# Numerical methods

Lecture notes

Prof. Dr. W. Benz

Bern, February 2008

# Contents

# Chapter 1

# Basics

## 1.1 Introduction

A computer is an electronic machine that only does what it is told to do and nothing else. Hence, there is no miracle. No key on the keyboard that one can press to solve a differential equation, to integrate or to find the zeroes of a function. Everything has to be programmed first and fed into the machine. Its only strength is to be able to perform what it has been told extremely fast and reliably. This means that the machine, except in case of a hardware failure, never does any mistakes. Of course, we do not mean to say that computer output is always correct, quite to the contrary. In fact, the immense weakness of computers is to execute instructions without being able to think wether or not they make sense! In this respect, the human being is vastly superior even if she/he cannot add as fast as the machine.
Unfortunately, the computer users (especially beginners) sometimes lose their ability to think critically and blindly trust what is being printed or plotted. Obviously, no bigger mistake can be made and it cannot be repeated enough that using computers *requires* thinking!

## 1.2 Variables, functions and derivatives

Space-time is a continuum in which variables can be defined at arbitrary positions. Hence, all mathematical constructs such as derivatives can also be defined at arbitrary positions in this continuum. Unfortunately, this is not the case when this space-time has to be mapped onto a computer. The latter cannot handle a continuum but only a discrete number of points. Depending upon the information available, the computational power at hand, the desired accuracy, etc. the number of points can/must be relatively large.
The question of defining variables and other mathematical functions on such a discrete

space-time may seem somewhat an academic issue. It is not. In fact, we shall see later that proper centering is often key to stability and/or precision. In this chapter we begin by first defining some basic approaches that will be of use in later chapters.

## 1.3 Centering

The first step in any numerical approach is to transform the space-time continuum into a finite number of points. In the present text, we shall use the standard convention that superscript always refer to time and subscript to space coordinates. In addition, integer values for subscripts or superscripts denoted values taken at zone edges while half-integer values denote cell centered quantities. We shall see later that centering is a critical issue in numerical work.
With these definitions, we write the coordinate along the $x$-axis at time $t^n$ and position $x_k$ by

$$x_k^n = \sum_{l=0}^{k-1} \Delta x_{l+\frac{1}{2}}^n \tag{1.1}$$

where $k = 1, 2, 3, \ldots, K$. K being the total number of cells considered. Note that in this expression with have accounted for the possibility for the spatial step size to vary with time and space which implies time and space indices on $\Delta x$. Similarly, the elapsed time is written

$$t^n = \sum_{l=0}^{n-1} \Delta t^{l+\frac{1}{2}}. \tag{1.2}$$

where $l = 1, 2, 3, \ldots$. Here again we notice the time indices on $\Delta t$ indicating the possibility for the time step to change with time. Finally, by looking at Eq. 1.1 and 1.2, we notice that the coordinates themselves are defined at zone edges (integer indices) while the increments are defined at zone centers (half-integer indices). This is illustrated in Fig. 1.1.
We can also define finite difference approximations to physical variables. These variables will be associated with zone corners, edges, or centers depending on the nature of the variable and/or the scheme used. As an example, let us look at a function g(x,t) that is associated with zone centers. Its value at time $t^{n+1}$ at spatial location $k + \frac{1}{2}$ will be given by

$$g(x, t) = g(x_{k+\frac{1}{2}}^{n+1}, t^{n+1}) = g_{k+\frac{1}{2}}^{n+1} \tag{1.3}$$

This value is represented by the upper most black dot in Fig. 1.1. The same function associated with zone edges at time $t^n$ would be written as

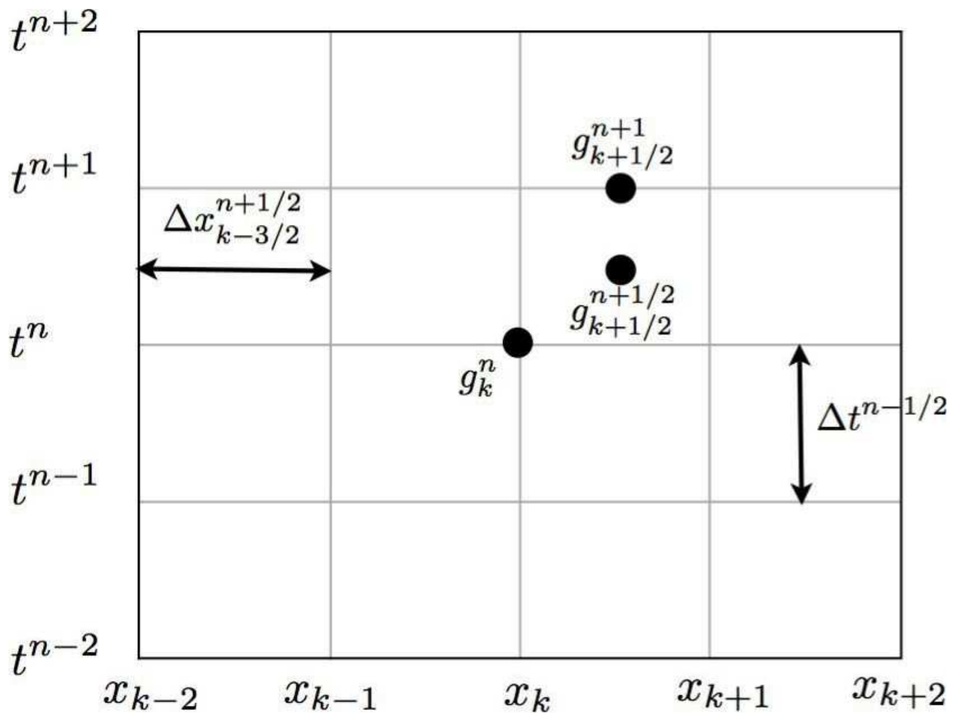


**Figure 1.1:** Space time centering. Note the cell-edge definition of the coordinates and the corresponding cell-center definition of the increments. Values of functions can be defined at cell center or boundary depending upon problem.

$$g(x,t) = g(x_k, t^n) = g_k^n \tag{1.4}$$

This value is also displayed in Fig. 1.1. Other definitions can be imagined especially in multi-dimensional problems. We shall see later that the centering of the variables and derivatives is actually critical to accuracy and stability of a given integration scheme. We can now use these definitions to write derivatives. Let us consider a function (still in one spatial dimension) $f(x,t)$. For a fixed time $t$, we can perform a Taylor expansion of $f$ centered about $x_0$

$$f(x_0 + \Delta x) = f(x_0) + \left(\frac{\partial f}{\partial x}\right)_{x_0} \Delta x + \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_{x_0} \Delta x^2 + O(\Delta x^3) \tag{1.5}$$

where $O(\Delta x^l)$ stands for all terms of order $\Delta x^l$ and higher. Using our notations we replace Eq. 1.5 by the following expression

$$f_{k+1}^n = f_k^n + \left(\frac{\partial f}{\partial x}\right)_k^n \Delta x_{k+1/2}^n + \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_k^n \left(\Delta x_{k+1/2}^n\right)^2 + O(\Delta x^3) \tag{1.6}$$

We can solve this expression for the first order spatial derivative of $f$

$$\left(\frac{\partial f}{\partial x}\right)^n_k = \frac{f^n_{k+1} - f^n_k}{\Delta x^n_{k+1/2}} - \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)^n_k \Delta x^n_{k+1/2} + O(\Delta x^2) \tag{1.7}$$

From this expression, we see that we can write an approximation of the first order derivative of $f$, the so-called *forward difference*, as

$$\left(\frac{\delta f}{\delta x}\right)^n_k = \frac{f^n_{k+1} - f^n_k}{\Delta x^n_{k+1/2}} \tag{1.8}$$

where we have used the symbol $\delta f/\delta x$ to indicate that this is a finite difference analog to the actual derivative of the function $f$. Given that the first term we have neglected to obtain this approximation is of order $\Delta x$, the error term is of order $O(\Delta x)$.
We obtained this expression by Taylor expanding the function about $x_0$. We could also expand it backward to obtain another representation, the so-called *backward difference* of the same derivative

$$\left(\frac{\delta f}{\delta x}\right)^n_k = \frac{f^n_k - f^n_{k-1}}{\Delta x^n_{k-1/2}} \tag{1.9}$$

Note that to obtain this expression we are using the preceding grid point rather than the following one. Of course, this also means that the proper $\Delta x$ has to be used. However, in case of uniform zoning all spatial increments are equal. We not that the error term for this expression is also $O(\Delta x)$. Hence both approximations have comparable errors. We could define yet another way of write still the same derivative by subtracting the forward and the backward Taylor expansions and we obtain

$$\left(\frac{\delta f}{\delta x}\right)^n_k = \frac{f^n_{k+1} - f^n_{k-1}}{2\Delta x^n_k} - \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)^n_k \frac{\left(\Delta x^n_{k+1/2}\right)^2 - \left(\Delta x^n_{k-1/2}\right)^2}{2\Delta x^n_k} + O(\Delta x^2) \tag{1.10}$$

where we have defined $\Delta x^n_k = \frac{1}{2}(\Delta x^n_{k+1/2} + \Delta x^n_{k-1/2})$. Now, in case of uniform zoning ($\Delta x = \Delta x^n_{k+1/2} = \Delta x^n_{k-1/2}$), the quadratic differences between the spatial increments vanishes and we are left with the following expression

$$\left(\frac{\delta f}{\delta x}\right)^n_k = \frac{f^n_{k+1} - f^n_{k-1}}{2\Delta x} \tag{1.11}$$

As indicated by Eq. 1.10, the remaining leading error term in this expression is $O(\Delta x^2)$. Hence, we notice that for small $\Delta x$, the derivative given by Eq. 1.11, the so-called *centered-difference* scheme, is one order in $\Delta x$ more accurate than the other two derivatives we have seen so far (Eqs. 1.9 and 1.8).

We can generalize the method to obtain derivatives by writing the derivatives as a weighted combination of function values. For example, a general expression for a scheme involving the value of the function at three points (a so-called "three-level scheme") can be obtained by setting

$$\left(\frac{\delta f}{\delta x}\right)_k^n = a f_{k-1}^n + b f_k + c f_{k+1}^n \tag{1.12}$$

with $a, b, c$ constants to be determined. We can determine these constants by expanding the value of the function at $n, k$. We obtain the equation

$$\begin{aligned} a f_{k-1}^n + b f_k + c f_{k+1}^n = (a+b+c) f_k^n + (c-a)\Delta x \left(\frac{\partial f}{\partial x}\right)_k^n + \\ \frac{1}{2}(a+c)\Delta x^2 \left(\frac{\partial^2 f}{\partial x^2}\right)_k^n + \frac{1}{6}(c-a)\Delta x^3 \left(\frac{\partial^3 f}{\partial x^3}\right)_k^n + ... \end{aligned} \tag{1.13}$$

For this combination to yield the first order derivative, we set the following conditions: $a+b+c=0$ and $(c-a)\Delta x = 1$. This implies the following relations:

$$a = c - \frac{1}{\Delta x} \qquad b = -2c + \frac{1}{\Delta x} \tag{1.14}$$

Choosing $c = -a$ has the additional virtue to cancel the third term in Eq. 1.13 (second order derivative) leading to the most accurate scheme possible involving only three points. It is apparent that this corresponds to the so-called centered difference approximation already obtained in Eq. 1.11.
From the few examples above, we notice that centered finite-difference approximations for derivatives are generally more accurate than non-centered approximations. Of course, for non-uniform zoning centered schemes involving more than two points are impossible to construct. The idea is therefore to construct centered schemes using only two mesh points. This naturally brings the concept of staggered mesh.
To illustrate this point, let us consider the case of a mesh where the function itself is defined on cell edges but the derivative is cell centered. To obtain a suitable expression for the derivative, we proceed again by Taylor expanding $f_{k+1}^n$ and $f_k^n$ but this time centered at $k+1/2$

$$f_{k+1}^n = f_{k+1/2}^n + \left(\frac{\partial f}{\partial x}\right)_{k+1/2}^n \frac{\Delta x_{k+1/2}}{2} + \left(\frac{\partial^2 f}{\partial x^2}\right)_{k+1/2}^n \frac{\Delta x_{k+1/2}^2}{4} + O(\Delta x^3) \tag{1.15}$$

and

$$f_k^n = f_{k+1/2}^n - \left(\frac{\partial f}{\partial x}\right)_{k+1/2}^n \frac{\Delta x_{k+1/2}}{2} + \left(\frac{\partial^2 f}{\partial x^2}\right)_{k+1/2}^n \frac{\Delta x_{k+1/2}^2}{4} + O(\Delta x^3) \tag{1.16}$$

We can now obtain a suitable expression for the first order derivative of $f$ by subtracting the second expression from the first we obtain

$$\left(\frac{\delta f}{\delta x}\right)^n_{k+1/2} = \frac{f^n_{k+1} - f^n_k}{\Delta x_{k+1/2}} \tag{1.17}$$

Since only the points at the beginning and end of the zone defined by $k, k+1$ enter in this expression, this error affecting this expression are clearly $O(\Delta x^2)$ even in the case of non-uniform zoning. However, we remind the reader that the derivative calculated in this fashion is defined at cell centers and not edges. Of course, we now realize how important the centering of variables, functions and derivatives are to ensure future accuracy of the numerical results.
Finite-difference approximations to second derivatives can be derived in a similar manner. Using a three point symmetric expression, we obtain the following expression in the case of uniform zoning

$$\left(\frac{\delta^2 f}{\delta x^2}\right)^n_{k} = \frac{f^n_{k+1} + f^n_{k-1} - 2f^n_k}{\Delta x^2} \tag{1.18}$$

A simple expansion of this expression reveals that the leading error term is $O(\Delta x^2)$.

## 1.4 Accuracy

As we have already seen in the section above, centering has a great deal to do with accuracy of the method. However, we shall see later that there is more to it since centering also directly determines the stability of a given scheme.
In general accuracy of a scheme can be determined to a good approximation by Taylor expanding all terms at one point in space time and evaluating the leading error term at this point.
As a concrete example, let us consider the simple exponential function $f(x) = e^x$. The derivative is given by $df/dx = e^x$. We shall now compare various estimates of the derivative at $x = 1$ with the exact value. The results from this comparison are given in Tab. 1.1. Note that since the function is not depending on $t$, the superscript $n$ has been omitted.
From looking at Tab. 1.1, it is evident that in the case studied the leading error term is actually a good estimate of the actual error. Higher order schemes (schemes involving more points) and/or symmetric schemes are also, as one intuitively expects, more accurate. However, we shall see shortly that these two comments are only true as long as one is dealing with a smooth function. In the case of discontinuities or sharp features the higher order terms in the Taylor expansion start to matter and the one can no longer consider that the first term dropped from the expansion is the leading term.

**Table 1.1:** Accuracy of different finite-difference schemes for solving the first order differential equation $df/dx = e^x$ at $x = 1$ using $\Delta x = 0.1$. Note that $f_{xx}$ and $f_{xxx}$ stand for the second and third derivative of $f$ with respect to $x$ respectively.

| Case | $\left(\frac{df}{dx}\right)_k$ | Actual error | Truncation error |
|---|---|---|---|
| Exact | 2.7183 | - | - |
| 3pt symmetric<br>$(f_{k+1} - f_{k-1})/2\Delta x$ | 2.7228 | $4.533 \times 10^{-3}$ | $4.531 \times 10^{-3}$<br>$\Delta x^2 f_{xxx}/6$ |
| forward difference<br>$(f_{k+1} - f_k)/\Delta x$ | 2.8588 | $1.406 \times 10^{-1}$ | $1.359 \times 10^{-1}$<br>$\Delta x f_{xx}/2$ |
| backward difference<br>$(f_k - f_{k-1})/\Delta x$ | 2.5868 | $-1.315 \times 10^{-1}$ | $-1.359 \times 10^{-1}$<br>$-\Delta x f_{xx}/2$ |
| 3 pt asymmetric<br>$(-\frac{3}{2}f_k + 2f_{k+1} - \frac{1}{2}f_{k+2})/\Delta x$ | 2.7085 | $-9.773 \times 10^{-3}$ | $-9.061 \times 10^{-3}$<br>$-\Delta x^2 f_{xxx}/3$ |
| 5 pt symmetric<br>$(f_{k-2} - 8f_{k-1} + 8f_{k+1} - f_{k+2})/12\Delta x$ | 2.7183 | $-9.072 \times 10^{-5}$ | $-9.061 \times 10^{-5}$<br>$-\Delta x^4 f_{xxx}/30$ |

Finally, its is quite clear that first order accurate approximations lead to appreciable errors.
Similar remarks are valid for second order derivatives. Using the same example as for the first order derivative above, we obtain the resultsb given in Table 1.2
Again first order schemes lead to large errors that can only be compensated for by taking correspondingly smaller step sizes ($\Delta x$ in our case). From this comparison one could also conclude that higher order schemes are always to be preferred to lower order ones. This is however deceptive for several reasons. From a practical perspective, the accuracy that can be achieved in a given amount of CPU time is usually more important than accuracy alone and higher order schemes are intrinsically more expensive to use. On a coarse grid (low resolution), high order formulae are not necessarily better. They might be on fine grids but require significantly more memory in addition to CPU. Finally, if the solution is discontinuous (a shock would be an example of a discontinuity) higher order solutions are not significantly better.

## 1.5 Discontinuities

To illustrate the problem of getting approximations of derivatives near discontinuities (or even strongly varying functions), we consider the case illustrated in Fig. 1.2
We can now estimate the first order derivative of the function $f(x)$ at $x = 1$ using to schemes of different order and compare the value obtained with the theoretically

**Table 1.2:** Accuracy of different finite-difference schemes for solving the second order differential equation $d^2f/dx^2 = e^x$ at $x = 1$ using $\Delta x = 0.1$

| Case | $\left(\frac{d^2f}{dx^2}\right)_k$ | Actual error | Truncation error |
|---|---|---|---|
| Exact | 2.7183 | - | - |
| 3pt symmetric $(f_{k-1} - 2f_k + f_{k+1})/\Delta x^2$ | 2.7205 | $2.266 \times 10^{-3}$ | $2.265 \times 10^{-3}$ $\Delta x^2 f_{xxxx}/12$ |
| 3pt asymmetric $(f_k - 2f_{k+1} + f_{k+2})/\Delta x^2$ | 3.0067 | $2.884 \times 10^{-1}$ | $2.718 \times 10^{-1}$ $\Delta x f_{xxx}/2$ |
| 5pt symmetric $(-f_{k-2} + 16f_{k-1} - 30f_k + 16f_{k+1} - f_{k+2})/12\Delta x^2$ | 2.7183 | $3.023 \times 10^{-5}$ | $3.020 \times 10^{-5}$ $-\Delta x^4 f_{xxxxxx}/90$ |

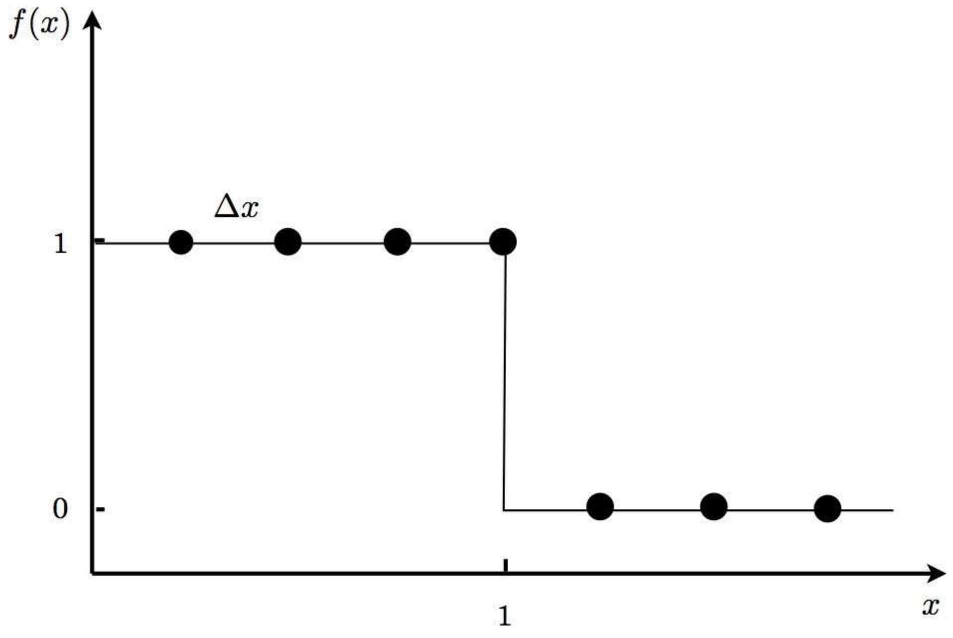


**Figure 1.2:** Accuracy in presence of discontinuities

expected value of $\infty$.

$$\left(\frac{\delta f}{\delta x}\right)_1 = \begin{cases} -\dfrac{0.5}{\Delta x} & \text{3 pt. sym. formula} \\ -\dfrac{7}{12\Delta x} & \text{5 pt. sym. formula} \end{cases} \tag{1.19}$$

Of course, such an example is only illustrative since true discontinuities are seldom in nature. Nevertheless, the point is made that in case of discontinuities, the high order terms in the Taylor expansion can no longer be neglected.
Another example is given by the function $f(x) = tanh(k(x-1))$ which is illustrated in Fig. 1.3. The parameter $k$ allows to control the steepness of the jump occurring at $x = 1$. While a value $k = 1$ gives a very smooth transition, a value of $k = 20$ gives rise to quite a sharp jump.

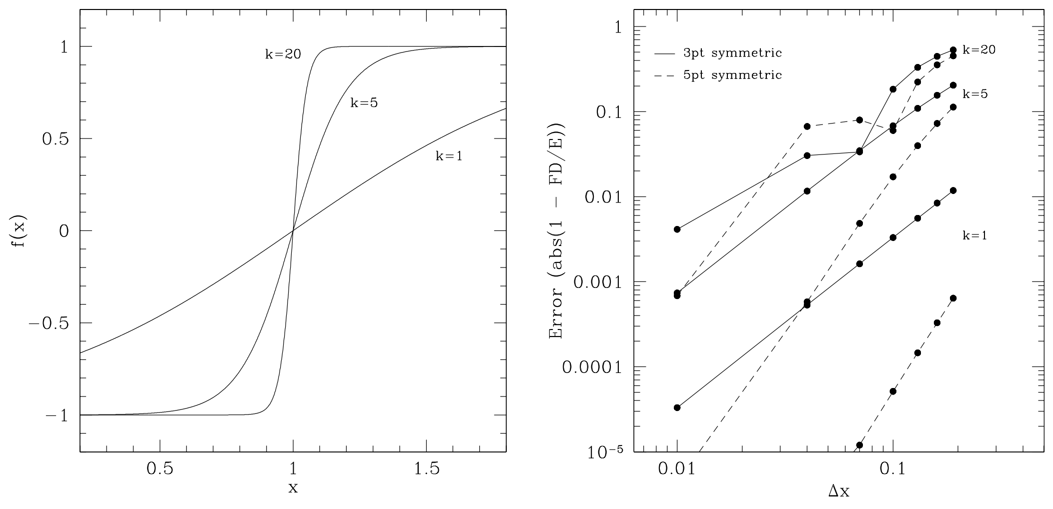


**Figure 1.3:** Accuracy in presence of a discontinuity. The function $f(x) = tanh(k(x-1))$ (left panel) is used to study the accuracy of two approximations (3 and 5 point symmetric) of the first order derivative of $f(x)$ at $x = 1$ as a function of the steepness of the discontinuity. The right panel shows the relative error as a function of resolution.

Several comments can be made by looking at this figure. First, for relatively smooth functions, the error is (as expected) decreasing with increasing resolution. The (almost) linear relationship between error and resolution indicates that both are related by a simple power law. The slope of the relation corresponds to the order of the scheme as given by the leading term neglected from the Taylor expansion. It is also evident that in the presence of a discontinuity, high resolution is needed to obtain accurate results. High order schemes do not improve accuracy if the resolution is too low.

## 1.6 Waves

Besides discontinuities, waves are often found as solutions to partial differential equations. They present problems of their own which we illustrate here briefly. Consider the following simple wave equation

$$f(x,t) = \cos(m(x - qt)) \tag{1.20}$$

where $m$ is the wave number and $q$ the propagation speed in the $x$ direction. Clearly, at a fixed point, the motion is periodic with a period: $P = 2\pi/qm$. We can compute the exact first derivative of this function at node $(n, k)$

$$\left(\frac{\partial f}{\partial x}\right)_k^n = -m\sin(m(x_k - qt^n)) \tag{1.21}$$

and the second order derivative at the same node

$$\left(\frac{\partial^2 f}{\partial x^2}\right)_k^n = -m^2\cos(m(x_k - qt^n)) \tag{1.22}$$

We can now compare these solutions with the algebraic expressions obtained using finite difference schemes. For example, using the 3 points symmetric expression, we obtain

$$\begin{aligned}\left(\frac{\delta f}{\delta x}\right)_k^n &= \frac{f_{k+1}^n - f_{k-1}^n}{2\Delta x} \\ &= \frac{1}{2\Delta x}\left[\cos(m(x_k - qt^n) + m\Delta x) - \cos(m(x_k - qt^n) - m\Delta x)\right] \\ &= \frac{\sin(m(x_k - qt^n))\sin(m\Delta x)}{\Delta x}\end{aligned} \tag{1.23}$$

To evaluate the error, we now take the ratio between the approximate and the exact derivative and obtain

$$\begin{aligned}R_3 &= \frac{\left(\frac{\delta f}{\delta x}\right)_k^n}{\left(\frac{\partial f}{\partial x}\right)_{exact}} \\ &= \frac{\sin(m\Delta x)}{m\Delta x} = \frac{\sin(2\pi\frac{\Delta x}{\lambda})}{2\pi\frac{\Delta x}{\lambda}} \quad (\lambda = \frac{2\pi}{m})\end{aligned} \tag{1.24}$$

This shows that for representing waves, high resolution is required. For example for the derivative to be accurate to 2%, one needs $\lambda > 20\Delta x$. In other words, one needs more than twenty resolution elements per wavelength! Conversely, if we only have $\lambda = 4\Delta x$, we obtain a ratio $R_3 \approx 0.63$!

Using the same approach, we can investigate by how much the situation improves by using a higher order scheme. Using the 5 point symmetric scheme, we obtain for the ratio after some algebra the following expression

$$R_5 = \left(\frac{4}{3} - \frac{1}{3}\cos(m\Delta x)\right)\frac{\sin(m\Delta x)}{m\Delta x} \tag{1.25}$$

For the same value of the resolution, namely $\lambda = 20\Delta x$, we obtain $R_5 = 0.996$ compared to $R_3 = 0.9836$. This is a significant improvement. However, in the low resolution case ($\lambda = 4\Delta x$) we obtain $R_5 = 0.8488$ which is also better but still quite bad!
The bottom line is that wave representation requires high resolution and that higher order schemes on a coarse grid are not obviously better.

## 1.7 References

- Benz, W., in *Numerical Modeling of Non Radial Stellar Pulsation: Problems and Prospects'*, Ed. J.R. Buchler (Dordrecht: Kluwer Academic Publishers, 1989)
- Bowers, R.L., Wilson, J.R., Numerical Modeling in Applied Physics and Astrophysics" (Boston: Jones and Bartlett, 1990)
- Fletcher, C.A.J., Computational Techniques for Fluid Dynamics", (Berlin: Springer-Verlag, 1988)

# Chapter 2

# Root finding in one dimension

## 2.1 Introduction

We consider here one of the most basic problem physicists are generally confronted to: Finding solutions to equations. For simplicity and pedagogical reasons, we limit ourselves here to cases where we only have one independent variable. Hence, we are basically, concerned with finding the roots of a function. Mathematically, we write

$$f(x) = 0 \tag{2.1}$$

and ask for which value(s) of $x$ this equation is satisfied.
Except in linear problem, the root finding always involves some form of iteration. Starting from a guess, the method improves the solution until some predefined convergence criterion is met. For reasonably smooth functions, convergence is almost guaranteed provided the initial guess is reasonably good but there can be cases for which the convergence is very difficult to achieve. Finally, some other pathologies can make the root finding process very difficult. For example, multiple or very close roots can fool a program easily. This points to the importance of the initial guess(es) and consequently to a good prior knowledge of the function itself. As always, a computer cannot replace a good brain, it can just help.
The aim of this chapter is not to provide an exhaustive review of all or even most root finding algorithms. It rather serves the purpose of providing a brief glimpse at the techniques being used for this purpose and thereby wetting the appetite of the reader for a more advanced study of this field. Hence we shall limit our discussion to the two following cases

- the derivatives of the function cannot be obtained
- the derivatives of the function can be obtained

## 2.2 Bisection method

We shall say that a root is bracketed in the interval $(a, b)$ if $f(a)$ and $f(b)$ have different signs, in mathematical notation if $f(a) \cdot f(b) < 0$. If the function is continuous, there must be at least one root in the interval.
Once an interval containing a root has been found, several procedure exist to further improve the determination of the root. They differ essentially by the rate at which they converge towards the solution and hence by the amount of numerical work that needs to be done in order to obtain a reasonable estimate of the solution. Unfortunately, the method that converge the fastest to the solution are also those that are the most likely to go wrong. In this respect, this story could very much end like the story of the rabbit and the tortoise...
The *bisection* method is the slowest converging method but one that cannot fail. In that sense, it makes it a very robust method which often serves as safeguard when the other more sophisticated approaches fail. The idea is very simple. Since we have identified an interval that contains a solution, we evaluate the function value at the midpoint and examine its sign. We replace which ever limit that has the same sign by the midpoint. After each iteration, we will have reduced the interval containing the root by a factor of two. If after $n$ iteration the root is within an interval $\epsilon_n$, then after the next iteration it will be bracketed in an interval

$$\epsilon_{n+1} = \frac{1}{2}\epsilon_n \tag{2.2}$$

We notice that the error on the root is diminishing linearly with the number of steps taken, i.e. the computational effort. Hence given the initial width of the bracketing interval $\epsilon_0$, we can compute the number of iterations $n$ required to achieve a given tolerance in the solution $\epsilon$

$$n = \log_2 \frac{\epsilon_0}{\epsilon} \tag{2.3}$$

Depending on the initial guess for the interval bracketing the root and the accuracy required, the number of iteration steps may become quite large which, if the function is easy to evaluate, is not a real problem. However, if the function is complicated to evaluate requiring significant amount of computer time, this can become a time consuming process.
The good news is that the method will always find a root. If the interval happens to contain more than a root, this approach will find one of them. If the interval contains no root but straddles a discontinuity, it will converge on the discontinuity.
It remains to discuss which practical criterion to use for convergence. For this, it is paramount to remember that computer use a finite number of binary digits to reproduce floating point numbers. Hence, while the true analytical function actually takes the value of zero, the value of the function represented by the computer may never take

this value. Hence, the value of the accuracy must be compatible with the number of digits used by the computer to reproduce floating point numbers.

## 2.3 Newton-Raphson method

We consider here a method that improves the convergence of the root finding method in case not only the value of the function $f(x)$ but also its derivatives $f'(x)$ can be evaluated at arbitrary locations. The most well known method of this type is the so-called *Newton-Raphson* method.
The method can be derived from a suitable Taylor expression

$$f(x+\delta) \approx f(x) + f'(x)\delta + \frac{1}{2}f''(x)\delta^2 + \dots \tag{2.4}$$

For small enough values of $\delta$ and well-behaved functions we can obtain a good approximation of $\delta$ by neglecting the high order terms and write

$$\delta \approx -\frac{f(x)}{f'(x)} \tag{2.5}$$

The Newton-Raphson consists geometrically of extending the tangent line at the current point $x_i$ to find where it crosses the abscissa and to use this new point as the start of the next iteration (see Fig. 2.1).
While the Newton-Raphson method is very powerful, it also has its problems. These can be seen by looking at Eq. 2.5. Indeed, in case the first derivative vanishes (or becomes very small), the correction term can become arbitrarily large. In order for this to happen, the function does not have to be discontinuous or complicated. In fact, it suffice that the function has a local minimum or maximum in the domain for this problem to potentially occur. This is illustrated graphically in Fig. 2.2.
To demonstrate that the Newton-Raphson method is indeed much more powerful than the bisection method (when everything goes well), we can compute its rate of convergence from the relevant Taylor expansions

$$\begin{aligned} f(x+\epsilon) &= f(x) + \epsilon f'(x) + \frac{1}{2}\epsilon^2 f''(x) + \dots \\ f'(x+\epsilon) &= f'(x) + \epsilon f''(x) + \dots \end{aligned} \tag{2.6}$$

From the Newton-Raphson formula, we have

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{2.7}$$

so that we must also have

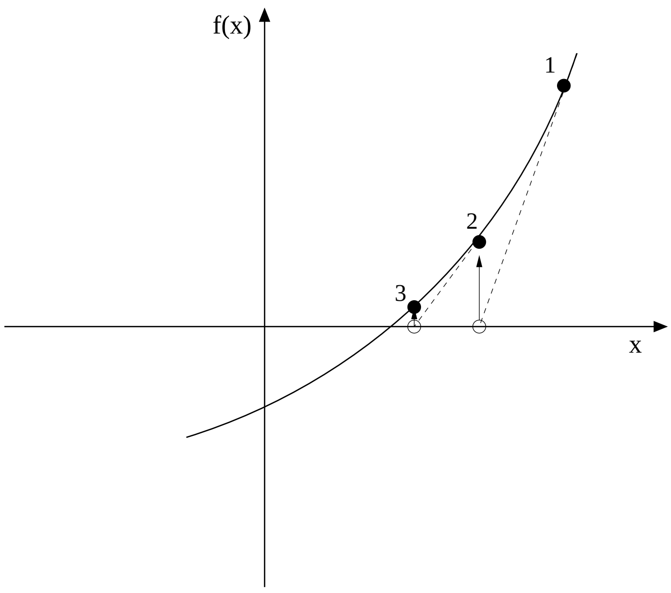


**Figure 2.1:** The Newton-Raphson method extrapolates the local derivative to find the next estimate of the root.

$$\epsilon_{i+1} = \epsilon_i + (x_{i+1} - x_i) = \epsilon_i - \frac{f(x_i)}{f'(x_i)} \tag{2.8}$$

When a trial solution $x_i$ differs by $\epsilon_i$ from the the true root, we can use Eq. 2.6 to estimate the value of $f(x_i)$ and $f'(x_i)$ expressing the derivative at the true root (at this point $f(x) = 0$ by definition of the root). We obtain the following relation

$$\epsilon_{i+1} \approx \epsilon_i^2 \frac{f''(x)}{2f'(x)} \tag{2.9}$$

Hence, Eq. 2.9 tells us that the Newton-Raphson method is converging quadratically as opposed to linearly as in the bisection method (see Eq. 2.2). This convergence property makes the Newton-Raphson method a very efficient method indeed.
One could be tempted to apply the Newton-Raphsom method in all cases even when no analytical expression for the derivative is available. The latter could indeed be calculated using one of the many expressions derived in Chapter 1. However, this is not to be recommended for several reasons. The main one being that the numerical evaluation of the derivatives are not accurate and, depending upon the function, can be quite noisy. This tends to seriously limit the convergence rate and/or introduce serious round-off problems.

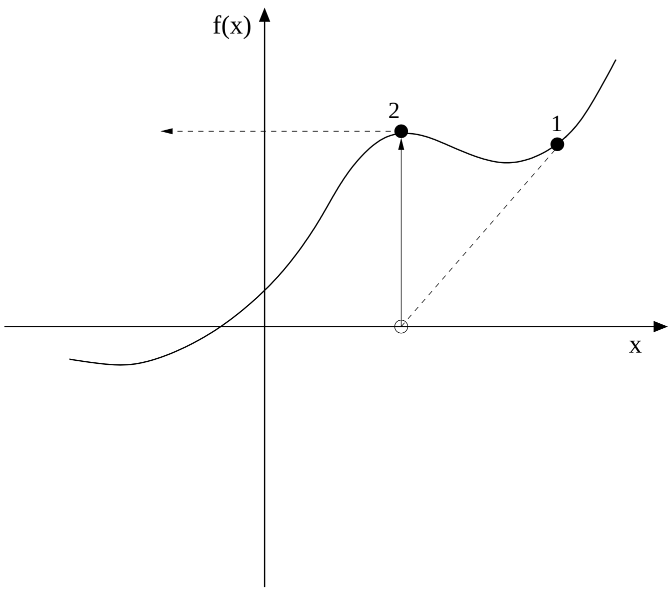


**Figure 2.2:** Unfortunate case where the Newton-Raphson method encounters a local extremum which leads to an ill-defined correction term.

## 2.4 References

- Forsythe, G.E., Malcolm, M.A., & Moler, C.B., Computer methods for mathematical computations (Prentice-Hall: Englewood-Cliffs, N.J., 1977) Section 7
- Press, W.H., Flannery, B.P., Teukolsky, S.A., & Vetterling, W.T., Numerical Recipes: The art of scientific computing (Cambridge University Press: Cambridge, 1988) Section 9

# Chapter 3

# Numerical integration

## 3.1 Introduction

The basic problem of interest in this chapter is to evaluate numerically the integral of a function $f(x)$ over an interval $[a, b]$. In other words, we want to compute

$$I = \int_a^b f(x)dx \tag{3.1}$$

Note that the result is a single number. In fact, the problem at hand is exactly identical to solving for $I = y(b)$ the differential equation

$$\frac{dy}{dx} = f(x) \tag{3.2}$$

with the boundary condition $y(a) = 0$. There are many different methods to compute this integral some good and some less appropriate. In fact, a particular choice of method to apply will essentially depend upon the information available about the function to integrate. We shall distinguish two different broad categories:

- Function values $f(x_i)$ are available only for a *fixed*, finite set of points $x_i$ in the interval $[a, b]$.
- The function $f(x)$ is defined and can be evaluated for any $x$ in the interval $[a, b]$.

Functions in the first category typically result from experimental data measured at some fixed points $x_i$, or they may be obtained from tables where the $x_i$ are regularly spaced. Functions in the second category provide the additional degree of freedom to be able to chose the location at which to evaluate the function. This advantage leads to integrations that, for a fixed number of evaluation points, are generally more accurate. We note in passing that whenever the function can be integrated analytically, the results will always be of equal or better precision than any numerical approach.

Hence, numerical integration is *not* a replacement for analytical integration including with the help of softwares such as *Mathematica* or *Maple*, but rather a solution of last resort when everything else has failed!
The integration, or quadrature, methods presented here are all in a way or another based on adding up the value of the integrand at a number of points within the interval $[a, b]$. The game is to minimize the number of points required while achieving a desired accuracy. Finally, we distinguish between integration formulas that make use of the end point of the interval and those which dont. The first type is called closed while the second open formulas. The advantage of using one or the other depends upon the behavior of the function at the end points. Obviously, in case of a divergence or an ill defined situation, an open formula will have a definite advantage.

## 3.2 Equally spaced abscissas

Let us consider a function $f(x)$ for which the values are given at a number of abscissas denoted $a = x_1, x_2, \ldots, x_N = b$ which are spaced apart by a constant stepsize $h$ and ranging in the interval $[a, b]$. The value of the function at these points is written in our usual notations $f(x_i) = f_i$. Our task is to to integrate this function over $[a, b]$. Depending upon we want/can use the value at the end points, i.e. $f(a)$ and $f(b)$ we may want to use an open or closed formula or a mixture of both.

### 3.2.1 Closed Newton-Cotes formulas

We begin by recalling the *Lagrange* interpolating polynomial. This polynomial is a polynomial of degree $(n-1)$ that passes through the $n$ points given by $(x_1, y_1 = f(x_1)), (x_2, y_2 = f(x_2)), \ldots, (x_n, y_n = f(x_n))$. Mathematically, we can write the polynomials as

$$P_n(x) = \sum_{j=1}^{n} L_j(x) \tag{3.3}$$

with

$$L_j(x) = f_j \prod_{k=1, k\neq j}^{n} \frac{x - x_k}{x_j - x_k} \tag{3.4}$$

For example we can write the case for $n = 2$ explicitly as

$$P_2(x) = f_1 \frac{x - x_2}{x_1 - x_2} + f_2 \frac{x - x_1}{x_2 - x_1} \tag{3.5}$$

Note that this expression is nothing else than a linear interpolation between these two points. For the case $n = 3$, we have

$$P_3(x) = f_1 \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} + f_2 \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} + f_3 \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} \quad (3.6)$$

Suppose now that we want to compute the integral over the first two points at our disposal

$$\int_{x_1}^{x_2} f(x)dx \quad (3.7)$$

We can use a Lagrange interpolating polynomial to approximate the function $f(x)$ over this interval and since we have two points at our disposal, we use $P_2$ and write

$$f(x) \approx P_2(x) = L_1(x) + L_2(x) \quad (3.8)$$

where the explicit expression for $P_2$ is given by (following Eq. 3.5)

$$P_2(x) = f_1 \frac{x-x_2}{x_1-x_2} + f_2 \frac{x-x_1}{x_2-x_1} \quad (3.9)$$

Inserting this approximation in the expression for the integral, we can obtain an expression that we can readily integrate without much difficulty (polynomials are easy to integrate). We therefore obtain an estimate of the value of the integral of $f(x)$ over the first interval

$$\int_{x_1}^{x_2} f(x)dx \approx \int_{x_1}^{x_2} P_2(x)dx = \frac{1}{2}(x_2-x_1)(f_1+f_2) \quad (3.10)$$

To evaluate the integral over the entire domain $[a,b]$, we make use of the composite rule for integrals and write

$$\int_a^b f(x)dx = \int_{x_1}^{x_2} f(x)dx + \int_{x_2}^{x_3} f(x)dx + \ldots + \int_{x_{N-1}}^{x_N} f(x)dx = \sum_{i=1}^{N-1} \int_{x_i}^{x_{i+1}} f(x)dx \quad (3.11)$$

Inserting the estimate of the integral over one interval (Eq. 3.10), we write

$$\int_a^b f(x)dx = \sum_{i=1}^{N-1} \int_{x_i}^{x_{i+1}} f(x)dx \approx \sum_{i=1}^{N-1} \frac{1}{2}(x_{i+1}-x_i)(f_i+f_{i+1}) \quad (3.12)$$

For equally spaced points, we can define $h = (b-a)/(N-1) = (x_{i+1}-x_i)$ and we obtain the so-called *trapezoidal rule*

$$\int_a^b f(x)dx = h\left(\frac{1}{2}f_1 + f_2 + \ldots + f_{N-1} + \frac{1}{2}f_N\right) + O(\frac{(b-a)^3}{N^2}f'') \quad (3.13)$$

Note that the error term was written in terms of $(b-a)$ and $N$ instead of $h$ (the $1/N^2$ rather than $1/N^3$ dependance comes from the fact that we are summing $(N-1)$ error terms).
Higher order formulas can easily be built using the same approach. For example, using three points and using $P_3$, one obtains *Simpson's rule*:

$$\int_{x_1}^{x_3} f(x)dx = \frac{h}{3}\left(f_1 + 4f_2 + f_3\right) + O(h^5 f^{(4)}) \tag{3.14}$$

where $f^{(4)}$ is the fourth derivative of $f$ with respect to $x$. Note that because of the symmetry of the formula, the error term is one order higher than one would have naively expected. Hence Simpson's formula is accurate for polynomials up to order 3.
This formula can be extended by considering a succession of *pairs* of intervals

$$\int_{x_1}^{x_N} f(x)dx = \frac{h}{3}\left(f_1 + 4f_2 + 2f_3 + 4f_4 + \ldots + 2f_{N-2} + 4f_{N-1} + f_N\right) + O(1/N^4) \tag{3.15}$$

Obviously, all sorts of combinations can be tried. For example, we could apply the trapezoidal rule for the first step followed by simpson's rule and finish again with a trapezoidal step:

$$\int_{x_1}^{x_N} f(x)dx = h\left(\frac{1}{2}f_1 + \frac{5}{6}f_2 + \frac{4}{3}f_3 + \frac{2}{3}f_4 + \ldots + \frac{4}{3}f_{N-2} + \frac{5}{6}f_{N-1} + \frac{1}{2}f_N\right) + O(1/N^3) \tag{3.16}$$

Note that while the trapezoidal rule as written in Eq. 3.13 has an error term of order $1/N^2$, the error here is order $1/N^3$ because we used the trapezoidal rule only twice and not $N$ times.
We can also add different combinations together to get new possibilities. For example we can average Eq. 3.16 and Eq. 3.15 to obtain

$$\int_{x_1}^{x_N} f(x)dx = h\left(\frac{5}{12}f_1 + \frac{13}{12}f_2 + f_3 + f_4 + \ldots + f_{N-2} + \frac{13}{12}f_{N-1} + \frac{5}{12}f_N\right) + O(1/N^3) \tag{3.17}$$

Here again and for the same reasons than above, the error is $O(1/N^3)$

### 3.2.2 Open and semi-open formulas

As we already mentioned, open or semi-open formulas can be quite handy in the case when the function to integrate is somehow ill-defined at one or both ends of the interval. To derive such formulas, we proceed in a similar fashion as we have done for the

closed formulas except for the starting and/or end points. As discussed above, we can begin/terminate by using a method one order lower without affecting the overall accuracy of the method.
It is convenient to begin again by considering various formulas to estimate the integral over the single interval $[x_1, x_2]$ since these are the building blocks for the general formulas. Here are a few formulas in increasing order of accuracy which can be obtained relatively straight forwardly with the help of Taylor expansions

$$\int_{x_1}^{x_2} f(x)dx = hf_2 + O(h^2 f') \tag{3.18}$$

$$\int_{x_1}^{x_2} f(x)dx = h\left(\frac{3}{2}f_2 - \frac{1}{2}f_3\right) + O(h^3 f'') \tag{3.19}$$

$$\int_{x_1}^{x_2} f(x)dx = h\left(\frac{23}{12}f_2 - \frac{16}{12}f_3 + \frac{15}{12}f_4\right) + O(h^4 f^{(3)}) \tag{3.20}$$

$$\int_{x_1}^{x_2} f(x)dx = h\left(\frac{55}{24}f_2 - \frac{59}{24}f_3 + \frac{37}{24}f_4 - \frac{9}{24}f_5\right) + O(h^5 f^{(4)}) \tag{3.21}$$

We can now build the complete formulas by summing the relevant formulas above. For example, for an interval open at both end, Eq. 3.18 and the trapezoidal rule as in Eq. 3.13 to obtain

$$\int_{x_1}^{x_N} f(x)dx = h\left(\frac{3}{2}f_2 + f_3 + \ldots + f_{N-2} + \frac{3}{2}f_{N-1}\right) + O(1/N^2) \tag{3.22}$$

Combining Eq. 3.19 together with Eq. 3.17 yields

$$\begin{aligned}\int_{x_1}^{x_N} f(x)dx &= h\left(\frac{23}{12}f_2 + \frac{7}{12}f_3 + f_4 + \ldots\right.\\ &\quad \left. + f_{N-3} + \frac{7}{12}f_{N-2} + \frac{23}{12}f_{N-1}\right) + O(1/N^3)\end{aligned} \tag{3.23}$$

Finally, the highest order formula we shall look at is obtained by combining Eq. 3.20 with the extended Simpson rule, Eq. 3.15

$$\begin{aligned}\int_{x_1}^{x_N} f(x)dx &= h\left(\frac{27}{12}f_2 + 0 + \frac{13}{12}f_3 + \frac{4}{3}f_4 + \frac{2}{3}f_5 + \ldots\right.\\ &\quad \left. + \frac{4}{3}f_{N-4} + \frac{13}{12}f_{N-3} + 0 + \frac{27}{12}f_{N-1}\right) + O(1/N^4)\end{aligned} \tag{3.24}$$

## 3.3 Unequally spaced abscissas

So far we have considered integration formulae in which the points, at which the value of the function was computed, were held fixed. We computed the weights to give to each value of the function at these points to obtain an estimate of the value of the integral. The accuracy of this estimate was found to depend on the number of points used and on the value of these weights.
In this section, we will also consider the abscissa of the points where the function is calculated as well as the weight themselves as free parameters. Hence, we allow us more degrees of freedom and expect therefore a higher degree of accuracy. In fact, we have seen in the previous section that for $n$ points one could integrate exactly a polynomial of degree $n-1$. Using the additional degrees of freedom provided by the free abscissa, we shall be able to integrate accurately polynomials of degree $2n-1$ using $n$ points.
The body of literature on this subject is vast. Generally, the problem is written as

$$\int_a^b w(x)f(x)dx \tag{3.25}$$

where the function $w(x)$ is introduced to capture some discontinuities or other particular behavior of the function in the specified interval. The general theory for computing this integral is based on orthogonal polynomials. Depending upon the function itself or on the integration interval different polynomials are chosen. Table 3.1 gives the most frequently used approaches.

**Table 3.1:** Various forms of Gaussian quadrature

| Interval | $w(x)$ | Orthogonal polynomials |
|---|---|---|
| $[-1,1]$ | $1$ | Legendre polynomials |
| $(-1,1)$ | $\frac{1}{\sqrt{1-x^2}}$ | Cebyshev polynomials (first kind) |
| $[-1,1]$ | $\sqrt{1-x^2}$ | Cebyshev polynomials (second kind) |
| $[0,\infty)$ | $e^{-x}$ | Laguerre polynomials |
| $(-\infty,\infty)$ | $e^{-x^2}$ | Hermite polynomials |

For our purpose, we shall limit ourselves to the case where $w(x)=1$ but show how it is possible to extend the integral interval to arbitrary $[a,b]$. We begin by determining the quadrature rule in a very intuitive way without resorting to the theory of orthogonal polynomials. The rules which allows the determination of an arbitrary order quadrature rule in case of Gauss-Legendre polynomials is given at the end of the section.

### 3.3.1 Gaussian quadrature

We begin with a very simple example, the so-called two-point Gaussian quadrature rule. This is very analogous to the trapezoidal rule derived above. However, since we have now the additional degree of freedom provided by the abscissa of the evaluation points, we can write

$$\int_a^b f(x)dx \approx c_1 f(x_1) + c_2 f(x_2) \tag{3.26}$$

where $c_1, c_2, x_1, x_2$ are the four parameters to be determined. Since we have four parameters, we can by a suitable choice assume that this formula can integrate a polynomial up to order 3 accurately (4 unknowns). Therefore, to compute our 4 unknown parameters, we begin by computing the exact value of the integral of the third order polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

$$\begin{aligned}
\int_a^b f(x)dx &= \int_a^b \left(a_0 + a_1 x + a_2 x^2 + a_3 x^3\right) dx \\
&= \left[a_0 x + a_1 \frac{x^2}{2} + a_2 \frac{x^3}{3} + a_3 \frac{x^4}{4}\right]_a^b \\
&= a_0(b-a) + a_1\left(\frac{b^2-a^2}{2}\right) + a_2\left(\frac{b^3-a^3}{3}\right) + a_3\left(\frac{b^4-a^4}{4}\right) \quad (3.27)
\end{aligned}$$

From Eq. 3.26, we must also have

$$\int_a^b f(x)dx = c_1\left(a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3\right) + c_2\left(a_0 + a_1 x_2 + a_2 x_2^2 + a_3 x_2^3\right) \tag{3.28}$$

Equating Eq. 3.27 to Eq. 3.28 yields the following

$$\begin{aligned}
&a_0(b-a) + a_1\left(\frac{b^2-a^2}{2}\right) + a_2\left(\frac{b^3-a^3}{3}\right) a_3\left(\frac{b^4-a^4}{4}\right) = \\
&\qquad a_0\left(c_1 + c_2\right) + a_1\left(c_1 x_1 + c_2 x_2\right) + a_2\left(c_1 x_1^2 + c_2 x_2^2\right) + a_3\left(c_1 x_1^3 + c_2 x_2^3\right) \quad (3.29)
\end{aligned}$$

By identifying the terms multiplying each of the coefficients, we obtain a set of four equations

$$\begin{aligned} b-a &= c_1+c_2 \\ \frac{b^2-a^2}{2} &= c_1x_1+c_2x_2 \\ \frac{b^3-a^3}{3} &= c_1x_1^2+c_2x_2^2 \\ \frac{b^4-a^4}{4} &= c_1x_1^3+c_2x_2^3 \end{aligned} \tag{3.30}$$

This set of equations can be solved for the four unknown parameter: $c_1, x_1, c_2, x_2$. We obtain

$$\begin{aligned} c_1 &= \left(\frac{b-a}{2}\right) \; x_1 = \left(\frac{b-a}{2}\right)\left(-\frac{1}{\sqrt{3}}\right) + \left(\frac{b+a}{2}\right) \\ c_2 &= \left(\frac{b-a}{2}\right) \; x_2 = \left(\frac{b-a}{2}\right)\left(\frac{1}{\sqrt{3}}\right) + \left(\frac{b+a}{2}\right) \end{aligned} \tag{3.31}$$

and our expression for the approximation of the integral is written

$$\begin{aligned} \int_a^b f(x)dx \;\approx\; & \left(\frac{b-a}{2}\right) f\left(\left(\frac{b-a}{2}\right)\left(-\frac{1}{\sqrt{3}}\right)+\left(\frac{b+a}{2}\right)\right) + \\ & \left(\frac{b-a}{2}\right) f\left(\left(\frac{b-a}{2}\right)\left(\frac{1}{\sqrt{3}}\right)+\left(\frac{b+a}{2}\right)\right) \end{aligned} \tag{3.32}$$

The method can easily be generalized to higher accuracy by using more points. The general formula is easily written as

$$\int_a^b f(x)dx \approx c_1f(x_1) + c_2f(x_2) + \ldots + c_nf(x_n) \tag{3.33}$$

The computation of the coefficient and of the abscissa on the other hand becomes increasingly difficult. In many handbooks (for example Abramowitz & Stegun), one finds tabulated the coefficients related to the following integral

$$\int_{-1}^{1} f(t)dt \approx \sum_{i=1}^{n} c_i f(t_i) \tag{3.34}$$

Table 3.2 provides the weighting factors and function arguments up to $n = 6$.
Note that even though Table 3.2 provides the information relevant for an integration over the domain $[-1, 1]$, it is easily extended over an arbitrary interval $[a, b]$ by a simple change of variable.

**Table 3.2:** Weighting factors $c$ and function argument $t$ used in Gauss quadrature formulae

| Points | Weighting factors | Function arguments |
|---|---|---|
| 2 | $c_1 = 1.000000000$ | $t_1 = -0.577350269$ |
| | $c_2 = 1.000000000$ | $t_2 = 0.577350269$ |
| 3 | $c_1 = 0.555555556$ | $t_1 = -0.774596669$ |
| | $c_2 = 0.888888889$ | $t_2 = 0.000000000$ |
| | $c_3 = 0.555555556$ | $t_3 = 0.774596669$ |
| 4 | $c_1 = 0.347854845$ | $t_1 = -0.861136312$ |
| | $c_2 = 0.652145155$ | $t_2 = -0.339981044$ |
| | $c_3 = 0.652145155$ | $t_3 = 0.339981044$ |
| | $c_4 = 0.347854845$ | $t_4 = 0.861136312$ |
| 5 | $c_1 = 0.236926885$ | $t_1 = -0.906179846$ |
| | $c_2 = 0.478628670$ | $t_2 = -0.538469310$ |
| | $c_3 = 0.568888889$ | $t_3 = 0.000000000$ |
| | $c_4 = 0.478628670$ | $t_4 = 0.538469310$ |
| | $c_5 = 0.236926885$ | $t_5 = 0.906179846$ |
| 6 | $c_1 = 0.171324492$ | $t_1 = -0.932469514$ |
| | $c_2 = 0.360761573$ | $t_2 = -0.661209386$ |
| | $c_3 = 0.467913935$ | $t_3 = -0.171324492$ |
| | $c_4 = 0.467913935$ | $t_4 = 0.171324492$ |
| | $c_5 = 0.360761573$ | $t_5 = 0.661209386$ |
| | $c_6 = 0.171324492$ | $t_6 = 0.932469514$ |

Indeed, let us set $x = mt + c$ and require that for $t = -1$ we have $x = a$ and for $t = 1$ we have $x = b$. We find easily that $m = (b - a)/2$ and $c = (b + a)/2$. Therefore, we can write the integral

$$\int_a^b f(x)dx = \int_{-1}^1 f\left(\frac{b-a}{2}t + \frac{b+a}{2}\right)\frac{b-a}{2}dt \tag{3.35}$$

In fact, there is much more to Gaussian quadrature than what we mentioned here. There exist a body of literature on this subject based on orthogonal polynomials that vastly exceeds the purpose of these notes. However, let us mention for generality that the following integral formula

$$\int_{-1}^1 f(x)dx \approx \sum_{i=1}^n w_i f(x_i) \tag{3.36}$$

is called the Gauss-Legendre quadrature. This is because the weights and the abscissas for evaluating the function are given in terms of Legendre polynomials. We recall here that the recurrence formula for the Legendre polynomials is given by

$$(i+1)P_{i+1} = (2i+1)xP_i - iP_{i-1} \quad \text{with} \quad P_0(x) = 1, P_1(x) = x \tag{3.37}$$

and the corresponding weights are computed from

$$w_i = \frac{2}{(1-x^2)[P_n'(x_i)]^2} \tag{3.38}$$

where $P_n'(x_i)$ is the derivative of the $n$-th Legendre polynomial at $x_i$. The abscissas correspond to the roots of the corresponding Legendre polynomial. In practice, the m-th root of $P_n(x)$ are found starting from the following asymptotic approximation

$$x_i \approx \cos\left(\frac{(4m-1)\pi}{4n+2}\right) \tag{3.39}$$

followed by Newton-Raphson iterations (see section 2.3) to obtain a more accurate estimate of the roots.

## 3.4 References

- Abramowitz, M., & Stegun, I.A., eds., Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables (Dover: New York, 1972) Section 25.4
- Forsythe, G.E., Malcolm, M.A., & Moler, C.B., Computer methods for mathematical computations (Prentice-Hall: Englewood-Cliffs, N.J., 1977) Section 5
- Press, W.H., Flannery, B.P., Teukolsky, S.A., & Vetterling, W.T., Numerical Recipes: The art of scientific computing (Cambridge University Press: Cambridge, 1988) Section 4
- Stoer, J., & Bulirsch, R., Introduction to Numerical Analysis (New York: Springer-Verlag, 1980) Section 3.6

# Chapter 4

# Ordinary differential equations

## 4.1 Introduction

Solving an ordinary differential equation (ODE) can always be reduced to solving sets of first order differential equations. For example, the second order equation for $y(x)$

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x) \tag{4.1}$$

can be reduced by introducing a new variable $z(x)$ to the following two first order equations

$$\begin{aligned} \frac{dy}{dz} &= z(x) \\ \frac{dz}{dx} &= r(x) - q(x)z(x) \end{aligned} \tag{4.2}$$

A maybe more concrete example might be the equation of motion of a small body in the gravitational potential of a body of mass $M$

$$\frac{d^2\vec{r}}{dt^2} = -\frac{GM}{r^3}\frac{\vec{r}}{r} \tag{4.3}$$

This equation can quite evidently be transformed into the following two first order equations

$$\begin{aligned} \frac{d\vec{r}}{dt} &= v(\vec{t}) \\ \frac{d\vec{v}}{dt} &= -\frac{GM}{r^3}\frac{\vec{r}}{r} \end{aligned} \tag{4.4}$$

Hence, the generic problem of solving an ordinary differential equation is reduced to solving a set of $N$ coupled first order differential equations for the functions $y_i$, $i = 1, 2, 3, \ldots, N$ having the general form

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, y_2, \ldots, y_N), \quad i = 1, 2, \ldots, N \tag{4.5}$$

Unfortunately, specifying the set of first order differential equations to solve is not enough to actually solve the system. Indeed, for the system to be solved, boundary conditions have to be specified whose nature can render the solution of the problem quite complicated. Boundary conditions are algebraic conditions on the values of the $y_i$ in Eq. 4.5. In general they can be satisfied at discrete specified points but not necessarily in between. They can be as simple as requiring that certain variables have certain numerical values, or as complicated as a set of non-linear algebraic equations among the variables.
Boundary conditions can be divided into two broad categories

- *Initial value problems*: All the $y_i$ are given at some starting value $x_s$ and the problem consists are finding the value of the $y_i$'s at some final value $x_f$ or at some specified intermediate points. A typical example of such a problem would be the trajectory of a projectile.
- *Boundary value problems*: The boundary conditions are specified at more than one $x$. Typically, some will be defined at $x_s$ and some at $x_f$. Heat transport would be a typical example of such problems.

In general, boundary value problems are more complicated to solve. We shall examine essentially two-classes of ordinary differential equation solver. The so-called multi-step methods and the multi-stage methods. In the former one, information from previous steps is used in order to obtain the value at the new step. Hence, no extra computations are required except a suitable bookkeeping of the previous steps. In the second approach, intermediate stages between the current and the next step are computed in order to advance the solution. Previous steps are not used and once the step has been completed, the intermediate stages are no longer used as well. Hence, significant amount of extra work is required for this approach. However, as we shall see, this approach does not require any special starting procedure which might be of great advantage.

## 4.2 Multi-step methods

Suppose we are given an initial value problem for which we want to compute the solutions on some interval $[0, T]$. Let us define the time steps as $t_n = n\Delta t$ where $n$ is

an integer ($n = 1, 2, 3, \dots$) and the corresponding numerical solutions to the equations by $y^n = y(t_n)$.
To solve such an equation, we can use the various expressions for the derivatives discussed above in order to find a algebraic representation of the differential equation. For example, the most straight forward method of integration of such a problem is the so-called *Euler method* which is based on the forward difference approximation of the first order derivative (Eq. 1.8). In this method, the equation is integrated by

$$y^{n+1} = y^n + \Delta t f(t_n, y^n) \tag{4.6}$$

where we have defined $f = \delta y/\delta x$. This equation advances the solution from $t_n$ to $t_{n+1} = t_n + \Delta t$ where $\Delta t$ is the step size (see Fig. 4.1) provided that the appropriate initial conditions $y(t = 0)$ are given in order to start the iterations. As we already noted in our comments of the forward difference approximation, this equation is not symmetrical in the sense that it uses the derivatives at step $n$ to advance the solution to step $n + 1$. By Taylor expanding $y^{n+1}$ about $y^n$, it is easy to show that the error in the above expression is of order $O(\Delta t^2)$, hence the method is only first order accurate.

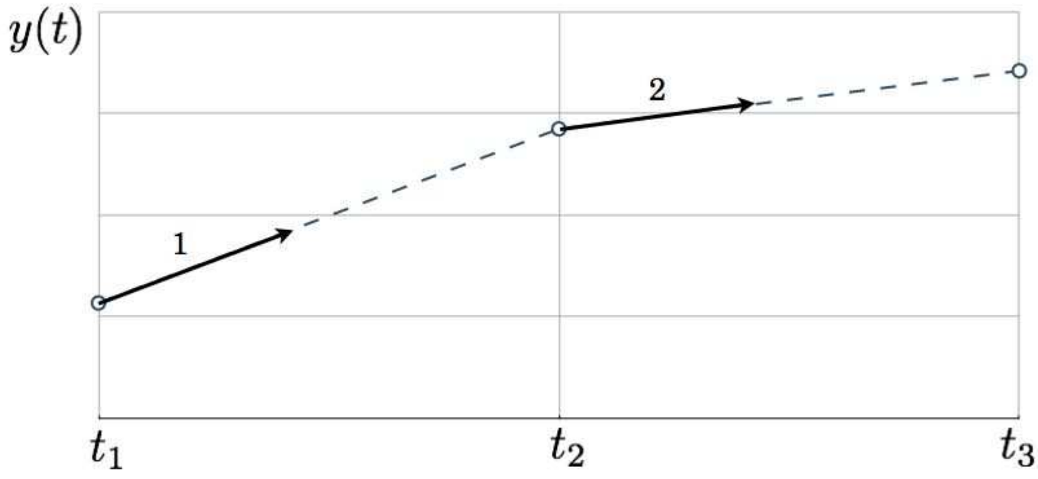


**Figure 4.1:** Euler's method. The derivative at each starting point is used to obtain the value of the next point. The method is first order accurate.

The method is quite fast because it requires only one evaluation of the derivatives per step. However, the method is clearly based upon extrapolation and hence can lead to severe problems depending upon the complexity of the function to be integrated. Since we can write the new value as a function of the old values (which are known either from previous steps or from the initial conditions) this method is also an *explicit* method.

Another, apparently very similar one step method can be written as

$$y^{n+1} = y^n + \Delta t f(t_{n+1}, y^{n+1}) \tag{4.7}$$

Contrary to the previous expression, here the derivative is estimated at the new time $t_{n+1}$ rather than at the beginning of the time step. Since the actual solution at the new time is still unknown, this modification is far from trivial since we now have an *implicit* scheme. To solve such an equation one has generally to resort to some iterations which makes the implicit approach not only more complicated but also more computationally expensive. One can therefore ask why such an approach should even be considered? It turns out that implicit schemes have often the advantage to be stable (we shall discuss stability in a later chapter) even with relatively large time steps while explicit schemes can be grossly unstable or have incredibly severe constraints on the size of the time step to be used. Hence, in practice the best approach is always based on a tradeoff between complexity, accuracy and stability.
To improve on the accuracy of the scheme while at the same time keeping the possibility to solve explicitly for the new value, we can use the centered difference approximation defined by Eq. 1.11. We obtain in this case the explicit *two-step formula*:

$$y^{n+1} = y^{n-1} + 2\Delta t f(t_n, y^n) \tag{4.8}$$

It follows from our discussion about the error associated with the expression of the derivative that the error in this equation is actually $O(\Delta t^3)$. Hence, Eq. 4.8 is second order accurate. While this is certainly an improvement in accuracy, in practice we have the problem associated with the initialization. If initial conditions are given at time $t = 0$, how do we provide the two levels ($n$ and $n-1$) required to compute the new solution? This initialization problem is a generic problem in multi-step methods. Simply ignoring it by providing twice the same value for $y^n$ and $y^{n+1}$ is not the solution as we would simply propagate the initial error further.
There are essentially two ways to circumvent this difficulty. Either one uses a lower order scheme such as the Euler scheme (Eq. 4.8) possibly with smaller time steps to minimize error, or one reverts to a Runge-Kutta method (see section 4.3).
While the schemes we have seen up to now have important applications in the more complicated problem of solving partial differential equations, ordinary differential equations are generally solved with approaches yielding more accurate solutions. One such popular method is the so-called fourth order *Adams-Bashforth* formula

$$y^{n+1} = y^n + \frac{\Delta t}{24}\left(55f^n - 59f^{n-1} + 37f^{n-2} - 9f^{n-3}\right) \tag{4.9}$$

or the fourth order *Adams-Moulton* formula, an implicit three-step approach

$$y^{n+1} = y^n + \frac{\Delta t}{24}\left(9f^{n+1} + 19f^n - 5f^{n-1} + f^{n-2}\right) \tag{4.10}$$

To illustrate these formulae, lets us integrate the following differential equation $dy/dt = y$ with $t \in [0, 2]$ and $y(0) = 1$. Obviously, the analytical solution to this equation is given by $y(t) = \exp(t)$. We compare the results between the theoretical solution and three different numerical solution at $t = 2$ in Table 4.1.

**Table 4.1:** Comparison between three different numerical solutions to the differential equation $dy/dt = y$. Show are the errors ($e_i = |\tilde{y}(t = 2) - y(t = 2)|$ where $\tilde{y}(t = 2)$ is the analytical solution at $t = 2$ and $y(t = 2)$ the corresponding numerical solution) as a function of step size $\Delta t$. The last column gives the ratio of the errors indicating the order of the method.

| Case | $e_1(\Delta t = 0.2)$ | $e_2(\Delta t = 0.1)$ | $e_3(\Delta t = 0.05)$ | $e_2/e_3)$ |
|---|---|---|---|---|
| Euler | 1.19732 (16%) | 0.66156 (8.9%) | 0.34907 (4.7%) | 1.90 |
| Midpoint | 0.09055 (1.2%) | 0.02382 (0.3%) | 0.00607 (0.08 %) | 3.92 |
| Adams-Bashforth | 0.00422 (0.05%) | 0.00038 (0.005%) | 0.00003 (0.0005%) | 12.7 |

A look at this Table confirms that first order accurate schemes such as the Euler method lead to sizable errors even with relatively small steps. Increasing the order of the method improves dramatically not only the error itself but also, as expected, the convergence rate towards the exact solution as one reduces the step. The ratio $e_2/e_3$ suggests that if the Euler method is $O(\Delta t)$ than the midpoint formula is $O(\Delta t^2)$ while the Adams-Bashforth is $O(\Delta t^4)$.
We have listed here a few formula with rather non-trivial coefficients. Clearly an infinite number of combinations can be found. We can write the following general expression for an a *s-step* formula

$$\sum_{j=0}^{s} \alpha_j y^{n+1-j} = \Delta t \sum_{j=0}^{s} \beta_j f^{n+1-j} \tag{4.11}$$

where the constants $\alpha_0 = 1$ and $\alpha_j$ and $\beta_j$ have to be determined. Note that if $\beta_0 = 0$ the formula is explicit while for $\beta_0 \neq 0$ the formula is implicit. The remaining constants can be determined by considerations about the order of the method. This can be done similarly as for the example in Eq. 1.12. We choose the number of steps $s$ of the method and Taylor expand the resulting formula about a common point (typically $n$). The constants can then be determined by optimizing the order of the method, i.e. by canceling the error terms to the highest order in $\Delta t$ possible.

## 4.3 Runge-Kutta method with fixed step

The multi-step methods advance the solution by using the solution computed at previous steps, the number of steps used determining essentially the accuracy of the method.

We have seen that a major drawback of this approach is related to the starting conditions. Unless we are using the first order Euler scheme, multi-step methods require special starting procedure. Depending upon the number of steps involved, this starting procedure can be quite complicated and require significant of extra-computations. Furthermore, if not done with the required accuracy, the errors will be propagated further along during the integration rendering the efforts at higher order integration useless.
In the *Runge-Kutta* approach previous time steps are not used but intermediate steps are computed to advance the solution. Hence, initial conditions specified at $t_0$ are sufficient to integrate the equation to the accuracy required without any other consideration. It is this ease of use that have made Runge-Kutta schemes so popular even though they are not particularly efficient since the intermediates steps are discarded at the end of the step.
Let us consider the following first order differential equation

$$\frac{dy}{dx} = f(x, y) \tag{4.12}$$

We can begin with a very simple extension of the Euler method by simply considering a additional step at mid-point of the interval. This point can then be used to compute the desired value at the end of the interval. Using this idea, we write

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\ y_{n+1} &= y_n + k_2 \end{aligned} \tag{4.13}$$

where $h$ is the step size along the $x$ coordinate (note the use of the subscript to express space coordinates). Since the additional point is computed at mid-point this approach is also called the *mid-point* method or *second order Runge-Kutta*. With the help of a Taylor expansion, it is easy to show that the error is $O(h^3)$, hence the method is second order accurate which explains the name. A graphical representation of this method is shown in Fig. 4.2.
Just as with the multi-step method we can derive an infinite number of formulae by varying the position of the intermediate steps and the corresponding weight given to the function evaluated at these points. Of course, the more intermediate points are being used the higher the order of the method but the more computationally expensive it becomes (more function evaluations are required which are then discarded at the end of the timestep). As we have already noted in section 1.5, higher order does not necessarily imply higher accuracy especially in the case where the function includes discontinuities or is rapidly varying. So, as usual, some advance knowledge of the solution is of great advantage in order to adapt the order and hence the computational effort to the expected gain...

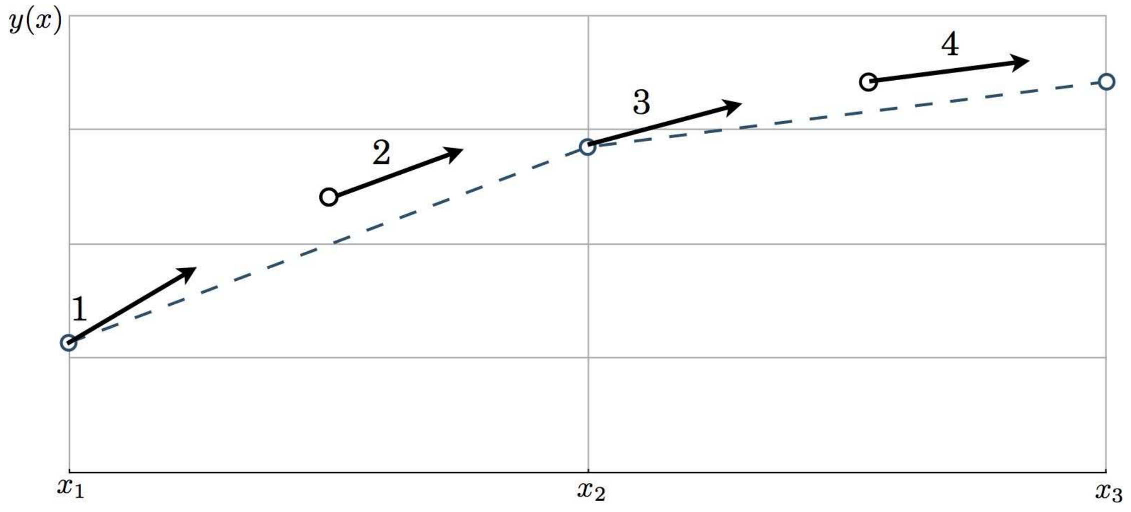


**Figure 4.2:** Mid-point or second order Runge-Kutta method. The derivative at mid-point is used to compute the final value. The method is second order accurate.

Various schemes can be found in all the major textbooks on the subject, but probably the most popular one is the so-called *fourth-order Runge-Kutta* formula

$$\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\
k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\
k_4 &= hf(x_n + \frac{1}{2}h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \qquad (4.14)
\end{aligned}$$

As suggested by its name, this formula has an error $O(h^5)$ and hence the solution is fourth order accurate. Note that it required 4 estimate of the derivative $f$ for a single time step. This is twice the amount required for the mid-point formula. Hence, from a computational point of view, this method has an advantage only if for the same accuracy steps twice as big can be taken. Often, for sufficiently well behaved function this might be the case. However, as we have just said before, higher order does not necessarily mean higher accuracy and the order of the method should always be adapted to the problem at hand. Having said that, it is also true that this fourth-order scheme generally represents a good compromise between efficiency and accuracy and, unless the computation of the derivative $f$ is especially time consuming represents a safe bet.

For completeness, we give a few general remarks on Runge-Kutta methods. For this, lets us define the real numbers $b_1, b_2, \dots, b_s$ and $a_{i,j}$ with $i, j = 1, \dots, s$ and let $c_i = \sum_j a_{i,j}$. Then the following method

$$\begin{aligned} k_i &= f(x_n + c_i h, y_n + h \sum_{j=1}^{s} a_{i,j} k_j) \\ y_{n+1} &= y_n + h \sum_{i=1}^{s} b_i k_i \end{aligned} \tag{4.15}$$

is an $s-$stage Runge-Kutta. Sometimes in the literature Runge-Kutta methods are expressed by means of the so-called Butcher or Runge-Kutta tableau.

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

Explicit methods are those for which the tableau is top-diagonal. In other words, explicit methods are iven by

$$\begin{array}{c|cccc} 0 & 0 & & & \\ c_2 & a_{21} & 0 & & \\ \vdots & \vdots & & & \\ c_s & a_{s1} & \cdots & a_{ss-1} & 0 \\ \hline & b_1 & \cdots & b_{s-1} & b_s \end{array}$$

Furthermore, one can show that the method is consistent provided that

$$\sum_{j=1}^{i-1} a_{ij} = c_i \; for \; i = 2, \dots, s \tag{4.16}$$

Additional requirements on the coefficients come from considerations about the order of the method. These so-called order conditions can be derived by considering the corresponding truncation error which is obtained by Taylor expanding at a single point the method. This is, however, not such a trivial matter because it rapidly involves a significant amount of computations and becomes rapidly tedious since the number of these order conditions rapidly increases. Indeed, while there are 2 conditions for the

second order method, there are 8 for the fourth order method, 37 for the sixth order method and 1025 for the tenth order method!
As an example, we give the relevant tableau for the well known fourth order method:

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}$$

## 4.4 Runge-Kutta method with variable step

In the previous section we have considered Runge-Kutta methods with a fixed step size $h$. Depending upon the problem at hand, this is not satisfactory. Indeed, depending upon the behavior of the function this step size might be too large (rapidly varying function) or too small (slowly varying function). In the former case, this could lead to large errors while in the latter unnecessary computational resources are waisted.
A better approach is one that adapts the step size in order to achieve a given accuracy in the solution. In other words, we are looking for an approach that uses a variable step size. Of course, for this to be possible the algorithm must return some estimate of the error which can then be used to adapt the step. This is certainly additional computational work, but it generally pays off generous dividends.

### 4.4.1 Step doubling

With the workhorse 4th order Runge-Kutta method the most straightforward approach is the so-called *step doubling* method. In essence, we take each step twice once as a full step and once as two full half steps as illustrated in Fig. 4.3
Since the first point at $x_{n-1}$ is common for both, it is easy to count that this procedure will require 11 evaluations to complete the step. The error estimate is then computed from considering the solution obtained using a single large step $y_1$ with the solution obtained using the two steps $y_2$. Since we deal with a fourth order method, the leading error term will be of order $h^5$ and we can write

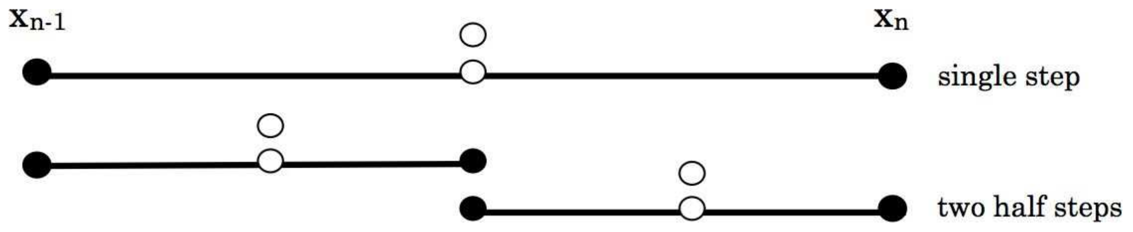


**Figure 4.3:** Step doubling as a means to control the accuracy in the case of the 4th order Runge-Kutta method (Eq. 4.14).

$$\begin{aligned} y(x+2h) &= y_1 + (2h)^5\phi + O(h^6) + \ldots \\ y(x+2h) &= y_2 + 2(h^5)\phi + O(h^6) + \ldots \end{aligned} \tag{4.17}$$

where $\phi$ stands for a term of order $f^{(5)}/5!$. We estimate the error from

$$\Delta = y_2 - y_1 \tag{4.18}$$

From Eq. 4.17 we know that $\Delta \propto h^5$. Therefore, if some choice of the step size, say $h_1$, produces results in an error estimate $\Delta_1$ while the desired accuracy is $\Delta_0$, we can compute the new step $h_0$ as

$$h_0 = h_1 \left|\frac{\Delta_0}{\Delta_1}\right|^{1/5} \tag{4.19}$$

In other words, if the estimated accuracy $\Delta_1$ is larger than the desired accuracy $\Delta_0$, we *reject* the timestep and try again with a new step size given by Eq. 4.19. On the other hand, if the estimated accuracy is smaller than the desired one, we increase the step size using Eq. 4.19 for the *next* step.

### 4.4.2 Runge-Kutta-Fehlberg

As we have shown in the previous section, the step doubling method allows the control of the step size but at the expense of a significant overhead in computational effort. Even if this effort eventually pays back in either keeping the error under control or allowing a significant speed-up, it would still be advantageous if the number of evaluation could be significantly reduced.
While $M$-th order methods generally require more than $M$ evaluations of the derivatives, Fehlberg discovered a 5-th order method that only requires 6 evaluations while the same combination of evaluations but with different factors yields a 4-th order scheme. Therefore this approach allows the error estimation at a much reduced computational

cost (with 6 evaluations only). The general form of this 5-th order method is the following

$$\begin{aligned} k_1 &= hf(x_n) \\ k_2 &= hf(x_n + a_2 h, y_n + b_{21} k_1) \\ &\dots \\ k_6 &= hf(x_n + a_6 h, y_n + b_{61} k_1 + \ldots + b_{65} k_5) \\ y_{n+1} &= y_n + c_1 k_1 + c_2 k_2 + c_3 k_3 + c_4 k_4 + c_5 k_5 + c_6 k_6 + O(h^6) \end{aligned} \tag{4.20}$$

while the corresponding 4-th order method is given by

$$y^*_{n+1} = y_n + c^*_1 k_1 + c^*_2 k_2 + c^*_3 k_3 + c^*_4 k_4 + c^*_5 k_5 + c^*_6 k_6 + O(h^5) \tag{4.21}$$

Table 4.2 gives the value or the coefficients to be used in Eqs. 4.20 and 4.21 as preferred by Press et al. which are somewhat different from the one originally derived by Fehlberg.

**Table 4.2:** Coefficients for the Runge-Kutta-Fehlberg method of order 4/5

| $i$ | $a_i$ | $b_{ij}$ | | | | | $c_i$ | $c^*_i$ |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | $\frac{37}{378}$ | $\frac{2825}{27648}$ |
| 2 | $\frac{1}{5}$ | $\frac{1}{5}$ | | | | | 0 | 0 |
| 3 | $\frac{3}{10}$ | $\frac{3}{40}$ | $\frac{9}{40}$ | | | | $\frac{250}{621}$ | $\frac{18575}{48384}$ |
| 4 | $\frac{3}{5}$ | $\frac{3}{10}$ | $-\frac{9}{10}$ | $\frac{6}{5}$ | | | $\frac{125}{594}$ | $\frac{13525}{55296}$ |
| 5 | 1 | $-\frac{11}{54}$ | $\frac{5}{2}$ | $-\frac{70}{27}$ | $\frac{35}{27}$ | | 0 | $\frac{277}{14336}$ |
| 6 | $\frac{7}{8}$ | $\frac{1631}{55296}$ | $\frac{175}{512}$ | $\frac{575}{13824}$ | $\frac{44275}{110592}$ | $\frac{253}{4096}$ | $\frac{512}{1771}$ | $\frac{1}{4}$ |
| $j =$ | | 1 | 2 | 3 | 4 | 5 | | |

The error estimate for step size control is given by

$$\Delta = y_{n+1} - y^*_{n+1} = \sum_{i=1}^{6} (c_i - c^*_i) k_i \tag{4.22}$$

and the step size adjusted as in Eq. 4.19. Note that with the Fehlberg approach, only 6 evaluations are required to achieve the step control. Depending upon the computational effort required to compute these evaluations, this can represent a significant amount of savings.

## 4.5 References

- Buther, J.C., The numerical analysis of ordinary differential equations (J. Wiley & sons: Chichester, 1987)
- Forsythe, G.E., Malcolm, M.A., & Moler, C.B., Computer methods for mathematical computations (Prentice-Hall: Englewood-Cliffs, N.J., 1977) Section 6
- Press, W.H., Flannery, B.P., Teukolsky, S.A., & Vetterling, W.T., Numerical Recipes: The art of scientific computing (Cambridge University Press: Cambridge, 1988) Section 15

# Chapter 5

# Partial differential equation

We have seen in chapter 1 how to write algebraic representations of derivatives on a grid. In chapter 4 we have seen how to use these expressions to obtain formulae that allow to integrate ordinary differential equations.
In this chapter, we shall begin to address the more general case of the integration of partial differential equations. These equations combine partial derivatives of a function of several variables and therefore the methods developed in chapter 4 are not directly applicable. Furthermore, solving systems of partial differential equations is generally a computer time consuming task and efficiency becomes also a major concern.
Our task is to chose and combine the different expressions for the derivatives we have at our disposal to again replace the partial differential equations by a set of algebraic equations. We shall see that while apparently a trivial matter, this is a tricky issue which can lead to considerable difficulties or even unusable schemes even though each of the individual term is written correctly.

## 5.1 Stability

While accuracy of the individual derivatives is of importance to obtain accurate solutions, it is not enough. Stability is an equally important concept. By stability, we mean wether or not any perturbation introduced at some stage in the calculation will actually grow without bound or rather be damped. A scheme for which the perturbations decay with time is a *stable* scheme while one for which perturbations grow with time is call an *unstable* scheme. We note that in numerical work, perturbations are present at all stages due to the unavoidable round-off errors. Obviously, unstable schemes will be of no use even if the individual derivatives are calculated to high precision. We shall see that accuracy and stability are two different things. It is not enough to discretize derivatives to some order, one must also ensure that the combination of derivatives can be used together! This somewhat surprising result illustrates the difficulties involved in numerical simulations. Although we shall use one particular method to analyze the

stability of a particular scheme (see below), analytical tools only exist for linear systems. For non-linear systems, only a trial-and-error approach is available coupled with the experience of the user...
Several methods are at hand to investigate mathematically the stability properties of a given scheme

- *Von Neumann method.*
  This is the easiest and most dependable method. Unfortunately it only works for initial value problems with constant (or slowly varying) coefficients.

- *Discrete perturbations analysis*

It is worth pointing out that usually a complete stability analysis of a algebraic representation is impossible because in practice the system becomes rapidly too complicated or, in case of the Von Neumann method, non-linear. This does not mean that this type of analysis is useless, quite to the contrary. They provide invaluable insights into the discretization process and allow to set necessary conditions for stability. For this reasons, we shall introduce such a stability analysis for a number of basics schemes below.

## 5.2 Methodology

Based on what we have seen so far, we can outline the basic methodology for developing numerical solutions to a set of partial differential equations. This is outlined in Fig. 5.1

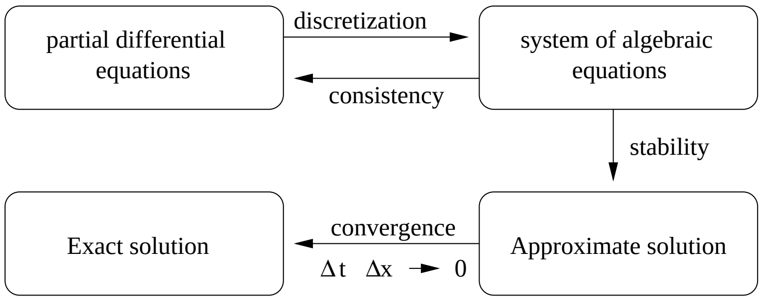


**Figure 5.1:** Methodology for developing numerical solutions to a set of partial differential equations

There is even a theorem, the so-called Lax theorem, that says: "Given a properly posed linear initial value problem and a finite difference to it that satifies the consistency condition, stability is the necessary and sufficient condition for convergence". While this theorem sounds quite powerfull, in practice most problems are non-linear and while consistency and stability are necessary there are no longer sufficient to ensure convergence.

## 5.3 One-dimensional diffusion equation

We are interested in solving the one-dimensional diffusion equation with constant diffusion coefficient. Mathematically, we want to solve the following equation

$$\frac{\partial f}{\partial t} = \alpha \frac{\partial^2 f}{\partial x^2} \tag{5.1}$$

between $x = 0$ and $x = L$. To do this, we have to specify suitable initial and boundary conditions. These can be specified either as Dirichlet boundary conditions such as for example, $f(0,t) = a(t)$ and $f(L,t) = b(t)$, or as von Neumann conditions such as $\frac{\partial f}{\partial x}(0,t) = c(t)$ and $\frac{\partial f}{\partial x}(L,t) = d(t)$. Either one of these conditions (together of course with initial conditions) is sufficient.
We shall now investigate different possibilities to transform this continuum differential equation into a suitable algebraic equation that can be solved on the computer.

### 5.3.1 The forward time centered space (FTCS) approximation

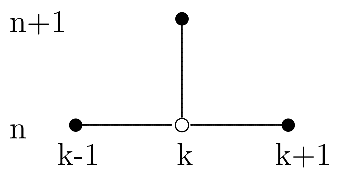
An obvious possible representation for the diffusion equation is obtained using a simple forward difference in time and centered in space (FTCS).

$$\frac{f_k^{n+1} - f_k^n}{\Delta t} = \alpha \frac{f_{k-1}^n - 2f_k^n + f_{k+1}^n}{\Delta x^2} \tag{5.2}$$

This scheme can be solved explicitly for the value at the new timestep

$$f_k^{n+1} = s f_{k+1}^n + (1-2s) f_k^n + s f_{k-1}^n \tag{5.3}$$

where $s = \alpha \Delta t / \Delta x^2$. To visualize the nodes that are active in a particular scheme it is instructive to draw the nodes that are actually used by the scheme as follows:



The truncation error can be computed by expanding everything in Taylor expansions centered at $(k,n)$. We obtain

$$E_k^n = \left( \frac{\Delta t}{2} \frac{\partial^2 f}{\partial t^2} - \alpha \frac{\Delta x^2}{12} \frac{\partial^4 f}{\partial x^4} \right)_k^n + O(\Delta t^2, \Delta x^4) \tag{5.4}$$

Not surprisingly, the FTCS scheme is only first order accurate in time while second order accurate in space. The scheme is consistent with the original equation since the truncation error vanishes in the limit of small $\Delta x$ and $\Delta t$.
It is interesting to note that we can eliminate the second order time derivative in Eq. 5.4 by using the original diffusion equation

$$\frac{\partial^2 f}{\partial t^2} = \alpha \frac{\partial}{\partial t} \frac{\partial^2 f}{\partial x^2} = \alpha \frac{\partial^2}{\partial x^2} \frac{\partial f}{\partial t} = \alpha^2 \frac{\partial^4 f}{\partial x^4} \tag{5.5}$$

Using this equation, we can rewrite the truncation error in the following way

$$E_k^n = \frac{1}{2} \alpha \Delta x^2 (s - \frac{1}{6}) \left( \frac{\partial^4 f}{\partial x^4} \right)_k^n + O(\Delta t^2, \Delta x^4) \tag{5.6}$$

Interestingly, for the special value of $s = \frac{1}{6}$ the first term in the truncation error vanishes making the FTCS scheme second order accurate in time and fourth order accurate in space! Unfortunately, in practice it is only possible to impose this value of $s$ on all the computational grid if the grid is uniform which is seldomly the case.
We now proceed and make a stability analysis of this scheme in order to derive how perturbations will propagate. Due to roundoff errors the actual solution computed is not $f_k^n$ (the actual true numerical solution of the algebraic equation) but $f_k^{*n}$ the actually computed solution. The error can be written as the difference between the two

$$\xi_k^n = f_k^n - f_k^{*n} \tag{5.7}$$

Since $f_k^{*n}$ is the actually computed solution, our FTCS scheme in Eq. 5.3 should in fact we written for $f_k^{*n}$. Using Eq. 5.7, we can replace $f_k^{*n}$ by $f_k^{*n} = f_k^n - \xi_k^n$ and insert this in Eq. 5.3. We obtain

$$\xi_k^{n+1} = \xi_k^n + s \left( \xi_{k+1}^n + \xi_{k-1}^n - 2\xi_k^n \right) \tag{5.8}$$

Clearly, this equation is of the same type as the original one (note that this will always be true for linear equations). A stability analysis is now concerned with the time behavior of $\xi$. In particular, we would like to know whether $\xi$ grows out of bounds or eventually decays.
In the von Neumann stability analysis, we assume that the errors are periodic in the interval of interest and expand the corresponding error terms in a Fourier series

$$\xi_k^n = \sum_{m=1}^{J} a_m^n e^{i\varphi_m k} \quad k = 2, 3, \ldots, J-1 \tag{5.9}$$

where $\varphi_m$ is the phase angle ($\varphi_m = m\pi\Delta x$). As long as the equation is linear, it is enough to consider only one Fourier mode and we can write (dropping index $m$):

$$\xi_k^n = a^n e^{i\varphi k} \tag{5.10}$$

Inserting this into Eq. 5.8 yields

$$a^{n+1}e^{i\varphi k} = a^n e^{i\varphi k} + s\left(a^n e^{i\varphi(k+1)} + a^n e^{i\varphi(k-1)} - 2a^n e^{i\varphi k}\right) \tag{5.11}$$

This expression can be simplified (recalling that $e^{i\varphi} + e^{-i\varphi} = 2\cos\varphi$) to yield

$$G = \frac{a^{n+1}}{a^n} = 1 + 2s\left(\cos\varphi - 1\right) \tag{5.12}$$

where $G$ is the so-called *amplification factor*. Clearly, when $|G| \leq 1\,, \forall\varphi$, errors will damp (or at least not grow) while when $|G| > 1\,$, for at least one $\varphi$, errors will grow out of bounds. The condition for stability therefore reads

$$-1 \leq 1 + 2s\left(\cos\varphi - 1\right) \leq 1 \tag{5.13}$$

The inequality to the right is always satisfied since $\cos\varphi \leq 1$ and $s \geq 0$. The inequality to the left implies

$$-1 \leq s\left(\cos\varphi - 1\right)\ \forall\varphi \quad \rightarrow \quad s \leq \frac{1}{2} \tag{5.14}$$

Recalling the definition of $s$, this yields the following *stability constraint* on the timestep

$$\Delta t \leq \frac{1}{2}\frac{\Delta x^2}{\alpha} \tag{5.15}$$

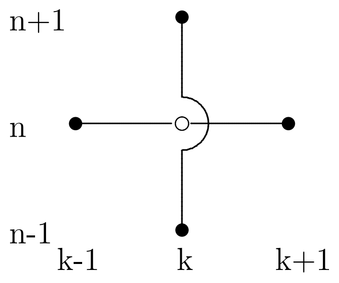
Note that $\Delta x^2/\alpha$ is nothing else than the actual diffusion time through on cell of width $\Delta x$. Hence, stability requires that the time step be less than the diffusion timescale over one cell. This can be intuitively understood in the sense that all cells need to be aware of what is going on. If the timestep would exceed the diffusion timescale of a cell, the diffusion process could bypass a cell all together.
We also note that this is a severe constraint on the time step. Doubling the spatial resolution implies a time step 4 times smaller hence 8 times more computing expenses! Furthermore, should the grid be non-uniform the stability constraint implies that the smallest $\Delta x$ over the whole grid must be used in determining the timestep according to Eq. 5.15.

### 5.3.2 The Richardson approximation

We can try to improve on the accuracy by combining two second order derivatives. In this spirit, we write the following centered approximation, the so-called *Richardson* approximation:

$$\frac{f_k^{n+1} - f_k^{n-1}}{2\Delta t} = \alpha \frac{f_{k-1}^n - 2f_k^n + f_{k+1}^n}{\Delta x^2} \tag{5.16}$$

We recognize in this expression on the left side the second order accurate centered difference expression (Eq. 1.11) and on the right side the second second order accurate expression for second order derivatives (Eq. 1.18). Since both derivatives are centered, this expression is second order accurate in both space and time. We can visualize the active nodes in the graphical representation as follows:



Eq. 5.16 can be explicitly solved for the value of the function at the new time as a function of the value at the previous time

$$f_k^{n+1} = f_k^{n-1} + \frac{2\alpha\Delta t}{\Delta x^2}(f_{k-1}^n - 2f_k^n + f_{k+1}^n) \tag{5.17}$$

While apparently satisfactory, this expressions turns out to be unconditionally unstable as we shall now deonstrate it using the von Neumann stability analysis. Introducing the Fourier decomposition for the error and introducing it in Eq. 5.17 as in the FTCS case, we obtain

$$a^{n+1} = a^{n-1} + a^n \left(2s(\cos\varphi - 1)\right) \tag{5.18}$$

Contrary to the FTCS case, we cannot directly write the amplification factor since the scheme actually involves three different time levels. However, we can re-write the system as follows

$$\begin{pmatrix} a^{n+1} \\ a^n \end{pmatrix} = \underbrace{\begin{pmatrix} 2s\left(\cos\varphi - 1\right) & 1 \\ 1 & 0 \end{pmatrix}}_{G} \begin{pmatrix} a^n \\ a^{n-1} \end{pmatrix} \tag{5.19}$$

Where $G$ is no longer an amplification factor but an amplification matrix. If we want the errors to decay with time, we have to require that

$$|\lambda_m| \leq 1 \ , \ \forall m \tag{5.20}$$

where $\lambda_m$ is the $m$th eigenvalue of the amplification matrix $G$. To find the eigenvalues, we set

$$\begin{vmatrix} 2s\left(\cos\varphi - 1\right) - \lambda & 1 \\ 1 & -\lambda \end{vmatrix} = 0 \tag{5.21}$$

which yields the characteristic equation

$$\lambda^2 - 2s\left(\cos\varphi - 1\right)\lambda - 1 = 0 \tag{5.22}$$

for which the two solutions are given by

$$\lambda_{\pm} = \underbrace{s\left(\cos\varphi - 1\right)}_{\leq 0} \pm \underbrace{\sqrt{s^2\left(\cos\varphi - 1\right)^2 + 1}}_{\geq 1} \tag{5.23}$$

Let us look at $\lambda_-$ and let us chose a value $\varphi = \pi$. This yields

$$\lambda_-(\varphi = \pi) = -2s - \sqrt{4s^2 + 1} < -1 \tag{5.24}$$

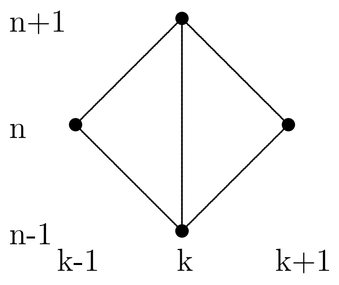
This clearly contradicts the stability condition given by Eq. 5.20. Hence, we must conclude that the Richardson approximation is unconditionally unstable. Even though this scheme looked quite promising, it is essentially useless in this form.

### 5.3.3 The Dufort-Frankel approximation

After this somewhat disappointing results, lets us turn towards a scheme that actually works. To illustrate how small differences can make a large difference and the difficulty in determining *a priori* which combination of derivatives work, let us modify the Richardson scheme discussed above slightly by writing $f_k^n = 0.5(f_k^{n+1} + f_k^{n-1})$ which yields the following scheme

$$\frac{f_k^{n+1} - f_k^{n-1}}{2\Delta t} = \alpha \frac{f_{k-1}^n - f_k^{n-1} - f_k^{n+1} + f_{k+1}^n}{\Delta x^2} \tag{5.25}$$

An expression which is know as the Dufort-Frankel approximation. Graphically, we can illustrate this scheme as follows



We can again solve explicitly Eq. 5.25 for the value of the function $f$ at the new time as a function of the old value. This yields:

$$f_k^{n+1} = \frac{2s}{1+2s}(f_{k-1}^n + f_{k+1}^n) + \frac{1-2s}{1+2s} f_k^{n-1} \tag{5.26}$$

with $s = \alpha \Delta t / \Delta x^2$. Contrary to the Richardson approximation, it turns out that this approximation is now unconditionally stable! This illustrates how seemingly small changes can make a huge differences in the results.
We now demonstrate the stability of the Dufort-Frankel scheme by using our Fourier decomposition. We obtain

$$\begin{pmatrix} a^{n+1} \\ a^n \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{4s\cos\varphi}{1+2s} & \frac{1-2s}{1+2s} \\ 1 & 0 \end{pmatrix}}_{G} \begin{pmatrix} a^n \\ a^{n-1} \end{pmatrix} \tag{5.27}$$

We find again the eigenvalue by setting $\det(G - \lambda I) = 0$. We obtain the characteristic equation

$$\lambda^2 - \frac{4s\cos\varphi}{1+2s}\lambda - \frac{1-2s}{1+2s} = 0 \tag{5.28}$$

which has the following two solutions

$$\lambda_\pm = \frac{2s\cos\varphi \pm \sqrt{1 - 4s^2\sin^2\varphi}}{1+2s} \tag{5.29}$$

One can easily convince oneself that the two eigenvalues above always satisfy the stability criterion for all values of $\varphi$ as long as $s \geq 0$. Hence the Dufort-Frankel scheme is indeed unconditionally stable.
In fact, although the change we made appeared small, they were not as small as that if we take a deeper look at it. To investigate this point, let us expand all terms in a Taylor expansion centered at $(k, n)$. We obtain the following equation

$$\left( \frac{\partial f}{\partial t} - \alpha \frac{\partial^2 f}{\partial x^2} \right)_k^n + E_k^n = 0 \tag{5.30}$$

where the truncation error is given by

$$E_k^n = \left( \alpha \left( \frac{\Delta t}{\Delta x} \right)^2 \frac{\partial^2 f}{\partial t^2} \right)_k^n + O(\Delta x^2, \Delta t^2) \tag{5.31}$$

Hence what appeared at first glance as a small change actually changed the entire nature of the original equation. Adding a term proportional to the second order time derivative transformed the originally parabolic equation into a hyperbolic equation! This obviously changes fundamentally the nature of the solutions. This example serves the purpose to warn about changes that may hide large consequences.
A given scheme is said to be consistent if in the limit of small $\Delta x$ and $\Delta t$ the original equation is recovered. In the case of the Dufort-Frankel scheme, we notice that the error is proportional to $\Delta t/\Delta x$ hence the original equation is recovered only if in addition to $\Delta x \to 0$ and $\Delta t \to 0$ we also have $\Delta t/\Delta x \to 0$. In practice, since the limit is never actually taken these constraints are rather vague. However, even though stability does not limit the timestep, consistency requires that

$$\alpha \left( \frac{\Delta t}{\Delta x} \right)^2 \ll 1 \tag{5.32}$$

### 5.3.4 The Crank-Nicholson approximation

All the schemes we have been considering up to now are characterized by the fact that the value of the function at the new time can be written explicitly as a function of the values at past times. We shall see in a later chapter that the so-called *explicit* schemes all suffer from limitations on the size of the time-step to ensure stability (with the noted exception of the Dufort-Frankel scheme). These limitations, in particular in the case of the diffusion equation, can turn out to be quite severe rendering some of these schemes very expensive in CPU time. It is possible to write schemes which are *a priori* not subject to such timestep limitations for stability reasons. Unfortunately, in such schemes it is no longer possible to explicitly solve for the value of the function at the new time but a system of equations has to be solved. Such schemes are called *implicit* schemes.
The simplest and well known implicit scheme, the so-called *Crank-Nicholson* scheme is obtained by a combination of the forward time differencing and an averaging over time the second order centered space derivative operator. Let us define this operator as follows

$$L_{xx}\, f_k^n = \frac{f_{k-1}^n - 2f_k^n + f_{k+1}^n}{\Delta x^2} \tag{5.33}$$

The Crank-Nicholson scheme is then written

$$\frac{f_k^{n+1}-f_k^n}{\Delta t} = \alpha\left(\frac{1}{2}(L_{xx}f_k^n + L_{xx}f_k^{n+1})\right) \tag{5.34}$$

Expanding the terms and regrouping them leads to the following algebraic equations

$$-\frac{1}{2}sf_{k-1}^{n+1} + (1+s)f_k^{n+1} - \frac{1}{2}sf_{k+1}^{n+1} = \frac{1}{2}sf_{k-1}^n + (1-s)f_k^n + \frac{1}{2}sf_{k+1}^n \tag{5.35}$$

A Taylor expansion of all terms shows that this expression is indeed second order accurate in space and first order in time. However, because on the left hand side the value of the function at the new time appears at three different spatial position, it is not possible to explicitly solve this equation. Rather, we have a system of linear equations that need to be solved simultaneously. We will show later that this added complexity pays off since this scheme has no longer any stability constraints on the size of the timestep. However, clearly for the solution to remain accurate we need $\Delta x$ and $\Delta t$ to remain small even in the case of implicit schemes.

Before we go ahead with solving this set of equations, we note that we could easily generalize the Crank-Nicholson scheme by writing

$$\frac{f_k^{n+1}-f_k^n}{\Delta t} = \alpha\left((1-\beta)L_{xx}f_k^n + \beta L_{xx}f_k^{n+1}\right) \tag{5.36}$$

Setting $\beta = 0$, we recover the explicit FTCS scheme while $\beta = \frac{1}{2}$ yields the Crank-Nicholson scheme. Setting $\beta = 1$ yields a fully implicit scheme as only the value of the $f$ at the new time appears in the right hand side of the equation. Additional variations can be created by adding a further level in time to the scheme

$$(1+\gamma)\frac{f_k^{n+1}-f_k^n}{\Delta t} - \gamma\frac{f_k^n - f_k^{n-1}}{\Delta t} = \alpha\left((1-\beta)L_{xx}f_k^n + \beta L_{xx}f_k^{n+1}\right) \tag{5.37}$$

The additional memory and CPU time is often offset by the fact that most oscillations in the solutions computed using the Crank-Nicholson scheme disappear using such a three level scheme. A popular choice for the free parameters is $\gamma = \frac{1}{2}$ and $\beta = 1$.

Turning now to the solution of the system of equations described by Eq. 5.35 which is nothing else than Eq. 5.37 with $\gamma = 0$ and $\beta = 1/2$, we start by writing this system in a matrix form as

$$\underbrace{\begin{pmatrix} (1+s) & -\frac{1}{2}s & 0 & 0 & ... & 0 \\ -\frac{1}{2}s & (1+s) & -\frac{1}{2}s & 0 & ... & 0 \\ 0 & -\frac{1}{2}s & (1+s) & -\frac{1}{2}s & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & -\frac{1}{2}s & (1+s) \end{pmatrix}}_{A} \begin{pmatrix} f_2^{n+1} \\ f_3^{n+1} \\ f_4^{n+1} \\ \vdots \\ f_{J-1}^{n+1} \end{pmatrix} = \begin{pmatrix} d_2 \\ d_3 \\ d_4 \\ \vdots \\ d_{J-1} \end{pmatrix} \tag{5.38}$$

with the $d$'s given by the following expressions

$$\begin{aligned} d_2 &= \frac{1}{2}sf_1^n + (1-s)f_2^n + \frac{1}{2}sf_3^n + \frac{1}{2}sf_1^{n+1} \\ d_j &= \frac{1}{2}sf_{j-1}^n + (1-s)f_j^n + \frac{1}{2}sf_{j+1}^n \\ d_{J-1} &= \frac{1}{2}sf_{J-2}^n + (1-s)f_{J-1}^n + \frac{1}{2}sf_J^n + \frac{1}{2}sf_J^{n+1} \end{aligned}$$

Note how the boundary conditions enter in the vector $d$. The values $f_1^{n+1}$ and $f_J^{n+1}$ must be given at all times.
The matrix in Eq. 5.38 is a tridiagonal matrix. Such matrices can be solved very efficiently using the so-called *Thomas* algorithm also known as forward sweep backward substitution algorithm. To see how it works consider the following system

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & ... & 0 \\ a_2 & b_2 & c_2 & 0 & ... & 0 \\ 0 & a_3 & b_3 & c_3 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & a_J & b_J \end{pmatrix} \begin{pmatrix} f_1^{n+1} \\ f_2^{n+1} \\ f_3^{n+1} \\ \vdots \\ f_J^{n+1} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_J \end{pmatrix} \tag{5.39}$$

The forward sweep consist in eliminating the $a_i$ and normalizing the $b_i$ to unity by setting

$$\begin{aligned} c'_1 &= \frac{c_1}{b_1} \\ c'_i &= \frac{c_i}{b_i - a_i c'_{i-1}} \\ d'_1 &= \frac{d_1}{b_1} \\ d'_i &= \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} \end{aligned}$$

With these changes, the system can now be written as

$$\begin{pmatrix} 1 & c'_1 & 0 & 0 & ... & 0 \\ 0 & 1 & c'_2 & 0 & ... & 0 \\ 0 & 0 & 1 & c'_3 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & 0 & 1 \end{pmatrix} \begin{pmatrix} f_1^{n+1} \\ f_2^{n+1} \\ f_3^{n+1} \\ \vdots \\ f_J^{n+1} \end{pmatrix} = \begin{pmatrix} d'_1 \\ d'_2 \\ d'_3 \\ \vdots \\ d_{J'} \end{pmatrix} \tag{5.40}$$

And the second step, the backward substitution proceeds as follows

$$\begin{aligned} f_J^{n+1} &= d'_J \\ f_i^{n+1} &= d'_i - f_{i+1}^{n+1} c'_i \end{aligned}$$

This algorithm is clearly very efficient. In practice, for more complicated systems or multi-dimensional problems one always tries to obtain such a tridiagonal matrix in order to invert the problem without too many difficulties.
As usual with matrix inversions, ill-conditioned systems can lead to severe problems. In fact, for the inversion to work well, a necessary condition is that matrix $A$ in Eq. 5.38 must be diagonally dominant. This means that if $A = [\alpha_{lm}]$ the necessary condition can be written

$$|\alpha_{ll}| \geq \sum_{m=1}^{J} |\alpha_{lm}| \ , \ l = 1, 2, \dots, J \ ; \ l \neq m \tag{5.41}$$

In our case, this means that

$$\begin{aligned} |b_1| &\geq |c_1| \\ |b_i| &\geq |a_i| + |c_i| \\ |b_J| &\geq |a_J| + |c_J| \end{aligned}$$

The stability analysis of this scheme is left to the reader. Without any difficulties it is possible to show that the Crank-Nicholson scheme is unconditionally stable.

### 5.3.5 Boundary and initial conditions

Depending on the type of boundary conditions specified, their implementation can be more or less complicated. Clearly, in the case of Dirichlet boundary conditions, we have no problems since $f(x_b, t) = c(t)$ and we can specify the value of the function at the boundary at any time. Von Neumann boundaries create a bigger problem since in this case only the derivative at the boundary is given $\partial f / \partial x = c(t)$. One frequently used method to work around this problem is to introduce a *ghost* cell on the other since of the boundary. For example, if the number of spatial zones are indexed $1, \dots, J$, we would define the two ghost zones by index 0 and $J + 1$. Hence we can write

$$\frac{f_2^n - f_0^n}{2\Delta x} = c(t) \tag{5.42}$$

Combining this expression with our FTCS scheme leads to

$$f_1^{n+1} = -2s\Delta x c^n + (1-2s) f_1^n + 2s f_2^n \tag{5.43}$$

Initial conditions are generally no problem except if they contradict the boundary conditions. In this case, it is customary to average both for the first time step and then to use the correct boundary conditions.

## 5.4 Multi-dimensional diffusion equation

Multi-dimensional systems become rapidly complicated. We shall here again concentrate only on a simple example namely the diffusion equation in two space dimensions. Let us consider

$$\frac{\partial f}{\partial t} = \alpha_x \frac{\partial^2 f}{\partial x^2} + \alpha_y \frac{\partial^2 f}{\partial y^2} \tag{5.44}$$

with $\alpha_x$ and $\alpha_y$ constants and the following boundary conditions: $f(0,y,t) = a(y,t)$ ; $f(X,y,t) = b(y,t)$ ; $f(x,0,t) = c(x,t)$ ; $f(x,Y,t) = d(x,t)$.

### 5.4.1 The forward time centered space (FTCS) approximation

A straightforward extension of the one dimensional cases we have seen above is to use the FTCS scheme and write

$$\frac{f_{j,k}^{n+1} - f_{j,k}^n}{\Delta t} = \alpha_x L_{xx} f_{j,k}^n + \alpha_y L_{yy} f_{j,k}^n \tag{5.45}$$

where the $L_{xx}$ and $L_{yy}$ operators are the second order derivative operator defined in Eq. 5.33 in the $x$ and $y$ direction respectively. By convention, the first subscript refers to the $x$ direction while the second subscript to the $y$ direction.
As in the 1D case, the FTCS scheme allows to explicitly solve for the new value of the function at $j,k$. As in the 1D case this explicit scheme also suffers from a limitation on the timestep which is given by the expression:

$$\Delta t \leq \frac{1}{2} \frac{1}{\dfrac{\alpha_x}{\Delta x^2} + \dfrac{\alpha_y}{\Delta y^2}} \tag{5.46}$$

which turns out to be more restrictive than in the 1D case even if $\alpha_x = \alpha_y$. Here again, implicit schemes allow to bypass this potentially disastrous restriction.

## 5.4.2 The fully implicit scheme

The fully implicit scheme already mentioned in Eq. 5.36 can be applied here too. We easily obtain

$$\frac{f_{j,k}^{n+1} - f_{j,k}^{n}}{\Delta t} = \alpha_x L_{xx} f_{j,k}^{n+1} + \alpha_y L_{yy} f_{j,k}^{n+1} \tag{5.47}$$

which translates in the following equation

$$-s_x f_{j-1,k}^{n+1} + (1 + 2s_x + 2s_y) f_{j,k}^{n+1} - s_x f_{j+1,k}^{n+1} - s_y f_{j,k-1}^{n+1} - s_y f_{j,k+1}^{n+1} = f_{j,k}^{n} \tag{5.48}$$

with $s_x = \alpha_x \Delta t/\Delta x^2$ and $s_y = \alpha_y \Delta t/\Delta y^2$. It can be shown that this scheme has a truncation error $O(\Delta t, \Delta x^2, \Delta y^2)$.
The problem is that the system of equations described by Eq. 5.48 cannot be grouped into a tri- nor a pentadiagonal matrix. Hence, the Thomas algorithm cannot be used to invert the system. Conventional Gauss elimination has to be used which is extremely computer intensive. This scheme is a good example of a perfectly valid approximation but useless in practice because of CPU requirements!

## 5.4.3 Splitting methods

A classical approach to solve the multidimensional diffusion equation is to use the so-called *Alternating Drirection Implicit* (ADI) method of Peaceman and Rachford (1950). In this scheme, we proceed in two half steps beginning by an implicit step in the $x$ direction only

$$\frac{f_{j,k}^{\star} - f_{j,k}^{n}}{\Delta t/2} = \alpha_x L_{xx} f_{j,k}^{\star} + \alpha_y L_{yy} f_{j,k}^{n} \tag{5.49}$$

Followed by an implicit step in the $y$ direction

$$\frac{f_{j,k}^{n+1} - f_{j,k}^{\star}}{\Delta t/2} = \alpha_x L_{xx} f_{j,k}^{\star} + \alpha_y L_{yy} f_{j,k}^{n+1} \tag{5.50}$$

Notice that each step is taken over only half the timestep. Each of the half step leads to a tridiagonal matrix which can be easily inverted. Furthermore, this approach has the additional advantage to have a truncation error $O(\Delta t^2, \Delta x^2, \Delta y^2)$. The second order in time comes from the symmetry of the scheme. To ensure second order in time one must be very careful with the implementation of the boundaries at $f^{\star}$.

## 5.5 The one-dimensional linear transport equation without damping

The diffusion equation is an important equation which is encountered in one form or another quite often in physics and the problems and particularities associated with it are quite characteristics for all parabolic equations. However, in physics other types of equations are encountered quite frequently like for example the fluid equations. We can obtain a first glimpse of the problems associated with such equations by considering the one dimensional linear transport or convection equation which is written

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} = 0 \tag{5.51}$$

with $v$ a constant. We note that this is nothing else than a disguised wave equation since

$$\frac{\partial^2 f}{\partial t^2} = -v\frac{\partial}{\partial x}\frac{\partial f}{\partial t} = v^2\frac{\partial^2 f}{\partial x^2} \tag{5.52}$$

In such a case, we know that a general solution to the equation will be given by

$$f(x,t) = g(x - vt) \tag{5.53}$$

where the initial conditions are given by $f(x,0) = g(x)$. We shall now have a look and analyze various schemes to discretize this wave equation.

### 5.5.1 The forward time centered space (FTCS) approximation

The appropriate discretization in the FTCS approach to Eq. 5.51 is written

$$\frac{f_k^{n+1} - f_k^n}{\Delta t} + v\frac{f_{k+1}^n - f_{k-1}^n}{2\Delta x} = 0 \tag{5.54}$$

This equation can be solved explicitly for the value of $f$ at the new time

$$f_k^{n+1} = f_k^n - \frac{1}{2}c\left(f_{k+1}^n - f_{k-1}^n\right) \tag{5.55}$$

where the constant $c$ is given by

$$c = v\frac{\Delta t}{\Delta x} \tag{5.56}$$

Clearly, this constant $c$ is nothing else as the fractional number of cells of width $\Delta x$ crossed by the wave during a timestep $\Delta t$. We shall see later that this number plays a critical role in determining the stability of the schemes.

We begin by determining the truncation error of our discretization by expanding all terms at $(n, k)$. Keeping terms up to third order, we obtain

$$\left(\frac{\partial f}{\partial t}\right)_k^n + v\left(\frac{\partial f}{\partial x}\right)_k^n = -\frac{1}{2}\Delta t\left(\frac{\partial^2 f}{\partial t^2}\right)_k^n - \frac{1}{6}\Delta t^2\left(\frac{\partial^3 f}{\partial t^3}\right)_k^n - \frac{1}{6}v\Delta x^3\left(\frac{\partial^3 f}{\partial x^3}\right)_k^n \tag{5.57}$$

Clearly, we recover on the left side the original equation. The right side is therefore the truncation error $E_k^n$. From this analysis, we recover the fact that the FTCS scheme is indeed $O(\Delta t, \Delta x^2)$ accurate. It is convenient to replace the time derivatives by the spatial derivatives using the original equation. We obtain

$$E_k^n = -\frac{cv\Delta x}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_k^n + \frac{v\Delta x^2}{6}\left(c^2 - 1\right)\left(\frac{\partial^3 f}{\partial x^3}\right)_k^n \tag{5.58}$$

We can now perform a stability analysis and we obtain the following amplification factor

$$G = 1 - ic\sin\varphi \tag{5.59}$$

$G$ turns out to be complex. For stability, we therefore require that $GG^* \leq 1$. That is, we must have

$$GG^* = 1 + c^2\sin^2\varphi \leq 1 \quad \forall\varphi \tag{5.60}$$

which is clearly impossible. Hence we must conclude that the FTCS scheme is unconditionally unstable. Interestingly, the scheme was found to be conditionally stable in the case of the diffusion equation (see section 5.3.1). This shows that the stability properties (and hence their usefulness) depends upon the equation to be solved. Hence, a universal scheme that works in all cases is something that simply doesnt exist. The appropriate discretization scheme must be adapted to the problem at hand.

### 5.5.2 The upwind approximation

In the upwind scheme, we use the value of the function downwind to integrate in the upwind direction. Since this obviously depends upon the sign of the velocity, we write

$$\frac{f_k^{n+1} - f_k^n}{\Delta t} = \begin{cases} -v\dfrac{f_k^n - f_{k-1}^n}{\Delta x} & \text{if } v > 0 \\ -v\dfrac{f_{k+1}^n - f_k^n}{\Delta x} & \text{if } v < 0 \end{cases} \tag{5.61}$$

Both being equivalent, let us consider only the case in which $v > 0$ which we can again explicitly solve

$$f_k^{n+1} = (1 - c)f_k^n + cf_{k-1}^n \tag{5.62}$$

where $c = v\Delta t/\Delta x$ as before. We begin with a stability analysis of this scheme and find an amplification factor

$$G = 1 - c(1 - \cos\varphi) - ic\sin\varphi \tag{5.63}$$

Requiring again that $GG^* \leq 1$, we obtain the following condition

$$2c^2 - 2c^2\cos\varphi - 2c(1 - cos\varphi) \leq 0 \quad \forall\varphi \tag{5.64}$$

This inequality is satisfied for

$$c = v\frac{\Delta t}{\Delta x} \leq 1 \quad \rightarrow \quad \Delta t \leq \frac{\Delta x}{v} \tag{5.65}$$

In other words, stability requires that the fluid (or what ever is transported) does not cross more than one cell ($\Delta x$) during a single timestep. We recall that a similar condition was already found in the case of the diffusion equation where the restriction was based on the diffusion time over one cell (cf. Eq. 5.5.1). Such a condition is called the *Courant condition* after the name of a famous fluid dynamicist.
Let us now turn and examine the truncation error of this scheme. Expanding again all the terms at $(n, k)$, we obtain

$$\left(\frac{\partial f}{\partial t}\right)_k^n + v\left(\frac{\partial f}{\partial x}\right)_k^n = -\frac{1}{2}\Delta t\left(\frac{\partial^2 f}{\partial t^2}\right)_k^n + \frac{1}{2}\Delta x\left(\frac{\partial^2 f}{\partial x^2}\right)_k^n + O(\Delta t^2, \Delta x^2) \tag{5.66}$$

Eliminating the time derivative on the right side of the equation using the original equation yields

$$E_k^n = \frac{1}{2}v\Delta x(1 - c)\left(\frac{\partial^2 f}{\partial x^2}\right)_k^n + O(\Delta t^2, \Delta x^2) \tag{5.67}$$

Interestingly, the functional form of the truncation error involves the second order space derivative of the function. Defining $\alpha^* = \frac{1}{2}v\Delta x(1 - c)$, we see that the upwind scheme could be seen as a second order accurate approximation to

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} - \alpha^*\frac{\partial^2 f}{\partial x^2} = 0 \tag{5.68}$$

Thus, we see that the upwind approximation results in adding to the original transport equation a diffusion term. This diffusion coefficient $\alpha^*$ is purely numerical since it involves quantities like $\Delta x$ and $\Delta t$ and will therefore depend upon the resolution and the time stepping used. This numerical viscosity is clearly a problem since it can severely affect the solution by damping and spreading all sharp features. However, it can be seen that the diffusion terms disappear for $c = 1$ which means looking at Eq. 5.62 that $f_k^{n+1} = f_{k-1}^n$ which is obviously the exact solution. In practice, it is very seldom that we can chose $c = 1$ over the whole grid since either $\Delta x$ or $v$ is likely to vary and unwanted diffusion will be present somewhere in the problem.

### 5.5.3 The leapfrog and Lax-Wendroff approximation

In these approximations, we try to increase the accuracy of the discretization by introducing centered differences. For example

$$\frac{f_k^{n+1} - f_k^{n-1}}{2\Delta t} + v\frac{f_{k+1}^n - f_{k-1}^n}{2\Delta x} = 0 \tag{5.69}$$

which we can solve explicitly for the value at the new time

$$f_k^{n+1} = f_k^{n-1} - c\left(f_{k+1}^n - f_{k-1}^n\right) \tag{5.70}$$

Using a Taylor expansion centered at $(n, k)$ it is easy to show that this expression is $O(\Delta x^2, \Delta t^2)$ accurate. Since the scheme leaps over $(n, k)$ to get $(n + 1, k)$ it is called the *leapfrop* scheme. In fact, the scheme handles the space-time as a chess board, the solution on the white squares evolving almost independently from the one on the black ones.

Performing a stability analysis, we obtain the following amplification matrix

$$G = \begin{pmatrix} -2ic\sin\varphi & 1 \\ 1 & 0 \end{pmatrix} \tag{5.71}$$

which yields the characteristic equation

$$\lambda_\pm = -ic\sin\varphi \pm \sqrt{1 - c^2\sin^2\varphi} \tag{5.72}$$

Looking at this expression, we realize that if $c^2\sin^2\varphi \geq 1$ than the eigenvalues are purely imaginary with module that can be greater than 1. Hence, for stability, we must require that $c^2\sin^2\varphi \leq 1$. In this case, we obtain $|\lambda| = 1$, and the scheme is marginally stable. In this case, the solution will not be damped which corresponds to the property of the original equation. Taking the maximum possible value $c = 1$, we obtain

$$f_k^{n+1} = f_k^{n-1} - f_{k+1}^n + f_{k-1}^n \tag{5.73}$$

However, from the initial conditions we know that we must have $f_{k+1}^n = f_k^{n+1}$ and $f_{k-1}^n = f_{k-2}^{n+1}$ and therefore the leapfrog scheme for $c = 1$ yields $f_k^{n+1} = f_{k-2}^{n-1}$ which is nothing else than the exact solution. The leapfrog scheme provides the exact solution when $c = 1$ provided the starting conditions are themselves correct. This illustrates that care must be given in setting up initial conditions if the accuracy of a given scheme is to be preserved.

Another quite popular scheme is the so-called *Lax-Wendroff* approximation. To derive the algorithm, let us start with an approximation to the derivative

$$\frac{\partial f}{\partial t} \approx \frac{f_k^{n+1} - f_k^n}{\Delta t} - \frac{1}{2}\Delta t\frac{\partial^2 f}{\partial t^2} \tag{5.74}$$

now using the original equation, we can replace the second order time derivative by a second order spatial derivative and we get

$$\frac{\partial f}{\partial t} \approx \frac{f_k^{n+1} - f_k^n}{\Delta t} - \frac{1}{2}\Delta t v^2 \frac{\partial^2 f}{\partial x^2} \tag{5.75}$$

We can use our second order accurate expression to discretize the spatial derivative and solve the equation for the value at the new time

$$f_k^{n+1} = f_k^n - \frac{1}{2}c\left(f_{k+1}^n - f_{k-1}^n\right) + \frac{1}{2}c^2\left(f_{k-1}^n - 2f_k^n + f_{k+1}^n\right) \tag{5.76}$$

which is the Lax-Wendroff approximation with an error $O(\Delta x^2, \Delta t^2)$. In fact, the truncation error can be calculated by means of a Taylor expansion at $(n, k)$, we obtain after eliminating the time derivatives

$$E_k^n = v\frac{\Delta x^2}{6}\left(1 - c^2\right)\frac{\partial^3 f}{\partial x^3} + vc\frac{\Delta x^3}{8}\left(1 - c^2\right)\frac{\partial^4 f}{\partial x^4} \tag{5.77}$$

A stability analysis shows that the amplification factor is given by

$$G = 1 - ic\sin\varphi - 2c^2\sin\frac{\varphi}{2} \tag{5.78}$$

which implies $c \leq 1$ for stability.
It is interesting to compare the Lax-Wendroff scheme Eq. 5.76 with the FTCS scheme Eq. 5.55. While the first one is conditionally stable, the second one was found to be unconditionally unstable. The difference between the two schemes resides in the term

$$\frac{1}{2}c^2\left(f_{k-1}^n - 2f_k^n + f_{k+1}^n\right) \tag{5.79}$$

This term must therefore be responsible for the difference in properties between the two approaches. To investigate this further, we can Taylor expand the difference at $(n, k)$ and obtain

$$\frac{1}{2}c^2\left(f_k^n - \Delta x\frac{\partial f}{\partial x} + \frac{1}{2}\Delta x^2\frac{\partial^2 f}{\partial x^2} - 2f_k^n + f_k^n + \Delta x\frac{\partial f}{\partial x} + \frac{1}{2}\Delta x^2\frac{\partial^2 f}{\partial x^2}\right) = \frac{1}{2}c^2\Delta x^2\frac{\partial^2 f}{\partial x^2} \tag{5.80}$$

Hence, we see that the additional term is nothing else than a diffusion term characterized by a "viscosity" $\alpha = \frac{1}{2}c^2\Delta x^2$.

### 5.5.4 The Crank-Nicholson approximation

To finish this section on the transport equation and for completeness, we derive the scheme corresponding to the Crank-Nicholson approach already encountered in section 5.3.3. Applied to the transport equation the scheme reads

$$\frac{f_k^{n+1} - f_k^n}{\Delta t} = -v \left( \frac{1}{2} L_x f_k^n + \frac{1}{2} L_x f_k^{n+1} \right) \qquad \text{where} \quad L_x f_k^n = \frac{f_{k+1}^n - f_{k-1}^n}{2\Delta x} \tag{5.81}$$

which yield the algebraic system of equations

$$-\frac{1}{4} c f_{k-1}^{n+1} + f_k^{n+1} + \frac{1}{4} c f_{k+1}^{n+1} = \frac{1}{4} c f_{k-1}^n + f_k^n - \frac{1}{4} c f_{k+1}^n \tag{5.82}$$

This system of equations can again be put in form of a tridiagonal matrix and therefore solved efficiently. It is easy to show that the truncation error of this approximation is $O(\Delta t^2, \Delta x^2)$. The stability analysis yields an amplification factor

$$G = \frac{1 - \frac{1}{2} i c \sin \varphi}{1 + \frac{1}{2} i c \sin \varphi} \tag{5.83}$$

It is easy to show that $GG^* \leq 1$ for all $\varphi$. Hence, the Crank-Nicholson scheme is indeed unconditionally stable when applied to the transport equation as well.

### 5.5.5 Concluding remarks

The transport equation is a much more difficult equation to solve than the diffusion equation. Moving things through the computational grid is always a delicate problem which requires a careful balance between accuracy and stability. First, we need sufficiently fine zoning to resolve the features (see also section 1.6). Second, we obviously need a stable scheme which is only possible if some damping is present which results in spreading and damping sharp features...
Besides damping and spreading another type of error can be present in numerical solutions: Dispersive errors. In this case, just like in optics, the signal propagates at a speed which is a function of its wavelength.
To illustrate these difficulties, we show in Fig. 5.2 four different simulation of the propagation of a truncated sine wave. The initial signal is given by

$$f(x, t = 0) = \begin{cases} \sin(10\pi x) & \text{for } 0 \leq x \leq 0.1 \\ 0 & \text{otherwise} \end{cases} \tag{5.84}$$

Quite clearly, none of the scheme is capable to reproduce the analytical solution correctly and the results are discouraging! All schemes show severe damping of the peak and speed propagation errors. Even worse, the solution has unphysical oscillations and $f(x,t)$ can even become negative. This can lead to severe problems is for example $f(x,t)$ would be the temperature (in Kelvin) or the density. In theses cases, negative values have no physical meaning and would surely bring the program to a crash. Tracing the origin of these oscillations, we find that they are connected with the second order nature of all these approximations. First order schemes produce no such oscillations but at the expense of much more damping.

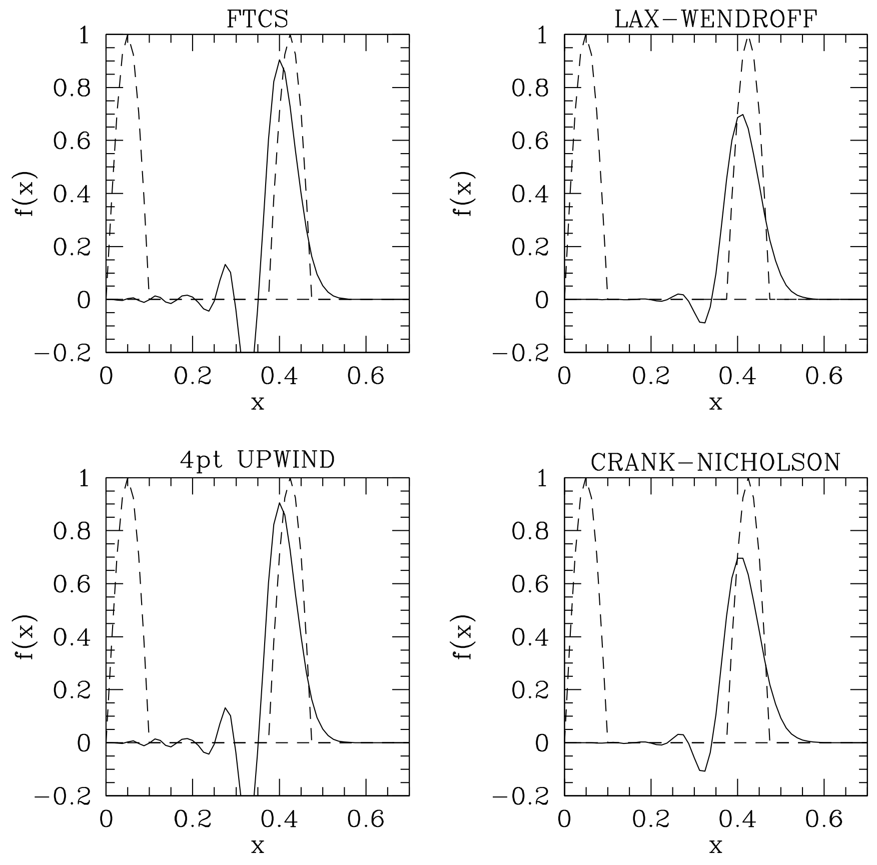


**Figure 5.2:** Propagation of a truncated sine wave using 4 different approximations. The dashed lines represent the exact solution at $t = 0$ and $t = 0.75$ whereas the solid lines show the numerical solutions at $t = 0.75$. In all cases, there are 80 computational cells between 0 and 1 and $c = 0.15$.

## 5.6 The linear transport equation with damping

In the previous chapter we considered various forms of algebraic representation for the transport equation and noticed the difficulty in representing moving waves or bodies across a computational grid. Here, we now combine the transport equation with the diffusion equation by considering the transport equation with damping. To do this, we consider the following equation:

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} - \alpha\frac{\partial^2 f}{\partial x^2} = 0 \tag{5.85}$$

For the moment we will assume that both $v$ and $\alpha$ are constants. From a theoretical point of view, this equation is strictly parabolic. However, if we look at the order of magnitude of the different terms, we note

$$v\frac{\partial f}{\partial x} \approx v\frac{f}{L} \tag{5.86}$$

whereas

$$\alpha\frac{\partial^2 f}{\partial x^2} \approx \alpha\frac{f}{L^2} \tag{5.87}$$

where $L$ is a typical length scale. Taking the ratio between the two, we obtain

$$\frac{v\dfrac{\partial f}{\partial x}}{\alpha\dfrac{\partial^2 f}{\partial x^2}} = \frac{vL}{\alpha} = R_e \tag{5.88}$$

Thus, for large Reynolds number $R_e$ the convective term dominates and the equation becomes hyperbolic. On the other hand, for low Reynolds numbers, diffusion dominates and the equation becomes parabolic in nature. We have seen in previous sections the behavior of both types of equations separately, the purpose of this section is to examine what happens when both are involved in the same equation.

### 5.6.1 The forward time centered space (FTCS) approximation

The discretized version in the FTCS approximation of Eq. 5.84 writes

$$\frac{f_k^{n+1} - f_k^n}{\Delta t} + v\frac{f_{k+1}^n - f_{k-1}^n}{2\Delta x} - \alpha\frac{f_{k-1}^n + f_{k+1}^n - 2f_k^n}{\Delta x^2} = 0 \tag{5.89}$$

which we write in algorithmic form

$$f_k^{n+1} = (s + \frac{1}{2}c)f_{k-1}^n + (1-2s)f_k^n + (s - \frac{1}{2}c)f_{k+1}^n \tag{5.90}$$

with the usual expressions $s = \alpha\Delta t/\Delta x^2$ and $c = v\Delta t/\Delta x$. A stability analysis yields the following amplification factor

$$G = (1-2s) + 2s\cos\varphi - ic\sin\varphi \tag{5.91}$$

This expression can be recognized as the equation of an ellipse centered at $(1-2s)$ on the real axis and of semi-major axis $2s$ and $c$ as illustrated in Fig. 5.3
Clearly, for stability necessary conditions are that $2s \leq 1$ and $c \leq 1$ but these are not sufficient. A detailed analysis is necessary to determine the maximum of $G$ for all $\varphi$. Doing so, one finds that

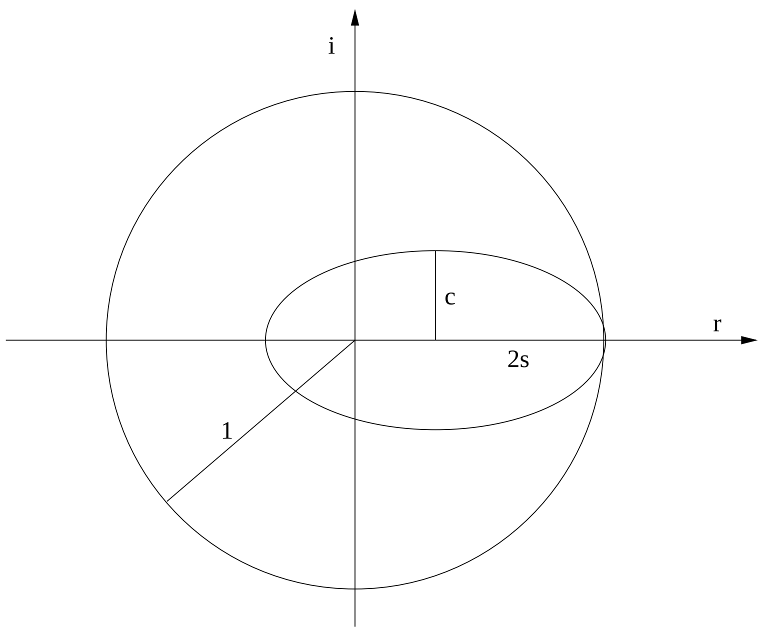


**Figure 5.3:** Stability of the FTCS scheme for the 1D transport equation with damping

$$R_{cell} = \frac{v\Delta x}{\alpha} \leq 2 \tag{5.92}$$

In the next step we want to look at the truncation error. After a Taylor expansion of all terms at $(k, n)$ and replacing all time derivatives by spatial derivatives using the original equation, we obtain

$$\frac{\partial f}{\partial t}+v\frac{\partial f}{\partial x}-(\alpha-\alpha^{'})\frac{\partial^2 f}{\partial x^2}-\left(\alpha v\Delta t+\frac{1}{3}v^3\Delta t^2-\frac{1}{6}v\Delta x^2\right)\frac{\partial^3 f}{\partial x^3}+(\ldots)\frac{\partial^4 f}{\partial x^4}+\ldots=0 \tag{5.93}$$

with $\alpha^{'} = \frac{1}{2}v^2\Delta t$. We see that this first order scheme has a first order dissipation error associated with it. In addition, we must also have that $\alpha^{'} \ll \alpha$ if we want the solution obtained to have something to do with the real solution. This condition implies that we also must have $\Delta t \ll 2\alpha/v^2$. This condition confirms the fact that this scheme is useless in the case $\alpha = 0$ that is for the transport equation without damping.
Finally, recalling the definition of $c$ and $s$, we can combine both and the inequality above to obtain $\Delta t = c^2\alpha/sv^2 \ll 2\alpha/v^2$ which implies that $c^2 \ll 2s$ or $R_{cell} \ll 2/c$. This shows that the stability condition derived above (Eq. 5.92) does not imply accuracy!

### 5.6.2 The upwind approximation

For $v > 0$, the upwind approximation for the transport equation with damping reads

$$f_k^{n+1} = (s+c)f_{k-1}^n + (1-2s+c)f_k^n + sf_{k+1}^n \tag{5.94}$$

A Taylor expansion at $(k,n)$ yields the truncation error

$$E_k^n = -\frac{1}{2}v\Delta x(1-c)\frac{\partial^2 f}{\partial x^2} - \left(c\alpha\Delta x - \frac{1}{6}v\Delta x^2\left(1-3c+2c^2\right)\right)\frac{\partial^3 f}{\partial x^3} \tag{5.95}$$

which shows that the upwind scheme is $O(\Delta x, \Delta t)$ accurate. It introduces a first order dissipation $\alpha^{'} = -\frac{1}{2}v\Delta x(1-c)$ which means that for accurate solution ($\alpha^{'} \ll \alpha$) we must have $R_{cell} \ll 2/(1-c)$. As for stability, we obtain an amplification factor

$$G = 1-(2s+c)(1-\cos\varphi) - ic\sin\varphi \tag{5.96}$$

for which the condition $GG^* \leq 1$ translate into the stability requirement $c+2s \leq 1$. Note that for the diffusion equation alone, the requirement was $s \leq \frac{1}{2}$ while when combined with a convection term we need $s \leq (1-c)/s$ which can be significantly smaller! In terms of timestep, stability demands a maximum timestep size of

$$\Delta t \leq \frac{0.5\Delta x^2}{1+\frac{1}{2}R_{cell}}\frac{1}{\alpha} \tag{5.97}$$

### 5.6.3 The Crank-Nicholson approximation

In the case of a pure diffusion equation we have seen that all explicit schemes suffer from a limitation of the timestep (which can be quite severe) for stability reasons. Implicict schemes such as the Crank-Nicholson scheme do not suffer from this problem and therefore are quite efficient for solving diffusion problems. In this section, we would like to analyze whether this advantage remains true when both transport and diffusion are combined.
We write the Crank-Nicholson algorithm as

$$\frac{f_k^{n+1}-f_k^n}{\Delta t} + \frac{1}{2}v\left(L_x f_k^{n+1} + L_x f_k^n\right) - \frac{1}{2}\alpha\left(L_{xx}f_k^{n+1} + L_{xx}f_k^n\right) \tag{5.98}$$

with $L_x$ and $L_{xx}$ the first and second order spatial derivative operator already encountered. As usual, this expression cannot be solved explicitly for the new time and we obtain a system of equations given by

$$-(s+\frac{1}{2}c)f_{k-1}^{n+1} + 2(1+s)f_k^{n+1} - (s-\frac{1}{2}c)f_{k+1}^{n+1} = (s+\frac{1}{2}c)f_{k-1}^n + 2(1-s)f_k^n + (s-\frac{1}{2}c)f_{k+1}^n \tag{5.99}$$

A Taylor expansion at $(k, n)$ shows that this expression is $O(\Delta t^2, \Delta x^2)$ accurate. A stability analysis yields the following amplification factor

$$G = \frac{1 - s(1-\cos\varphi) - \frac{1}{2} ic \sin\varphi}{1 + s(1-\cos\varphi) + \frac{1}{2} ic \sin\varphi} \tag{5.100}$$

It is easy to check that $GG^* \leq 1 \ \forall \varphi$. The Crank-Nicholson scheme does not suffer from timestep restrictions due to stability requirements. One could be tempted to say that because the Crank-Nicholson approach never suffers from timestep restrictions, this scheme should always be preferred. This is not true in general. First, implicit schemes such as this one require considerable more computer resources and make for much more complicated procedures with matrices to invert. Second, while there are no stability requirements on the timestep, there are often accuracy conditions that in practice limit its size. To see this, we recall that the system of equations given by Eq. 5.99 can be put in a tridiagonal matrix form

$$G = \begin{pmatrix} b & c & 0 & 0 & \dots & 0 \\ a & b & c & 0 & \dots & 0 \\ 0 & a & b & c & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \dots & b \end{pmatrix} \tag{5.101}$$

One can show that such a matrix has eigenvalues given by

$$\lambda_j = b + 2\sqrt{ac} \cos\left(\frac{j\pi}{J-1}\right) \ , \quad j = 1, 2, 3, \dots, J-2 \tag{5.102}$$

From this expression we see that if $ac < 0$, the eigenvalues will be imaginary. In such a case, the solutions will have an oscillatory behavior. Therefore, in order to avoid oscillations in the solutions, we request that $ac \geq 0$. In the case of the Crank-Nicholson scheme, this implies that

$$(s + \frac{1}{2}c)(s - \frac{1}{2}c) = s^2 - \frac{1}{4}c^2 \geq 0 \tag{5.103}$$

Since $c/s = R_{cell}$, this condition also implies that $R_{cell} \leq 2$. Hence, the Crank-Nicholson scheme has also limitations not for stability but for accuracy reasons.

## 5.7 Non-linear equations

So far we have only considered linear equations which allowed us to use a number of mathematical tools to study exactly the properties of the algebraic representations

chosen. Unfortunately, the situation is much more difficult for non-linear equations as these tools become unpractical. To make matter worse, most of the equations we actually want to solved in practice are non-linear!
To provide a short glimpse into the real world, let us consider two prototypes non-linear equations only. We begin with the so-called *inviscid Burger* equation

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} = 0 \tag{5.104}$$

and the *viscous Burger* equation

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} - \alpha\frac{\partial^2 v}{\partial x^2} = 0 \tag{5.105}$$

The non-linearities clearly enter through the second term which is called advection in the case of fluid dynamics. Notice that we could define a quantity $F$ by $F = \frac{1}{2}v^2$ and rewrite the equations above

$$\frac{\partial v}{\partial t} + \frac{\partial F}{\partial x} = 0 \quad \text{or} \quad \frac{\partial v}{\partial t} + \frac{\partial F}{\partial x} - \alpha\frac{\partial^2 v}{\partial x^2} = 0 \tag{5.106}$$

which is generally called the conservative form for reasons we shall see later.

## 5.7.1 The FTCS approximation

The FTCS approximation applied to the inviscid Burger equation is trivial:

$$\frac{v_k^{n+1} - v_k^n}{\Delta t} + \frac{F_{k+1}^n - F_{k-1}^n}{2\Delta x} = 0 \tag{5.107}$$

Once $v_k^{n+1}$ is computed, the corresponding $F_k^{n+1} = \frac{1}{2}v_k^{n+1}$ can be computed and the scheme can proceed. A detailed stability analysis is impossible due to the non-linear nature of the equation. In practice, we would use the stability criterion for the linearized equation as a necessary condition and proceed by trial and error until the stability condition on $dt$ is determined experimentally.

## 5.7.2 The Lax-Wendroff approximation

To derive the Lax-Wendroff scheme for the inviscid Burger equation, we replace

$$\frac{\partial v}{\partial t} = \frac{v_k^{n+1} - v_k^n}{\Delta t} \tag{5.108}$$

by

$$\frac{\partial v}{\partial t} = \frac{v_k^{n+1} - v_k^n}{\Delta t} - \frac{1}{2}\Delta t\frac{\partial^2 v}{\partial t^2} \tag{5.109}$$

From the original equation, we have

$$\frac{\partial^2 v}{\partial t^2} = -\frac{\partial}{\partial x}\left(\frac{\partial F}{\partial t}\right) = -\frac{\partial}{\partial x}\left(\frac{\partial F}{\partial v}\frac{\partial v}{\partial t}\right) = \frac{\partial}{\partial x}\left(A\frac{\partial F}{\partial x}\right) \tag{5.110}$$

where $A = \partial F/\partial v$ is the Jacobian. The original equation is therefore written

$$\frac{v_k^{n+1} - v_k^n}{\Delta t} - \frac{1}{2}\Delta t\frac{\partial}{\partial x}\left(A\frac{\partial F}{\partial x}\right) \tag{5.111}$$

Approximating

$$\frac{\partial F}{\partial x} = \frac{F_{k+1}^n - F_{k-1}^n}{2\Delta x} \tag{5.112}$$

and

$$\frac{\partial}{\partial x}\left(A\frac{\partial F}{\partial x}\right) = \frac{A_{k+\frac{1}{2}}(F_{k+1}^n - F_k^n) - A_{k-\frac{1}{2}}(F_k^n - F_{k-1}^n)}{\Delta x^2} \tag{5.113}$$

we obtain our final expression

$$v_k^{n+1} = v_k^n - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(F_{k+1}^n - F_{k-1}^n\right) + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2\left(A_{k+\frac{1}{2}}(F_{k+1}^n - F_k^n) - A_{k-\frac{1}{2}}(F_k^n - F_{k-1}^n)\right) \tag{5.114}$$

which is accurate $O(\Delta t^2, \Delta x^2)$. In this example $A$ is in fact the Jacobian which we define at midpoint by $A_{k+\frac{1}{2}} = \frac{1}{2}(v_{k+1} + v_k)$. As it appears, this scheme is not so easy to use especially since for systems of equations the Jacobian is a matrix which we must then invert.
A more economical method is to transform this scheme in a two step scheme. The first step is written

$$v_{k+\frac{1}{2}}^* = \frac{1}{2}(v_{k+1}^n + v_k^n) - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(F_{k+1}^n - F_k^n\right) \tag{5.115}$$

Followed by a second step

$$v_k^{n+1} = v_k^n - \frac{\Delta t}{\Delta x}\left(F_{k+\frac{1}{2}}^* - F_{k-\frac{1}{2}}^*\right) \tag{5.116}$$

It can be shown that this approximation is also $O(\Delta t^2, \Delta x^2)$ accurate. In fact, both schemes are equivalent but the second one is much easier to implement and to use.

### 5.7.3 The Crank-Nicholson approximation

Using the Crank-Nicholson approach, we write the inviscid Burger equation

$$\frac{v_k^{n+1} - v_k^n}{\Delta t} = -\frac{1}{2} L_x \left( F_k^n + F_k^{n+1} \right) \tag{5.117}$$

A straight forward approach would result in a non-tridiagonal matrix because of the presence of the non-linear terms. Hence, a full matrix inversion would be required which would be computationally prohibitive. However, we can circumvent this difficulty by noting that

$$F_k^{n+1} = F_k^n + \Delta t \left( \frac{\partial F}{\partial t} \right)_k^n + O(\Delta t^2) = F_k^n + \Delta t \left( \frac{\partial F}{\partial v} \frac{\partial v}{\partial t} \right)_k^n + O(\Delta t^2) \tag{5.118}$$

We define again $A = \partial F / \partial v$ and write Eq. 5.117 as

$$\frac{v_k^{n+1} - v_k^n}{\Delta t} = -\frac{1}{2} L_x \left( 2F_k^n + A_k^n (v_k^{n+1} - v_k^n) \right) \tag{5.119}$$

The system of equation described by this expression can be put in a tridiagonal form of the type

$$a_k^n v_{k-1}^{n+1} + b_k^n v_k^{n+1} + c_k^n v_{k+1}^{n+1} = d_k^n \tag{5.120}$$

with

$$\begin{aligned} a_k^n &= -\frac{\Delta t}{4\Delta x} v_{k-1}^n \\ b_k^n &= 1 \\ c_k^n &= -\frac{\Delta t}{4\Delta x} v_{k+1}^n \\ d_k^n &= v_k^n \end{aligned}$$

Note that these matrix coefficients have to be re-computed at all timesteps.

## 5.8 Numerical diffusion and dispersion

In the previous chapter we have seen the first hints that the algebraic equations solved are *not* exactly equivalent to the original differential equations but that they contain higher order terms whose properties can modify the solution. A good numerical scheme is therefore a scheme that manages to keep these additional, unwanted, terms as small as possible.

In this chapter, we are interested in making the influence of these higher order terms visible in order to understand their effect and to be able to identify later potential

numerical troubles in the solutions obtained. For this we begin with two differential equations which properties we are going to analyze in more details.
Let us consider the one-dimensional transport equation including a viscous term

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} - \alpha\frac{\partial^2 f}{\partial x^2} = 0 \tag{5.121}$$

and the linear Korteweg de Vries equation

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} + \beta\frac{\partial^3 f}{\partial x^3} = 0 \tag{5.122}$$

A solution to these equation is the plane wave solution which we write

$$f(x,t) = \Re\left(f_0 e^{-p(m)t} e^{im(x-q(m)t)}\right) \tag{5.123}$$

where $p(m)$ gives the the damping of the wave and $q(m)$ its speed. The wave number is given by $m = 2\pi/\lambda$.
Introducing this solution in Eq. 5.121 yields $p(m) = \alpha m^2$ and $q(m) = v$ while introducing it in Eq. 5.122 yields $p(m) = 0$ and $q(m) = v - \beta m^2$. We conclude the following

- Second order spatial derivatives (and we shall see all even spatial derivatives) are introducing damping while third order spatial derivatives (and all odd derivatives) are introducing dispersion.
- The damping is proportional to $m^2$. Hence, short waves are attenuated much more rapidly than longer ones.
- The difference in wave speed is also strongly dependent on wave number. If $\beta > 0$, short wavelength propagate slower.

The dispersive and dissipative properties of a scheme are determined by the higher order derivatives: even order derivatives implying dissipation and odd order derivatives dispersion. Note that the presence of these terms is due to the discretization process, they are therefore inherent to any numerical scheme.
For linear equations, the dissipative and dispersive properties can be examined more rigorously using a Fourier representation of the function

$$f(x,t) = \sum_{-\infty}^{\infty} f_m e^{-p(m)t} e^{im(x-q(m)t)} \tag{5.124}$$

as each Fourier component can be studied separately. During a time $\Delta t$ the $m$th component will have decreased by $e^{-p(m)\Delta t}$ and will have advanced by a distance $q(m)\Delta t$. Inserting the Fourier components into the algebraic equations and comparing with these

exact solutions allows the determination of the dissipative and dispersive properties of the scheme.
Let us consider again our upwind approximation in the case $v \geq 0$, which we write

$$f_k^{n+1} = (1-c) f_k^n + c f_{k-1}^n \tag{5.125}$$

The amplitude ratio of the function at two consecutive times is found

$$G = \frac{f_k^{n+1}}{f_k^n} = (1-c) + c \frac{f_{k-1}^n}{f_k^n} \tag{5.126}$$

From the Fourier decomposition, we must have

$$\frac{f_{k-1}^n}{f_k^n} = e^{-im\Delta x} = \cos(m\Delta x) - i\sin(m\Delta x) \tag{5.127}$$

Inserting this expression into Eq. 5.126 yields

$$G = 1 - c\,(1-\cos(m\Delta x)) - ic\sin(m\Delta x) \tag{5.128}$$

which is the same expression we already obtained for the amplification in our stability analysis. From the Fourier decomposition, the value of the amplitude ratio is given by

$$\frac{f_k^{n+1}}{f_k^n} = e^{-(p(m)+imq(m))\Delta t} \tag{5.129}$$

Equating the two amplitude ratio, we have for the $m$ Fourier component

$$|G_m| = e^{-p(m)\Delta t} = (1 - c(1-\cos(m\Delta x)))^2 + c^2\sin^2(m\Delta x) \tag{5.130}$$

which yields after some elementary simplifications

$$|G_m| = 1 - 4\left(c - c^2\right)\sin^2\left(\frac{m\Delta x}{2}\right) = e^{-p(m)\Delta t} \tag{5.131}$$

For stable solution, we must have $p(m) \geq 0$ which means that we must ensure that

$$0 \leq 4\left(c - c^2\right)\sin^2\left(\frac{m\Delta x}{2}\right) \leq 1 \tag{5.132}$$

or in other words $c \leq 1$. Hence, we recover our stability condition and demonstrate that stability is indeed associate with dissipation. The dissipation will be largest for $m\Delta x/2 = \pi/2$ or $m\Delta x = \pi$, that is for the shortest wavelength.
We can also look at the phase of the amplitude ratio which is directly connected with the phase error. We have for the phase of the $m$ Fourier component

$$\phi_m = -mq(m)\Delta t = \tan^{-1}\frac{-c\sin(m\Delta x)}{1 - c(1-\cos(m\Delta x)} \tag{5.133}$$

Now recalling that $\Delta t = c\Delta x/v$ and $q_{theo} = v$, we obtain

$$\frac{q(m)}{q_{theo}} = -\frac{1}{cm\Delta x}\tan^{-1}\frac{-c\sin(m\Delta x)}{1 - c(1 - \cos(m\Delta x))} \tag{5.134}$$

This relation is shown graphically on Fig. 5.4.

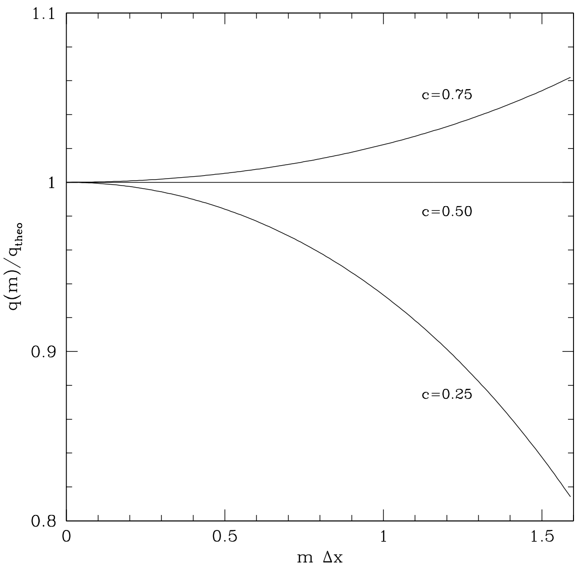


**Figure 5.4:** Phase error in the upwind scheme. Plotted is the ratio of the signal speed as computed by the upwind difference scheme to the exact speed as a function of $m\Delta x$ for three different Courant number $c$

We note that errors are small for small $m\Delta x$ that is for long wavelength. For $c < 0.5$, the numerically computed speed is smaller than it should be while for $c > 0.5$ the numerical solution moves faster. For $c = 0.5$ there is no speed error.

## 5.9 References

- Benz, W., in *Numerical Modeling of Non Radial Stellar Pulsation: Problems and Prospects'*, Ed. J.R. Buchler (Dordrecht: Kluwer Academic Publishers, 1989)
- Bowers, R.L., Wilson, J.R., Numerical Modeling in Applied Physics and Astrophysics" (Boston: Jones and Bartlett, 1990)
- Fletcher, C.A.J., Computational Techniques for Fluid Dynamics", (Berlin: Springer-Verlag, 1988)

- Richtmyer, R.D., Morton K.W., (1967), Difference Methods for Initial-Value Problems (New York: Interscience Publishers, Inc., 1967)