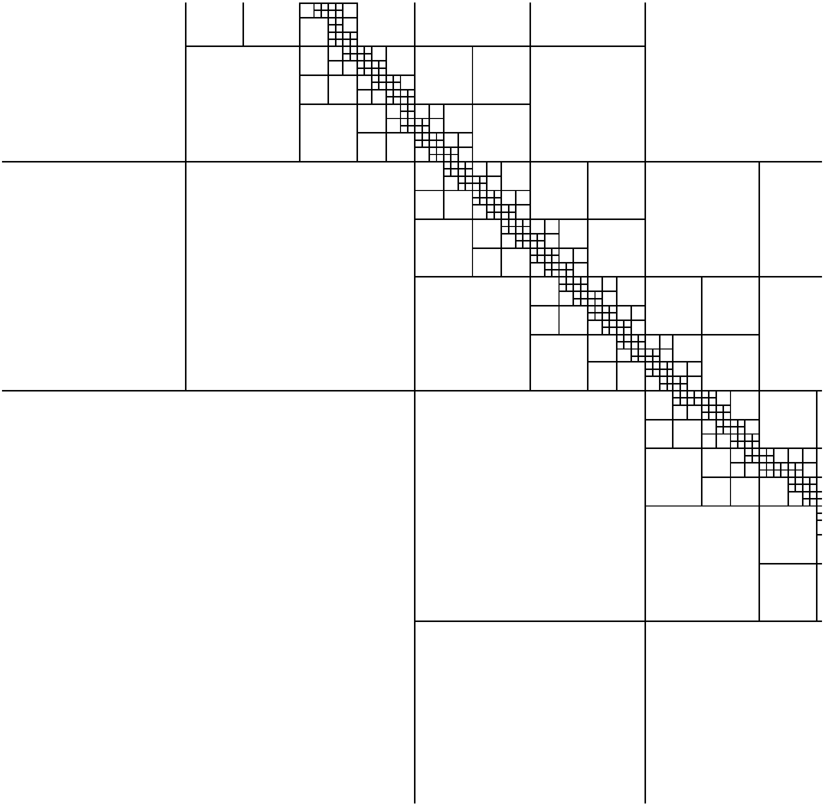

Figure 6.30. A zoomed in version of Figure 6.29 showing that the circle is not actually drawn in the figure and is really captured by the mesh, giving the illusion of it being drawn

Figure 6.29 shows another example of the unstructured Cartesian mesh. Figure 6.30 shows a zoomed in version of the same figure. As you can see, the circle has not been drawn, though from the mesh it (the circle that is) is apparent.

**Unstructured Quadrilateral Meshes.** Before we go on to structured grids we will see one more kind of an unstructured grid. Like the Cartesian mesh this is also an unstructured mesh. However, it is akin to an unstructured triangulation. The easiest way to generate an unstructured quadrilateral mesh is to first generate an unstructured triangulation and from there generate a quadrilateral mesh. There are many ways by which a triangulation can be converted to a quadrilation. One way is to choose two triangles and remove a shared edge. The other, more robust method is two decompose a triangle into three quadrilaterals. This is robust since a given triangle is decomposed without having to hunt for a mating triangle. Figure 6.31, shows how this

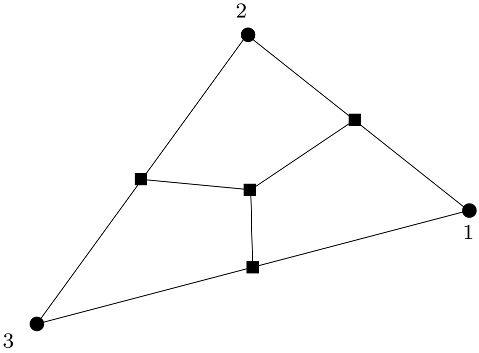

**Figure 6.31. Three quadrilaterals formed from a triangle. The mid-points are connected to the centroid**

can be done. This scheme, however, results in the insertion of nodes on the sides. This is particularly important on the boundary of the domain. Given no other constraints, we would like all the quadrilaterals to be squares. Laplacian smoothing, as was done with unstructured triangulation, can be used to improve the quality of the quadrilaterals.

## Three Dimensional Problems

How do we handle three dimensional domains? One is to see if we can generate a two dimensional grid and stack it somehow in the third direction. In reality, in the third dimension one can generate an algebraic grid. This allows us to cluster the grids very easily. Note that the two dimensional grid that is stacked can be of any nature: structured, unstructured...

We can also generate tetrahedra in three dimensions. The idea here is to describe the bounding surfaces in terms of a triangular mesh and then to fill the volume with tetrahedra. The advantage with this as with all unstructured meshes is that we can capture very complex geometries very easily. The disadvantage is that it requires more information about the grid to be stored as in the two dimensional case.

One could also generate an unstructured Cartesian mesh in three dimensions. This would involve dividing a given cube into eight octets. A tree representation would be made up of node representing the given cube and eight child nodes representing the eight child cubes of the parent cube. This would be an Oct-tree. As was done in two dimensions, the grid would be initially refined to pick up the boundaries to our satisfaction. Subsequently, one can refine the grid as required by the solution.

**Stacking Two-dimensional grids.** Let us consider the simple case of a cylinder. We can generate the grid on the circular face of the cylinder using any of the techniques that we have seen so far. Figure 6.15 is would do quite nicely. This grid provides us with $(x_i, y_j)$ in a circular region for a $7 \times 7$ grid. If the length of the cylinder is $L$, and we want $n$ grid points uniformly along the length, we need to define $z_k = k * L/(n-1)$. The grid $(x_i, y_j, z_k)$ then discretises the cylindrical volume.

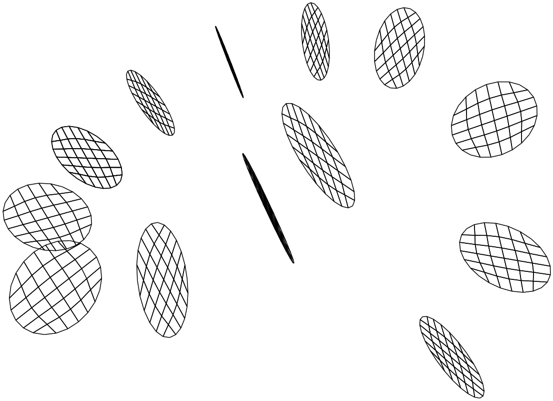

**Figure 6.32. A $7 \times 7$ elliptic grid generated in a unit circle in the xy plane is scaled and stacked along the helix $(0.2t, 0.1\cos t, 0.1\sin t)$ to generate a three-dimensional grid in a helical tube. There are twenty eight grid planes stacked along the helix**

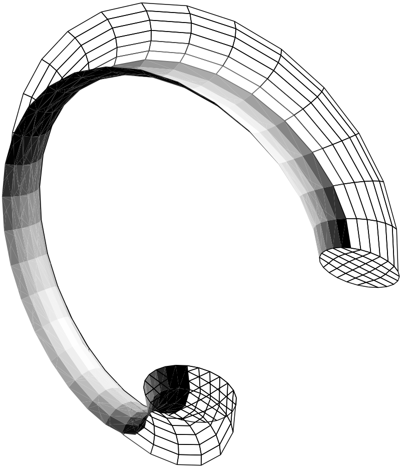

Figure 6.33. A $7 \times 7$ elliptic grid generated in a unit circle in the xy plane is scaled and stacked along the helix $(0.2t, 0.1\cos t, 0.1\sin t)$ to generate a three-dimensional grid in a helical tube. There are twenty eight grid planes stacked along the helix. Only grids on the inlet, exit, bottom, and side coordinate surfaces are shown for clarity

Now we can extend this further. Consider a situation where the axis of the cylinder is some space curve, say a helix. In this case, at any point along the helix, the Frenet-Serret triad are known. The grid plane that we have generated needs to be translated to that point and rotated so that the normal to the plane being stacked is along the tangent at that point. Whether the orientation of the grid plane is rotated in plane to account for the torsion really depends on the user. Grids generated in this fashion are shown in Figures 6.32, 6.33, 6.35, and 6.36. Figure 6.35 uses a curve with no torsion, resulting in the quarter of a torus. A full torus is show in Figure 6.36.

Now consider a different extension of the same cylinder problem. The cylindrical surface of the cylinder is a surface of revolution. How would we handle other objects of revolution. One way would be to generate the grid as done before on a circle and the grid is stacked, it can also be scaled. This would also work for a conical nozzle as shown in Figure 6.34. In this case,

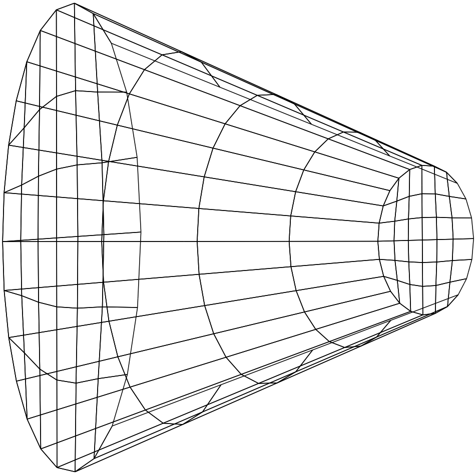

**Figure 6.34. A $7 \times 7$ elliptic grid generated in a unit circle in the xy plane is scaled and stacked in the $z$-direction to generate a three-dimensional grid in a converging conical nozzle. There are five grid planes along the $z$-axis. Only grids on the inlet, exit, bottom, top, and back faces are shown for clarity**

not only do we stack the grid along the $z$-coordinate direction, we also scale the grid down to match the nozzle dimensions.

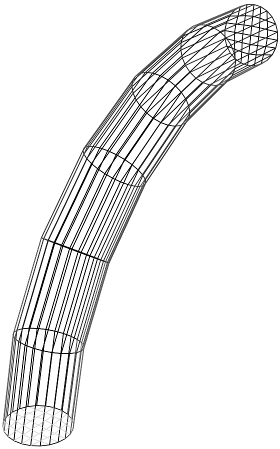

**Figure 6.35. A $7 \times 7$ elliptic grid generated in a unit circle in the xy plane is scaled and stacked in the $\theta$-direction to generate a three-dimensional grid in a torus. A quarter of the torus is shown. There are seven grid planes along the $\theta$-axis. Only grids on the inlet, exit, and boundaries are shown for clarity**

The other possibility is to generate a grid between the generating curve for the surface of revolution and the axis about which it is revolved. This is a two dimensional grid. This grid can now be stacked in appropriate intervals in the $\theta$-direction. That is the grid can be revolved about the axis so as to fill the volume.

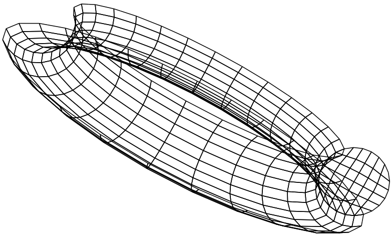

Figure 6.36. A $7 \times 7$ elliptic grid generated in a unit circle in the xy plane is stacked along a circle to generate a three dimensional grid in a full torus. The $7 \times 7$ grid is shown. Grids are also shown on the bottom and back faces for clarity.

## Hybrid Grids

An obvious example of a hybrid grid is an unstructured grid stacked in the third direction. Here, the hybridness came really from the convenience of the stacking. However, there may be other reasons and other mechanisms by which we generate hybrid grids.

All the grids that we have generated so far have advantages and disadvantages. The idea is to use the right tool for a given job. Once we accept that we see the possibility of using different kinds of grids in different regions. In boundary layers, for instance, we need grids with large aspect ratio. We want long thing grids aligned with the flow direction, which is aligned to the boundary. In such cases, triangles are not very good. It is better to use quadrilaterals. On the other hand, a structured grid is difficult to use around complex geometries. For instance, if one used the Rivara's algorithm to refine the grid in a boundary layer, one would halve the characteristic size in the flow direction, every time one halved the direction in the transverse direction. This results in a very rapid increase in the number of grids that are required. On the other hand if one had a structured grid near the surface, the grid can very easily be stretched perpendicular to the wall and left alone parallel to the wall. The grid in the general case can be generated using a hyperbolic/parabolic grid generator. This then provides the boundary on which a conforming unstructured mesh can be generated. Now, the grids in the boundary layer can truly be refined and stretched as required.

### Overset grids

There are lots of situations where we may just decide to create grids around various parts of the domain separately. In the process, we attempt to stitch the various grids together having generated them in a fashion that they conform to the geometries around them and each other.

The other option is to abandon the attempt to have the grids conform to each other and just focus on what the grid is

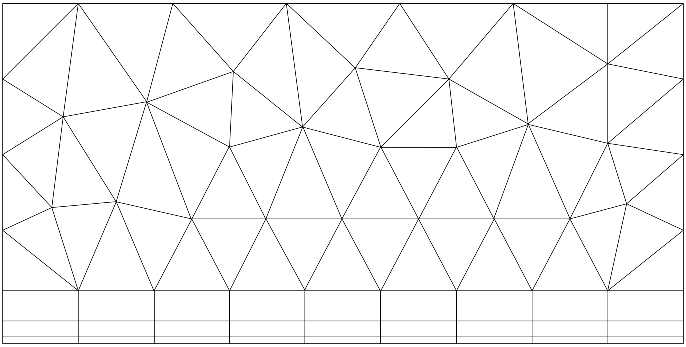

**Figure 6.37. A two-dimensional hybrid grid made up of a structured quadrilateral mesh and an unstructured conforming triangular mesh**

meant to do. This of course will result in an overlap of grids. The following two points need to be borne in mind.

(1) Always solve the governing equations on the finest of the overlapping grids
(2) Interpolate to get the data on the other grids
(3) Transfer of data from one grid to the other needs to conform to the propagation due to the underlying flow.

Let us consider an example where this may work well. Consider the the flow over any rotating machinery, a helicopter for example. One could consider the possibility that a grid is generated about the helicopter and that another grid is generated about each of the blades of the helicopter main rotor. Now, the grid around a blade is likely to be completely embedded in the grid about the fuselage. This has the advantage that the fuselage grid is generated to conform to the needs of the flow about the fuselage. The rotor can independently rotate, the blades can go through a flapping motion or a lead lag motion with the blade grid following the motion and conforming to the blade surface.

It is clear that transferring data from one grid to another is the most important piece of technology required. This can be achieved again by generating the appropriate unstructured Cartesian mesh to find out for a given point in the coarse grid, which are all the grid points on the fine grid that are likely to contribute to the value on the coarse grid. Once these points are identified, the data can appropriately transfered.

### Important ideas from this chapter

- The finite volume method derives directly from the conservation equations derived in integral form.
- The accurate evaluation of the flux at the boundary determines the quality of the solution that we obtain.
- Boundary conditions can be applied either directly as conditions on the flux, or by employing ghost volumes.
- Unstructured grids are easy to generate about complex geometries.
- Structured grids work well in simple geometries. One could decompose a complex geometry into many simple geometries and employ a structured grid.
- Structured grids tend to be better in capturing flow features near boundaries. A hybrid grid consisting of a structured grid near the boundary and an unstructured grid elsewhere often may be a better solution rather than using an unstructured grid everywhere.

- The preceding two points essentially partition the domain. Once a decision is made to partition the domain, one can consider different ways to do it. Overset grids are easy to generate, require a little effort to use. They can easily handle moving/deforming geometries.

# Advanced Topics

This chapter is a combination of many topics. Each topic is worthy of a book in its own right. Such books are available. Given that the philosophy of this book is to give you the essential elements of CFD, these topics are grouped together here. The chapter is essentially in three parts. The first two sections deal with a different view and techniques for solving our equations. The second part focuses on various acceleration techniques. Finally we will close this chapter with a look at some special techniques to solve unsteady problems.

Most of the material presented so far has been based on finite difference methods and finite volume methods. The finite volume method essentially solves the integral form of the conservation equations. We will look at other techniques to do the same.

## Variational Techniques

Variational methods form the basis of much more specialised and popular class of methods called finite element methods(FEM). I am not going to dwell on FEM here since there is a whole machinery of jargon and techniques that have been developed over the years. However, this section on variational techniques should lay the foundation for any future study of FEM pursued by you.

**Three Lemmas and a Theorem.** [**GF82**] As the section title indicates, we are going to look at three Lemma's that will allow us to derive the Euler-Lagrange equation.

Here is the motivation for the mathematics that follows. Let's say that you wanted to find out whether the route you normally take from the library to your class room is the shortest path. You could perturb the current path to see if the length decreases. The change $h(x)$ that you make to the path $y(x)$ has to ensure that the new path $y(x) + h(x)$ starts at the library and arrives at your classroom. So we see that $h(\textbf{library}) = h(\textbf{class room}) = 0$. Since you are not likely to teleport from one point to another point, the path needs to be a continuous path. If we identify the library as "$a$" and the class room as "$b$" then we are looking for the shortest continuous path from amongst all the continuous paths going from "$a$" to "$b$". We will call the set containing all these paths as $C[a, b]$. We can go further and define the subset of $C[a, b]$ which has zero end points as $C_0[a, b]$

The first Lemma: For $\alpha(x)$, a continuous function on the interval [a, b], that is $\alpha(x) \in C[a, b]$ if we can say that

$$\int_a^b \alpha(x)h(x)dx = 0 \tag{7.1}$$

for **any** h(x) which is continuous and $h(a) = h(b) = 0$, that is $h(x) \in C_0[a, b]$ then

$$\alpha(x) = 0 \tag{7.2}$$

How can we prove this to be true? Can we find an $\alpha(x) \in C[a, b]$ which satisfies (7.1) but is not zero? Let us assume we managed to find such an $\alpha(x)$. Now this $\alpha(x)$ must be non-zero for some value of $x$, say it is positive. In that case since it is continuous it must be positive in a neighbourhood say $(x_1, x_2)$. Since (7.1) is supposed to work with all $h(x)$ we need only find one case where it fails. Consider the function

$$h(x) = (x - x_1)(x_2 - x) \text{ for } x \in (x_1, x_2) \tag{7.3}$$

$$= 0 \text{ otherwise} \tag{7.4}$$

We see immediately that

$$\int_a^b \alpha(x)h(x)dx = \int_{x_1}^{x_2} \alpha(x)(x - x_1)(x_2 - x)dx > 0$$

which contradicts the assertion that the integral is zero. So, the $\alpha(x)$ could not have been non-zero as we imagined. We could have chosen $h(x)$ to be a hat function on the interval $[x_1, x_2]$ taking unit value at the midpoint of that interval.

Just as we had the set of functions that were continuous labelled $C[a, b]$, we call the set of functions that have continuous derivatives as $C^1[a, b]$ and if we have a function which is zero at the end points that belongs to $C^1[a, b]$, we say it belongs to $C_0^1[a, b]$.

The second Lemma: For $\alpha(x) \in C[a, b]$ if we have

$$\int_a^b \alpha(x)h'(x)dx = 0 \tag{7.5}$$

for **any** $h(x) \in C_0^1[a, b]$ then

$$\alpha(x) = \text{constant} \tag{7.6}$$

How can we prove this to be true? Can we find an $\alpha(x) \in C[a, b]$ which satisfies (7.5) but is not a constant? Let us assume we managed to find such an $\alpha(x)$. What worked effectively last time was to find an $h(x)$ that violated the given integral condition. We need to construct such an $h(x)$. Let us first define a constant $c$ as

**(7.7)**
$$c = \frac{1}{b-a}\int_a^b \alpha(\xi)d\xi$$

or more useful to us would be to take the $c$ into the integral and write it as

**(7.8)**
$$\int_a^b (\alpha(\xi) - c)d\xi = 0$$

We are essentially taking $c$ to be the mean value of $\alpha(x)$ on the interval $[a, b]$. We then define our $h(x)$ to be

**(7.9)**
$$h(x) = \int_a^x (\alpha(\xi) - c)d\xi$$

What we have achieved is that $h(a) = h(b) = 0$ and the derivative $h'(x)$ exists! We now look at the integral.

**(7.10)**
$$\int_a^b (\alpha(x) - c)h'(x)dx = \underbrace{\int_a^b \alpha(x)h'(x)dx}_{\textbf{(7.5)}} - c\int_a^b h'(x)dx = 0$$

on the other hand

**(7.11)**
$$\int_a^b (\alpha(x) - c)h'(x)dx = \int_a^b (\alpha(x) - c)^2 dx$$

which according to equation (7.10) is zero. This is possible only if $\alpha(x) = c$

Our Third Lemma: For $\alpha(x) \in C[a,b]$ and $\beta(x) \in C[a,b]$ if we have

**(7.12)**
$$\int_a^b (\alpha(x)h(x) + \beta(x)h'(x))\, dx = 0$$

for **any** $h(x) \in C_0^1[a,b]$ then $\beta$ is differentiable and

**(7.13)**
$$\beta'(x) = \alpha(x)$$

We take a cue from equation (7.13). Define

**(7.14)**
$$A(x) = \int_a^x \alpha(\xi) d\xi$$

Then applying integration by parts to

**(7.15)**
$$\int_a^b \alpha(\xi)h(\xi)d\xi = A(\xi)h(\xi)\big|_a^b - \int_a^b A(\xi)h'(\xi)d\xi$$

Since $h(a) = h(b) = 0$, the first term on the right hand side of equation (7.15) is zero. Substituting back into equation (7.12) we get

**(7.16)**
$$\int_a^b (-A(x) + \beta(x))\, h'(x)dx = 0$$

We know from our last lemma that this means

**(7.17)**
$$-A(x) + \beta(x) = \text{constant} \quad \Rightarrow \beta'(x) = \alpha(x)$$

Finally, given a function

**(7.18)**
$$J(y) = \int_a^b F(x, y, y')dx$$

We would like to find its extremum. If we consider a perturbation $h$ on $y$ we get

**(7.19)**
$$J(y+h) = \int_a^b F(x, y+h, y'+h')dx$$

The difference between the two would be

**(7.20)**
$$\begin{aligned} J(y+h)-J(y) &= \int_a^b F(x,y+h,y'+h')dx - \int_a^b F(x,y,y')dx \\ &= \int_a^b \left(F(x,y+h,y'+h') - F(x,y,y')\right) dx \end{aligned}$$

Well we have the difference now. Remembering that a derivative consists of a linear transformation and a direction, we see that we already have the "direction" in which the derivative is being taken, namely $h$. We need to obtain the linear transformation. For this reason we will expand $F(x, y+h, y'+h')$ using Taylor's series and only retain the linear terms. We get

**(7.21)**
$$\delta J(y) = \int_a^b \left( F(x,y,y') + \frac{\partial F}{\partial y} h + \frac{\partial F}{\partial y'} h' - F(x,y,y') \right) dx$$

So, if $y$ were an extremum, then $\delta J(y)$ would be zero for any perturbation $h$. We immediately see that the last lemma is applicable and as a consequence $\partial F/\partial y'$ is differentiable with respect to $x$ and that

**(7.22)**
$$\frac{\partial}{\partial x}\frac{\partial F}{\partial y'} = \frac{\partial F}{\partial y}$$

This is called the Euler-Lagrange equation. So how do we use it? Consider the following problem. In two dimensions, we want to find the shortest path between two points $A$ and $B$ as shown in the Figure 7.1.

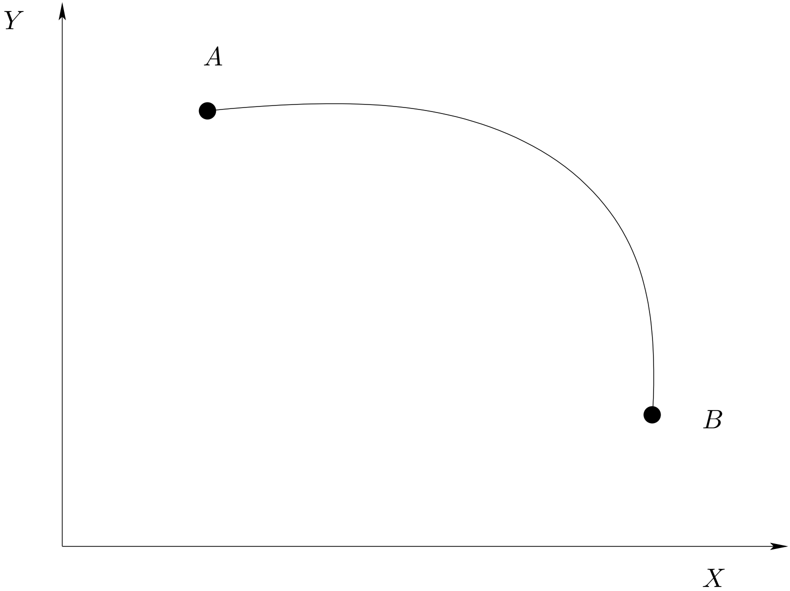

**Figure 7.1. What is the shortest path between $A$ and $B$?**

So what is the length of the path that is shown in the figure. From calculus we know that the length of the curve $y(x)$ from $x = a$ to $x = b$ is given by

$$J(y) = \int_a^b \sqrt{1 + y'^2}dx \tag{7.23}$$

We want to find the path, $y(x)$, which has the shortest length. We need to find the Euler-Lagrange equation for equation (7.23)

**(7.24)** $$F(x, y, y') = \sqrt{1 + y'^2}; \quad F_y = 0, \quad F_{y'} = \frac{y'}{\sqrt{1 + y'^2}}$$

so,

**(7.25)**
$$\frac{d}{dx}\left\{\frac{y'}{\sqrt{1 + y'^2}}\right\} = 0 \Rightarrow \frac{y'}{\sqrt{1 + y'^2}} = c \Rightarrow y'^2 = c(1 + y'^2)$$

If we were to integrate the equation from $x = a$ to $x$ we get

**(7.26)** $$\left.\frac{y'}{\sqrt{1 + y'^2}}\right|_a^x = 0 \Rightarrow \frac{y'}{\sqrt{1 + y'^2}} = \frac{y'(a)}{\sqrt{1 + y'^2(a)}}$$

squaring both sides and subtracting one from both sides we get

**(7.27)** $$\frac{-1}{\sqrt{1 + y'^2}} = \frac{-1}{\sqrt{1 + y'^2(a)}} \Rightarrow y'(x) = y'(a)$$

Which tells us that the slope of the curve is a constant along the curve which gives us a straight line. In fact, integrating one more time from $a$ to $x$ we get

**(7.28)** $$y(x) = y'(a)[x - a] + y(a)$$

Consider the problem of the shortest distance between two points on the surface of a sphere of radius $R$. Without loss of generality we can assume that the radius of the sphere $R$ is one. The coordinate map on the sphere can be taken in terms of $\theta$ and $\phi$ which are the standard coordinates used in the spherical polar coordinate system. Since we are given two points on the sphere, taken along with the origin this gives us three distinct points in space through which we can find a unique plane. We will align our coordinate system so that the $\phi$ value of the two points is the same. That is the $x-z$ plane of our coordinate system passes through these points. The $x-z$ plane can be placed in such a fashion that the $z$-axis passes through the initial point and consequently $\theta$ of the start point is zero. The initial and final points have coordinates $(0,0)$ and $(\theta_f,0)$. The length of a differential line element on the surface of the sphere can be obtained as

**(7.29)**
$$ds^2 = d\theta^2 + \sin^2\theta d\phi^2$$

So, the length of the curve from $\theta_i$ to $\theta_f$ is given by

**(7.30)**
$$L(\phi) = \int_0^{\theta_f} \sqrt{1+(\phi')^2\sin^2\theta} d\theta$$

where $\phi'$ indicates differentiation with respect to $\theta$. We want a $\phi(\theta)$ which gives us the minimum $L(\phi)$. The Euler-Lagrange equation corresponding to this problem is

**(7.31)**
$$\frac{d}{d\theta}\left\{\frac{\phi'}{\sqrt{1+(\phi')^2\sin^2\theta}}\right\} = 0$$

Integrating with respect to $\theta$ once we get

**(7.32)**
$$\frac{\phi'}{\sqrt{1+(\phi')^2\sin^2\theta}} = \left.\frac{\phi'}{\sqrt{1+(\phi')^2\sin^2\theta}}\right|_0 = \phi'(0) = c$$

Rearranging terms and integrating one more time with respect to $\theta$ we get

**(7.33)**
$$\phi(\theta) = \phi(0) + \int_0^\theta c\sqrt{1 + (\phi')^2 \sin^2\theta}\, d\theta$$

At this point we do not need to evaluate the integral. We know that $\phi(0) = \phi(\theta_f)$. So the above equation gives when integrating to $\theta_f$

**(7.34)**
$$\int_0^{\theta_f} c\sqrt{1 + (\phi')^2 \sin^2\theta}\, d\theta = 0$$

Now, $c$ is a constant, the rest of the integrand is always positive. This leaves us with a situation where equation (7.34) is satisfied only when $c = 0$. This tells us that $\phi(\theta) = \phi(0)$. The shortest distance between two points is a segment of a great circle on the surface of the sphere passing between those points.

This is all very nice. How do we solve differential equations using these methods? Look at the variational problem of minimising

**(7.35)**
$$J(u) = \frac{1}{2}\int_a^b u_x^2 dx$$

with $u(a) = u_a$ and $u(b) = u_b$. The Euler-Lagrange equation for this is given by

**(7.36)**
$$u_{xx} = 0$$

with the boundary conditions with $u(a) = u_a$ and $u(b) = u_b$.

**This is the essential point of this section. We have two different but related mathematical models for the same problem.**

We are then free to choose which of these two mathematical models we use to obtain the solution. The variational problem does not seek a solution which has second derivatives, however we do have the minimisation to perform. The differential equation on the other hand has its own set of problems as we have seen so far and requires that the solution have second derivatives in this case. Let us look at how we would solve the variational problem numerically.

If we represent the solution in terms of the hat functions from chapter 2.2.7, we can write it as

$$u = \sum_{i=0}^{N} u_i N_i \tag{7.37}$$

The derivative of $u$ is given by

$$u' = \sum_{i=0}^{N} u_i N_i' \tag{7.38}$$

where the prime denotes differentiation with respect to $x$. We see that the integral in equation (7.35) can be interpreted as the dot product. So, this equation can be rewritten as

$$J^h(u) = \frac{1}{2}\langle u', u' \rangle = \frac{1}{2}\left\langle \sum_{i=0}^{N} u_i N_i', \sum_{j=0}^{N} u_j N_j' \right\rangle \tag{7.39}$$

The superscript $h$ on the $J$ is to indicate it is a discrete version of the one given in equation (7.35). If we assume that we are going to take grid points at equal intervals $h$ apart, we can then get an expression for the derivative of the hat function. The hat function centred about the point $i$ is given by

**(7.40)**

$$N_i(x) = \begin{cases} 0 & x \leq x_{i-1} \\ \dfrac{x - x_{i-1}}{x_i - x_{i-1}} = \dfrac{x - x_{i-1}}{h} & x \in (x_{i-1}, x_i] \\ \dfrac{x - x_{i+1}}{x_i - xi + 1} = -\dfrac{x - x_{i+1}}{h} & x \in (x_i, x_{i+1}] \\ 0 & x > x_{i+1} \end{cases}$$

Now, the derivative is found to be

**(7.41)**
$$N_i(x)' = \begin{cases} 0 & x < x_{i-1} \\ \dfrac{1}{h} & x \in (x_{i-1}, x_i) \\ -\dfrac{1}{h} & x \in (x_i, x_{i+1}) \\ 0 & x > x_{i+1} \end{cases}$$

This is nothing but the Haar function. We can now find the dot product between these functions. It is clear that the only non-zero terms for the dot product are going to be for the terms corresponding to $j = i - 1$, $j = i$, and $j = i + 1$.

**(7.42)**
$$\langle N_i(x)', N_j(x)' \rangle = \begin{cases} 0 & j < i-1 \\ -\dfrac{1}{h} & j = i-1 \\ \dfrac{2}{h} & j = i \\ -\dfrac{1}{h} & j = i+1 \\ 0 & j > i+1 \end{cases}$$

Substituting back into equation (7.39) we get

**(7.43)**
$$J^h(u) = \frac{u_a^2}{2h} + \frac{u_b^2}{2h} + \frac{1}{2h} \sum_{i=1}^{N-1} \left\{ -u_i u_{i-1} + 2u_i^2 - u_i u_{i+1} \right\}$$

To find the extremum of this functional we differentiate it and set it equal to zero. Differentiating equation (7.43) with respect to $u_j$ we get

**(7.44)**
$$\frac{\partial}{\partial u_j} J^h(u) = \sum_{i=1}^{N-1} \frac{1}{2h} \left[ -u_{i-1} \frac{\partial u_i}{\partial u_j} - u_i \frac{\partial u_{i-1}}{\partial u_j} + 4u_i \frac{\partial u_i}{\partial u_j} \right.$$
$$\left. -u_{i+1} \frac{\partial u_i}{\partial u_j} - u_i \frac{\partial u_{i+1}}{\partial u_j} \right] = 0,$$

The five terms in the expression on the right hand side of this equation are non-zero for a specific value of $i$ for each of those terms. Meaning, this is not the time to factor and simplify the expression. We need to inspect each one and determine the value of $i$ in terms of $j$ so that the derivative is non-zero. We do this below.

**(7.45)**

$$\frac{1}{2h}\left[-u_{i-1}\underset{\underset{i=j\Rightarrow\quad i-1=j-1}{\nwarrow}}{\frac{\partial u_i}{\partial u_j}}-u_i\overset{\overset{i-1=j\Rightarrow i=j+1}{\downarrow}}{\frac{\partial u_{i-1}}{\partial u_j}}+4u_i\underset{\underset{i=j}{\uparrow}}{\frac{\partial u_i}{\partial u_j}}-u_{i+1}\underset{\underset{i+1=j+1}{\nwarrow}}{\frac{\partial u_i}{\partial u_j}}-u_i\overset{\overset{i+1=j\Rightarrow i=j-1}{\searrow}}{\frac{\partial u_{i+1}}{\partial u_j}}\right]$$

**(7.46)**

$$\frac{\partial}{\partial u_j}J^h(u)=\frac{1}{2h}\left\{-2u_{j-1}+4u_j-2u_{j+1}\right\}=0,\quad j=1,...,N-1$$

This gives us a tri-diagonal system of equations that we identify as the finite difference discretisation to the one-dimensional Laplace's equation.

In fact, we can derive this directly from the variational problem by substituting for the derivative of $u$ with a one sided differences and employ rectangle rule to the job of approximating the integral. In that case we can rewrite equation (7.35) as

$$J^h(u)=\sum_{i=1}^{N}\left\{\frac{u_i-u_{i-1}}{h}\right\}^2 h \tag{7.47}$$

Again in order to get the extremum we differentiate equation (7.47) with respect to $u_i$ and set the resulting expression to zero.

$$\frac{\partial}{\partial u_i} J^h(u) = 2\frac{u_i - u_{i-1}}{h} - 2\frac{u_{i+1} - u_i}{h} \\ = \frac{-2u_{i-1} + 4u_i - 2u_{i+1}}{h} = 0 \tag{7.48}$$

Again, we see that we get the same answer. The advantage of doing it with the hat functions as a basis and going through whole process is that we know that we are using linear interpolants and have an idea as to how our function is represented. We now know that just using finite differences in this case involves the use of linear interpolants.

How does this method of using hat functions work if we had a forcing function / source term on the right hand side? That is, the equation that we want to solve is given by

$$u_{xx} = p(x), \quad u(a) = u_a, \quad u(b) = u_b \tag{7.49}$$

The variational problem corresponding to this is given by

$$J(u) = \int_a^b \left( \frac{1}{2} u_x^2 + pu \right) dx \tag{7.50}$$

to be extremised with $u(a) = u_a$ and $u(b) = u_b$. How did we get this? Is it correct? The second question actually answers the first. The process we use is exactly like that of integration, we guess and check whether the answer is correct or not. Earlier we had written $u$ in terms of the hat function. In a similar fashion we need to project $p$ on to the hat functions so that we can write it as

**(7.51)**
$$p = \sum_{i=0}^{N} p_i N_i$$

Again the discrete version of the variational problem becomes

**(7.52)**
$$J^h(u) = \frac{1}{2}\left\langle \sum_{i=0}^{N} u_i N_i', \sum_{j=0}^{N} u_j N_j' \right\rangle + \left\langle \sum_{i=0}^{N} p_i N_i, \sum_{j=0}^{N} u_j N_j \right\rangle$$

We have a new set of terms here from the $pu$ term.

**(7.53)**
$$\langle N_i(x), N_j(x) \rangle = \begin{cases} 0 & j < i-1 \\ \frac{h}{6} & j = i-1 \\ \frac{2h}{3} & j = i \\ \frac{h}{6} & j = i+1 \\ 0 & j > i+1 \end{cases}$$

We differentiate the discrete functional with respect to $u_i$ and set it equal to zero.

**(7.54)**
$$\frac{\partial}{\partial u_i} J^h(u) = \frac{1}{2h}\left\{-2u_{i-1} + 4u_i - 2u_{i+1}\right\} + h\frac{p_{i-1} + 4p_i + p_{i+1}}{6} = 0, i = 1, ..., N-1$$

This gives us on simplification

**(7.55)**

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} + \frac{p_{i-1} + 4p_i + p_{i+1}}{6} = 0, \quad i = 1, ..., N-1$$

Interestingly, we see that the function $p$ is also averaged over three neighbouring points. This term appears because the integrals which have a $u_i$ in them correspond to $p_{i-1}$ with a part overlap, $p_i$ with a full overlap and $p_{i+1}$ with a part overlap.

---

**Assignment 7.1**

(1) Since the problem stated equation (7.39) is in one spatial dimension, another way to determine the functional $J^h$ is to actually enumerate the inner product term by term. Show that it can also be written as

$$J^h(u) = \frac{u_a^2}{2h} - \frac{u_b^2}{2h} + \frac{1}{h}\sum_{i=1}^{N} u_i^2 - u_i u_{i-1}$$

---

**Representing Functions revisited.** We can employ linear interpolants on a grid without enforcing continuity of the function representation as done in the case of the hat function. We could use the original basis from which the hat functions were constructed and ask the question which is the best straight line on this interval to approximate the given function.

On the interval $(x_i, x_{i+1})$ if we look at the function given by equation (2.64) and rewrite it here in a slightly different form as

**(7.56)**

$$f(x; a_i, b_i) = a_i\alpha_i(x) + b_i\left(1 - \alpha_i(x)\right) = (a_i - b_i)\alpha_i(x) + b_i, \quad \text{for } x$$

where the ";" indicates that the actual function depends on the choice of $a_i$ and $b_i$. Now, if we want to represent some function $F(x)$ on the interval $(x_i, x_{i+1})$ we can pick some $a_i, b_i$ pair to get a linear representation on that interval. The error in the representation would be

**(7.57)** $$E(a_i, b_i) = \sqrt{\int_{x_i}^{x_{i+1}} \left(F(x) - f(x, a_i, b_i)\right)^2 dx}$$

We need to find the pair $a_i, b_i$ for which this error is minimised. It should be noted here that we are not going to insist that the representation is continuous at the end points just like in the box functions and the Haar functions. This allows us to solve for one interval without bothering about the neighbouring intervals. So to get the $a_i, b_i$ pair we differentiate equation (7.57) with respect to $a_i$ and set the derivative to zero and repeat the process with $b_i$. This will give us two equations in two unknowns which we solve for the $a_i$ and $b_i$.

I used the package wxmaxima to perform the symbolic integration and differentiation to find the general expression for $a_i$ and $b_i$ for $F(x) = \sin x$. I obtained the following expressions.

**(7.58)**

$$a_i = -\frac{6\left(\sin x_{i+1} - \sin x_i\right) - 2dx_i\left(\cos x_{i+1} + 2\cos x_i\right)}{dx_i^2}$$

and

$$b_i = \frac{6\left(\sin x_{i+1} - \sin x_i\right) - 2dx_i\left(2\cos x_{i+1} + \cos x_i\right)}{dx_i^2} \tag{7.59}$$

where, $dx_i = x_{i+1} - x_i$.[1] We use eleven grid points as we did earlier and graph these line segments in Figure 7.2 to see what the representation look like. This representation looks quite good. It is shown in the same scale as the graphs in section 2.9. It almost looks as though the lines segments meet and that the representation is continuous everywhere. This is not really so. We look at a zoomed in view of the above graph at the point $x = \pi$. This is shown in Figure 7.3. We can clearly see the jump in the representation. What do we do at the point where the jump occurs? Take the average of the two values. We can do this for the function and the derivative at the nodal points.

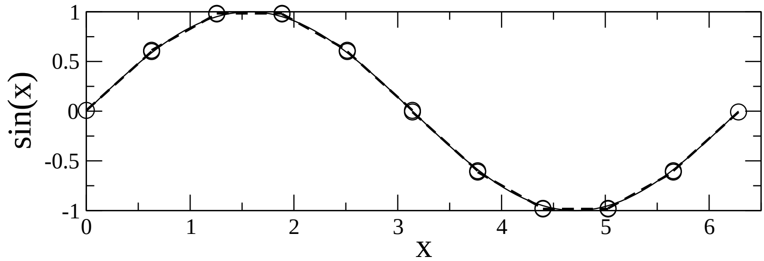

**Figure 7.2. An approximation for $\sin x$ using ten optimal straight line segments**

[1] I want to point this out just as a curiosity. If you squint at the expression for $a_i$ and $b_i$ can you see something that looks like the finite difference approximation to the second derivative of $\sin x$?

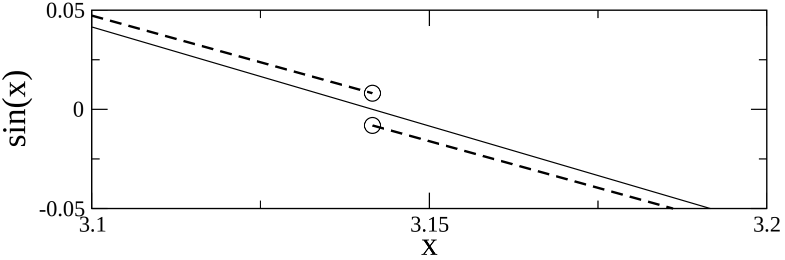

**Figure 7.3.** **Zoomed view near $x = \pi$ of the approximation for $\sin x$ using ten optimal straight line segments. Note the jump in the representation at $x = \pi$.**

In fact the general expression for $\sin nx$ turns out to be ( thanks again to wxmaxima, though in view of the footnote we could have actually guessed this expression given the one for $\sin x$. )

**(7.60)**

$$a_i = -\frac{6\left(\sin nx_{i+1} - \sin nx_i\right) - 2ndx_i\left(\cos nx_{i+1} + 2\cos nx_i\right)}{n^2 dx_i^2}$$

and

**(7.61)**

$$b_i = \frac{6\left(\sin nx_{i+1} - \sin nx_i\right) - 2ndx_i\left(2\cos nx_{i+1} + \cos nx_i\right)}{n^2 dx_i^2}$$

---

## Assignment 7.2

(1) As we did in section 2.9, plot these graph for various values of the wave number $n$.

(2) Find this kind of a representation for $F(x) = x^2$ on the interval $[-1, 1]$. How does that compare to the representation employing hat functions.

(3) Repeat this process for a cubic, that is for $F(x) = x^3$. Check the values at the nodes. What can you conclude?

(4) Repeat the exercise with a different number of intervals. How about eleven intervals?

---

I have included plots for four cases of $n = 5, 10, 11, 12$ here from the first question in the assignment. We are looking at these cases since we expect the representation to be poor for these cases. You can look at these graphs and compare them to the ones we plotted using hat functions. In the case of the hat functions we lost all information regarding frequency and amplitude for the $\sin 5x$ case ( see Figure 2.21 ). Here, the amplitude is off, however, the frequency is the same, meaning we are able recover the fact that the wave number was five. This is so even for the $\sin 10x$ case. How are we able to get the correct information on the frequency all the way up to wave number

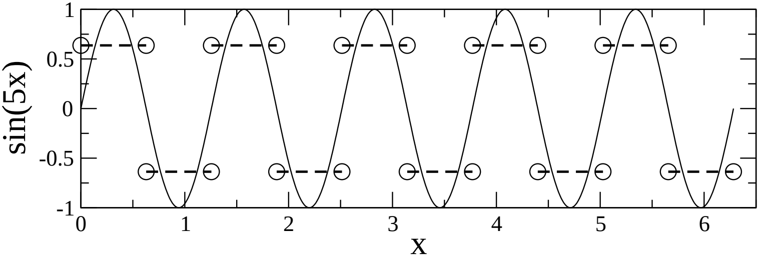

**Figure 7.4. Representation of $\sin 5x$**

$n = 10$ with ten intervals? Actually we are using 22 $a_i$ and $b_i$ values and that is the reason why we can capture frequency information to such a high wave number. If we are interested

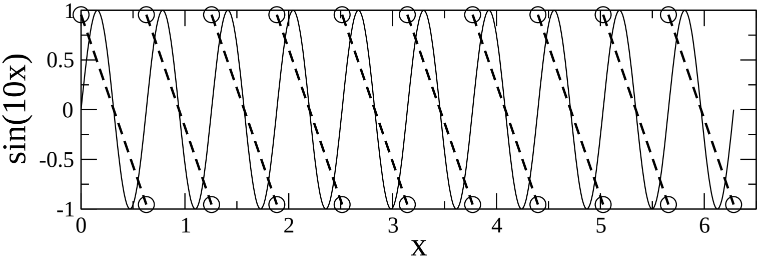

**Figure 7.5. Representation of** $\sin 10x$

only in the frequency content and not the amplitudes we can use ten intervals to go to a wave number ten. However, in CFD, we are very often interested in an accurate reconstruction of the function. In that case we really want to use at least 40 intervals in the smallest wavelength to be able to reconstruct the function with some semblance of fidelity. One of the easiest ways to ensure that you have enough grid points is to double the number of grid points and see if the solution changes significantly. The two Figures 7.6 and 7.7 are given here just to show the degeneration in the representation if the grid is too coarse. There is loss of both amplitude and frequency information in these two cases.

It would be fair to assume at this point that if we were to come up with a variational formulation with this linear basis function for our gas dynamical equations we could pick up

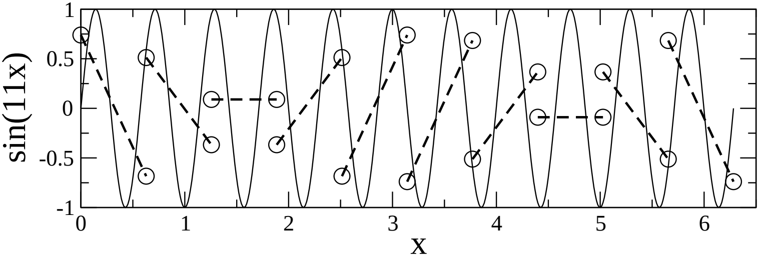

Figure 7.6. Representation of $\sin 11x$

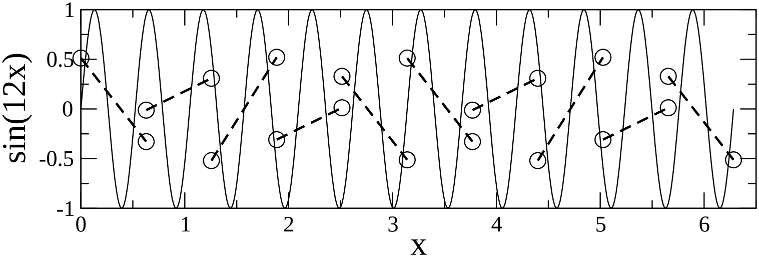

Figure 7.7. Representation of $\sin 12x$

shocks quite easily since the set of basis functions already allow for jumps to occur in the solution.

We have so far seen problems in one independent variable. In the next section we will look at problems in which we have more than on independent variable.

**Extension To Two Dimensional Problems.** Can we, by observation extend the Euler-Lagrange equation to a case of two independent variable $(x, y)$ and a dependent variable $u(x, y)$? The variational problem in two spatial dimensions would be stated as follows

**(7.62)**
$$J(u) = \int_A F(x, y, u, u_x, u_y) dA.$$

where $u_x$ and $u_y$ are derivative of $u$ with respect to $x$ and $y$, $A$ is the region over which the integration is performed. The corresponding Euler-Lagrange equation would be something like this

**(7.63)**
$$\frac{\partial}{\partial x}\frac{\partial F}{\partial u_x} + \frac{\partial}{\partial y}\frac{\partial F}{\partial u_y} = \frac{\partial F}{\partial u}$$

See if you can derive it by just following the earlier derivation.

Let us consider an application, given the functional

**(7.64)**
$$J(u) = \frac{1}{2}\int_A (u_x^2 + u_y^2) dA$$

We see that the Euler-Lagrange equation is nothing but the Laplace's equation. Try it. We can define hat functions in two dimensions and in fact in multiple dimensions in exactly the same manner in which we derived them in one dimension. We define the hat function at a grid point as taking a value one at that grid point and dropping off to zero toward the adjacent grid points and the edges connecting those grid points.

We have seen that quite often, problems have a differential representation, an integral representation or a variational representation. Now looking at the example above for Laplace's equation it may not be obvious that one representation is better than the other. The example is chosen simply to demonstrate that there are various ways of representing the same problem. We will now consider a different problem which is initially posed as a variational problem and we subsequently derive the partial differential equation

**The Soap Film Problem.** This problem is also called the Plateau problem. Simply, it is this. If we loop a wire around and immerse it into soap water so that on removing it from the water a film of soap is formed on loop, what is the shape of the soap film. Again, it is clear that we are seeking a function which satisfies certain boundary conditions. Namely the film is supported by the loop. There are many functions that satisfy this requirement. We look for the one which has the minimum area and for a thin film this turns out to be a minimum energy state. If $\mathcal{D}$ defines the region enclosed by the look we as that

$$J(u) = \int_{\mathcal{D}} \sqrt{1 + u_x^2 + u_y^2} dA, \tag{7.65}$$

where $dA$ is a differential area element at the point $(x, y) \in \mathcal{D}$. We need to find $u$ so that $J(u)$ is minimum. If we were to find the corresponding Euler-Lagrange equation ( I would suggest that you verify it ) we get the equation

$$\left(1 + u_y^2\right) u_{xx} - 2u_x u_y u_{xy} + \left(1 + u_x^2\right) u_{yy} = 0 \tag{7.66}$$

with $u = f(x, y)$, with $(x, y)$ on the boundary of $\mathcal{D}$ as the boundary condition. In this case, posing the problem as a variational problem definitely looks more attractive then the corresponding partial differential equation.

## Random Walk

The idea of random walk will first be introduced in an intuitive fashion[**Fel68**]. We will start with a probabilistic view of a game played by two gamblers. Let us say that two gambler's, A & B, start playing a game. Let us say at the beginning they have 100 chips between them [ a chip is a unit of currency ]. One of them, A, has $x$ chips the other, B, has $100 - x$ chips. The game played by the gamblers goes in steps called "rounds". At the end of each round one of them gains a chip which the other loses. So clearly, we need only look at A, the gambler with $x$ chips. We can infer the fate of B from the story of A.

So, our gambler has $x$ chips. What is the probability that A is ruined? Let us call this probability $q_x$. The game may not be a fair game, B may be cheating. Let us say that the probability that A wins a round is $p$ and the probability of a loss is $q$. Then, given $x$ chips, the probability that A will have $x+1$ chips at the end of a round is $p$ and the probability that A will have $x - 1$ chips at the end of the same round is $q$.

Therefore, the probability of ruin with $x$ chips can be written as

**(7.67)**
$$q_x = qq_{x-1} + pq_{x+1}$$

That is there is a probability $q$ that A will end up with $x - 1$ chips and the probability of ruin with $x - 1$ chips is, of course, $q_{x-1}$. Similarly, with a probability $p$, A will have $x + 1$ chips given that A has $x$ chips before the round and the probability of ruin with $x + 1$ chips is $q_{x+1}$. Note that here, $p + q = 1$.

This equation is valid for all values of $x$ except for $x = 0$ and $x = 100$. In the first case A is ruined and hence $q_0 = 1$.

In the second case B has been ruined and the probability of A's ruin is zero. That is $q_{100} = 0$. So, given $p$ and $q$ one can find the probability of ruin for $x = 1$ to $x = 99$ by iterating on the equations given by (7.67) When it has converged, we will have the $q_x$ for all the $x$'s.

Take a good look at equation (7.67), with $p = q = 1/2$. We see that it becomes

**(7.68)**
$$q_x = \frac{q_{x-1} + q_{x+1}}{2}$$

It is possible to use the prescribed boundary conditions to iterate this equation to get the probability distribution $q(x)$. The iterates are related by

**(7.69)**
$$q_x^n = \frac{q_{x-1}^{n-1} + q_{x+1}^{n-1}}{2}$$

subtracting $q_x^{n-1}$ from both sides we get

**(7.70)**
$$q_x^n - q_x^{n-1} = \frac{q_{x-1}^{n-1} - 2q_x^{n-1} + q_{x+1}^{n-1}}{2}$$

Which is nothing but our solution to the one-dimensional Laplace equation or the heat equation. So, it looks like we could some how use this problem of the ruin of a gambler to study the solution to the heat equation.

Since we need to generate a random sequence of $+1$ and $-1$, we will take a small detour, and look at how to write a simple random number generator.

**Random Number Generators.**

> The generation of random numbers is too important to be left to chance - Anonymous

We will look at how we can generate a sequence of random numbers[**Knu81**]. You could also look to use a built in random number generator, however, you really should check how good it is. One has to remember that we are usually generating pseudo random numbers. That is, they are not truly random. In fact from the point of view of debugging a program that employs a random number generator, we would like the numbers to be the same sequence each time we run the code. Otherwise, we would not be able to figure out whether the change in the program behaviour was due to a change that we have performed or due to the random number generator.

Most random number generators will take an initial "seed" number and generate a sequence of numbers. The sequence generated is determined by the seed. If one were to change the seed the sequence would change. Since we are dealing with a finite number of digits, it is clear that numbers will eventually start to repeat. We typically want the size of this loop to be as large as possible.

**Doing the Random Walk.** Now that we know how to generate random numbers. How do we have a go at solving the ruin problem? Here is one way to do it. We can create one hundred and one bins. We put one hundred marbles in the first bin and none in the others. Each marble represents a player. As we have seen earlier, each player plays a round with someone who need not be shown in the bins, since if our player is ruined the other player wins and vice-versa.

The recipe for the solution is quite simple. Any player in a bin tosses a fair coin, $p = q = 0.5$. If the player wins he moves to the right. If the player looses he is out of the game. We top off the first bin to maintain a hundred players there.

Any player who ends up in the bin at the far right is removed as a victor from the game. You can now run this simulation and see what happens to the players as a function of time and how many players there are in each bin.

- for each marble in a bin generate a random number which has two possible states:$(+, -)$. If it is a plus move the chip into the bin on the right. If it is a minus move the chip to the left. The chips in the first bin do not move left and the chips in the last bin do not move to the right.
- make sure there are 100 chips in the first bin and zero in the last.

This process can be repeated.

---

**Assignment 7.3** Implement the above algorithm. How does the solution evolve? What happens if you generate ten solutions with ten different seeds and then take the average of the ten solutions to get one solution. Try this for 4, 16, 64 sets of solution. What can you say about the quality of the solution? If you let the solution evolve by a hundred time steps. What does the average of the last ten steps look like. Do this for the average solution obtained earlier.

---

**Stochastic Differential Equations.** Stochastic differential equation occur in various field. In the applied disciplines, this is most probably encountered for the first time through kinetic theory of gases and the billiard ball model. However, the sole objective there is not to solve for the motion of the individual molecules, the stochastic differential equation would have never really been stated. Here is a simple stochastic differential equation.

**(7.71)** $$df = \mu dt + \sigma dz$$

Here, $\mu dt$ is called the drift term. $\sigma^2 dt$ is the variance. To ascribe some physical meaning to these terms, the first term has a $\mu$ in it which is basically the mean velocity of the flow and the second term is usually captured by temperature in gas dynamics.

---

**Assignment 7.4**

Do the following.

(1) Generate random numbers that are normally distributed with mean zero and variance one, that is $N(0,1)$.

(2) These random numbers that you generate at each time step are the $dz$ values. Use these $dz$ values along with the drift term to integrate the stochastic differential equation (7.71). Use the Euler explicit scheme for the deterministic part. The resulting scheme is called the Euler scheme for the stochastic differential equation. For one set of random number you generate one path. You can repeat this trial many times.

(3) Plot a few sample paths and see what they look like. What is the expectation of these paths?

---

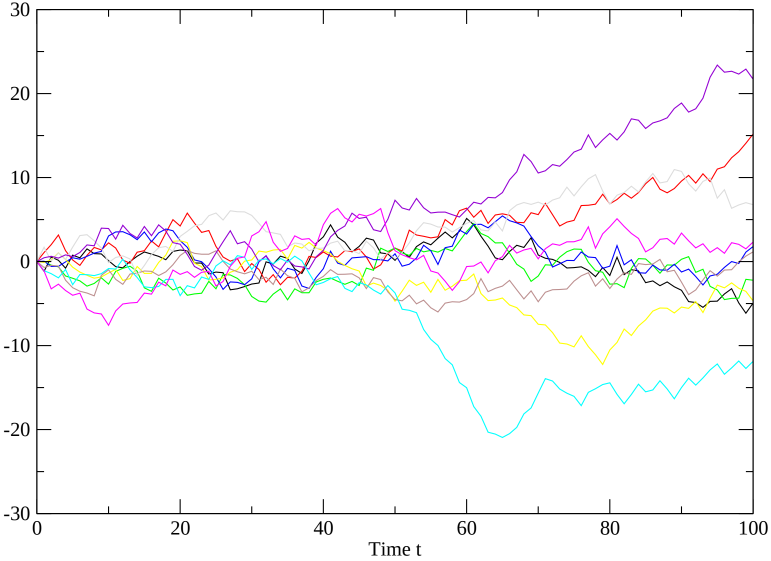

**Figure 7.8. Ten paths with $\mu = 0$ and $\sigma = 1$ obtained by integrating equation (7.71) over a hundred time steps using the explicit Euler's scheme and a $\Delta t = 1$**

It is also important to see the relationship between the solution to the heat equation and this random process. Any bin in the heat equation solution contains chips corresponding to the number of paths that pass through that bin at that time.

## Multi-grid techniques

Multi grid schemes, somehow, are not as popular as they should be. An excellent start can be made by reading Briggs[**Bri87**] Let's look at a simple introduction to the multi grid technique.

The basic objective of the multi-grid scheme is to accelerate a given numerical method. We saw earlier that we could get an improvement in convergence by employing the successive over relaxation scheme. However, we also saw that there was a short coming, in the sense that one had to perform an exploratory study to obtain the optimal relaxation parameter and that as the grid became finer the relaxation parameter got closer to 2.0 rendering the scheme less useful. Here accelerated convergence is obtained using a multiplicity of grids rather than just one grid.

Why use multiple grids to solve the same problem? High frequency is defined with reference to the grid. A function is said to have high frequency content if it is composed of frequencies comparable to the highest frequency that can be represented on that grid. As was seen in the earlier demonstration, for a given grid, high frequencies are damped out faster than low frequency. This is especially true if some kind of smoothing is involved either in the process represented by the equation being solved or the smoothing is done deliberately to eliminate the high frequency component of the function.

These two features are used to develop the multi grid method. We will use the Laplace equation on a unit square as an example.

We have seen that the Laplace equation on a unit square when dicretised results in a system of equations given by

**(7.72)** $$A^h \phi^h = f^h$$

where the superscript "$h$" indicates the grid size. Remember now, that if we solve the equation and obtain a $\phi^h$, it show satisfy the equation (7.72) and approximate the solution to the Laplace equation.

In the process of trying to obtain $\phi^h$, we have a candidate approximation $\Phi^h$. Now this candidate solution to equation (7.72) differs from the actual solution. This difference is defined simply as

**(7.73)** $$e^h = \phi^h - \Phi^h$$

Here $e^h$ is also called the correction as we can obtain $\phi^h$ from $\Phi^h$ as follows

**(7.74)** $$\phi^h = \Phi^h + e^h$$

As is usually done, we define the residual for equation (7.72) as

**(7.75)** $$r^h = f^h - A^h \Phi^h$$

Multiplying equation (7.73) by $A^h$ we get

**(7.76)** $$A^h e^h = A^h(\phi^h - \Phi^h) = A^h \phi^h - A^h \Phi^h = f^h - A^h \Phi^h$$

substituting from equation (7.75) we get

**(7.77)** $$A^h e^h = r^h$$

This is an important equation. It shows that the error or correction satisfies the same equation as the original problem.

So here is an alternative way of looking at the solution to Laplace's equation.

(1) **Assume** a $\Phi^h$

(2) **Compute** the residue $r^h = f^h - A^h \Phi^h$
(3) **Perform** relaxation sweeps to (7.77) and get $e^h$
(4) **Obtain** the corrected solution from (7.74)

These steps are the same as the earlier algorithm. . It is rewritten to accommodate our discussion on multi-grid schemes.

It also suffers from the same disadvantage that it decimates the higher frequencies and has a lot of difficulty with the lower frequencies. This disadvantage can be turned around to speed things up if we remember that a low frequency on a fine grid [ small $h$] will be a higher frequency on a coarse grid [ larger interval, say $2h$]

So, what we need to do, is to transfer our problem to a coarser grid once we have eliminated the high frequencies on the grid $h$.

We can look at this alternative series of steps.

(1) **Assume** a $\Phi^h$
(2) **Compute** the residue $r^h = f^h - A^h \Phi^h$
(3) **Perform** relaxation sweeps to (7.77) to eliminate the high frequencies
(4) **Transfer** the residue $r^h$ on the grid $h$ to the grid $2h$.
(5) With this residue, we can rewrite (7.77) on the grid $2h$ to obtain $e^{2h}$.
(6) **Transfer** $e^{2h}$ back to the grid $h$ to get $e^h$.
(7) **Obtain** the corrected solution from (7.74)
(8) back to step (1) till the required convergence on the fine grid is achieved.

So, what happens when the iterations in step (5) wipe out the high frequencies with reference to grid $2h$ and are now slogging along slowly with the lower frequencies. We have a solution.

This statement of the modified algorithm was made deliberately vague on the steps (4) and (5). Remember again, that the equation for the correction [ often referred to as the correction equation] looks the same as the original equation.

So, when we transfer the residue, $r^h$, from grid $h$ to $2h$, we choose the name $f^{2h}$. Then the algorithm is

(1) **Assume** a $\Phi^h$
(2) **Compute** the residue $r^h = f^h - A^h\Phi^h$
(3) **Perform** relaxation sweeps to (7.77) to eliminate the high frequencies
(4) **Transfer** the residue $r^h$ on the grid $h$ to the grid $2h$ to obtain $f^{2h}$.
(5) With this residue, we can rewrite (7.77) on the grid $2h$ as $A^{2h}\Phi^{2h} = f^{2h}$
(6) **Compute** the residue $r^{2h} = f^{2h} - A^{2h}\Phi^{2h}$
(7) **Perform** relaxation sweeps to $A^{2h}e^{2h} = r^{2h}$ and get $e^{2h}$
(8) **Obtain** the corrected solution as $\phi^{2h} = \Phi^{2h} + e^{2h}$
(9) **Transfer** $\phi^{2h}$ back to the grid $h$ to get $e^h$.
(10) **Obtain** the corrected solution from (7.74)
(11) back to step (1) till the required convergence on the fine grid is achieved.

We are now ready to do a preliminary design for an implementation of this algorithm that will answer our question of what to do with the problem on the grid $2h$.

First write a simple solver called

ComputePhi( $A^h$, $f^h$, $\Phi^h$ ):

**Compute** the residue $r^h = f^h - A^h\Phi^h$
**Perform** relaxation sweeps to (7.77) to eliminate the high frequencies

**Update** $\Phi^h = \Phi^h + e^h$ **return** $\Phi^h$

Here is how you would do a multi-grid computation.

(1) Form $A^h$, $f^h$
(2) Guess a $\Phi^h$
(3) invoke the function ComputePhi( $A^h$, $f^h$, $\Phi$ ) to obtain $e^h$
(4) find the new improved $\Phi$ from $\Phi^h = \Phi^h + e^h$
(5) find $r^h$ and transfer it to $f^{2h}$
(6) transfer $e^h$ to $\Phi^{2h}$ and form $A^{2h}$
(7) invoke the function ComputePhi( $A^{2h}$, $f^{2h}$, $\Phi^{2h}$ ) to obtain $e^{2h}$
(8) find $r^{2h}$ and transfer it to $f^{4h}$
(9) transfer $e^{2h}$ to $\Phi^{4h}$ and form $A^{4h}$
(10) invoke the function ComputePhi( $A^{4h}$, $f^{4h}$, $\Phi^{4h}$ ) to obtain $e^{4h}$
(11) **This can be taken to as many levels as required**
(12) transfer $e^{4h}$ back to $e^{2h}$.

**if** you want to go to a coarser grid **then**

```
    Transfer the residue $r^h$ on the grid $h$ to the grid $2h$ to obtain $f^{2h}$.

    Transfer the correction $e^h$ from grid $h$ to grid $2h$ to obtain $\Phi^{2h}$.

    ComputePhi( $A^{2h}$, $f^{2h}$, $\Phi^{2h}$ )

    Transfer $e^{2h}$ back to $e^h$

    Perform  relaxation sweeps to (7.77)

return $e^h$
```

This is called the V–cycle. This is because we start at the finest grid make our way down to the coarsest grid with the residuals and work out way back to the finest grid with the corrections.

There are other cycles or grid sequences that can be used. To motivate this let us consider the number of operations performed in order to get to the solution. We can then see if we can try to reduce the total number of operations to get down to converge to the solution by changing the grid sequencing.

The number of grid points at the finest grid are $n$. The number of operations per grid point for one iteration are $\omega$. Then the number of operations at the finest grid $h$ for one iteration are $n\omega$. This is termed one work unit. At the next coarsest level there are approximately $n/4$ grid points. Which means that an iteration at the second level is only $1/4$ of a work unit. Again an iteration at the third level corresponds to $1/16$ of a work unit. As we would expect the number of work units on a coarse grid are less per iteration than on a fine grid. That is, it is cheaper iterating on a coarse grid rather than on a fine grid. So on the way back from a coarse grid to the fine grid we may be tempted to go back to the coarse grid once more before returning to the fine grid. This is called the W–cycle. The two cycles are illustrated in the figure.

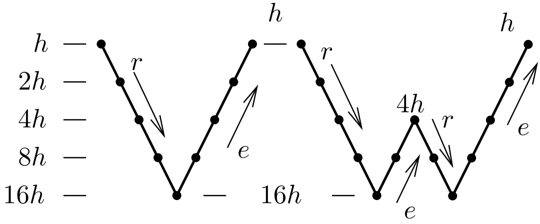

**Figure 7.9. The V–cycle and the W–cycle. $r$, the residue, is passed down a left limb from a fine grid to a coarse grid and $e$, the correction, is passed up the right limb from a coarse grid to a fine grid. $h$, $2h$, $4h$, $8h$, and $16h$, indicate the grid size at the dots at the corresponding level on the limbs**

We need to remember that finally we want our convergence on the fine grid. So, what ever grid sequencing we do, we need to ensure that the final test for convergence is done based on iterates from the fine grid.

This idea of the work unit now allows us to compare the effort involved in solving a problem with various multi grid algorithms and an old fashioned single grid algorithm. We just look at or plot the error on the finest grid versus the number of work units.

Finally of course, the easiest and most obvious thing to do is to actually start the iterations with the coarse grid rather than the fine grid.

How do we now apply this to the one-dimensional Euler equations. The first point to bear in mind is that we always linearise the equations. This multi grid algorithm can be applied almost directly to the linearised equation. The second point to bear in mind is that we always transfer the residue from the finest grid to the coarsest grid and the corrections/solutions from the coarsest grid to the finest grid. The equations are written in delta form and the test for convergence is always made on the fine grid, meaning the residue is evaluated on the fine grid.

---

**Assignment 7.5** First redo the Laplace's equation solver you have written already to accelerate convergence using the various multi grid methods.

---

**Applying the Multi-Grid Scheme to the Euler's Equation.** How do we apply this to the Euler's equation? Lets start with the one-dimensional first order wave equation and see what we get. We will do this in the context of looking for a steady state solution.

We have seen that the time-marching equations can be written in the delta form as

**(7.78)** $$\boldsymbol{S}\Delta u = -\Delta t \boldsymbol{R}(t)$$

We will now apply the multi-grid scheme to this equation. The point to be borne in mind is that we always transfer the residual from the fine grid to the coarse grid and transfer the correction / solution from the coarse grid to the fine grid.

So given the discrete equation (7.78) and an initial condition, we do the following to obtain the solution.

(1) Take a few time steps with the equation (7.78).
(2) Compute the residual $R^h$. The residual can be smoothed if required especially if we have taken only one or two time steps.
(3) Transfer $R^h$ and $u^h$ to the grid $2h$ to get $R^{2h}$ and $u^{2h}$.
(4) Take time steps on the grid $2h$.
(5) Transfer to the grid $4h$ and so on.
(6) transfer the $u$ back to the fine grid.
(7) repeat this process.

---

**Assignment 7.6** Do the same with the Euler equation solver. Again try out different levels of grids. Also find the effect of the number of time steps taken on a given grid level.

---

## Unsteady Flows

So far, we have seen numerical schemes that involved solutions to equilibrium equations like the Laplace's equation and time varying equations like the wave equation, heat equation and the one-dimensional Euler equations. Along the way we changed our focus to computing steady state solutions to the unsteady equations. We have built up quite a bit of "machinery" to handle time marching schemes leading to steady state solutions of these equations with time varying terms.

Consider the simple problem of the viscous flow past a cylinder. From fluid mechanics we are aware that one of the important parameters characterising this flow is the Reynold's number defined as

**(7.79)**
$$Re = \frac{\rho U D}{\mu}$$

For extremely small Reynold's number value flows, the flow is laminar and turns out to be dominated by viscous effects. The governing equations for a two dimensional incompressible flow at these Reynold's number in fact degenerates to the Laplace's equation in the velocity components. Creep flow, as it is called, throws up a flow governed by some kind of a potential. As we keep increasing the Reynold's number, the inertial effects start to grow and the viscous effects slowly start to get important near the surface of the cylinder. Put another way, as the Reynold's number increases, the viscous effects become less important away from the cylinder. The perceptible and increasingly strong viscous effects are confined to a region near the cylinder called the boundary layer. It turns out that this boundary layer can in fact separate from the cylinder causing a recirculation region to form. So far everything seems fine. The problem is that the boundary layer and the recirculation region effect the pressure field around the cylinder which in turn effects the boundary layer. We could hope that all of this settles down to an equilibrium situation giving a steady state solution. However, in real life the recirculation region may slosh back and forth responding to the pressure field that it creates in a sense the system hunting for that equilibrium and never finding it. Worse still, the recirculation region may break off from the cylinder and head out into the flow. From fluid mechanics we recognise this as vortex shedding and then the problem is anything but steady. Look what this vortex shedding did to the pressure field of the Tacoma narrows bridge with catastrophic consequences. The bridge problem was complicated by the fact that the bridge also deformed in response to the pressure distribution.

**The conclusion**: There may be problems that are inherently unsteady.

How do we compute flows that vary in time. Or, how do we calculate the transient to a flow that eventually makes it to the steady state. That is, we are interested in the variation of our flow parameters in time.

The flow past a cylinder example given above is one where we have unsteady flow without a geometrical change. The bridge example is a case where the unsteady flow entails changes in geometry. We will not address the geometry change issue here. The unsteady flow problem is one on which whole tomes can actually be written. When we sought the steady state solution with no reference to the transient, we were interested in getting to the steady state as quickly as possible. If the flow had different time scales we did not bother to capture all of them accurately as long as the transients with those scales were quickly eliminated. We went out of our way to get the time scales comparable to each other so that they could be efficiently eliminated. Now we want to pick up the transients. There may be many different time scales that are important. This creates an interesting problem as we have already seen that dissipation in our numerical scheme can change amplitudes and that dispersion can change speeds of propagation. The dissipation and dispersion can depend on the time scales. On the other hand, the physical system that we are trying to model may have its own dissipation and dispersion characteristics. That is, the behaviour of decay and dispersion that we have seen are not only properties of numerical schemes, they are also exhibited by real life. Just look at a bag of potato chips, you open a bag, offer your friends some and then find that all the piece left are smaller and tending towards crumbs. An initial even distribution of various length scales of potato chips will on transit from the manufacturer to your hands be shaken enough times to cause the smallest length scales to end up at the bottom of the bag. This process is dispersion. It exists. Clearly, we need a numerical scheme that is accurate and can incorporate/capture the correct dissipation and dispersion.

Why are these parameters critical. Imagine trying to model a tsunami in the ocean after an earthquake has occurred. It is important to be able to predict the magnitude of the tsunami and the time of landfall. Dissipation and dispersion will introduce errors in both. Consequently, we may give a false alarm which will cause people to ignore future alarms or we may decide that there is no danger and not give an alarm when we should.

## Standard Schemes?

Can we use the schemes that we have employed so far with some success to get time accurate solutions. If you go back to the section on the modified equation, you will remember that FTBS gave the "exact solution" when $\sigma = 1$. What can we do to make sure that we get the best possible solution within the constraint of resources available?

If we were to try setting $\sigma = 1$ for the one-dimensional Euler's equation, which $\sigma$ would we use? Clearly this is not going to be easy. This is not getting us anywhere, so let us just jump and look at a scheme with which we are already familiar: BTCS.

BTCS has the following advantages

- It is unconditionally stable
- . . .

What of the disadvantages

- It involves the solution of a system of equations
- It is first order accurate in time

So, clearly, we would like to get a scheme that is second order accurate in time. This can be done by employing a central difference representation in time. The Crank-Nicholson method does exactly this.

It should be pointed out here that the BTCS scheme can be viewed as a central difference scheme for the spatial coordinate with a backward Euler scheme or a rectangle rule for quadrature in time. Similarly, the Crank-Nicholson scheme is equivalent to a trapezoidal quadrature rule employed in time, again in conjunction with a central difference scheme in space. They are also identified an first and second order Runge-Kutta schemes.

We could go to higher order schemes. There are a whole slew of higher order Runge-Kutta schemes. There are numerous predictor corrector schemes that are second order accurate in time.

## Pseudo Time stepping

We have seen that we can march to a steady state solution by marching time from some guessed initial condition. We have accumulated a lot of analysis tools and know how for this class of time marching schemes. In fact, we would go as far as adding a time derivative term of our choice to an equation that is formulated to be unsteady, just to employ our time marching technology. We will try to bring that to bear on the problem of solving unsteady flows.

Consider the first order, linear, one-dimensional wave equation again.

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{7.80}$$

We will create an extra coordinate called "pseudo time", $\tau$, and include a derivative with respect to $\tau$ in our equation.

$$\frac{\partial u}{\partial \tau} + \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{7.81}$$

Now, we can take a second order backward difference in time-$t$ and use some time marching scheme to proceed in $\tau$. In theory, each time we reach some kind of a steady state in $\tau$, we should have satisfied our unsteady equations.

Let us discretise this equation using central difference for the spatial derivatives. We will take a two point backward difference for the time derivative. For the sake of simplicity, we will use a forward difference in pseudo time, $\tau$. This gives us

**(7.82)**
$$\frac{u_{pq}^{r+1} - u_{pq}^{r}}{\Delta\tau} + \frac{3u_{pq}^{r} - 4u_{p(q-1)} + u_{p(q-2)}}{2\Delta t} + a\frac{u_{(p+1)q}^{r} - u_{(p-1)q}^{r}}{2\Delta x} = 0$$

We are now able to march in pseudo time using the automaton

**(7.83)**
$$u_{pq}^{r+1} = u_{pq}^{r} - \Delta\tau\frac{3u_{pq}^{r} - 4u_{p(q-1)} + u_{p(q-2)}}{2\Delta t} - a\Delta\tau\frac{u_{(p+1)q}^{r} - u_{(p-1)q}^{r}}{2\Delta x}$$

Rearranging terms we get

**(7.84)**
$$u_{pq}^{r+1} = \frac{1}{2}\sigma_\tau u_{(p-1)q}^{r} + (1 - \frac{3}{2}\Theta)u_{pq}^{r} - \frac{1}{2}\sigma_\tau u_{(p+1)q}^{r} + \mathcal{D}$$

where,

**(7.85)**
$$\sigma_\tau = a\frac{\Delta\tau}{\Delta x}, \quad \Theta = \frac{\Delta\tau}{\Delta t}, \quad \mathcal{D} = \frac{1}{2}\theta\left\{4u_{p(q-1)} - u_{p(q-2)}\right\}$$

**Stability Analysis using Pseudo Time stepping.** Let us perform a stability analysis to see how things are going to work out. We know that as we proceed in pseudo time $\tau$ if all is going well that

**(7.86)**
$$u_{pq}^{r} \to u_{pq}$$

At some intermediate iteration we define the error in our current pseudo time step solution as

**(7.87)**
$$\epsilon_{pq}^r = u_{pq} - u_{pq}^r$$

Use this in equation (7.83) to get an equation for $\epsilon_{pq}$ as

**(7.88)**
$$\epsilon_{pq}^{r+1} = \epsilon_{pq}^r - \Delta\tau \frac{3\epsilon_{pq}^r + 3\Delta u_{p(q-1)} - \Delta u_{p(q-2)}}{2\Delta t} - a\Delta\tau \frac{\epsilon_{(p+1)q}^r - \epsilon_{(p-1)q}^r}{2\Delta x} + a\Delta\tau \frac{u_{(p+1)q} - u_{(p-1)q}}{2\Delta x}$$

As it turns out that when the pseudo time steps have converged we would have solved the equation

**(7.89)**
$$\frac{3u_{pq} - 4u_{p(q-1)} + u_{p(q-2)}}{2\Delta t} + a\frac{u_{(p+1)q} - u_{(p-1)q}}{2\Delta x} = 0$$

If we add and subtract $4u_{pq}$ to the first term of equation (7.89) we get

**(7.90)**
$$\frac{3\Delta u_{p(q-1)} - \Delta u_{p(q-2)}}{2\Delta t} + a\frac{u_{(p+1)q} - u_{(p-1)q}}{2\Delta x} = 0$$

Multiplying through by $\Delta\tau$ we substitute the resulting equation back in to equation (7.88) to get an equation in terms of $\epsilon$ alone.

**(7.91)**
$$\epsilon_{pq}^{r+1} = \epsilon_{pq}^r - \Delta\tau \frac{3\epsilon_{pq}^r}{2\Delta t} - a\Delta\tau \frac{\epsilon_{(p+1)q}^r - \epsilon_{(p-1)q}^r}{2\Delta x}$$

The gain $g_\tau$ turns out to be

**(7.92)**
$$g_\tau = (1 - \frac{3}{2}\Theta) - i\sigma_\tau \sin\theta$$

Stability requires that the modulus of the gain be less than one. That is

**(7.93)** $$|g_\tau|^2 < (1 - \frac{3}{2}\Theta)^2 + \sigma_\tau^2 \sin^2\theta < 1$$

$g_\tau$ takes its maximum value for $\theta = \frac{\pi}{2}$. From here it is clear that

**(7.94)** $$-\frac{3}{2}\Theta(2 - \frac{3}{2}\Theta) + \sigma_\tau^2 < 0$$

would ensure that the automaton is stable.

To see this in terms of our original time step we expand out all the terms to get

**(7.95)** $$\frac{9}{4}\left(\frac{\Delta\tau}{\Delta t}\right)^2 - 3\frac{\Delta\tau}{\Delta t} + a^2\left(\frac{\Delta\tau}{\Delta x}\right)^2 < 0$$

If we divide through by $\Theta^2$ we get

**(7.96)** $$\frac{9}{4} - 3\frac{\Delta t}{\Delta\tau} + \sigma^2 < 0, \quad \sigma = a\frac{\Delta t}{\Delta x}$$

We really need to derive the modified equation for discretisation given in equation (7.89) to find out when, if at all, all the dispersive and dissipative terms disappear. We will for now look at the case when $\sigma = 1$. This tells us that we will have a stable iteration for the pseudo time marching if

**(7.97)** $$\Delta\tau < \frac{12}{13}\Delta t$$

Just for fun, we could ask the question when can we take $\Delta\tau = \Delta t$. Substituting back into equation (7.96) we get

$$\sigma^2 < \frac{3}{4} \tag{7.98}$$

**One-Dimensional Euler's Equation.** Let us find out how it works with the one-dimensional Euler's equation

$$\frac{\partial Q}{\partial \tau} + \frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} = 0. \tag{7.99}$$

The naive thing to do is to just add the unsteady term in pseudo time to the Euler equation as we have just done. This gives us an automaton to generate a sequence of $Q$ in pseudo time. Every pseudo time step, one needs to then extract the flow parameters as required to calculate the other terms of the equation on order to take the next pseudo time step. Since, most of our work here is in taking pseudo time steps and the pseudo time derivative goes to zero anyway, why not add a term that makes the computation easier. We could instead look at the equation

$$\frac{\partial W}{\partial \tau} + \frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} = 0. \tag{7.100}$$

where $W$ is some convenient set of state variables like $Q$, $\tilde{Q}$, and so on.

Let us consider different ways that one could proceed to discretise these equations. First, we can repeat the process used for the linear wave equation. We can use an implicit scheme in time and an explicit scheme in pseudo-time so as to get an explicit scheme. Again to illustrate we will use a second order backward-time representation of the time derivative to get the following equation.

$$\textbf{(7.101)}\quad \frac{W^{r+1}_{pq}-W^{r}_{pq}}{\Delta\tau}+\frac{3Q^{\textcircled{r}}_{p\underline{q}}-4Q_{p(q-1)}+Q_{p(q-2)}}{2\Delta t}+\frac{E^{\textcircled{r}}_{(p+1)\underline{q}}-E^{\textcircled{r}}_{(p-1)\underline{q}}}{2\Delta x}=0.$$

Implicit in physical time

Explicit in pseudo time

Implicit in physical time

Solving for $\tilde{Q}^{r+1}_{pq}$ we get

$$\textbf{(7.102)}\quad W^{r+1}_{pq}=W^{r}_{pq}-\Delta\tau\left(\frac{3Q^{r}_{pq}-4Q_{p(q-1)}+Q_{p(q-2)}}{2\Delta t}+\frac{E^{r}_{(p+1)q}-E^{r}_{(p-1)q}}{2\Delta x}\right)=0.$$

We can now march in $\tau$, that is, advance in index $r$ till we are satisfied that we have a steady state in $\tau$. A good initial condition in $\tau$ ( $r = 0$ ) is $Q^{0}_{pq} = Q_{p(q-1)}$. Or, an explicit step can be taken in physical time to get an initial condition for time-marching in pseudo time.

A second possible mechanism is to use an implicit scheme in pseudo time. In this case we rewrite equation (7.101) as

$$\frac{W_{pq}^{r+1}-W_{pq}^{r}}{\Delta\tau}+\frac{3Q_{pq}^{\boxed{r+1}}-4Q_{p(q-1)}+Q_{p(q-2)}}{2\Delta t}+\frac{E_{(p+1)q}^{\boxed{r+1}}-E_{(p-1)q}^{\boxed{r+1}}}{2\Delta x}=0. \tag{7.103}$$

Implicit in pseudo time

As done in the earlier case, to get the Delta form we expand $E$ in using Taylor's series and retain the first two terms alone. This taken along with chainrule gives us

$$E_{pq}^{r+1}=E_{pq}^{r}+A_W\frac{\partial W}{\partial\tau}\Delta\tau,\quad A_W=\left.\frac{\partial E}{\partial W}\right|_{pq}^{r} \tag{7.104}$$

Substituting back into equation (7.103) and adding and subtracting $3Q_{pq}^r$ from the physical time derivative term we get

$$\begin{aligned}\frac{\Delta W_{pq}^{r}}{\Delta\tau}&+\frac{3}{2\Delta t}(Q_{pq}^{r+1}-Q_{pq}^{r})\\&+\frac{3Q_{pq}^{r}-4Q_{p(q-1)}+Q_{p(q-2)}}{2\Delta t}+\frac{\partial A_W\Delta W}{\partial x}=-\frac{\partial E}{\partial x}.\end{aligned} \tag{7.105}$$

If we define the Jacobian $P_W = \dfrac{\partial Q}{\partial W}$ and take the second order time derivative to the right hand side we get

$$\textbf{(7.106)}\quad \frac{\Delta W_{pq}^r}{\Delta \tau} + \frac{3}{2\Delta t} P_W (\Delta W_{pq}^r) + \frac{\partial A_W \Delta W}{\partial x} = -\frac{\partial E}{\partial x} - \frac{3Q_{pq}^r - 4Q_{p(q-1)} + Q_{p(q-2)}}{2\Delta t}.$$

The right hand side is our residue $R$. Multiplying through by $\Delta \tau$ and factoring out the $\Delta W_{pq}^r$ we get

$$\textbf{(7.107)}\quad \left\{ I + \frac{3}{2}\frac{\Delta \tau}{\Delta t} P_W + \Delta \tau \frac{\partial A_W}{\partial x} \right\} \Delta W = -\Delta \tau R_{pq}^r.$$

Clearly, most schemes to accelerate convergence, whether it is local time stepping in $\tau$, residual smoothing, preconditioning the unsteady term can all now be included. For example, preconditioning the unsteady term would simply require the change in one term in equation (7.107) to get

$$\textbf{(7.108)}\quad \left\{ \Gamma_W + \frac{3}{2}\frac{\Delta \tau}{\Delta t} P_W + \Delta \tau \frac{\partial A_W}{\partial x} \right\} \Delta W = -\Delta \tau R_{pq}^r.$$

$\Gamma_W$ needs to be determined appropriately.

---

**Assignment 7.7** Apply the dual-time stepping scheme to the one-dimensional Euler's equation and solve the transient problem corresponding to 4.6.

---

## Important ideas from this chapter

- A variational formulation maybe possible for the given problem.
- It is possible that a discontinuous representation is more "faithful" to a function in comparison to a continuous representation.
- If a class of finite difference/element/volume equations, random walks, and molecules jiggling in a rod all have a continuum model which is the heat equation, they can be used as approximate models for each other.
- It is possible to use a hierarchy of grids to accelerate convergence of a numerical scheme as long as one ensures the solution on the finest grid.
- One can not just pick up any old scheme and assume that it can be used for computing unsteady flows.
- Typically, one should use at least a second order scheme. However, the results of the exercise shown in figure 2.31 should be borne in mind. One can not just keep taking smaller and smaller time steps.
- Just like one can try to compute the steady state solution to a problem by looking for the steady state asymptote of the unsteady equations for the problem. One can add a pseudo time term to any unsteady equation and seek the steady solution in pseudo time...which would give the unsteady solution in real time.
- It is not possible to capture all time scales with the same degree of accuracy. One has to decide which physical phenomenon is dominant and/or of interest and try to capture that as well as possible.

- Extra care must be taken in applying boundary conditions.

# Closure

**If you are right 95% of the time, there is no sense in worrying about the remaining 3%** - Anonymous

**Faith is a fine invention**
**For gentlemen who see;**
**But microscopes are prudent**
**In an emergency!**
-Emily Dickinson

We have looked at various techniques to represent entities of interest to us. We have seen how to get estimates of the error in the representation of mathematical entities. We will now look at this whole process in a larger context.

## Validating Results

How good are the approximations that we generate? How do we know they are correct? Remember where we started this discussion back in chapter 1. Also bear in mind what we want to do - answer the two questions that we just asked. We will recast the discussion from the first chapter in a broader setting.

We have the world around us and in a attempt to understand and predict this complex world we try to model it. Figure 8.1 tries to capture the processes involved. The figure consists of

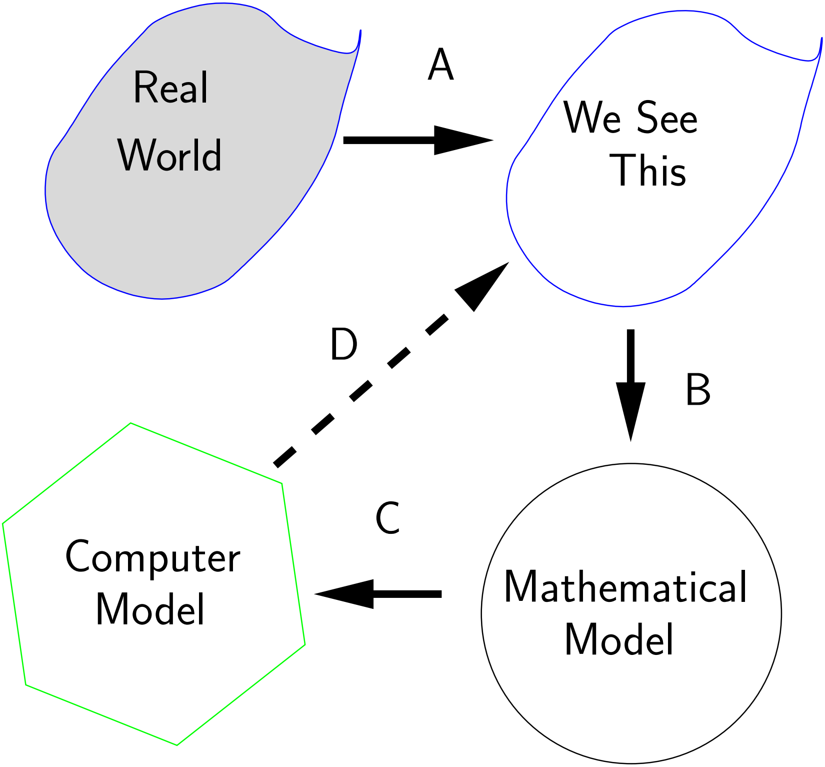

Figure 8.1. How CFD works. The polygon at the end is our approximation to the circle. The process marked "A" is perception. The process marked "B" maybe abstraction. The process "C" is the CFD. Finally the process marked "D" would be validation

arrows going between various shapes. The shapes have explanatory labels on them. One of the points that I am trying to make through the figure is that we have only a faint idea of what the real world is like. The arrow or limb marked as $A$ is the process of perception. Take vision for example. Our vision is limited to a range of frequencies of electro-magnetic radiation in which we can see. We call this range that we perceive "light". We can't see x-rays for example. However, by accident we may discover their existence and try to come up with an arrangement to see them. Here, once we have a theory for light as electro magnetic radiation, we can predict the existence of radiation not visible to us. The fact of the matter is that we usually cannot test limb $A$ in the figure. We may not know what we do not know. In this case, we do not see the grey shade in the figure, in fact no one sees it or perceives it and that is that.

We would then take what ever we see and try to model it. If we look at what we perceive, we see that it is quite complex. We try to model this by throwing away non-essentials and typically create a mathematical model. In the figure we choose to represent the mathematical model as a circle. This is to convey as clearly as possible, the loss of detail and our attempts, very often, to keep the model as simple as possible. Sometimes, we may be tempted and fascinated by the symmetries exhibited by our mathematical model. The first question to ask is "does the mathematical model represent the reality that we choose to represent?" That is, how do we test limb $B$ of the figure? Normally, we can't answer this question unless we use it to solve / model a problem that can then be used in an attempt to answer this question. We usually find that this mathematical model is in itself complicated and difficult to solve. By making appropriate assumptions and approximations we end up modelling the mathematical model on the computer. In our case we assume the Navier-Stokes equations are correct and then proceed to approximate these equations on the computer using the techniques of CFD to get and algorithm and techniques of programming to get an actual implementation. The discretisation process we have studied so far in this book is indicated by the polygon that is used to approximate the circle. Quite often, we can not get an exact representation of our mathematical model on the computer and hence we end up approximating the mathematical model. We ask, how well does the computer model represent the original mathematical model. This is an important question when we are trying to develop techniques to solve equations that arise in the modelling of nature. This also questions the competence of doing limb $C$. It is complicated by the fact that one needs to ensure that the program captures the algorithm appropriately.

The other question that is often asked, especially by someone who is interested in the application of the model to solve problems is: How well does the computer model represent reality? Here we are short circuiting all of the previous questions asked before and instead say, "Just tell me if the answer mirrors reality?" This would check whether limb $D$ exists or not. This is not always as easy to answer. A short cut would be to perform experiments and compare the results with the output of our program. However, always bear in mind that the errors in the experiment and the errors in the computer model may be such that the results agree. The issue then is: what confidence do we have that they will agree when we do not have experimental data?

This is the problem of validation[**Roa00**]. We now address different ways to evaluate or validate our programs.

Consider, for example, Laplace's equation. We know from complex analysis [**Ahl79**][**Chu77**] that any analytic function is a solution to Laplace's equation. We pick an analytic function ( say $z^2$ ) and use the real part as our solution. We evaluate the solution that we have picked on the boundaries and use this as the boundary condition for Laplace's equation. We now act as though we did not know the solution and solve the problem using any of our favourite techniques. Since, we actually know the solution we can figure out how each solution technique is doing.

What if we did not know any solution? Well, make up a function say

$$u(x,y) = x^2 + y^2 \tag{8.1}$$

As before one can evaluate this function on the boundaries and obtain the boundary conditions for the problem. If we were to substitute equation (8.1) into Laplace's equation it will leave a residue. You can check for your self that the residue is 4. This tells us that $u$ given by equation (8.1) is a solution to

$$\nabla^2 u = 4 \tag{8.2}$$

This does not change the essential nature of our equation. We now have a problem to which we know the solution. We can now compare the computed solution to actual solution of this equation.

How does one pick the solution? Looking at the differential equation, one knows that the solution to Laplace equation will have second derivatives in the various coordinate direction. So, we choose functions of our independent variables that have this property. If we know more, for example, that the equation or the original problem will allow for some discontinuity. We can then pick such problems. We did this for the wave equation, when we took the step function as the initial condition.

The choice of functions for the solution can be made due to some very sophisticated reasons. One may want to test the ability of the computer model to pick up some aspect. As has already been noted, discontinuities are one such artefact. One could test the ability to pick maxima, minima, the correct dispersion and so on.

## Computation, Experiment, Theory

The first part of this chapter spoke in generalities. We now look at this triad that makes up current scientific endeavour. Theory is the abstraction of exploratory experiment and computation. Abstraction is the process of extracting the general and throwing away the specifics relating to a particular phenomenon. Exploratory experiment gives the broad outlines of the terrain in which our problem resides. Theory abstracted from these experiments provides the direction for future experiments and computation that can be used to confirm or refute the theory. Experiments can also be used to refine theories and help make the theoretical framework more precise. They can help add detail to make the theory work for the particular. The experiment can be used to demarcate the boundaries of applicability of the theory. That is refute the theory in some instances and confirm it as acceptable in other instances.

For example, the theory that for a fluid stress is proportional to the deformation rate is a very general statement of a theory. It can be verified for a class of fluids ( limit the theory to a class of fluids ). In that class of fluids it can be verified for a class of flow conditions ( again limit the applicability of the theory). Having achieved this, the theory may be able to give clues as to other experiments that can be made to make the theory "useful". For example, in the case of fluids, the coefficient of viscosity may be measured using a rheological experiment like Couette flow. Thus experiment can be used to specialise the theory for the specific problem / material at hand. Computation can be used to perform whole scale experiments. Depending on the level of the computational model, one can explore a range of theoretical parameters that may be difficult to achieve with experiment. Computational studies can also be used to motivate the direction of experiment. Confidence in computation can be obtained by validation against experiment and theory. Failure of agreement in any two of the theory, experiment and computation can be used by the theoretician, experimentalist and the computer (as in the person who computes or uses computational techniques), to improve their respective models. Every human intellectual artefact is improved by extending its boundary of applicability. In order to extend the boundary of applicability, one needs to know the location of the boundary. With the advent of computational techniques, we have gone from the binocular vision of experiment and theory to trinocular vision. It is easy to do theory and computation with no reference to reality. It is easy to accumulate experimental data with no reference to theory or computation.

One observation must be made here with reference to this trinocular view. It is obviously effective only on the intersection of the three views. Herein lies the problem and source of tension. A theoretical observation that is easy to state and compute may be difficult to achieve in experiment. For example, we want to setup a flow field with zero velocity in room. This is possible to do, relatively easily in computation. Extremely easy for the theoretician, $\vec{V} = 0$. Very difficult if not impossible for the experimentalist to achieve. At the other extreme, to swirl a glass of water around and drink it would be a relatively easy experiment to perform, which involves considerable ( to understate the facts ) difficulties for the theoretician and computer. The point is that, in order for the trinocular vision to work, continuous effort must be made to seek test cases in the overlap region. The individual disciplines of theory, experiment and computation themselves are enhanced by an effort to increase the overlap.

# Computers

There are numerous books out there to explain the workings of computers with various degrees of detail. We will look at an analogy to understand the issues involved when it comes to high performance computing which is the realm in which CFD resides.

We will use the student registration system as it was run the first time it was started at Indian Institute of Technology Madras (IITM). For the sake of the analogy I have tweaked a few of the details.

**The Layout:** IITM is located in a 650 acre campus. At the centre is a five storied central administrative building (adblock). On the fifth floor of this building is the senate room where students are to be registered for the new semester.

**Cast of Characters:** The students are to be registered by the "Professor in Charge of the Unregistered". We will call this person the CPU. The CPU is a busy person and does not like to be kept idling. The students are located about a kilometre away in ten different hostels (or dormitories). Each hostel houses 192 students.

**The Program:** Each student

(1) needs some discussion with the CPU. (about 4 minutes)

(2) will fill out a registration form. (about 4 minutes)

(3) will hand the form to the CPU who will verify that everything is fine with the form and sign it. (about 4 minutes)

**The Time Element:** Each student takes ten minutes to bicycle down to the adblock. It takes a couple of minutes up the elevator into the senate room and then 12 minutes ($3 \times 4$ for each of the above steps) to finish registering.

This can be implemented in various ways.

(1) The first simple minded solution is to have the CPU call the hostel. The student bicycles down and registers after which the next student is called. Now while the student is bicycling down and taking the elevator the CPU is idling. Each student takes about 25 minutes to process and the CPU is going to be stuck there for days.

(2) As it turns out, the lobby of adblock can comfortably handle 32 students. We could use the institute minibus to fetch 32 people from the hostel to adblock. The bus has a capacity of 16 students. Corralling the 16 and bringing them in would take about ten minutes. Now all that the CPU has to do is call down to the lobby and ask for the next student to be sent up. The student takes a couple of minutes up the elevator and can then proceed with the registration. From the CPUs perspective, each student takes about 15 minutes to process. There is still idle time present. When sixteen of the students are done in the lobby they get bussed back to the hostel and the next sixteen replaces them.

(3) We can now add a wrinkle to the above scenario. Outside the senate room there is an ante-room which can accommodate four students. Now the CPU only rings a bell which is a signal to the next student to enter the senate room. In this fashion we have eliminated the ride up the elevator showing up on the CPUs clock.

(4) If we pay attention to the CPU, he/she talks to the student and then waits while the student fills up the form. If instead of waiting, the CPU gets another student in and talks to the second student while the first is filling up forms, that idle time is eliminated. The CPU can verify the first students registration form while the second student is busy filling out his/her form. The first student can then be replaced by another student. So, the CPU has a pipeline with at most two students in it at any given time. With this structure or architecture we have reduced the time between students leaving the senate room on completion of registration to about 8 minutes per student instead of 12. Can we do better?

(5) We could have two professors in the senate room. One advises one student after another. The students move over one seat and fill out the registration form. The second professor verifies the forms and signs them. The pipeline has three stages and we get a student out every 4 minutes.

(6) We could organise another pipeline in the conference room adjacent to the senate room. and process 30 students per hour instead of 15 students per hour.

The last is very clearly a great improvement from the first version of about two students per hour. Now the analogy. In the computer, the CPU would be the Central Processing Unit. The Hostels would be the computer memory. The minibus would be the memory bus connecting the CPU to the memory. Fetching students sixteen at a time and keeping them waiting in the lobby is called caching. Having a smaller cache on the fifth floor basically means we have a two level cache. Typically in computers the cache on the top floor is called the level one cache or the L1 cache. The cache on the ground floor would be the L2 cache. The cost of not having a data item in the L1 cache is high. The cost of not finding it in the L2 cache is very high.

Another point to consider: can we go to four rooms with four CPUs or eight rooms with eight CPUs? Remember the elevator can only handle 120 students per hour and the bus can handle about 96 students per hour. The two lessons we get from this, try to access the data in the order in which it is available in the cache. The rate at which data can come into the CPU is a strong determinant of the performance of the computer. We cannot just keep on increasing the number of CPUs.

A final note of caution here. The example given above fall into the category of an embarrassingly parallel problem. That is, the registration of one student did not depend on the other. The students could be registered in any order. We have not looked at the flow of the data relevant to each of the students to the CPU. Nor have we looked at the nature of the courses. If courses had constraints on the number of students that can register for a given course, then we start to see complications.

How does one deal with two students in two different rooms contending for the last seat in a given course. How does one know the state of the course at any time? For example, how do you know there is only one seat left?

The objective here was to give a reasonable picture of the computer so that the student can endeavour to write good programs. Warning: If you reached this appendix because of a reference from chapter 2 you may wish to go back now. You could continue reading. However, you may encounter terms that are defined in chapters that follow chapter 2. We can now see how we can go about writing programs on computers.

### How do we actually write these programs?

As I had indicated in the introduction we need to bring together different skills to this material. Fluid Mechanics, Mathematics and Programming. In an introductory book, the fluid mechanics maybe restricted and the mathematics may predominate the discussion. However, you really should code if you want to really understand the process.

So, how does one go about developing these codes. My intent is not to introduce serious design methodologies here. However, using a little common sense, one can easily create functionally correct code – meaning it works.

I would classify people who write code into five different categories based on an aerospace engineering analogy. Let's look at someone programming equivalent to designing and building an airplane.

**Duh:** Code? I know section 212 of the motor vehicles act.
Nothing wrong with being in this state. If you want

"to CFD" though you will need to put in the minimal amount of effort required to switch to the next level. I would suggest that you pick up a relatively high level programming language. My personal preference right now is Python.

**Novice:** Code is like the Wright flyer: It works. This is fine. You can get by with a little help from your friends. You can learn "to CFD" which is one of the aims of this book. On the other hand, anytime we start working on something, by the time we are done, we usually know a better way to do it, having now had the experience of doing it the first time. I have noticed that people in the novice state (remember, no one was born programming) tend to cling to code that they should discard, since they barely got it working. Practice and paying attention to the process of programming will get you to the next level.

**Proficient:** I can fly it. Not sure whether anyone else can operate it. At this point code writing should not be such a hassle that you cling to any particular piece of code. At this point you can not only learn "to CFD", you can actually do decent research in CFD without the need of a collaborator to take care of the coding. This level of proficiency gives you the confidence to check out new algorithms. Now, if you paid attention, you would realise that your confidence may be justified by the fact that you are able to capture many of the mistakes that you make. If you will recognise two things:

a. Every one makes mistakes, no one is infallible

b. Every one acquires habits and quirks. If you pay attention while you are learning something, (and here I will admit it helps if you have someone to help you out) you can make sure you acquire habits that make you less likely to make an error and consequently, more efficient.

**Reeeeealy Good:** My programs are robust. Can be used for scheduled flights of an airline, However, they need to be flown by experienced pilots. You have the right habits. You know that errors occur because of bad programming practice and that you the individual has only one responsibility: to learn from every mistake.

**Super Programmer:** Someone who is absolutely not tech-savvy can use this code; the average person can fly to work. Well, you have the gift, enjoy. Incidentally, I have this little program that I want written... Finally, in this stage it is very easy to stop learning, the process never ends, the day you cease learning is the day you no longer are.

So, where should you be as far as your programming skills are concerned. You really have to at least be at the novice level. You have to be able to create functionally correct code.

Here are some simple rules.

**First:** Don't Panic. Don't look at the whole problem and wonder: How am I ever going to get this done? Where do I begin?

**Second:** We begin, as always, with the analysis of the problem at hand. This is because we are attempting the creation of something that does not exist till we write it - a program. This is done by synthesis - or putting the parts that make up our program together. To this end try to see if you can identify what parts and steps make up your problem. For example, if you are looking at solving the one-dimensional first order Euler's equation, we see that

- We need to decide the scheme to be used: say FTCS
- FTCS requires data at the current time step and will generate data at the next time step. [ I need to have an **array** in which to store my current data and one in which to store the new data ]
- My algorithm will use $Q$ and $E$, whereas I am interested in $\tilde{Q}$. All of these may have to be stored at each grid point.
- Need to apply boundary conditions on the left and right.
- Need to be able to take one FTCS step for all the interior grid points.
- Need to take a lot of time steps and then decide when we are done.
- Need to visualise results or get some idea as at what is the state of the code
- Clearly, we need to interact with the user.
- What programming language should I use. [ Any thing in which you are proficient will do for now, remember the Wright flyer. You want to develop a functionally correct program not some super-duper code]

The actual process is a little more involved. This will suffice for now.

**Third:** Extract out stuff to be done. Sort them out so that you can identify things that can be done directly.

**Fourth:** You need to decide how you will handle $Q$ and $\tilde{Q}$. In Python they could be implemented as classes as follows:

(1) ConsFlowParm represents $Q$,
(2) NonConsFlowParm represents $\tilde{Q}$,
(3) FluxE represents the flux term $E$.

```
Gamma = 1.4 # Cp / Cv
Gm1 = Gamma - 1.

class ConsFlowParm:
    """ Manages the conservative flow parameters"""

    def __init__( self ):
        self.Q = numpy.zeros(3)

    def Rho( self ):
        return self.Q[0]

    def RhoU( self ):
        return self.Q[1]

    def RhoEt( self ):
        return self.Q[2]

    def U( self ):
        return self.Q[1] / self.Q[0]

    def P( self ):
        return Gm1*(self.Q[2]-0.5*self.Q[1]*self.U())
```

```
class NonConsFlowParm:
    """Manages non-conservative flow parameters"""

    def __init__( self ):
        self.Q_ = numpy.zeros(3)

    def Rho( self ):
        return self.Q_[0]

    def U( self ):
        return self.Q_[1]

    def P( self ):
        return self.Q_[2]


class FluxE:
    """Manages the x-component of the flux"""

    def __init__( self ):
        self.E = numpy.zeros(3)

    def MassFlux( self ):
        return self.E[0]

    def MomFlux( self ):
        return self.E[1]

    def EnergyFlux( self ):
        return self.E[2]

    def SetFlux( self, Q ):
        self.E[0] = Q.RhoU()
        self.E[1] = Q.RhoU() * Q.U() + Q.P()
        self.E[2] = ( Q.RhoEt() + Q.P() ) * Q.U()
    def GetFlux( self ):
        return self.E
```

**Fifth:** Having defined the two types of flow parameters $Q$, $\tilde{Q}$ and the flux term $E$, we look at evaluating these terms from each other as required in our algorithms. As we have seen earlier our equations are in terms of $Q$ and we are planning on marching in time and obtain $Q$ from the governing equation. To solve the equations, given $Q$ at some time step we need to evaluate $E$ at that step. We need $\tilde{Q}$ as they are typically the parameters in which we are actually interested.

We need to implement and test functions that

- given $\tilde{Q}$ return $Q$ is done using NonCons2Cons,
- given $Q$ return $\tilde{Q}$ is done using Cons2NonCons,
- given $Q$ return $E$, this has actually been implemented as a method in the class FluxE.
- given $Q$ find $a$, the speed of sound

```
def Cons2NonCons( Q ):

    """Converts from the Conservative flow parameters
    to the Non-Conservative flow parameters"""

    tmp = NonConsFlowParm()

    tmp.Q_[0] = Q.Rho()
    tmp.Q_[1] = Q.RhoU() / Q. Rho()
    tmp.Q_[2]=Gm1*(Q.RhoEt()-0.5*Q.RhoU()*tmp.Q_[1])

    return tmp

def NonCons2Cons( Q_ ):
    """Converts from the Non Conservative flow
    parameters to Conservative flow parameters"""

    tmp = ConsFlowParm()

    tmp.Q[0] = Q_.Rho()
    tmp.Q[1] = Q_.Rho() * Q_.U()
    tmp.Q[2] = Q_.P()/Gm1 + 0.5*tmp.Q[1]*Q_.U()

    return tmp
```

What do all of these items have in common? They have nothing to do with grids or the kind of solver one is going to use. In fact they form a part of a core structure that one should implement. Each of these functions should be tested. Once you know they work, set them aside to be used by any of your solvers. Make sure you test them thoroughly.

**Sixth:** We will now need to create two arrays of $Q$, $E$ and $\tilde{Q}$. Where are these arrays created? Right now that can be in your main program. What is the size of theses arrays? We may need to interact with the user to find out.

**Seventh:** Determine all the interactions with the user. Try to keep all of this in one place in your program. Typical interactions for the one-dimensional Euler's equation solver would be

- Physical extent of the problem?
- Inlet condition: Subsonic? Data?
- Exit condition: Subsonic? Data?
- How many grid points?

  - $\sigma$?
  - Convergence criterion: How many time steps?

  This interaction is simple enough that it may be done through one function.

**Eighth:** Go ahead and create all the arrays. The ones at the current time step need to be initialised. The creation and initialisation can be done a function.

**Ninth:** Now we can write the main part of the code. Write a function

  - given $Q$ and $\sigma$, find time step.
  - Given $Q$ at the current time step takes a step using our chosen scheme.

**Tenth:** Code the program that will call all of these functions to achieve our goal.

```
class FlowManager:
    """Manages the full problem...uses some solver"""

    def __init__( self ):

        self.N = input( " How many grid points \
        including the boundary? " )

        # Should really generalise this later
        # to take care of supersonic and other
        self.InletBC = SubSonicInlet()
        self.ExitBC = SubSonicExit()

        self.x = []   # to store the coordinate locations
        self.dx = []  # This is used for computations

        self.Qlist = []
        self.Q_list = []
        self.E_list = []
        self.SetInitialCond( ) # Will populate above list
        self.SingleStep = FTCS
        self.pm = plotmanager.PlotManager()

    def SetInitialCond( self ):
        """ Set the conditions at time t = 0"""

        DX = 1.0 / ( self. N - 1. )
        for i in range( self.N ):

            CQ =  ofp.ConsFlowParm()

            # ambient pressure throughout the duct
            CQ.Q[0] = self.ExitBC.Rhoa
            CQ.Q[1] = 0.0 #Initial velocity is zero...
            # uses the above fact here...
            CQ.Q[2] = self.ExitBC.Pa / Gm1

            NQ=ofp.NonConsFlowParm()# create object
            NQ=ofp.Cons2NonCons( CQ )# initialise object

            self.Qlist.append( CQ )
            self.Q_list.append( NQ )

            self.dx.append( DX )
            self.x.append( DX * i )

    def Step( self, L = 0.0002 ):
        e2 = 0.01 # added to in book
        e4 = 0.001# to help format line
        dQ = self.SingleStep(self.Qlist,L)
        dQ2 = SecondOrderDissipation(self.Qlist)
        dQ4 = FourthOrderDissipation(self.Qlist)
        for i in range( len( self.Qlist )):
          self.Qlist[i].Q-=dQ[i]-e2*dQ2[i]+e4*dQ4[i]

        self.Qlist = self.InletBC.ApplyBC(self.Qlist)
        self.Qlist = self.ExitBC.ApplyBC(self.Qlist)

        self.pm.ShowX( self.Qlist, self.x )
```

Where PlotManager allows one to see plots of $\rho$, $\rho u$, and $\rho E_t$ in a graphical form on the screen.

**An example using the four step Runge-Kutta Scheme.** This algorithm will be given in a general form for the equation

**(A.1)**
$$\frac{d}{dt}\int_\sigma Q d\sigma = -\int_S \vec{f}\cdot\hat{n} dS$$

If we were to discretise our domain into small volumes with volume $\Delta V$ having $m$ faces each with area $\vec{A}_i = s_i\hat{n}_i$, [ no sum over $i$], we could write for a volume

**(A.2)**
$$\frac{dQ}{dt}\Delta V = -\sum_{i=1}^{m}\vec{f_i}\cdot\vec{A}_i = -F(Q)$$

Where, $Q$ now represents the mean values in the volume or cell and $F(Q)$ represents the net efflux from the volume.

We are given the initial condition $Q^0$. In order to take one time step of magnitude $\Delta t$ we do the following. note: read the symbol $\forall$ as "for all".

Given $Q^0$, compute $\vec{f^0}$, and the artificial dissipation $D^0$.

(1) $\forall$ FVolumes use given $Q^n$, $\vec{f^n}$ and, $D^0$

   (a) Compute $G(Q^n) = F(Q^n) + D^n$.
   (b) Set $RQ = G$.
   (c) Compute $Q^* = Q^n - 0.5\times\Delta t G(Q^n)$.

(d) Compute $\vec{f}^*$

(2) $\forall$ FVolumes

(a) Compute $G(Q^*) = F(Q^*) + D^n$.

(b) Set $RQ = RQ - 2G$.

(c) Compute $Q^* = Q^n - 0.5 \times \Delta t G(Q^*)$.

(d) Compute $\vec{f}^*$

(3) $\forall$ FVolumes

(a) Compute $G(Q^*) = F(Q^*) + D^n$.

(b) Set $RQ = RQ - 2G$.

(c) Compute $Q^* = Q^n - 0.5 \times \Delta t G(Q^*)$.

(d) Compute $\vec{f}^*$

(4) $\forall$ FVolumes

(a) Compute $G(Q^*) = F(Q^*) + D^n$.

(b) Set $RQ = RQ - G$.

(c) Compute $Q^{n+1} = Q^n - \Delta t RQ/6$.

(d) Compute $\vec{f}^{n+1}$

(e) Compute $D^{n+1}$

(5) $\forall$ FVolumes compute the square of the error contributed by that volume to the whole as

**(A.3)**

$$\Delta E_{\mathsf{sqr}} = \sum_i \left( \frac{RQ_i}{Q_i} \right)^2, \qquad \text{do not divide by } Q_i \text{ when } |Q_i| > \epsilon$$

If $\Delta E_{\mathsf{sqr}}$ is greater than a predetermined convergence criterion then proceed back to step (1) to take the next time step.

We will use one the various versions of the Runge-Kutta scheme to implement it in Python. We inherit from the earlier FlowManager class. This simply says that the RK4FlowManager is a FlowManager except that we have modified how a time step is taken and consequently added a method: IntermediateStep and a data element:Qnew to reach that end.

```
OneSixth = 1. / 6.
class RK4FlowManager(FlowManager):
    """
    A flow manager that uses the four step
    Runge-Kutta method to march in time.
    """
    def __init__( self ):
        if hasattr( FlowManager, "__init__" ):
            FlowManager.__init__( self )
        self.Qnew = copy.deepcopy( self.Qlist )

    def IntermediateStep( self, L, alpha ):
        dQ = self.SingleStep( self.Qnew, L )
        dQ2 = SecondOrderDissipation( self.Qnew )
        dQ4 = FourthOrderDissipation( self.Qnew )
        for i in range( len( self.Qlist )):
            self.Qnew[i].Q = self.Qlist[i].Q -\
            ( dQ[i] - 0.01*dQ2[i] + 0.001 * dQ4[i] ) * alpha

        self.Qnew = self.InletBC.ApplyBC( self.Qnew )
        self.Qnew = self.ExitBC.ApplyBC( self.Qnew )


    def Step( self, L = 0.0002 ):
        self.IntermediateStep( L, 0.25 )
        self.IntermediateStep( L, OneSixth )
        self.IntermediateStep( L, 0.375 )
        self.IntermediateStep( L, 0.5 )
        self.IntermediateStep( L, 1. )
        self.Qlist = copy.deepcopy( self.Qnew )
        self.pm.ShowX( self.Qlist, self.x )
```

## Programming

I will just collect things here as I remember them and then collate later.

As computers evolve, some of these guidelines will change. Even the ones that start with “Never”.

- Try not to subtract [ I should say add things of opposite sign ]
- Add in preference to multiply
- multiply in preference to division
- square roots are expensive, trancendentals are worse.
- store multidimensional arrays as vectors.
- if you insist on using a multidimensional array, at least access it the same way that your computer stores it.
- Readability matters. This means the srtucture of your code is transparent, the choice of names (variables or otherwise is logical and informative) and all of these do what they claim to do and nothing else.
- Document! When an arbitrary choice is made (example: store two dimensional arrays row wise) - document! Do not drown the code and logic in documentation.

# Parallel Programming

In this day and age of cheap computers and multi-core CPUs, One necessarily has to pay attention to parallel computing

There is a lot going in your computer in parallel. Computer architecture has advanced to a stage where a lot of parallelisation is done in the CPU. One can benefit from this by simply helping the modern CPU do its job.

Beyond this is the realm where we want to deliberately run our program on multiple computers. We have so far seen the kind of problems that we would like to solve. How about the kinds of computers you are likely to encounter. Computers can be classified in many ways. From the point of view of this discussion we will restrict ourselves to simple classifications like "tightly coupled" versus "loosely coupled" or "Shared Memory" versus "distributed memory". An extreme example of a tightly coupled computer would be a vector computer. It has the capability of applying a single instruction of a vector of data [ some times called Single Instruction Multiple Data, SIMD for short ]. Such a computer would definitely be a plus where our problem is dominated by large number vector operations. It would however pay a penalty if we did not have many vector operations.

At the other extreme we could have a bunch of computers connected by a network of some sorts. They are individual computers that are capable of performing a part of a task and exchanging the data as required. These are clearly very loosely coupled. They also have their own memory and other resources. So these would also be called distributed memory systems and the use of these together to solve a single problem is referred to as distributed computing. In between these extremes we have multiple CPU machines that share the same memory. Each CPU sees the whole memory and can read and write to it. However, they are not vector machines. Each will have its own "thread of execution"

The fundamental problem of parallel computing is to break up the problem into sub-problems. These sub-problems may be related to each other very closely or may be completely independent of each other. If they are completely independent of each other we have an embarrassingly parallel problem. On the other hand if they are related very closely, we should check to see how we can loosen this coupling.

Here is a concrete example. Consider solving Laplace's equation on a unit square (see 3). We have seen how this problem can be solved numerically. We have also seen numerous ways by which the convergence to the solution can be speeded up.

Now we will see if can gain further speed up using parallel computing. Remember that every point was the average of its neighbours. Let us look again at that $[6 \times 6]$ grid we used earlier. It is redrawn with some embellishments in figure A.1.

The grid points are marked squares and circles. The interior points are indicated using filled circles and squares. The boundary points are indicated using open ( that is unfilled ) squares and circles. If you recollect, any interior grid point is obtained as the average of its neighbours. Here, we take this description one step further. Any interior square point the average of the neighbouring circle points. Any interior circle point is the average of the neighbouring square points. This means that all the

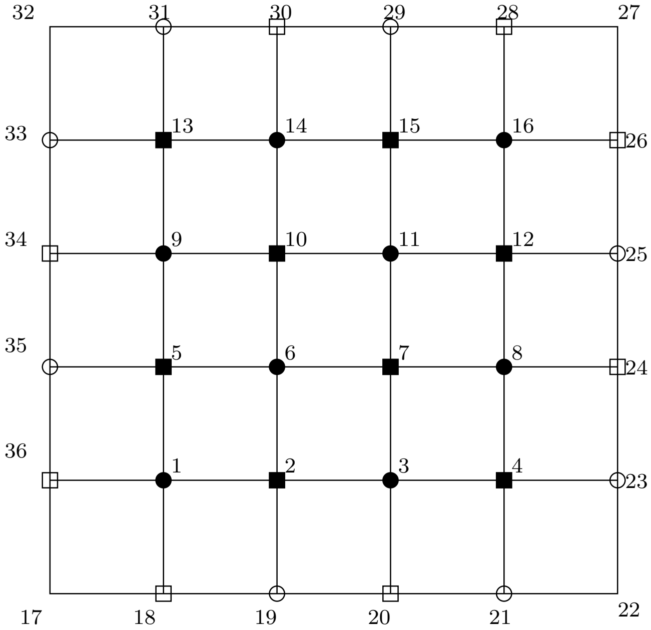

**Figure A.1. Grid points used to solve Laplace's equation on a unit square at the origin in the first quadrant. Any square depends on its circular neighbours and similarly any circle depends on the square neighbours. The boundary grids are shown as open (not filled ) circles and squares.**

square points can be evaluated and updated at the same time. That is

**(A.4)**
$$\phi_{i_\blacksquare}^{n+1} = \frac{\phi^n_{i+1_\bullet} + \phi^n_{i-1_\bullet} + \phi^n_{i+N_\bullet} + \phi^n_{i-N_\bullet}}{4}$$

and

$$\text{(A.5)} \qquad \phi_{i_\bullet}^{n+1} = \frac{\phi_{i+1_\blacksquare}^{n+1} + \phi_{i-1_\blacksquare}^{n+1} + \phi_{i+N_\blacksquare}^{n+1} + \phi_{i-N_\blacksquare}^{n+1}}{4}$$

We have captured an inherent parallelism in the Gauss-Seidel algorithm for this equation. On the other hand, if we would like to keep it simple, we could use Gauss-Jordan instead. Since the values are updated only after the full sweep, there is no coupling between the computation of the steps.

If you pay attention to what we are doing ( see equations (A.4), (A.5)) we are performing the same operations or instructions. The data or domain is divided to provide the parallelism. This clearly falls into the category of SIMD or SPMD [ Single Program Multiple Data] category. The process we are following is called domain decomposition. Once we realise that we are doing domain decomposition we can ask: Are there other ways of decomposing the domain to get our parallelism?[1]

Figure A.2 shows another easy way to decompose the domain. In this case the domain is broken up into four parts. These can now be solved separately. It should be noted however, that grid points 2,6,5 require data from points 3,7,10, and 9. This data needs to be obtained from the adjacent domains.

We can now ask the question: Is it possible to decompose the problem in such a manner as to reduce the dependence on the neighbouring domains?

[1] I hope this situation here looks familiar. The domain is like the support of a function. The program here is analogous to the function. We seek orthogonality or independence. We typically get it when the domains/support is non-overlapping.

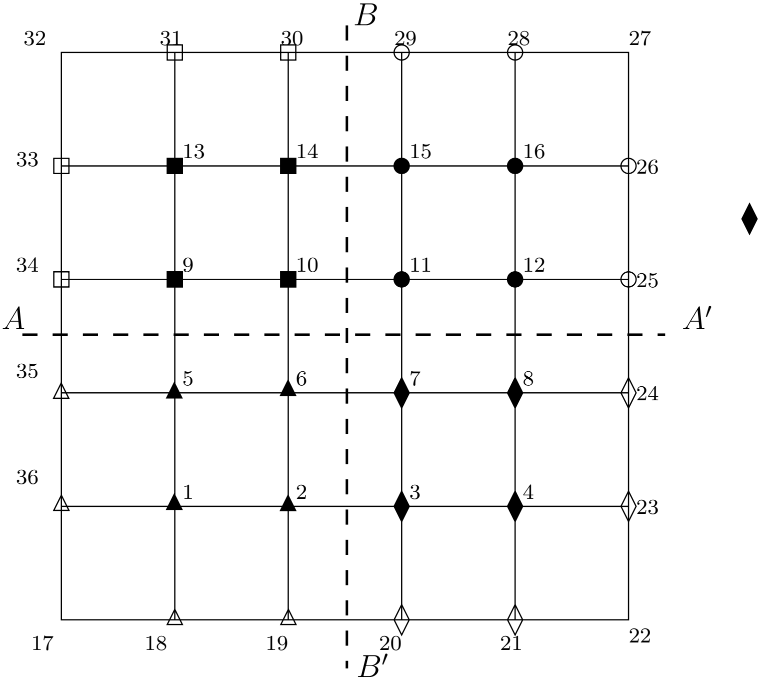

**Figure A.2.** Grid points used to solve Laplace's equation on a unit square at the origin in the first quadrant. The domain is shown divided into four parts using two broken lines: $A - A'$, $B - B'$. The grid points are indicated appropriately. It should be noted that to compute some of these grid points data from adjacent domains is required. This data needs to be made available as and when required.

# Some Mathematical Background

You really should go out and get your mathematical background from books on the appropriate topics. However, I will present the bare minimum here to get an understanding on why we use entities like complex variables and matrices.

## Complex Variables

We will do a quick review of complex variables to just about meet the requirements of this book. A complex number is a two dimensional entity that can be represented by a point on a plane called the complex plane or an Argand diagram. If the axes of the coordinates are labelled $x$ and $y$, then an arbitrary point on this complex plane is denoted by a complex variable defined as

$$z = x + iy, \quad \text{where } i = \sqrt{-1} \tag{B.1}$$

We will call this the **Cartesian form** of a complex number.

As we saw in section 2.4 the $i$ basically stop you from adding the $x$ to the $y$. However, the algebra is interesting since $i \times i = i^2 = -1$. Since, in our usual numbers we cannot imagine the product of two numbers giving us a negative number, $i$ is called an imaginary number. In conversing about complex numbers, $iy$ ( or moer often $y$ ) would be called the imaginary part of the complex number. The non-imaginary part is called the real part. A complex number where the real part is zero is called purely imaginary. Consider two complex numbers

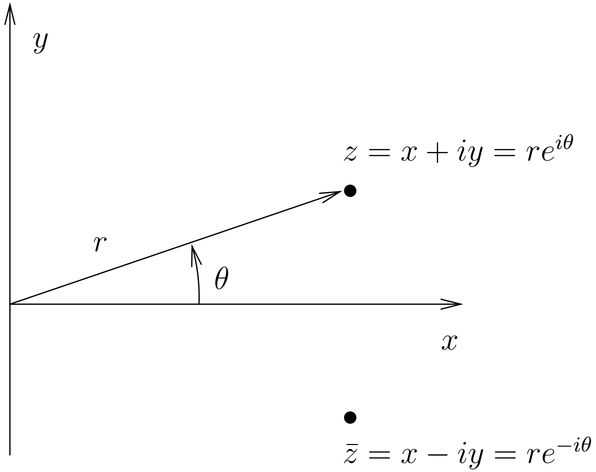
$y$
$z = x + iy = re^{i\theta}$
$r$
$\theta$
$x$
$\bar{z} = x - iy = re^{-i\theta}$

**Figure B.1. The complex plane or the Argand diagram. An arbitrary point $z = x + iy$ and its conjugate are shown. The polar form is also indicated.**

**(B.2)** $$z_1 = x_1 + iy_1, \quad \text{and } z_2 = x_2 + iy_2$$

You can verify the following

We define a conjugate corresponding to any complex number which geometrically is a reflection off the $x$-axis. Algebraically it involves just changing the sign of the imaginary part of the number. The conjugate of a complex number $z$ is denoted by a “bar” on top as in $\bar{z}$. In fact, placing the bar on a

| operation | result |
|---|---|
| $+$ | $(x_1 + x_2) + i(y_1 + y_2)$ |
| $-$ | $(x_1 - x_2) + i(y_1 - y_2)$ |
| $\times$ | $(x_1x_2 - y_1y_2) + i(x_1y_2 + y_1x_2)$ |
| $\div$ | $\dfrac{(x_1x_2 + y_1y_2) + i(x_2y_1 - y_1x_2)}{x_2^2 + y_2^2}$ |

**Table B.1. Various operations on two complex numbers $z_1 = x_1 + iy_1$, and $z_2 = x_2 + iy_2$**

complex number is the operation of conjugation on the number. So $z = \bar{\bar{z}}$. Now, we define

**(B.3)** $$|z|^2 = z\bar{z} = x^2 + y^2$$

$|z|$ is our idea of the Euclidean distance of the point $(x, y)$ from the origin. Since it is the distance from the origin we are talking about the position vector $\vec{r}$ and its magnitude is $|\vec{r}| = r = |z|$. This suggests that we can write in polar coordinates the complex number $z$ as

**(B.4)** $$z = re^{i\theta} = r\cos\theta + ir\sin\theta$$

where, $\theta$ is the angle measured from the positive $x$-axis to the position vector. This is called the **polar form** of the complex number. Both the forms of representing a complex number are useful. When adding or subtracting the standard for is useful. When one needs to multiply complex number then the polar form is very often easier to handle. The identity

**(B.5)**
$$e^{i\theta} = \cos\theta + i\sin\theta$$

is called **Euler's Formula**

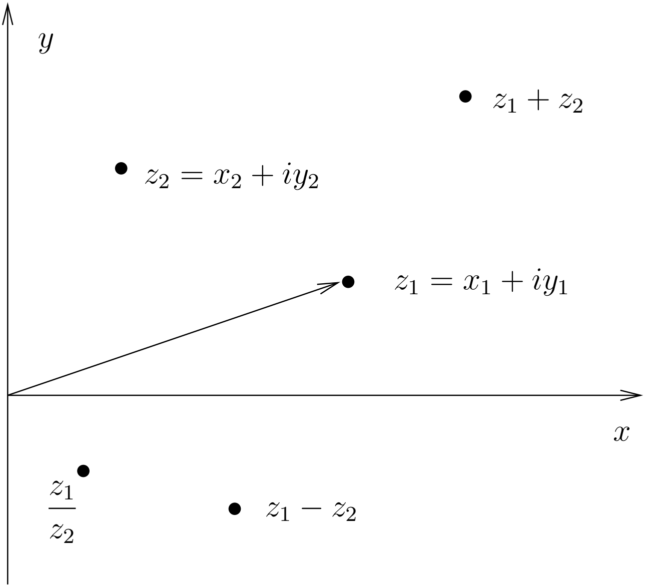

Figure B.2. Composition of two complex numbers using various operations.

**Assignment 2.1** Given $z_1 = 4.5 + i1.5$ and $z_2 = 1.5 + i3$,

(1) Find $z_1 + z_2$ and $z_1 - z_2$.
(2) Find $\dfrac{z_1}{z_2}$.
(3) Find $z_1 z_2$.
(4) Repeat the last two problems using the polar form.
(5) Compare with the results plotted in figure B.2.

## Matrices

A matrix is a mathematical mechanism to organise and operate upon $n$-tuples. For example, the complex numbers we have just encountered consist of a real part and an imaginary part which is basically two numbers or a "double". A position vector in three spatial dimensions would consist of three numbers or a "triple". Along these lines we could represent $n$ distinct numbers or an $n$-tuple. Mathematically, if an entity can be represented as a matrix, it is said to have a representation.

You have encountered many entities made up of $n$-tuples just like the examples given above. We will now look at how matrices organise and operate upon this data. A triple as we encounter in vector algebra can be organised in two possible ways in the matrix world, either a column matrix(column vector) or a row matrix(row vector). This is shown in the following equation

$$\textbf{(B.6)} \qquad \text{Row Matrix: } (x\ y\ z), \quad \text{Column Matrix: } \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

The row matrix is said to be the transpose of the column matrix and vice-versa. This is indicated as follows

$$\textbf{(B.7)} \qquad (x\ y\ z) = \begin{pmatrix} x \\ y \\ z \end{pmatrix}^T, \quad \text{and } \begin{pmatrix} x \\ y \\ z \end{pmatrix} = (x\ y\ z)^T$$

The superscript, $T$, is the transpose operator and its action is to generate the transpose of a matrix. If we have two vectors $\vec{a}$ and $\vec{b}$ represented by matrices as follows $(a_1, a_2, a_3)$ and $(b_1, b_2, b_3)$

our conventional operation of the dot product of two vectors defines the matrix product between a column matrix and a row matrix in that order as

**(B.8)**

$$a_1b_1 + a_2b_2 + a_3b_3 = (a_1\ a_2\ a_3)\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = (b_1\ b_2\ b_3)\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}.$$

Something that we would write as $\vec{a}\cdot\vec{b}$ we would write in matrix form as $\vec{a}\vec{b}^T$. Bear in mind that I have defined the vectors as row vectors. If I had defined them as column vectors then we would have $\vec{a}^T\vec{b}$. Simply put, we take the element of the row and the corresponding element of the column and multiply them and accumulate the products to get the product of a row matrix and a column matrix.

We can build other matrices from our row matrices (all of the same size of course) by stacking them up. If you stack up as many row matrices as you have elements in the row we get a square matrix. When we use the term matrix in this book, we will be referring to a square matrix. We will use two subscripts to index the matrix. For example $a_{ij}$. The first subscript, $i$, indicates which row vector we are referring, the second one, $j$, picks the element out of that row vector.

**(B.9)**

$$
\begin{array}{c|ccccccccc|}
 & 1 & 2 & 3 & \cdots & j-1 & j & j+1 & \cdots & n \\
\hline
1 & a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,j-1} & a_{1,j} & a_{1,j+1} & \cdots & a_{1,n} \\
2 & a_{2,1} & a_{2,2} & \cdots & \cdots & \cdots & a_{2,j} & \cdots & \cdots & a_{2,n} \\
3 & a_{3,1} & \vdots & a_{3,3} & \cdots & \cdots & a_{3,j} & \cdots & \cdots & a_{3,n} \\
\vdots & \vdots & \ldots & \ldots & \ldots & \ldots & \vdots & \ldots & \ldots & \vdots \\
i-1 & a_{i-1,1} & \cdots & \cdots & \cdots & \cdots & a_{i-1,j} & \cdots & \cdots & a_{i-1,n} \\
i & a_{i,1} & a_{i,2} & a_{i,3} & \cdots & a_{i,j-1} & a_{i,j} & a_{i,j+1} & \cdots & a_{i,n} \\
i+1 & a_{i+1,1} & \cdots & \cdots & \cdots & \cdots & a_{i+1,j} & \cdots & \cdots & a_{i+1,n} \\
\vdots & \vdots & \ldots & \ldots & \ldots & \ldots & \vdots & \ldots & \ldots & \vdots \\
n-1 & a_{n-1,1} & a_{n-1,2} & \cdots & \cdots & \cdots & a_{n-1,j} & \cdots & \cdots & a_{n-1,n} \\
n & a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,j-1} & a_{n,j} & a_{n,j+1} & \cdots & a_{n,n}
\end{array}
$$

If the row vectors that make up the matrix are independent of each other we have a matrix that is not singular. If we are given a vector $\vec{x}$ we can project it onto these independent vectors to get a kind of a projection $\vec{b}$ onto the space spanned by these vectors. If we labelled this matrix as $\boldsymbol{A}$ then we could write

**(B.10)**
$$\boldsymbol{A}\vec{x} = \vec{b}$$

For example, given the matrices $\boldsymbol{C}$ and $\vec{x}$ as given below in equation (B.11) how would we find $\vec{y} = \boldsymbol{C}\vec{x}$?

**(B.11)**
$$\boldsymbol{C} = \begin{pmatrix} 5 & 2 & 7 \\ 0 & 11 & 1 \\ 3 & 8 & 10 \end{pmatrix}, \quad \text{and } \vec{x} = \begin{pmatrix} 4 \\ 6 \\ 9 \end{pmatrix}$$

We just go through the motions.

$$y_1 = 5 \times 4 + 2 \times 6 + 7 \times 9 = 63 \tag{B.12}$$

$$y_2 = 0 \times 4 + 11 \times 6 + 1 \times 9 = 75 \tag{B.13}$$

$$y_1 = 3 \times 4 + 8 \times 6 + 10 \times 9 = 150 \tag{B.14}$$

Inspection of the equations (B.12) shows us that they could also be rewritten as

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 3 \end{pmatrix} \times 4 + \begin{pmatrix} 2 \\ 11 \\ 8 \end{pmatrix} \times 6 + \begin{pmatrix} 7 \\ 1 \\ 10 \end{pmatrix} \times 9. \tag{B.15}$$

That is, $\vec{y}$ can also be thought of as a linear combination of the vectors formed by the columns of the matrix. A given square matrix can be viewed as a set of vectors stacked on top of each other and would span a space called the row space of that matrix or a set of column vectors placed side-by-side and would then span a space called the column space of that matrix. Now that we see that the square matrix can also be treated as a composition of column vectors we can write, analogous to our previous operation, $\vec{z} = \vec{x}^T \boldsymbol{C}$. You can verify that

$$\begin{pmatrix} 4 & 6 & 9 \end{pmatrix} \begin{pmatrix} 5 & 2 & 7 \\ 0 & 11 & 1 \\ 3 & 8 & 10 \end{pmatrix} = \begin{pmatrix} 47 & 146 & 124 \end{pmatrix} \tag{B.16}$$

We have so far done the transpose operations to vectors. We can also perform this operation on any general matrix. To take the transpose of a square matrix $\boldsymbol{A}$ with entries $a_{ij}$ we just swap the elements $a_{ij}$ with the element $a_{ji}$ for all $i < j$.

## Assignment 2.2

(1) Given the matrix

$$\boldsymbol{A} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \tag{B.17}$$

Find $\vec{y}$ using equation the $\vec{y} = \boldsymbol{A}\vec{x}$ for various values $\vec{x}$ lying on a unit circle centred at the origin. The easiest way to pick various values of $\vec{x}$ is to take its components to be $\left(\begin{smallmatrix} \cos\theta \\ \sin\theta \end{smallmatrix}\right)$ for various values of $\theta$. Graph $\vec{x}$ and the corresponding $\vec{y}$. What can you say about the relationship of $\vec{x}$ to $\vec{y}$ ? Do the same for $\vec{z} = \vec{x}^T \boldsymbol{A}$ ?

(2) Repeat problem one with points taken from a circle of radius two instead a unit circle.

(3) Given the matrix

$$\boldsymbol{B} = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix} \tag{B.18}$$

Repeat the exercise in the first problem. What is the difference between the two matrices?

(4) Finally repeat these exercises for the matrices

$$\boldsymbol{C} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \tag{B.19}$$

You should have seen from the assignment that for a unit vector $\vec{x}$ the effect of $\boldsymbol{A}$ is to rotate the vector and to scale it to obtain the vector $\vec{y}$. This is true even in the case of $\vec{z}$. We will discuss this assignment. We start with the last problem first.

The matrix $\boldsymbol{C}$ is a bit strange. It rotated every vector enough to get it pointed in the same direction and scaled it in that direction. Effectively all the points on the plane collapse into a line. Taking two independent vectors $\vec{x}$ no longer gives you two independent $\vec{y}$s. The matrix is said to be singular. This happens as the two vectors that are stacked to form the matrix are linearly dependent on each other. In fact, in this case they were chosen so that one has components which are twice the other. You can confirm this. The easiest way to find out if two vectors are parallel to each other is to take the cross product. For a $2\times2$ matrix this is called the determinant of the matrix. In general for a matrix $\boldsymbol{A} = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ the determinant can be written as

**(B.20)**
$$\det \boldsymbol{A} = \det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

If the determinant of a matrix is zero, it is singular and vice-versa. In this book we will use determinants of a three by three matrix which we will now look at. You will need to know how to find determinants of larger matrices and can followup on references [**Str06**],[**Kum00**],[**BW92**],[**Hil88**]. If we have three independent vectors we could find the volume of the parallelepiped that have those three vectors as edges. The volume of the parallelepiped is the determinant of the matrix made up of those vectors. So, we have

**(B.21)**

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

Now, let us get back to the fourth problem of the assignment. The matrix $\boldsymbol{D}$ did stretch every vector. It did not rotate any vector. Its action is said to be spherical in nature. The matrix is variously referred to as an isotropic matrix or a spherical matrix. We have employed the last problem to define the determinants and spherical matrices. Let's see what we can get from the first two problems of the assignment.

In the first problem, before we look at the action of the matrix, let us make an observation about the structure of the matrix. You will notice that $\boldsymbol{A} = \boldsymbol{A}^T$. Such a matrix is said to be a **symmetric** matrix. Just for completeness, a matrix which is identical to the negative of the transpose, that is $\boldsymbol{A} = -\boldsymbol{A}^T$, is called a **skew-symmetric** matrix.

Now let us look at the action of the matrix on various vectors. What was the effect or action of $\boldsymbol{A}$ on the unit vector at an angle of 45 degree from the $x$-axis? It went through a pure stretch and no rotation. The effect of multiplying by a matrix in that direction turned out to be the same as multiplying by a scalar. To check this we need to look at the second problem. We see that the magnitude in this direction again doubled. Along the $\vec{x}_{(45^\circ)}$ matrix multiplication is like scalar multiplication. That is

**(B.22)**
$$\boldsymbol{A}\vec{x}_{(45)^\circ} = 2\vec{x}_{(45)^\circ}, \quad \vec{x}_{(45)^\circ} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

Is there another direction like this where the action of the matrix $\boldsymbol{A}$ is a pure "stretch". If you haven't found it check out the vector along the line through the origin at angle of 135 degrees from the $x$-axis. The direction is called a characteristic direction of $\boldsymbol{A}$ or an eigenvector of $\boldsymbol{A}$. Corresponding to the direction is the amount of stretch, did you check the stretch for the 135 degree direction? The amount of the stretch is called a characteristic value or an eigenvalue. The two matrices $\boldsymbol{A}$, $\boldsymbol{B}$ given above have two eigenvectors and each has a corresponding eigenvalue. The student should recollect that we have encountered the idea of a characteristic direction before. In that case the partial differential equation reduced to an ordinary differential equation along the characteristic direction.

In general the equation (B.22) gives us the clue to finding these directions. We rewrite it since we do not know the characteristic direction and would like to find out what the characteristics are. So, in general we want to find a number $\lambda_i$ and a corresponding direction $\vec{x}_i$ as

**(B.23)**
$$\boldsymbol{A}\vec{x}_i = \lambda_i \vec{x}_i$$

The subscript $i$ is used to remind ourselves here of the correspondence between the stretch $\lambda_i$ and the eigenvector $\vec{x}_i$. We can rearrange this equation as

**(B.24)**
$$(\boldsymbol{A} - I\lambda_i)\vec{x}_i = \vec{0}$$

Now, one obvious $\vec{x}$ that satisfies this equation is the zero vector. This is of no use to us. We could hope for a non-zero $\vec{x}$ if the matrix multiplying it were singular. The matrix is singular when

**(B.25)** $$|\boldsymbol{A} - I\lambda| = 0$$

This gives us the so called characteristic polynomial. We will work it out for the matrix from the first problem in the assignment B-2. To find the eigenvalues and the corresponding eigenvectors of the matrix $\boldsymbol{A} = \left(\begin{smallmatrix} 3 & -1 \\ -1 & 3 \end{smallmatrix}\right)$, solve the following equation for $\lambda$.

**(B.26)** $$\begin{vmatrix} 3-\lambda & -1 \\ -1 & 3-\lambda \end{vmatrix} = 0$$

This gives us the characteristic polynomial equation in $\lambda$

**(B.27)** $$\lambda^2 - 6\lambda + 8 = 0$$

Fortunately the left hand side of equation (B.27) can be factored as $(\lambda - 2)(\lambda - 4)$ which gives us two eigenvalues

**(B.28)** $$\lambda_1 = 2, \quad \text{or/and} \quad \lambda_2 = 4$$

Now we need to find the corresponding eigenvectors $\vec{x}_1$ and $\vec{x}_2$. Let us find $\vec{x}_1$ using $\lambda_1$. We can do this from equation (B.24). If we were to substitute $\lambda_1 = 2$, we would make the matrix singular. As a consequence we cannot get both the components of $\vec{x}_1$ independently. We rewrite equation (B.24) here for convenience after the substitution

**(B.29)** $$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} x_{1,1} \\ x_{2,1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

where $x_{1,1}$ is the first component of the first eigenvector and $x_{2,1}$ is the second component of the first eigenvector. Solving any one of the equations gives $x_{1,1} = x_{2,1}$ (normally I solve both equations, I check my algebra by solving the second equation and make sure that I get the same answer as from the first). In this case, if $x_{1,1} = a$, then an eigenvector corresponding to $\lambda = 2$ is $(a, a)$. We can set the vector $\vec{x}_1$ to the unit vector in that direction as

**(B.30)**
$$\vec{x}_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$

---

### Assignment 2.3

It is advisable to do these calculations employing a calculator instead of programming them or using a package to find the eigenvalues and eigenvectors. Pay attention to the relationship between the matrices in the two problems and the way they relate to the eigenvalues and eigenvectors.

(1) Find the eigenvalues and eigenvectors for the matrices given in the previous assignment B-2.

(2) Repeat problem one for the transpose of the matrices from the previous assignment.

---

Now, for each of the matrices you have the matrix $\boldsymbol{X}$ whose entries $x_{i,j}$ provide the $i^{th}$ component of the $j^{th}$ eigenvector corresponding to the eigenvalue $\lambda_j$. Notice that the eigenvectors are being stacked as columns, side-by-side. From the definition given in equation (B.23) we can write

**(B.31)**
$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix[1] having $\lambda_j$ on the diagonal. Make sure you understand the right hand side of the equation; $\boldsymbol{X}\boldsymbol{\Lambda} \neq \boldsymbol{\Lambda}\boldsymbol{X}$. If we pre-multiply equation (B.31) by $\boldsymbol{X}^{-1}$, we get

$$\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{\Lambda} \tag{B.32}$$

We have a scheme to diagonalise the matrix $\boldsymbol{A}$. It also interesting to see what happens when we post-multiply equation (B.32) by $\boldsymbol{X}^{-1}$. We get

$$\boldsymbol{X}^{-1}\boldsymbol{A} = \boldsymbol{\Lambda}\boldsymbol{X}^{-1} \tag{B.33}$$

This equation looks something like equation (B.31), but not quite. In general if we have a matrix $\boldsymbol{R}_e$ whose columns are made of vectors $\vec{r}_e$ such that

$$\boldsymbol{A}\boldsymbol{R}_e = \boldsymbol{R}_e\boldsymbol{\Lambda}. \tag{B.34}$$

The vectors $\vec{r}_e$ are called the right eigenvectors of $\boldsymbol{A}$. In a similar fashion if the the matrix $\boldsymbol{L}_e$ is a matrix whose rows are made up of vectors $\vec{l}_e$ such that

$$\boldsymbol{L}_e\boldsymbol{A} = \boldsymbol{\Lambda}\boldsymbol{L}_e. \tag{B.35}$$

the vectors $\vec{l}_e$ are called left eigenvectors. Now we can write using equations (B.34) and (B.35)

$$\boldsymbol{A} = \boldsymbol{R}_e\boldsymbol{\Lambda}\boldsymbol{R}_e^{-1} = \boldsymbol{L}_e^{-1}\boldsymbol{\Lambda}\boldsymbol{L}_e \tag{B.36}$$

If we were to calculate $\boldsymbol{R}_e$ and $\boldsymbol{L}_e$ they should relate as $\boldsymbol{R}_e = c\boldsymbol{L}_e^{-1}$, where $c$ is a constant. In fact we could set

$$\boldsymbol{L}_e = \boldsymbol{R}_e^{-1} \tag{B.37}$$

[1] **Note**: It may not always be possible to get a diagonal form for $\boldsymbol{A}$.

**Assignment 2.4**

(1) For the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ given in the earlier assignment find the left and right eigenvectors.

Is there a way for us to get an estimate of the magnitude of the eigenvalues? Yes there is. There is an extremely useful result called the Gershgorin's circle theorem. If we have a matrix $\boldsymbol{A}$ and we partition the matrix into two as follows

$$\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{F}, \tag{B.38}$$

where $\boldsymbol{D}$ is a diagonal matrix made up of the diagonal of $\boldsymbol{A}$. Consequently $\boldsymbol{F}$ has a zero diagonal and the off-diagonal entries of $\boldsymbol{A}$. If $d_i$ is the $i^{\text{th}}$ entry of $\boldsymbol{D}$ and $f_{ij}$ are the entries of $\boldsymbol{F}$ then we define an $R_i$ as

$$R_i = \sum_j |f_{ij}| \tag{B.39}$$

Remember that $f_{ii} = 0$. If $z$ is a complex number then the Gershgorin's circle theorem says that the circular disc $|z - d_i| < R_i$ has an eigenvalue in it if it does not overlap other discs. If a set of $\omega$ such discs overlap, then $\omega$ eigenvalues are contained in the union of those discs. Of course if the $d_i = 0$ then the circle is centred at the origin. To make sure you understand the meaning of this theorem try out the following exercise.

**Assignment 2.5**
Find / draw the circles as indicated by the Gershgorin's theorem for the matrices given below. Find the eigenvalues.

(1) $\begin{pmatrix} 3 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & -3 \end{pmatrix}$

(2) $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & a & 0 \end{pmatrix}, a = 1, 2, 3$

Matrices are a popular mode of representation for mathematical entities. This is so extensive that a mathematical entity is said to have a representation if it has a matrix representation. You will see in chapter 5 that tensors can be represented by matrices. As an example for now, consider the matrices

**(B.40)** $$\mathbf{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{i} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

What do these matrices represent? Evaluate the product $\boldsymbol{i} * \boldsymbol{i}$ and find out. Did you get it to be $-\mathbf{1}$. So, what does $5\mathbf{1} + 2\boldsymbol{i}$ represent? Check out the appendix B. A note of caution here. A mathematical entity may have a representation. That does not mean that every matrix is a representation. Specifically, a complex number can be written in terms of $\mathbf{1}$ and $\boldsymbol{i}$. Every matrix does not represent a complex number. $\left(\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right)$ is an obvious example. In a similar fashion, every tensor may have a matrix representation. Every matrix does not represent a tensor.

Since we have so many other applications of matrices, we will look at some useful decompositions of matrices. Any square matrix $\boldsymbol{A}$ can be written as the sum of

(1) a symmetric matrix and a skew symmetric matrix,

**(B.41)** $$\boldsymbol{A} = \frac{1}{2}\left(\boldsymbol{A} + \boldsymbol{A}^T\right) + \frac{1}{2}\left(\boldsymbol{A} - \boldsymbol{A}^T\right)$$

(2) a spherical and a deviatoric matrix,

**(B.42)** $$\boldsymbol{A} = \frac{1}{3}\mathsf{tr}\left(\boldsymbol{A}\right)\boldsymbol{I} + \left\{\boldsymbol{A} - \frac{1}{3}\mathsf{tr}\left(\boldsymbol{A}\right)\boldsymbol{I}\right\}$$

---

**Assignment 2.6**

Perform the decomposition on a few of the matrices given the previous assignments.

---

## Fourier Series

There are numerous resources for Fourier series – an indication of the importance of the topic. I am just citing a few here: [**Act90**], [**Lat04**], [**Bha03**],[**Tol76**], and [**Stu66**]. If you are encountering Fourier series for the first time it would help if you completed chapter 2. It would also help if you reviewed trigonometric identities and integration rules.

We have seen in chapter 2 that we can use a variety of functions as a basis to represent functions. The box functions and Haar functions had discontinuities in the representation. The hat function and the higher order spline functions resulted in a smoother representation, but lost the important property of orthogonality. We look at using trigonometric functions as a basis from the observation that on the interval $[0, 2\pi]$ the functions $1$, $\sin x$, $\cos x$, $\sin 2x$, $\cos 2x$, ..., $\sin nx$, $\cos nx$, ... are orthogonal to each other. Let us take a look at this more closely. We will look at the trigonometric functions now and look at the constant function later.

First, we remind ourselves of the definition of the scalar product of functions defined on the interval $(0, 2\pi)$.

**(B.43)**
$$\langle f, g \rangle = \int_0^{2\pi} f g dx$$

Using this definition,

**(B.44)**
$$\langle \sin x, \sin x \rangle = \int_0^{2\pi} \sin^2 x dx = \int_0^{2\pi} \frac{1 - \cos 2x}{2} dx = \pi$$

So, the norm of $\sin$ on the interval $(0, 2\pi)$ is $\sqrt{\pi}$. Is it obvious that this is the same for $\cos$? Check it out. What about the norm for an arbitrary wave number $n$?

**(B.45)**

$$\langle \sin nx, \sin nx \rangle = \int_0^{2\pi} \sin^2 nx dx = \int_0^{2\pi} \frac{1 - \cos 2nx}{2} dx = \pi$$

Since the norms of all of the functions are the same we have an interesting possibility here. We could redefine our dot product so that the functions have unit norm. We redefine the dot product as

**(B.46)**
$$\boxed{\langle f, g \rangle = \frac{1}{\pi} \int_0^{2\pi} f g dx}$$

I have boxed the definition so that it is clear that this is the one that we will use. Now, check whether $\sin$ and $\cos$ are orthogonal to each other.

**(B.47)**

$$\langle \sin x, \cos x \rangle = \frac{1}{\pi} \int_0^{2\pi} \sin x \cos x dx = \frac{1}{\pi} \int_0^{2\pi} \frac{\sin 2x}{2} dx = 0$$

You can verify that the sets $\{\sin nx\}$ and $\{\cos nx\}$ are orthonormal both within the set and between the sets.

What happens to the constant function which we have tentatively indicated as $1$?

**(B.48)**
$$\langle 1, 1 \rangle = \frac{1}{\pi} \int_0^{2\pi} dx = 2$$

To normalise the constant function under the new dot product, we define $C_0(x) = 1/\sqrt{2}$. So, the full orthonormal set is

**(B.49)**
$$S_0(x) = 0, C_0(x) = \frac{1}{\sqrt{2}}, S_1(x) = \sin x, C_1(x) = \cos x,$$
$$S_2(x) = \sin 2x, C_2(x) = \cos 2x, \cdots ,$$
$$S_n(x) = \sin nx, C_n(x) = \cos nx, \cdots$$

Now, a periodic function $f(x)$ can be represented using the Fourier series by projecting it onto this basis. By taking the dot product, we will find the components along the basis vectors, or the Fourier coefficients as they are called, as follows

**(B.50)**
$$a_n = \frac{1}{\pi}\int_0^{2\pi} f(x)C_n(x)dx$$

**(B.51)**
$$b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)S_n(x)dx$$

We then try to reconstruct the function using the Fourier components. The representation $\tilde{f}$ is given by

**(B.52)**
$$\tilde{f}(x) = \sum_{n=0}^{\infty} a_n C_n(x) + b_n S_n(x)$$

It will be identical to $f$, if $f$ is smooth. $\tilde{f}(x)$ will differ from $f$, when $f$ has a finite number of discontinuities in the function or the derivative. We will just use $f(x)$ instead of $\tilde{f}(x)$ as is usually done and write

$$f(x) = \sum_{n=0}^{\infty} a_n C_n(x) + b_n S_n(x) \tag{B.53}$$

A more convenient form of the Fourier series is the complex Fourier series. Now we can use the **deMoivre's Formula**[2] given by

$$e^{inx} = \cos nx + i \sin nx \tag{B.54}$$

In order to substitute into equation (B.53), we need to obtain expressions for $\cos nx$ and $\sin nx$. As it turns out we can use the fact that $\cos$ is an even function and that $\sin$ is an odd function by replacing $x$ in equation (B.54) by $-x$ to get

$$e^{-inx} = \cos nx - i \sin nx \tag{B.55}$$

From which we can see that

$$\cos nx = \frac{e^{inx} + e^{-inx}}{2} \tag{B.56}$$

$$\sin nx = \frac{e^{inx} - e^{-inx}}{2i} \tag{B.57}$$

Substituting back into equation (B.53) we get

$$f(x) = \sum_{n=0}^{\infty} a_n \frac{e^{inx} + e^{-inx}}{2} + b_n \frac{e^{inx} - e^{-inx}}{2i} \tag{B.58}$$

This is the same as

$$f(x) = \sum_{n=0}^{\infty} \frac{a_n - ib_n}{2} e^{inx} + \frac{a_n + ib_n}{2} e^{-inx} \tag{B.59}$$

[2] deMoivre's Formula is a variation of Euler's formula defined in section B

If we were to define

**(B.60)**
$$c_n = \frac{a_n - ib_n}{2}$$

**(B.61)**
$$c_{-n} = \frac{a_n + ib_n}{2} = \bar{c}_n$$

then, the Fourier series expansion can be written as

**(B.62)**
$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}$$

Since $e^{inx}$ is complex, we need to define the dot product appropriately so as to get a real magnitude. The dot product now is redefined as follows

**(B.63)**
$$\langle f, g \rangle = \frac{1}{2\pi} \int_0^{2\pi} f\bar{g}dx$$

where $\bar{g}$ is the complex conjugate of $g$. Note that $\langle f, f \rangle$ is real. If $g$ is real then the dot product degenerates to our original definition (apart from the normalising factor).

---

**Assignment 2.7**

(1) In the case where $f$ and $g$ are complex, what is the relation between $\langle f, g \rangle$ and $\langle g, f \rangle$?
(2) Verify that the set $\{e^{inx}\}$ is orthonormal using the norm given in equation (B.63)
(3) Find the Fourier components of the function $f(x) = x(x - 2\pi)$.

---

We see that

**(B.64)**
$$c_n = \frac{1}{2\pi}\int_0^{2\pi} f e^{-inx} dx$$

We can recover our original $a_n$ and $b_n$ using equations (B.60) as

**(B.65)**
$$a_n = c_n + c_{-n} = c_n + \bar{c}_n$$

**(B.66)**
$$b_n = i(c_n - c_{-n}) = i(c_n - \bar{c}_n)$$

It is clear that given a function $f(x)$ on an interval $[0, 2\pi]$, we can find the corresponding Fourier coefficients.

# Bibliography

[Act90] F. S. Acton, Numerical methods that work, The Mathematical Association of America, 1990.

[Act96] ______, Real computing made real : Preventing errors in scientific and engineering calculations, Princeton University Press, 1996.

[Ahl79] L. V. Ahlfors, Complex analysis, McGraw-Hill Book Company, 1979.

[Ame77] F. W. Ames, Numerical methods for partial differential equations, Academic Press Inc, 1977.

[Ari89] R. Aris, Vectors, Tensors, and the Basic Equations of Fluid Mechanics, Dover Publications, Inc, 1989.

[Arn04] V. I. Arnold, Lectures on partial differential equations, Springer-Verlag, 2004.

[Bha03] R. Bhatia, Fourier series, Hindustan Book Agency, 2003.

[BM80] W. R. Briley and H. McDonald, On the structure and use of linearised block implicit schemes, Journal of Computational Physics **34** (1980), 54–73.

[Boo60] G Boole, Calculus of finite differences, Chelsea Publishing Company, 1860.

[Bra00] R. N. Bracewell, The fourier transform and its applications, McGraw-Hill Book Company, 2000.

[Bri87] W. L. Briggs, A multigrid tutorial, SIAM Press, 1987.

[BW92] T. Banchoff and J. Wermer, Linear algebra through geometry, Springer, 1992.

[CFL67] R. Courant, K. Friedrichs, and H. Lewy, On the Partial Difference Equations of Mathematical Physics, IBM Journal (1967), 215:234.

[Cha90] S. C. Chang, A critical analysis of the modified equation technique of Warming and Hyett, Journal of Computational Physics **86** (1990), 107–126.

[Chu77] R. V. Churchill, Complex variables, McGraw-Hill Book Company, 1977.

[CJ04] R. Courant and F. John, Introduction to calculus and analysis, Springer-Verlag, 2004.

[Ded63] R. Dedekind, Essays on the theory of numbers, Dover Publications, Inc., 1963.

[ea99] J. F. Thompson et al, Handbook of grid generation, CRC Press, 1999.

[Fel68] W. Feller, An introduction to probability theory and its applications, vol. I, John Wiley and Sons, 1968.

[FH74] Warming R. F. and B. J. Hyett, The modified equation approach to the stability and accuracy analysis of finite difference methods, Journal of Computational Physics **14** (1974), 159–179.

[Gea71] C. W. Gear, Numerical initial value problem in ordinary differential equations, Prentice-Hall Inc., 1971.

[GF82] I. M. Gelfand and S. V. Fomin, Calculus of variations, McGraw-Hill, 1982.

[GL83] G. H. Golub and C. F. Van Loan, Matrix computations, The Johns Hopkins University Press, 1983.

[Gol91] D. Goldberg, What every computer scientist should know about floating-point arithmetic, ACM Computing Surveys **23** (1991), no. 1, 5–48.

[GU72] Goscinny and Uderzo, Asterix and the soothsayer, Dargaud, 1972.

[Ham73] R. W. Hamming, Numerical methods for scientists and engineers, Dover Publications, 1973.

[Hil88] D. R. Hill, Experiments in computational matrix algebra, Random House, 1988.

[HP03] J. L. Hennessy and D. A. Patterson, Computer architecture—a quantitative approach, Morgan Kaufmann Publishers, 2003.

[HY81] L. A. Hageman and D. M. Young, Applied iterative methods, Academic Press, 1981.

[Jr55] J. Douglas Jr, On the numerical integration $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t}$ by implicit methods, Journal of the Society for Industrial and Applied Mathematics **3** (1955), no. 1, 42–65.

[Knu81] D. E. Knuth, The art of computer programming - seminumerical algorithms, vol. II, Addison-Wesley, 1981.

[KPS97] P. E. Kloeden, E. Platen, and H. Schurz, Numerical solution of sde through computer experiments, Springer, 1997.

[Kre89] E. Kreysig, Introductory functional analysis with applications, Wiley, 1989.

[Kum00] S. Kumaresan, Linear algebra: A geometric approach, Prentice-Hall of India Private Limited, 2000.

[Lat04] B. P. Lathi, Linear systems and signals, Oxford University Press, 2004.

[Lax73] P. D. Lax, Hyperbolic systems of conservation laws and the mathematical theory of shock waves, SIAM, 1973.

[LR57] H. W. Liepmann and A. Roshko, Elements of gasdynamics, John Wiley & Sons, 1957.

[LW60] P. D. Lax and B. Wendroff, Systems of conservation laws, Communications on Pure and Applied Mathematics **13** (1960), no. 2, 217–237.

[Moo66] R. E. Moore, Interval analysis, Prentice-Hall, 1966.

[Moo85] ______, Computational functional analysis, Ellis-Horwood limited, 1985.

[NA63] Sarmin E. N. and Chudov L. A., On the stability of the numerical integration of systems of ordinary differential eqautions ariding in the use of the straight line method, USSR Computtaional Mathematics and Mathematical Physics **3** (1963), no. 6, 1537–1543.

[PJ55] D. W. Peaceman and H. H. Rachford Jr, The numerical solution of parabolic and elliptic differential equations, Journal of the Society for Industrial and Applied Mathematics **3** (1955), no. 1, 28–41.

[Roa00] P. Roache, Validation, verification, certification in cfd, ??, 2000.

[Roe81] P. L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics **43** (1981), 357–372.

[RS81] D. H. Rudy and J. C. Strikwerda, Boundary conditions for subsonic compressible Navier-Stokes calculations, Computers and Fluids **9** (1981), 327–338.

[Sam03] H. Samet, Spatial data structures, Morgan-Kaufmann, 2003.

[Sha53] A. H. Shapiro, The dynamics and thermodynamics of compressible fluid flow, The Ronald Press Company, 1953.

[Smi78] G. D. Smith, Numerical solution of partial differential equations: Finite difference methods, Clarendon Press, 1978.

[Sne64] I. Sneddon, Elements of partial differential equations, McGraw-Hill, 1964.

[SS82] J. L. Synge and A. Schild, Tensor calculus, Dover Publications, Inc, 1982.

[Str86] G. Strang, Introduction to applied mathematics, Wellesley-Cambridge Press, 1986.

[Str06] ______, Linear algebra and its applications, 4th ed., Thompson Brooks/Cole, 2006.

[Stu66] R. D. Stuart, An introduction to Fourier analysis, Chapman and Hall, 1966.

[Tol76] G. P. Tolstov, Fourier series, Dover Publications, 1976.

[Var00] R. S. Varga, Matrix iterative analysis, 2nd ed., Springer-Verlag, 2000.

[Wes04] P. Wesseling, Principles of computational fluid dynamics, Springer, 2004.

[Win67] A. M. Winslow, Numerical solution of the quasi-linear Poisson equation in a non-uniform triangle mesh, Journal of Computational Physics 1 (1967), 149–172.

[You93] E. C. Young, Vector and tensor analysis, Marcel-Dekker Ltd, 1993.

[ZT86] E. C. Zachmanoglou and D. W. Thoe, Introduction to partial differential equations with applications, Dover Publications, 1986.