International Master on Turbulence

Lille - Poitiers

Numerical Methods for Simulation of Fluid Mechanics

J.-P. Laval

November 29, 2012

Contents

1 Introduction to numerical methods

1.1 What is a model ?

The objective of modeling is to replace a complex system into an simple object or a simple operator replacing most of the aspects and of the behaviors of the original system (reduced models, thinking models, mathematical models, numerical models, ...) The real systems are usually too complex to be solved directly. The complexity can come from the physics of the problem, the size of the problem, the huge number of non-linear parameters in interaction. A good example is the meteorology. The physic is rather complex (geometry, size of the problem, chemical reaction, physic of clouds, highly non-linear problem) with a huge number of degrees of freedom ($> 10^{26}$).

One solution is to make use of a series of experiments to Analize all the parameters of the system and to deduce a model. However, this solution is not always accessible because of the cost of the experiments (flying tests for aircraft, expensive measuring tools, ...) or too dangerous (nuclear environment, spatial environment, ...). One more constrain may be the size of the problem which enable the possibility to do experiments (infinitely small, climatology, astrophysics, ...). An other option is to built a mathematical model to represent the physical phenomena. These models usually use systems of non-linear Partial Differential Equations (PDE) which are impossible to solve in order to obtain exact solutions (i.e. Navier Stokes equations). Therefore, one must solve the numerical problem by transforming the continuous equations of the physic into a discrete problem in a limited calculation domain. In some case, solving the numerical problem is the only possibility. In some other case, the numerical analysis can be conducted in parallel to some experiments. Numerical simulation encounter many different limitations which are different from the experiments (turbulent flows, complex chemical reactions, multi-phase flows). However many difficulties linked to the experimental set-up do not hold any more with numerical simulation (opaque flows, moving boundaries, ...).

1.1.1 From the modeling to the numerical simulation

The different steps to model a complex system are the followings:

- definition of a mathematical model to represent the physic of the system (transformation into equations)
- definition and meshing of the computational domain
- discretisation of the equations of the physic
- solving of discretised equations (usually a system of equations)
- transcription of the discretised equation into computing environment
- numerical simulation
- validation and exploitation of the results

1.1.2 Stability concept

One can distinguish between three kind of stability:

- stability of the physical problem
- stability of the mathematical problem
- numerical stability of the computational method.

Stability of the physical problem

A system is said "Chaotic" if a small variation of the initial conditions leads to an non deterministic variation of the results. This concept of Chaos is linked to the physics of the problem but is independent of the mathematical formulation and of the numerical method used to solve this mathematical problem. Many problem are chaotic. The most significant ones are the turbulent flows. A small variation of the initial conditions leads to completely different results (uncorrelated solutions) but which may have the same statistical behaviors (i.e. phase differences).

Stability of a mathematical model

A problem is ill-conditioned if a small variation of the parameters leads to a large variation of the results. This concept of conditioning linked to the mathematical problem is independent to the numerical method used to solve the problem. Solving a physical problem which is not chaotic requires a mathematical model with the best possible conditioning in order to reduce the possible errors.

Stability of a numerical method

A numerical method is said to be unstable if it can easily propagates the numerical discretisation errors or round-off errors. A problem may be well conditioned with a unstable numerical method to solve it. In this case, it is imperative to change the numerical method. However, fs the original mathematical problem is hill-conditioned, no numerical method will be able to solve it. In this case, one must look for a different mathematical formulation to solve the same problem (if the original physical problem is stable).

1.1.3 General comments on CFD

Numerical simulations, when the problem can be easily discretised, usually bring a new perception of the solution. However, as we are dealing with a solution of the discretised problem, one must be very careful when analyzing the results. First, the solution of the numerical problem is contaminated by errors of accuracy which are linked to the quality of the discretisation. Discretisation errors can be reduced by using more accurate interpolations or approximations or by applying the approximation to smaller regions, but this usually increases the time and cost of obtaining the solution. Compromise is usually needed.
Compromise is also needed in solving the discretised equations. Direct solvers are seldom used, because they are too costly. Iterative methods are more common but the errors due to stopping the iterations are to be taken into account. The accuracy of the results is not the only parameter

of a simulation and has to be related to the cost. CFD is usually a small ingredient of a larger problem (aircraft design). Its accuracy, and consequently its cost, has to be adapted to the cost of the complete problem. One of the benefit of numerical simulation is the richness of the results. The solution is usually known at any discretised nodes and may be known for any time. This is very useful to investigate finely the solution and to draw any plots useful to analyze the solution. However, the user must remember that the solution is the solution of a discrete equation system which may be soiled by accuracy error or any possible error due to the numerical method.

1.2 Concept of Consistency, Stability, Convergence and Accuracy

The resolution of partial derivative equations using discretised equations should have certain properties. the three major properties are: consistency, stability and convergence. These three properties allow to link the exact solution of the continuous equations to the exact solution of the discretised equation and to the numerical solution obtained by the numerical method.

1.2.1 Consistency

The consistency is the property which ensure that the exact solution of the discretised equations tend to the exact solution of the continuous equations when the time and space discretisations tend to zero ($\Delta t \to 0, \Delta x \to 0$)

The discretisation should become exact as the grid spacing tends to zero. The difference between the discretised equation and the exact one is called the *truncation error*. So, for a method to be consistent, the truncation error must become zero when the mesh spacing tends to zero. Some discretisation methods lead to truncation errors which are functions of the ratio Δx to Δt or vice versa. In such a case the consistency requirement is only conditionally fulfilled: Δx and Δt must be reduced in a way that allows the appropriate ratio to go to zero. Even if the approximations are consistent, it does not necessarily mean that the solution of the differential equation will become the exact solution in the limit of small step size. For this to happen, the solution method has to be stable.

1.2.2 Stability (of the numerical method)

The stability is the property which ensure that the difference between the numerical solution and the exact solution remains bounded.
In other words, a numerical solution method is said to be stable if it does not magnify the errors that appears in the course of numerical solution process. Stability can be difficult to investigate, especially when boundary conditions and non-linearity are present. For this reason, it is common to investigate the stability of a method for linear problems with constant coefficients and without boundaries. Experiences show that the results obtained from the linear stability analysis can often be used for more complex problems, but there are notable exception. Many solution schemes require that the time step be smaller than a certain limit or that under-relaxation be used.

1.2.3 Convergence

The convergence is the property which ensure that the numerical solution tends to the (or one) exact solution of the continuous equation when the grid spacing tends to zero.

For linear initial value problems, the *Lax equivalence theorem* [13] states that *"given a properly posed linear initial value problem and a finite differences approximation to it that satisfies the consistency condition, stability is a necessary and sufficient condition for convergence"*. For non-linear problems which are strongly influence by boundary conditions, the stability and convergence of a method are difficult to demonstrate. Therefore, convergence is usually checked a-posteriori, repeating the calculation on a series of successively refined grids. If the method is stable and if all approximations used in the discretisation process are consistent, we usually find that the solution does converge to a grid-independent solution.

1.2.4 Conservation

Since the equations to be solved are conservation laws, the numerical scheme should also, on both local and global basis, respect these laws. This means that, at steady state and in the absence of sources, the amount of a conserved quantity leaving a closed volume is equal to the amount entering that volume. This is an important property of the solution method, since it imposes a constraint on the solution error. In Navier-Stokes equations, if the conservation of mass, momentum and energy are insured, the error can only be improperly distributed over the solution domain. Non-conservative schemes can produce artificial sources and sinks, changing the balance locally and globally. However, non-conservative schemes can be consistent and stable and therefore lead to correct solutions in the limit of very fine grids.

1.2.5 Accuracy

Numerical solutions of fluid flow and heat transfer problems are only approximated solutions. Numerical solutions often include three kind of systematic errors:

- *Modeling errors*, which are defined as the difference between the actual flow and the exact solution of the mathematical model

- *Discretisation error*, defined as the difference between the exact solution of the Conservative equations and the exact solution of the algebraic system of equation obtained by discretising these equations

- *Iteration errors*, defined as the difference between the iterative and exact solutions of the algebraic equations systems (for iterative methods only)

Modelling errors can be due to a simplification of the geometry of the solution domain or the boundary conditions. It can also be due to an assumption made in deriving the transport equations or a simplified model for chemical reactions or turbulence models (if not DNS). Discretisation approximations introduce errors which decrease with the grid refinement, and the order of the approximation is a measure of accuracy. However, on a given grid, methods of the same order may lead to discretisation errors which differ by as much as one order of magnitude. This is because the order tells us about the rate at which the error decreases as the mesh spacing is reduced. However, it gives no information about the error on a single grid.

1.2.6 Well posed problem

Considering the following problem:

$$F(x, d) = 0 \tag{1}$$

where F is a functional relation between x and d where d is an ensemble of parameters. The Eq. (1) is a discrete problem if F and d are given and x is the unknown, an inverse problem if F and x are given and d is unknown, and a identification problem if x and d are given and the functional relation F is unknown.

The problem (1) is said *well conditioned* or *well posed* if the solution x exists, is unique, and continuously depends of the ensemble of parameters d. A problem which do not satisfy the above property is said *ill conditioned* and must be regularized before to solve it, this means that it has to be transformed into a *well conditioned* problem. It is not conceivable to count on the numerical method to solve the intrinsic pathology of a ill conditioned problem.

1.2.7 Mathematical classification of flows

Quasi-linear second order partial differential equations in two independent variables can be divided into three types: hyperbolic, parabolic and elliptic. This distinction is based on the nature of the *characteristics*, curves along which information about the solution is carried. Every equation of this type has two sets of characteristics.

- In the *hyperbolic* case, the characteristics are real and distinct, this means that the information propagates at finite speed in two set of directions. The two set of characteristics require two initial conditions (at the initial point of each of them).

- In *parabolic* equations, the characteristics degenerate to a single real set. Consequently, only one initial condition is normally required

- In the *elliptic* case, the characteristics are imaginary or complex so there are no special directions of information propagation. Indeed, information travels essentially equally well in all directions.

Incompressible flows

The mathematical properties of a model is directly connected to the physical properties of the flow. Any flow configuration is the outcome of a balance between the effect of convective fluxes, diffusive fluxes and the external or internal forces. The various approximation levels can be considered as resulting from a priori estimates of the relative influence and balance between the contribution of these various fluxes and forces. From a general point of view, the diffusive fluxes appear through second order derivative terms and as a tendency to smooth out the gradients. The convective fluxes appear as first-order derivative terms and express the transport properties

of the flow system. Therefore, each of these contribution will influence the mathematical nature of the equations, particularly the competition between the elliptic, parabolic and hyperbolic of the system of equations.

If we consider the x-component of the incompressible Navier Stokes equations

$$\rho\frac{\partial u}{\partial t} + \rho(\vec{v}\cdot\vec{\nabla})u = -\frac{\partial p}{\partial x} + \mu\Delta u \tag{2}$$

If all variable are non-dimensionalised using a reference length L, a time scale T, a velocity scale V for the velocity and ρV^2 for the pressure, one obtains the dimensionless equation (keeping the same notation for the variables)

$$\frac{VT}{L}\frac{\partial u}{\partial t} + \rho(\vec{v}\cdot\vec{\nabla})u = -\frac{\partial p}{\partial x} + \frac{1}{Re}\Delta u \tag{3}$$

where Re is the Reynolds number defined as

$$Re = \frac{\rho V L}{\mu} \tag{4}$$

For very small values of Reynolds number, that is for strongly viscous dominated flows, the convection term can be neglected with respect to the viscous term and we obtain the Stokes equation

$$\frac{-V^2T}{\nu}\frac{\partial u}{\partial t} + \Delta u = Re\frac{\partial p}{\partial x} \tag{5}$$

This equation is purely of an elliptic type in the steady state case and for a fixed pressure gradient, but parabolic in the unsteady case. The Laplace equation (or Poisson equation) can be considered as the standard form of a elliptic equation describing an isotropic diffusion in all space directions.

On the other hand, for large values of Re (and outside the boundary layer), the viscous term as a negligible influence and the flow is dominated by non-viscous transport terms describing the effect of the convective fluxes. Hence, the equation reduces to the Euler equation

$$\frac{\partial u}{\partial t} + (\vec{v}\cdot\vec{\nabla})u = -\frac{1}{\rho}\frac{\partial p}{\partial x} \tag{6}$$

which in a one-dimensional space takes the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -\frac{1}{\rho}\frac{\partial p}{\partial x} \tag{7}$$

which is a basic hyperbolic equation in space and time describing a propagation phenomena.

The numerical discretisation and the numerical methods will have to take into account the character of the equations. The diffusion property is essentially independent of the flow direction acting in all directions while the propagation property is essentially direction dominated and acts in a specific region of space defined by the wave propagation directions.

Between this two extremes, the parabolic type of equation (in space and time) for a time-dependent, diffusion-dominated system

$$\frac{V^2 T}{\nu}\frac{\partial u}{\partial t} = \Delta u \tag{8}$$

represents an intermediate situation between hyperbolic and elliptic. This equation describes a diffusion effect propagating in all space directions but damped in time.

Hence the system of time-dependent Navier-Stokes equations is essentially parabolic in space and time, although the continuity equation has an hyperbolic structure. Therefore they are considered as parabolic-hyperbolic. For the same reason, the steady state form of the Navier-Stokes equations leads to elliptic-hyperbolic properties.

Compressible flow

If we first consider the unsteady, inviscid compressible flow. A compressible fluid can support sound and waves and it is not surprising that these equations have essentially hyperbolic character. Most of the methods to solve them are based on the idea that the equations are hyperbolic, and, given sufficient care, they work quite well.

For inviscid steady compressible flow, the character depends on the speed of the flow. If the flow is supersonic, the equation are hyperbolic. However, subsonic flows are essentially elliptic.

The case of viscous compressible flows is more complicated. Their character is a mixture of the two situations above. They do not fit well into the classification scheme and numerical methods for them are difficult to construct.

1.2.8 Boundary conditions

The choice of the boundary conditions are important in the global behavior of a numerical method used to solve a problem modeled by partial differential equation. Several conditions may be possible for a specific problem, but the accuracy and the stability of the numerical method is directly link to the boundary conditions. The first observation is that one must respect the classification of the equations in order to distribute the boundary conditions.

The following basic example of a second order linear PDE

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = 0 \tag{9}$$

is called Laplace's equation. The corresponding inhomogeneous PDE

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = f(x, y) \tag{10}$$

is called the Poisson equation. These PDEs arise in a variety of mathematical and physical contexts. For instance, the pressure for incompressible fluid flow is modeled by a Poisson equation.

Associated with the two physical interpretations mentioned above are two special types of boundary conditions.

Dirichlet conditions

In a thermal equilibrium problem is seems reasonable to expect the equilibrium temperature distribution of a planar object to be completely determined by the temperature distribution imposed on its boundary. The corresponding mathematical problem would be phrased as follows: Let Ω be a closed region of the plane and let $\partial\Omega$ denote the boundary of Ω , find a function $\Phi(x, y)$ such that

$$\begin{aligned} \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} &= 0, \quad \forall(x, y) \in \Omega \\ \Phi(x, y) &= \Phi_o(x, y), \quad \forall(x, y) \in \partial\Omega \end{aligned} \tag{11}$$

Such a Partial Differential Equation problem is called a Dirichlet problem. The Dirichlet conditions (or first type conditions) are the conditions which specify the value of the solution at the boundary of the domain.

Neumann conditions

Consider the following physical problem: A planar object is surrounded by material capable of transferring heat at a prescribed rate $f(x, y)$; find the equilibrium temperature inside the object. The corresponding mathematical problem would be phrased as follows: Let Ω be a closed region of the plane and let $\partial\Omega$ denote the boundary of Ω, find a function $\Phi(x, y)$ such that

$$\begin{aligned} \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} &= 0, \quad \forall(x, y) \in \Omega \\ \frac{\partial\Phi}{\partial n}(x, y) &= kf(x, y), \quad \forall(x, y) \in \partial\Omega \end{aligned} \tag{12}$$

where $\frac{\partial\Phi}{\partial n}$ is the derivative of Φ in the direction normal to the boundary. Such a PDE is called a Neumann problem. The Neumann conditions specifies the values that the derivative of a solution is to take on the boundary of the domain. If the right hand side of the Neumann conditions is zero, the conditions are called *homogeneous* Neumann boundary conditions and in the more general case with non-zero conditions, they are called *non-homogeneous* Neumann boundary conditions.

For instance, in the case of the resolution of incompressible flow in a confined domain, when solving the Poisson equation for the pressure, non-homogeneous Neumann boundary conditions can be used. In this case the $\partial p/\partial n$ can be computed from the pressure gradient of the momentum equation. However, it is not possible to use Neumann boundary conditions on the entire boundaries of the simulation domain because the pressure would not be completely defined (one can add a constant value).

Robin conditions

The Neumann and Dirichlet boundary conditions are the most used. However, in some situation, it may be of interest to define mixed boundary conditions. The Dirichlet and Neumann

condition can be mixed into a single condition at the same position.

$$a\Phi(x,y) + \frac{\partial \Phi}{\partial n}(x,y) = f(x,y) \tag{13}$$

This type of conditions are called *Robin boundary conditions*. The Robin boundary condition is a general form of the insulating boundary condition for convection-diffusion equations. Here, the convective and diffusive fluxes at the boundary sum to zero:

$$-D\frac{\partial c(0)}{\partial x} + u_x(0)\, c(0) = 0 \tag{14}$$

where D is the diffusive constant, u is the convective velocity at the boundary and c is the concentration.

2 Interpolation methods

In order to evaluate a function, it is usual to have the values of the function only for a finite ensemble of points. In the simplest case of the uni-dimensional function, these points are usually arranged onto a regular 1D grid. If the points where the function needs to be evaluated is located inside a two consecutive points of the original grid, the approximation procedure by a simpler function (usually a polynomial) is called interpolation. However, if the point is located outside the range of the original grid, the procedure is called extrapolation.

The interpolation for a function which is initially evaluated on a given grid is based on the optimization of approximate function by using all the existing points of the original grid. However, two solutions are possible. The first one is to use the full data-set at the same time to define a single interpolation function valid on the whole domain. The second option is to define several interpolation function using only a fraction of the local data-set. We will see that the answer is not straightforward and depends on the regularity of the function to interpolate and to the type of the interpolation function. As a first guess, one would think that the polynomials are the best candidate to interpolate any function. However, we will see that using a polynomial of high order may deteriorate the quality of the approximation.

Given $n+1$ pair of values (x_i, y_i), the interpolation problem leads to find $\Phi = \Phi(x)$ such that $\Phi(x_i) = y_i$ for $i = 0, ..., m$ where y_i are given values. The function Φ is said to interpolate y_i at the x_i nodes. The interpolation is said an *polynomial interpolation* when Φ is a polynomial , or a *trigonometric approximation* when Φ is a trigonometric polynomial or *polynomial interpolation by piece* when Φ is a polynomial by piece. The values y_i can be the values of a function at nodes x_i or experimental data for instance. In the first case, the objective can be to replace the original function by a simpler one and in the second case, the objective is usually to represent the data with a simple function reducing the number of parameters.

2.1 Polynomial interpolation

2.1.1 Lagrange Polynomial

Considering $n + 1$ couples (x_i, y_i). The interpolation problem is to find a polynomial Π_m called an *interpolation polynomial* such as:

$$\Pi_m(x_i) = a_m x_i^m + ... + a_1 x_i + a_0 = y_i, \quad i = 0, ..., n \tag{15}$$

where the x_i are called the *interpolation nodes*. If $n \neq m$ the problem is under or over determined. However, if $n = m$, one can prove that there is a unique polynomial Π_m such that $\Pi_m(x_i) = y_i$, for $i = 0, ..., n$.

Considering the n order polynomial function l_i such as:

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}, \quad i = 0, ..., n. \tag{16}$$

The polynomial $(l_i, i = 0, ..., n)$ are a basis of the ensemble of polynomial of order n. By

decomposing Π_n on this basis, one have:

$$\Pi_n(x) = \sum_{j=0}^{n} b_j l_j(x), \tag{17}$$

and therefore:

$$\Pi_n(x_i) = \sum_{j=0}^{n} b_j l_j(x_i), \quad i = 0, ..., n. \tag{18}$$

As $l_j(x_i) = \delta_{ij}$, on can deduce that $b_i = y_i$. As a consequence, the interpolation polynomial exists and can be written as:

$$\Pi_n(x) = \sum_{i=0}^{n} y_i l_i(x). \tag{19}$$

It is possible to check that

$$\Pi_n(x) = \sum_{i=0}^{n} \frac{\omega_{n+1}(x)}{(x - x_i)\omega'_{n+1}(x)} y_i, \tag{20}$$

where ω_{n+1} is the $n+1$ order *nodal polynomial* defined by:

$$\omega_{n+1} = \prod_{i=0}^{n} (x - x_i). \tag{21}$$

The formula (19) are called *interpolation formula of Lagrange* and the polynomial $l_i(x)$ are the *characteristic polynomial of Lagrange*. The figure 1 shows the characteristic polynomial $l_2(x)$, $l_3(x)$, $l_4(x)$ and $l_5(x)$ for $n = 6$ with homogeneous repartition of the points on the interval $[-1, 1]$. If $y_i = f(x_i)$ for $i = 0, ..., n$, f being a given function, the interpolating polynomial $\Pi_n(x)$ will be noted $\Pi_n f(x)$.

2.1.2 Newton's method

The Lagrange Polynomial form is not the most useful interpolation method. There is a more efficient way to define an interpolation function. This alternative method is based on an iterative method. The objective is the following. Given $n+1$ couples (x_i, y_i), $i = 0, ..., n$, we would like to define Π_n (such as $\Pi_n(x_i) = y_i$ with $i = 0, ..., n$) as the sum of Π_{n-1} (such as $\Pi_{n-1}(x_i) = y_i$ with $i = 0, ..., n-1$) and a polynomial of order n which depends on nodes x_i and an additional unknown coefficient.

$$\Pi_n(x) = \Pi_{n-1}(x) + q_n(x) \tag{22}$$

As $q_n(x_i) = \Pi_n(x_i) - \Pi_{n-1}(x_i) = 0$ for $i = 0, ..., n-1$ we have

$$q_n(x) = a_n(x - x_0)...(x - x_{n-1}) = a_n \omega_n(x) \tag{23}$$

In order to find the coefficient a_n, let's suppose that $y_i = f(x_i), i = 0, ..., n$ where f is a given function (not necessary with an explicit form). As $\Pi_n f(x_n) = f(x_n)$ than we can deduced from Eq. (22) that

$$a_n = \frac{f(x_n) - \Pi_{n-1} f(x_n)}{\omega_n(x_n)} \tag{24}$$

The a_n coefficient is called the n^{th} *Newton's divided difference* and is usually written as:

$$a_n = f[x_0, x_1, ..., x_n] \tag{25}$$

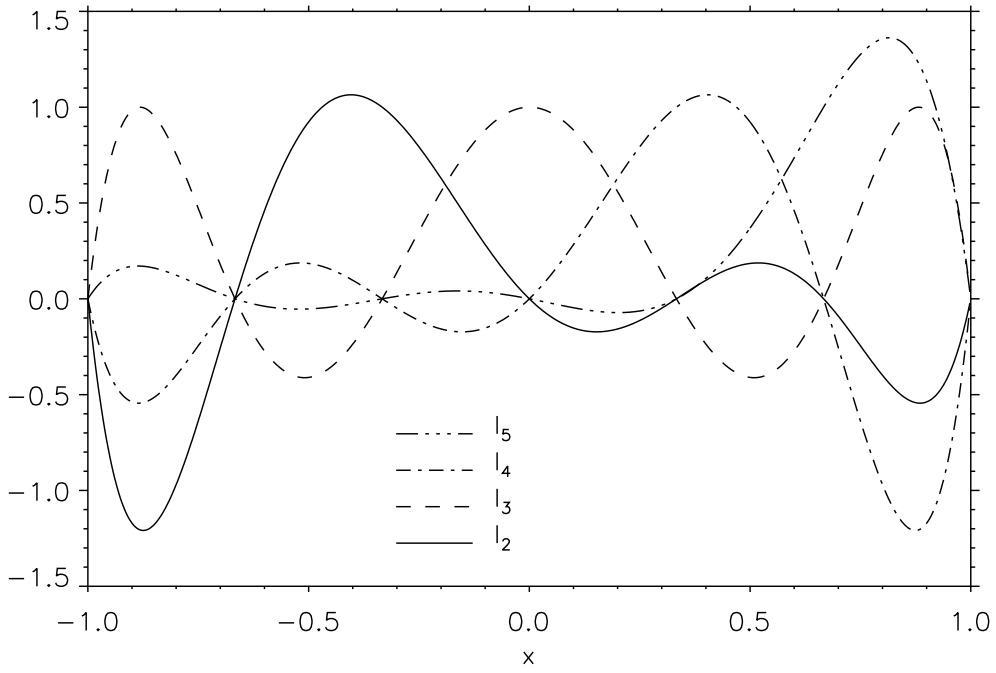

Figure 1: Characteristic Lagrange polynomials $l_2(x)$, $l_3(x)$, $l_4(x)$ and $l_5(x)$ for $n = 6$

The equation (22) becomes

$$\Pi_n(x) = \Pi_{n-1}(x) + \omega_n(x) f[x_0, x_1, ..., x_n]. \tag{26}$$

By using the notation $y_0 = f(x_0)$ and $\omega_0 = 1$ the recurrence formula becomes

$$\Pi_n(x) = \sum_{k=0}^{n} \omega_n(x) f[x_0, x_1, ..., x_k] \tag{27}$$

As a consequence of the unicity of the interpolation polynomial, this interpolation polynomial is identical to the Lagrange polynomial. This recurrence formula is called *Newton's divided difference formula*.

Some properties of the Newton's divided differences

One can noticed that the n^{th} divided differences $f[x_0, x_1, ..., x_k] = a_n$ is the coefficient of x^n in the interpolating polynomial $\Pi_n f$. By extracting this coefficient from the definition of Π_n (eq. (20)) and by identifying this coefficient with the one of the Newton's formula (eq. (26)), we obtain

$$f[x_0, x_1, ..., x_n] = \sum_{i=0}^{n} \frac{f(x_i)}{\omega'(x_i)}. \tag{28}$$

After manipulating this formula, we can obtain a recurrence formula to easily compute the divided differences

$$f[x_0, x_1, ..., x_n] = \frac{f[x_1, x_2, ..., x_n] - f[x_0, x_1, ..., x_{n-1}]}{x_n - x_0}, \quad n \geq 1 \tag{29}$$

The recurrence formula can summarized in a matrix d

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} f(x_0) & & & & \\ f(x_1) & f(x_0,x_1) & & & \\ f(x_2) & f(x_1,x_2) & f(x_0,x_1,x_2) & & \\ \vdots & \vdots & \vdots & \ddots & \\ f(x_n) & f(x_{n-1},x_n) & f(x_{n-2},x_{n-1},x_n) & \dots & f(x_0,...,x_n) \end{bmatrix} \tag{30}$$

The coefficients used for the Newton's formula are in the diagonal of the matrix.

Example: Fill the following table with the Newton's divided differences for the interpolation of the function $f = 1 + sin(3x)$ in the interval [0,2]

x_i	$f(x_i)$	$f(x_i, x_{i-1})$	.....	.....	.....	.....	.....
0	1.0000						
0.2	1.5646						
0.4							
0.8							
1.2							
1.6							
2.0							

(31)

2.1.3 Interpolation Errors

If one want to interpolate a function $f(x)$ at $n+1$ nodes in a closed interval $[a, b]$, what would be the best choice of the nodes ? The question is not so straightforward. What would think that taking the node equally- spaced would be the best choice, but this is not the case. One can illustrated this with an example:.
Let's take the function

$$f(x) = \frac{1}{1+x^2} \tag{32}$$

known as the *Runge function* and let's $\Pi_n f(x)$ being the interpolation function on n equally distributed point in the $[-5, 5]$ domain. Then:

$$\lim_{n\to\infty} \left(\max_{x\in[-5,5]} |\Pi_n f(x) - f(x)| \right) = \infty \tag{33}$$

This behavior is illustrated by Fig. 2 where increasing the number of equally spaced nodes leads to larger errors at the two bounds of the interpolation domain. It turns out that a much better choice is related to the *Chebyshev Polynomials*. For a closed interval of [-1,1], then interpolation nodes are defined by:

$$x_i = cos\left[\left(\frac{2i+1}{2n+2} \right) \pi \right], \quad 0 \le i \le n \tag{34}$$

These nodes can be seen as the projection of the nodes uniformly spaces on a semi-circle on its diameter as shown in Figure 3. The Chebyshev nodes distribution leads to a much better interpolation of the Runge function as can be seen in the example Fig. 4 for 15 and 25 nodes.

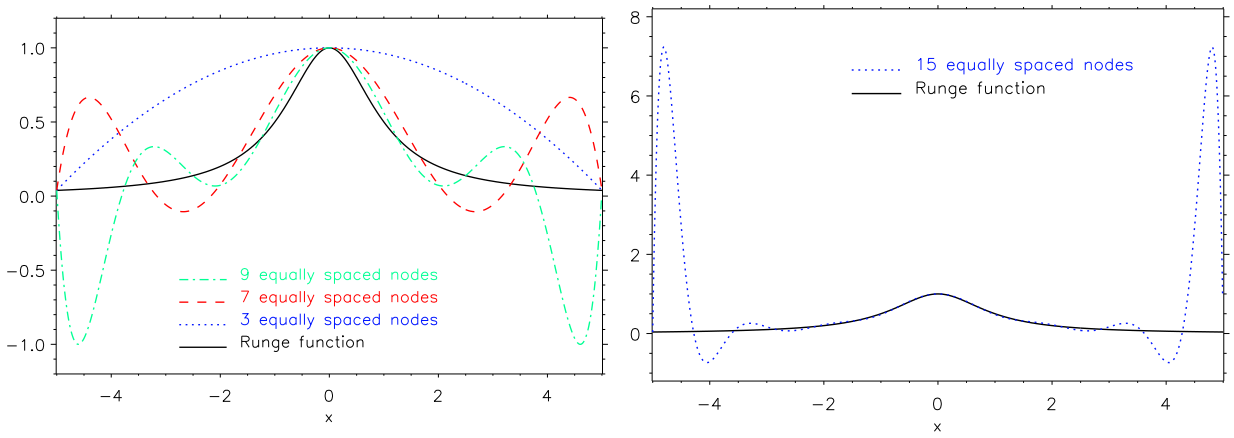

Figure 2: Interpolation function of the Runge function (Eq. (32)) using 3,7,9 and 15 equally space nodes in the domain $[-5, 5]$.

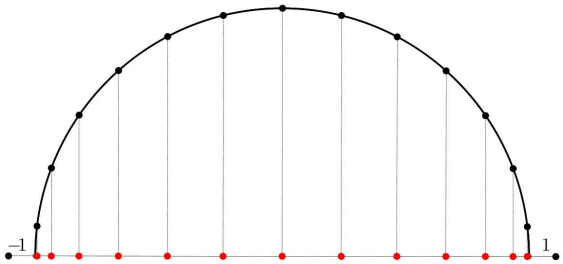

Figure 3: Distribution of the Chebyshev nodes on a closed interval [-1,1].

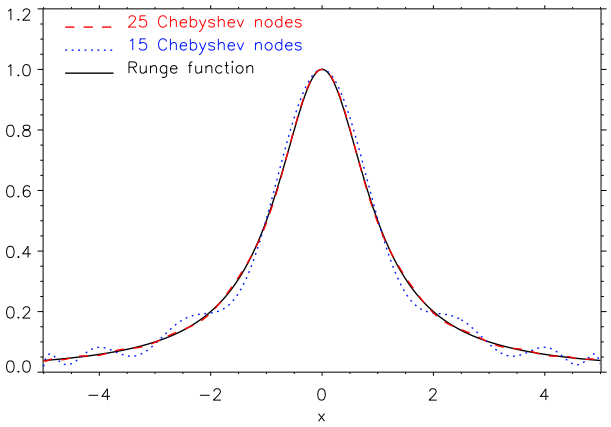

Figure 4: Interpolation of the Runge function using Lagrange polynomial based on Chebyshev nodes distribution.

Theorem 2.1. (Interpolation error Theorem) *Let p be the polynomial of degree at most n interpolating function f at $n+1$ nodes $x_0, x_1, x_2, ..., x_n$ on $[a,b]$. Let $f^{(n+1)}$ be continuous. Then, for each $x \in [a,b]$, there is some $\xi \in [a,b]$ such as:*

$$E_n(x) = f(x) - \Pi_n f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\,\omega_{n+1}(x), \tag{35}$$

where $\omega_{n+1}(x)$ is the nodal polynomial defined by Eq. (21).

Interpolation Errors for equally spaced nodes

One can try to bound the interpolation error for equally spaced nodes. We now consider bounding

$$\max_{x\in[a,b]} \left(\prod_{i=0}^{n} |x - x_i| \right), \tag{36}$$

where

$$x_i = a + h * i = a + \frac{(b-a)}{n} i, \quad i = 0, ..., n \tag{37}$$

and h is the node spacing. We can assume that x is not one of the nodes, otherwise, the product is zero. Let consider j such as x is between x_j and x_{j+1}. We can show that

$$|x - x_j||x - x_{j+1}| \leq \frac{h^2}{4} \tag{38}$$

We can now claim that $|x - x_i| \leq (j - i + 1)h$ for $i < j$ and $|x - x_i| \leq (i - j)h$ for $j + 1 < i$ then

$$\prod_{i=0}^{n} |x - x_i| \leq \frac{h^2}{4} \left[(j+1)!\, h^j\right] \left[(n-j)!\, h^{n-j-1}\right] \tag{39}$$

Moreover, it can be shown that $(j+1)!(n-j)! \leq n!$ and so we get an overall bound

$$\prod_{i=0}^{n} |x - x_i| \leq \frac{h^{n+1} n!}{4}. \tag{40}$$

The interpolation theorem gives us

$$|E_n(x)| = |f(x) - \Pi_n f(x)| \leq \frac{h^{n+1}}{4(n+1)} \max_{\xi\in[a,b]} \left| f^{(n+1)}(\xi) \right|, \tag{41}$$

with $h = (b-a)/n$.

The reason why the interpolation of the Runge function using equally spaced nodes are not bounded is due to the fact that $f^{(n)}$ for the Runge function becomes unbounded when $n \to \infty$.

Exercise: How many equally spaced nodes are required to interpolate the function $f(x) = cos(2x) + sin(x)$ to within 10^{-10} on the interval $[0, \pi]$?

Interpolation Errors for Chebyshev nodes

We have no control over the function f and its derivative $f^{(n+1)}$ and once the nodes and f are fixed, the interpolating polynomial is determined. thus, the only way to make the error $|f(x) - \Pi_n f(x)|$ small is by a judicious choice of the interpolation nodes x_i. The Chebyshev nodes on $[-1, 1]$ have the remarkable property that:

$$\left| \prod_{i=0}^{n} (t - x_i) \right| \leq 2^{-n} \tag{42}$$

for any $t \in [-1, 1]$. Moreover, it can be seen that, for any choice of nodes $x - i$ that

$$\max_{t \in [-1,1]} \left| \prod_{i=0}^{n} (t - x_i) \right| \geq 2^{-n} \tag{43}$$

Therefore, the Chebyshev nodes are considered as the best or polynomial interpolation.

The Chebyshev nodes can be rescaled and shifted to be used for general interval $[\alpha, \beta]$. In this case the nodes are given by

$$x_i = \frac{\beta - \alpha}{2} cos \left[\left(\frac{2i + 1}{2n + 2} \right) \pi \right] + \frac{\alpha + \beta}{2}, \;\; 0 \leq i \leq n \tag{44}$$

In this case, the rescaling changes the bound on $\prod_{i=0}^{n}(t - x_i)$ so the overall error bound becomes

$$E_n(x) = f(x) - \Pi_n f(x) = \frac{(\beta - \alpha)^{n+1}}{2^{2n+1}(n + 1)!} \max_{\xi \in [\alpha, \beta]} \left| f^{(n+1)}(\xi) \right|, \tag{45}$$

for $x \in [\alpha, \beta]$

Exercise: How many Chebyshev nodes are required to interpolate the function $f(x) = cos(2x) + sin(x)$ to within 10^{-10} on the interval $[0, \pi]$?

2.1.4 Interpolation by piecewise Lagrange polynomials

It has been shown previously that this is not possible to guarantee the uniform convergence of the Lagrange polynomials with equally spaced nodes. However, the interpolation with low order polynomial is usually accurate enough when used on a sufficiently small intervals. This is therefore natural to introduce an interpolation on K sub-intervals $I_j = [x_j, x_{j+1}]$ of length h_j using $k+1$ equally spaced nodes in each sub-interval. It can be shown that

$$\left\|E_h^k\right\|_\infty = \max_{[a,b]}(E_h^k) = \max_{[a,b]}\left(f - \Pi_h^k f\right) \leq Ch^{k+1} \max_{\xi\in[a,b]}\left(f^{(k+1)}(\xi)\right) \tag{46}$$

where

$$h = \max_{0\leq j\leq K-1}(h_j)$$

This shows that one can obtain a small interpolation error using small values of k but with sufficiently low values of h.

Example: Interpolation error for a first and second order piecewise Lagrange polynomials of the Runge function $f(x) = (1+x^2)^{-1}$ in the interval $[-5, 5]$ using increasing numbers of constant sub-intervals

h	$\|f - \Pi_h^1\|_\infty$	$\|f - \Pi_h^2\|_\infty$
5	0.4153	0.0835
2.5	0.1787	0.0971
1.25	0.0631	0.0477
0.625	0.0535	0.0082
0.3125	0.0206	0.0010
0.15625	0.0058	$1.382e-04$
0.078125	0.0015	$1.7715e-05$

(47)

2.2 Spline interpolation

As it has already been discussed, the interpolation using high order polynomial on a large interval can lead to bad results at the two bound of the interval. On the other hand, the approximation with piecewise first order or low order polynomials leads to a global interpolation with discontinuity of the derivative within the interval. The Spline method consist to a kind of "minimization of elastic energy" and impose the continuity of the derivative and the second order derivative in the interval for the cubic spline. The definition of a cubic spline is given by:

Definition: A function S is a spline of degree k on [a,b] if

1. The domain of S is [a,b]
2. $S, S', S^{(2)}, ..., S^{(k-1)}$ are continuous on [a,b]
3. There is a partition $\{t_i\}_{i=0}^n$ of $[a, b]$ such that on $[t_i, t_{i+1}]$, S is a polynomial of degree $\leq k$.

If the partition has $n+1$ knots, the spline of degree k is defined by $n(k+1)$ parameters. Thus we have n+1 constrains $s(x(i)) = y(i)$ for $i = 0, 1, ..., n$ and $k(n-1)$ constrains for the continuity of $S, S', S^{(2)}, ..., S^{(k-1)}$ at the inner knots $i = 1, ..., n-1$. Thus we have $k-1$ more

unknowns as equations. Therefore, one must add $k-1$ constrains to uniquely defined the splines. This constrains are called *degrees of freedom*. Often k is chosen as 3. This yields cubic splines. We must add 2 extra constrains to define the spline. The usual choice is to make

$$S''(t_0) = S''(t_n) = 0 \tag{48}$$

This yields the natural cubic spline. Another choice for the 2 degrees of freedom is to make S''' to be continuous at t_1 and t_n. This choice is called the *not-a-knot* condition.

The method to compute the coefficients of the cubic spline will be described. Let's define $n+1$ nodes in the $[a,b]$ interval $a = x_0 < x_1 < x_2 < ... < x_n = b$ and the corresponding values $f_i, i = 0, ..., n$. The objective is to find an efficient method to define the interpolating cubic splines for these values. As we are dealing with spline s of order 3, the second derivative must be continuous. Let's define the following notations:

$$f_i = s(x_i),\ m_i = s'(x_i),\ M_i = s''(x_i),\ i = 0, ..., n. \tag{49}$$

Let's also define the splines on the interval $[x_i, x_{i-1}]$ as $s_i(x)$ and $h_i = x_i - x_{i-1}$ for $i = 0, ..., n$. As the splines are of order 3, s'' is linear and we have:

$$s''_{i-1}(x) = M_{i-1}\frac{x_i - x}{h_i} + M_i\frac{x - x_{i-1}}{h_i} \quad \text{for} \quad x \in [x_{i-1}, x_i] \tag{50}$$

By integrating twice the formula, we obtain:

$$s_{i-1}(x) = M_{i-1}\frac{(x_i - x)^3}{h_i} + M_i\frac{(x - x_{i-1})^3}{h_i} + C_{i-1}(x - x_{i-1}) + \tilde{C}_{i-1}, \tag{51}$$

Where the two constants C_{i-1} and $\tilde{C}_{i-1}$ are determined by imposing the values at the bounds $s(x_{i-1}) = f_{i-1}$ and $s(x_i) = f_i$. This gives for $i = 1, ..., n-1$

$$\tilde{C}_{i-1} = f_{i-1} - M_{i-1}\frac{h^2}{6},\ \ C_{i-1} = \frac{f_i - f_{i-1}}{h_i} - \frac{h^2}{6}(M_i - M_{i-1}) \tag{52}$$

Let define the continuity of the first derivative at x_i:

$$\begin{aligned} s'(x_i^-) &= \frac{h_i}{6}M_{i-1} + \frac{h_i}{3}M_i + \frac{f_i - f_{i-1}}{h_i} \\ &= -\frac{h_{i+1}}{3}M_i - \frac{h_{i+1}}{6}M_{i+1} + \frac{f_{i+1} - f_i}{h_{i+1}} = s'(x_i^+) \end{aligned} \tag{53}$$

where

$$s'(x_i^{\pm}) = \lim_{t \to 0} s'(x_i \pm t) \tag{54}$$

This leads to the following linear system:

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i,\ \ i = 1, ..., n-1 \tag{55}$$

where we have defined:

$$\begin{aligned} \mu_i &= \frac{h_i}{h_i + h_{i+1}},\ \ \lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}} \\ d_i &= \frac{6}{h_i + h_{i+1}}\left(\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}\right),\ \ i = 1, ..., n-1 \end{aligned} \tag{56}$$

The system (55) has $n+1$ unknowns and $n-1$ equations. 2 additional conditions need to be fixed. Usually, these two conditions are:

$$2M_0 + \lambda_0 M_1 = d_0, \quad \mu_n M_{n-1} + 2M_n = d_n \tag{57}$$

where $\lambda_0 \geq 0$, $\mu_m \geq 1$ and d_0, d_n are given values. In order to obtain *natural splines* satisfying $s''(a) = s''(b) = 0$, one must choose $M_0 = M_n = 0$. An other usual choice is to take $\lambda_o = \mu_n = 1$ and $d_0 = d_n = d_{n-1}$. These last condition leads to consider the two points a and b as internal points. The equation (55) completed by the last two conditions can be expressed in a tri-diagonal matrix form:

$$\begin{bmatrix} 2 & \lambda_0 & 0 & \dots & 0 \\ \mu_1 & 2 & \lambda_1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \mu_{n-1} & 2 & \lambda_{n-1} \\ 0 & \dots & 0 & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{58}$$

This matrix system can be solved efficiently using the *Thomas algorithm* (see section 4.5.4).

3 Discretisation of Partial Derivative Equation

3.1 Spectral methods

3.1.1 Introduction

Spectral methods are a class of method less suited and much less used for general purpose in CFD than Finite differences and Finite Volumes. However, due to their high level of accuracy, these methods can be of interest for simulations using simple geometries and especially for DNS of such flows. The simulation with spectral methods will be discussed in a following section. In this section, we will only introduce the Fourier transform which is of interest for a spectral analysis of the accuracy of finite difference schemes.

3.1.2 Fourier Transform

In spectral methods, the derivations are performed with Fourier Transform or one of their generalizations. The simplest spectral methods deal with periodic functions specified by their values at a uniformly spaced set of points. It is possible to represent such function with a **Discrete Fourier Transform**.

$$f(x_i) = \sum_{q=-N/2}^{q=+N/2-1} \hat{f}(k_q) e^{ik_q x_i}, \tag{59}$$

where $x_i = i\Delta x,\ i = 1, 2, ..., N$ and $k_q = 2\pi q/\Delta_x N$. The equation (59) can be inverted by

$$\hat{f}(k_q) = \frac{1}{N} \sum_{i=1}^{i=N} f(x_i) e^{-ik_q x_i}, \tag{60}$$

The value of q is arbitrary as changing q from q to $q \pm lN$ (where l is integer), produce no change of $e^{\pm ik_q x_i}$ at the grid points. These propriety is know as **aliasing**. The aliasing is the main drawback of the spectral method as it can be an important source of errors in the non-linear equations such as the Navier Stokes equations. The aliasing is not restricted to the spectral method, but is also present, even if it is less critical, in other methods such as high order finite difference methods.

The equation (59) can be used for the interpolation of a function $f(x)$. We simply replace the discrete variable x_i by a continuous variable x and $f(x)$ can be defined for all x. When $f(x)$ is defined, one can differentiate the formula to obtain the Fourier series for the derivatives:

$$\frac{df}{dx} = \sum_{q=-N/2}^{q=+N/2-1} ik_q \hat{f}(k_q) e^{ik_q x}, \tag{61}$$

This shows that the Fourier coefficient of df/dx are simply $ik_q\hat{f}(k_q)$. This leads to a method to derive a function:

- Given $f(x_i)$, use the formula (60) in order to evaluate the Fourier coefficients $\hat{f}(k_q)$;
- Compute the Fourier coefficient of $g = df/dx$ by $\hat{g}(k_q) = ik_q\hat{f}(k_q)$;
- Evaluate the series (61) to obtain g at the grid points.

The method can easily be used for higher order derivatives by using:

$$\frac{d^p f}{dx^p} = \sum_{q=-N/2}^{q=+N/2-1} i^p k_q^p \hat{f}(k_q) e^{ik_q x}, \tag{62}$$

The cost of a derivative is in N^2 (cf eq. (59)). This would be prohibitively expensive without a Fast Fourier Transform algorithm which reduce the cost down to $N\, log_2\, N$. For this method to be useful, the function must be periodic and the grid points equally spaced. These are the main limitation of Fourier spectral method for general problems. These conditions can be relaxed by using other functions than complex exponential for complex geometries and different boundary conditions. However one has to pay the price by important modification of the method. This method is therefore usually restricted to simulations on simple geometries with homogeneous flows.

3.1.3 Pseudo-spectral method [1]

Let us consider the simple Burger equation periodic on $[0, 2\pi]$:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0 \tag{63}$$

In pseudo spectral method, the discrete Fourier transform is applied to the above equation which lead to:

$$\frac{\partial \hat{u_k}}{\partial t} + \widehat{\left(u\frac{\partial u}{\partial x}\right)}_k + \nu k^2 \hat{u}_k = 0 \tag{64}$$

where

$$\widehat{\left(u\frac{\partial u}{\partial x}\right)}_k \tag{65}$$

is the discrete Fourier transform of the nonlinear term. This term can be evaluated by a convolution product defined as

$$\widehat{(uv)}_k = \sum_{p+q=k} \hat{u}_p \hat{v}_q \tag{66}$$

However, the straightforward evaluation of this convolution product requires $O(N^2)$ operations.

Pseudo-spectral methods evaluate the nonlinear terms in physical space instead of by convolution product. Each variable involved in the product is inverse transformed in physical space before to be multiplied and a Fourier transformed is applied to the results. Let us introduce the discrete Fourier transform of the two quantities to multiply:

$$\begin{aligned} U_j &= \sum_{k=-N/2}^{N/2+1} \hat{u}_k e^{ikx_j} \quad j = 0, 1, ..., N-1 \\ V_j &= \sum_{k=-N/2}^{N/2+1} \hat{v}_k e^{ikx_j} \quad j = 0, 1, ..., N-1 \end{aligned} \tag{67}$$

and define

$$W_j = U_j V_j \quad j = 0, 1, ..., N-1 \tag{68}$$

and

$$\hat{W}_k = \frac{1}{N} \sum_{j=0}^{N-1} W_j e^{-ikx_j} \quad k = -\frac{N}{2}, ..., \frac{N}{2} - 1 \tag{69}$$

where

$$x_j = 2\pi j / N \tag{70}$$

Due to the discrete transform orthogonality relation

$$\frac{1}{N} \sum_{j=0}^{N-1} e^{-ipx_j} = \begin{cases} 1 & \text{if} \quad p = Nm, \quad m = \pm 1, \pm 2, ... \\ 0 & \text{otherwise} \end{cases} \tag{71}$$

we obtain

$$\hat{W}_k = \sum_{m+n=k} \hat{u}_m \hat{v}_n + \sum_{m+n=k\pm N} \hat{u}_m \hat{v}_n \tag{72}$$

$$= \hat{u}_k + \sum_{m+n=k\pm N} \hat{u}_m \hat{v}_n \tag{73}$$

The second term on the right hand side is the **aliasing error**. If the convolution is evaluated by the method described above, the method is not a true spectral Galerkin method. Orszag (1971) named it a **pseudo-spectral method**

3.1.4 Aliasing removal by Truncation

Several method can be used to remove the aliasing term which appear in pseudo-spectral methods. The simplest one and the more used is the truncation. The key to this de-aliasing technique is the use of discrete Fourier transform with M rather than N points with $M \geq 3N/2$

Let $x_j = 2\pi j / M$

$$U_j = \sum_{k=-N/2}^{N/2+1} \tilde{u}_k e^{ikx_j} \quad j = 0, 1, ..., M-1$$

$$V_j = \sum_{k=-N/2}^{N/2+1} \tilde{v}_k e^{ikx_j} \quad j = 0, 1, ..., M-1 \tag{74}$$

$$W_j = U_j V_j$$

where

$$\tilde{u}_k = \begin{cases} \hat{u}_k & |k| \leq N/2 \\ 0 & \text{otherwise} \end{cases} \tag{75}$$

Thus the $\tilde{u}_k$ coefficient are the $\hat{u}_k$ coefficients padded with zeros for the additional wavenumbers. Likewise, let

$$\tilde{W}_k = \frac{1}{M} \sum_{j=0}^{M-1} W_j e^{-ikx_j} \quad j = -\frac{M}{2}, ..., \frac{M}{2} - 1 \tag{76}$$

then

$$\tilde{W}_k = \sum_{m+n=k} \tilde{u}_m \tilde{v}_n + \sum_{m+n=k\pm N} \tilde{u}_m \tilde{v}_n \tag{77}$$

As we are only interested in $\tilde{W}_k$ for $|k| \leq N/2$ we can choose M such that the second term on the right-hand side vanishes for these k. Since $\tilde{u}_m$ and $\tilde{v}_m$ are zero for $|m| > N/2$, the worse case condition is

$$-\frac{N}{2} - \frac{N}{2} \leq \frac{N}{2} - 1 - M \tag{78}$$

or

$$M \geq \frac{3N}{2} \tag{79}$$

The operation count for this transform method is $(45/4)\ N\ log_2(3N/2)$ which is roughly 50% larger than the aliased method. The technique can be generalized for 2D and 3D problems.

3.1.5 Chebyshev

As the Fourier spectral method is mostly restricted to homogeneous discretisation and period boundary conditions, polynomial discretisation can be used. Chebyshev or Legendre approximation polynomials may be used instead of cosine and sine modes. The spectral method can be derived from a general framework of orthogonal polynomial approximation. We will only give the definition without any proof.

The Chebyshev polynomial are defined by:

$$T_k(x) = cos(k\theta), \quad \theta = \text{arccos}(x), k = 0, 1, 2, ... \tag{80}$$

where $T_k(x)$ is a polynomial of order k. One can prove the following recurrence formula:

$$\begin{cases} T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) \\ T_o(x) = 1, \quad T_1(x) = x \end{cases} \tag{81}$$

The Chebyshev expansion of a function f is:

$$f(x) = \sum_{k=0}^{\infty} \hat{f}_k T_k(x) \quad \hat{f}_k = \frac{1}{c_k} \int_{-1}^{1} f(x) T_k(x) (1 - x^2)^{-1/2} dx \tag{82}$$

Differentiation

The derivative of a function f expanded in Chebyshev polynomials according to (82) can be represented formally as

$$f' = \sum_{m=0}^{\infty} \hat{f}_m^{(1)} T_m \tag{83}$$

where

$$\hat{f}_m^{(1)} = \frac{2}{c_m} \sum_{\substack{p=m+1 \\ p+m\ odd}}^{\infty} p \hat{f}_p^{(1)} \tag{84}$$

this expression is a consequence of the relation

$$2T_k(x) = \frac{1}{k+1} T'_{k+1}(x) - \frac{1}{k-1} T'_{k-1}(x), \quad k \geq 1 \tag{85}$$

which is a consequence of a trigonometric relation. From (84) one has

$$2k\hat{f}_k = c_{k-1}\hat{f}^{(1)}_{k-1} - \hat{f}^{(1)}_{k+1}, \quad k \geq 1 \tag{86}$$

Since $\hat{f}^{(1)}_k = 0$ for $k \geq N$, the non-zero coefficients are computed in decreasing order by the relation

$$c_k\hat{f}^{(1)}_k = \hat{f}^{(1)}_{k+2} + 2(k+1)\hat{f}^{(1)}_{k+1} \quad 0 \geq k \geq N-1 \tag{87}$$

If the discrete Chebyshev transform are computed by FFT algorithm, the total number of operations to differentiate in physical space is $(5log_2N + 10)N$. The extra overhead in the cosine transform leads to 20% to 40 % grater cost for a Chebyshev derivative compared with Fourier derivative.

3.2 Finite differences

In finite difference methods, the derivatives of the equations of the physics are approximated using Taylor expansions. This is the oldest method for numerical solution of PDE's, believed to have been introduced by Euler and several other mathematicians (Taylor, Leibniz, ...) in the 18th century. The principle of the method is to replace the partial derivative by a combination of punctual values at a finite given number of discrete points or mesh points. The main advantages is that the method is very simple and effective on structured grid in which it is easy to obtain higher-order schemes and is requires a limited computing resources. The drawback of the method is that it is difficult to apply to unstructured grid usually used for complex geometries. The other disadvantage is the difficulty to take into account boundary condition of Neumann type.

3.2.1 Taylor development

Given a function $u(x, y, z, t)$ of space and time. By definition of the derivative :

$$\frac{\partial u}{\partial x} = \lim_{\Delta x \to 0} \frac{u(x + \Delta x, y, z, t) - u(x, y, z, t)}{\Delta x}$$

If Δx is small, a Taylor development of $u(x, y, z, t)$ in the vicinity of x is given by :

$$u(x+\Delta x, y, z, t) = u(x, y, z, t)+\Delta x\frac{\partial u}{\partial x}(x, y, z, t)+\frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2}(x, y, z, t)+\frac{\Delta x^3}{6}\frac{\partial^3 u}{\partial x^3}(x, y, z, t)+...$$

Truncating the expansion at the first order in Δx, one obtain:

$$\frac{u(x + \Delta x, y, z, t) - u(x, y, z, t)}{\Delta x} = \frac{\partial u}{\partial x}(x, y, z, t) + O(\Delta x)$$

The approximation of the derivative $(\partial u/\partial x)(x)$ is at the first order indicating that the truncation error $O(\Delta x)$ goes to zero as the first power of Δx. The power of Δx by which the truncation error tends to zero is called *order of the method*.

By considering the 1D case where we would like to determine a quantity $u(x)$ in the interval $[0, 1]$. Looking for the discrete solution of u leads to define a mesh of the interval of definition. We will consider the mesh composed with N+1 points x_i for i=0,...,N equably spaced with a spacing Δx. The points $x_i = i\Delta x$ are called *mesh points*.

Notations: the discrete value of $u(x)$ at point x_i will be noted u_i ($u_i = u(x_i)$). Similarly, $\left(\frac{\partial u}{\partial x}\right)_{x=x_i} = \left(\frac{\partial u}{\partial x}\right)_i$ will be used for the derivative of u at the point x_i.

The matrix notation of the finite difference scheme at the first order derived above is:

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_i - u_{i-1}}{\Delta x} + O(\Delta x) \tag{88}$$

This scheme is said *upwind* as it uses the value of the function at the point $i-1$ (upward the point i where the derivative is evaluated).

It is possible to define other first order finite difference schemes for the first derivative of $u(x)$:

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_i}{\Delta x} + O(\Delta x) \tag{89}$$

This last scheme is said "*forward*".

3.2.2 Higher order schemes

The finite differences of higher order can be obtained by combination of Taylor expansions in the vicinity of x_i:

$$u_{i+1} = u(x_i + \Delta x) = u(x_i) + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i + O(\Delta x^3) \tag{90}$$

$$u_{i-1} = u(x_i - \Delta x) = u(x_i) - \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i + O(\Delta x^3) \tag{91}$$

Subtracting the two expansions above leads to:

$$u_{i+1} - u_{i-1} = 2\Delta x \left(\frac{\partial u}{\partial x}\right)_i + O(\Delta x^3) \tag{92}$$

This gives the second order *central* scheme which is the approximation of the first derivative of u:

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x^2) \tag{93}$$

In order to obtain a higher order scheme, additional points in the neighborhood of x_i are required. The number of points required to write a finite difference scheme are called the *stencil*. For instance, a third order scheme for the first derivative reads:

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{-u_{i+2} + 6u_{i+1} - 3u_i - 2u_{i-1}}{6\Delta x} + O(\Delta x^3) \tag{94}$$

This scheme has a stencil of 4 (which corresponds to $u_{i+2}, u_{i+1}, u_i, u_{i-1}$). Other third order schemes are possible for the first derivative.

3.2.3 Higher order derivatives

The principle of the method is identical for higher order schemes. The method is still based on the Taylor expansion in the vicinity of x_i. For instance, in order to define a scheme for the second order derivative of u, one can write the two following expansions:

$$u_{i+1} = u(x_i + \Delta x) = u(x_i) + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{\Delta x^3}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + O(\Delta x^4) \quad (95)$$

$$u_{i-1} = u(x_i - \Delta x) = u(x_i) - \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{\Delta x^3}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + O(\Delta x^4) \quad (96)$$

Then, one must find the right combination of the two above equations which suppress the first derivatives. In the present case, taking the sum of the two equations leads to:

$$u_{i+1} + u_{i-1} - 2u_i = \Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i + O(\Delta x^4) \quad (97)$$

This gives the *central second order scheme* which approximates the second derivative of u

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + O(\Delta x^2) \quad (98)$$

Other formations such as *upwind* or *forward* schemes can be derived for the second order derivative. However, keeping the same stencil of 3, only first order schemes are possible:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+2} - 2u_{i+1} + u_i}{\Delta x^2} + O(\Delta x) \qquad \left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_i - 2u_{i-1} + u_{i-2}}{\Delta x^2} + O(\Delta x)$$

Using the same technique of combination of Taylor expansion, one can derive higher order finite difference schemes for any order of the derivative. However, the stencil (and therefore the number of Taylor expansions) must be adapted to the order of the scheme. The drawback of this methodology using Taylor expansion is that it can be very long, especially for higher order schemes and schemes for higher order derivatives as it requires a large number of Taylor expansions and the solving of a linear system of equation in order eliminate the lowest order terms of the expansions. However, the above procedure, with undetermined coefficients, can be put into a systematic framework in order to obtain finite difference approximations to all derivatives with a preselected order of accuracy. In order to achieve this, a formulation is to be defined via the introduction of appropriate difference operators.

3.2.4 General methods

General procedures to developed in order to generate finite difference schemes to any order of accuracy can be found in Hildebrand (1956) [3]. This approach is based on the definition of differences operators.

Displacement operators E:

$$E\, u_i = u_{i+1} \quad (99)$$

Forward difference operator δ^+:

$$\delta^+ u_i = u_{i+1} - u_i \tag{100}$$

Backward difference operator δ^-:

$$\delta^- u_i = u_i - u_{i-1} \tag{101}$$

Central difference operator δ:

$$\delta u_i = u_{i+1/2} - u_{i-1/2} \tag{102}$$

Averaging operator μ:

$$\mu u_i = \frac{1}{2}\left(u_{i+1/2} + u_{i-1/2}\right) \tag{103}$$

Differential operator D:

$$D u = \frac{\partial u}{\partial x} \tag{104}$$

From these definitions some obvious relations can be defined between these operators:

$$\delta^+ = E - 1 \tag{105}$$
$$\delta^- = 1 - E^{-1} \tag{106}$$

where

$$E^{-1} u_i = u_{i-1} \tag{107}$$

This leads to the following relations

$$\delta^- = E^{-1}\delta^+ \tag{108}$$

and

$$\delta^+\delta^- = \delta^-\delta^+ = \delta^+ - \delta^- = \delta^2 \tag{109}$$

but also

$$\delta\mu = (E^{+1/2} - E^{-1/2})\frac{1}{2}(E^{1/2} + E^{-1/2}) = \frac{1}{2}(E^1 - E^{-1}) \tag{110}$$

Using the general definition (n being positive or negative):

$$E^n u_i = u_{i+n} \tag{111}$$

We also have

$$\delta = E^{1/2} - E^{-1/2} \tag{112}$$
$$\mu = \frac{1}{2}(E^{1/2} + E^{-1/2}) \tag{113}$$

Any of the above difference operators taken to a given power n, is interpreted as n repeated actions of this operator. For instance:

$$\delta^{+2} = \delta^+\delta^+ = (E-1)^2 = E^2 - 2E + 1 \tag{114}$$
$$\delta^{+3} = \delta^+\delta^+\delta^+ = (E-1)^3 = E^3 - 3E^2 + 3E - 1 \tag{115}$$
$$\tag{116}$$

The key to the operator technique is to link the derivative operator D to the finite displacement operators $E, \delta^+, \delta^-, \delta, \mu, ...$ The relation are obtained using a Taylor expansion:

$$u(x+\Delta x) = u(x) + \Delta x \frac{\partial u}{\partial x}(x) + \frac{\Delta x^2}{2!}\frac{\partial^2 u}{\partial x^2}(x) + \frac{\Delta x^3}{3!}\frac{\partial^3 u}{\partial x^3}(x) + ...$$

which can be written in operators form:

$$Eu(x) = \left(1 + \Delta x D + \frac{(\Delta x D)^2}{2!} + \frac{(\Delta x D)^3}{3!} + ...\right) u(x) \tag{117}$$

Taking into account the Taylor expansion of the exponential function, this relation can be written formally as

$$Eu(x) = e^{\Delta x D} u(x) \tag{118}$$

or symbolically

$$E = e^{\Delta x D} \tag{119}$$

This relation can be inverted as:

$$\Delta x D = ln(E) \tag{120}$$

but also

$$(\Delta x)^n D^n = (ln(E))^n \tag{121}$$

Forward differences

Formulas for forward differences are obtained by introducing the relation (105) between E and the forward operator δ^+ in relation (120). After a formal development of the ln function, we obtain:

$$\begin{aligned} \Delta x D &= ln(E) = ln(1+\delta^+) && (122)\\ &= \delta^+ - \frac{\delta^{+2}}{2} + \frac{\delta^{+3}}{3} - \frac{\delta^{+4}}{4} + ... && (123) \end{aligned}$$

The order of the accuracy of the approximation increase with the number of terms kept in the right-and-side. The first neglected term gives the truncation error. Keeping only the first term will lead to the first order formula and a truncation error equal to $\frac{\Delta x}{2}\frac{\partial^2 u}{\partial^2 x}$. If the first two terms are considered, we obtain the second order formula and the truncation error is $\frac{\Delta x^2}{3}\frac{\partial^3 u}{\partial^3 x}$.

$$\left(\frac{\partial u}{\partial x}\right)_i = D\, u_i = \frac{-3u_i + 4u_{i+1} - u_{i+2}}{2\Delta x} + \frac{\Delta x^2}{3}\left(\frac{\partial^3 u}{\partial x^3}\right)_i \tag{124}$$

Since the forward difference operator can be written as

$$\delta^+ = \Delta x \frac{\partial u}{\partial x} + O(\Delta x^2) \tag{125}$$

the first neglected operator δ^{+n} is of order n, showing that the truncation error is $O(\Delta x^{n-1})$.

Backward differences

Similarly, backward difference formula at increasing order of accuracy can be obtained by application of the relation (120):

$$\begin{aligned} \Delta x D &= ln(E) = -ln(1-\delta^-) && (126)\\ &= \delta^- + \frac{\delta^{-2}}{2} + \frac{\delta^{-3}}{3} + \frac{\delta^{-4}}{4} + ... && (127) \end{aligned}$$

Considering the first two terms of the right-hand side, we obtain the second order formula:

$$\left(\frac{\partial u}{\partial x}\right)_i = D\,u_i = \frac{3u_i - 4u_{i-1} + u_{i-2}}{2\Delta x} + \frac{\Delta x^2}{3}\left(\frac{\partial^3 u}{\partial x^3}\right)_i \qquad (128)$$

Central differences

In order to obtain central finite difference formula, one must link the operator E to the central difference operator δ (or to δ and μ). From the definition of the central difference operator:

$$\delta\,u_i = u_{i+1/2} - u_{i-1/2} = (E^{1/2} - E^{-1/2})\,u_i \qquad (129)$$

and therefore using (119)

$$\delta = e^{\Delta x D/2} - e^{-\Delta x D/2} = 2\sinh\left(\frac{\Delta x D}{2}\right) \qquad (130)$$

which through inversion leads to:

$$\begin{aligned} \Delta x D &= 2\sinh^{-1}\left(\frac{\delta}{2}\right) && (131)\\ &= 2\left[\frac{\delta}{2} - \frac{1}{2\cdot 3}\left(\frac{\delta}{2}\right)^3 + \frac{1\cdot 3}{2\cdot 4\cdot 5}\left(\frac{\delta}{2}\right)^5 - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7}\left(\frac{\delta}{2}\right)^7 + ...\right] && (132)\\ &= \delta - \frac{\delta^3}{24} + \frac{3\delta^5}{640} - \frac{5\delta^7}{7168} + ... && (133) \end{aligned}$$

by using the expansion formula for $\sinh^{-1}$:

$$\begin{aligned} \sinh^{-1}(x) = \quad & x - \frac{1}{2\cdot 3}x^3 + \frac{1\cdot 3}{2\cdot 4\cdot 5}x^5 - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7}x^7 + ... \\ & +(-1)^n \frac{1\cdot 3\cdot 5\ldots(2n-1)}{2\cdot 4\cdot 6\ldots(2n)(2n+1)}x^{2n+1} + O(x^{2n+2}) \end{aligned} \qquad (134)$$

The same formula can be used for the n^{th} order derivative:

$$(\Delta x D)^n = \left(\delta - \frac{\delta^3}{24} + \frac{3\delta^5}{640} - \frac{5\delta^7}{7168} + ...\right)^n \qquad (135)$$

For odd values of n, the formula will used values of the function u at the half integer mesh points ($u^{i+k/2}$ where k an integer).

Example:

$$\left(\frac{\partial u}{\partial x}\right)_i = Du_i = \frac{1}{\Delta x}\delta u_i + O(\Delta x^2) = \frac{u_{i+1/2} - u_{i-1/2}}{\Delta x} + O(\Delta x^2) \qquad (136)$$

For odd values of derivative n, in order to use the values of the function at the node points (u^{i+k}), one must link the operator E to δ and μ. From the definition of δ and μ (Eqs. (102) and (103)), we can write the following relation:

$$1 + \frac{\delta^2}{4} = \mu^2 \tag{137}$$

or

$$1 = \mu\left(1 + \frac{\delta^2}{4}\right)^{-1/2} = \mu\left(1 - \frac{\delta^2}{8} + \frac{3\delta^4}{128} - \frac{5\delta^6}{1024} + ...\right) \tag{138}$$

by using the following relation:

$$\begin{aligned}(1+x)^a = \quad & 1 + a\,x + \frac{a(a-1)}{2!}x^2 + \frac{a(a-1)(a-2)}{3!}x^3 + ... \\ & + \frac{a(a-1)(a-2)\ldots(a-n+1)}{n!}x^n + O(x^{n+1})\end{aligned} \tag{139}$$

After multiplying the eq. (132) by eq. (138) we obtain:

$$\Delta x D = \mu\left(\delta - \frac{1}{3!}\delta^3 + \frac{1^2 \cdot 2^2}{5!}\delta^5 - ...\right) \tag{140}$$

Hence we obtain the central difference formula for the first order derivative with integer mesh point values.

Example:

The second order accurate central difference approximations of the first derivative is given by:

$$\left(\frac{\partial u}{\partial x}\right) = Du_i = \frac{1}{\Delta x}\left(\mu\delta - \frac{1}{3!}\mu\delta^3\right)u_i \tag{141}$$

$$= \frac{1}{\Delta x}\left(\frac{1}{2}(E^1 - E^{-1})(1 - \frac{1}{6}\delta^2)\right)u_i \tag{142}$$

$$= \frac{1}{2\Delta x}\left((E^1 - E^{-1})(1 - \frac{1}{6}(E^1 + E^{-1} - 2))\right)u_i \tag{143}$$

$$= \frac{1}{12\Delta x}\left(8(E^1 + E^{-1}) - (E^2 - E^{-2})\right)u_i \tag{144}$$

$$= \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x} \tag{145}$$

Higher order derivatives

The same techniques using operator can be applied to higher order derivatives. Starting from

the formula (131), the central formula of the n order derivative is obtained by:

$$D^n u_i = = \left(\frac{2}{\Delta x}\,\mathsf{sinh}^{-1}\left(\frac{\delta}{2}\right)\right)^n u_i \tag{146}$$

$$= \frac{1}{(\Delta x)^n}\left[\delta - \frac{\delta^3}{24} + \frac{3\delta^5}{640} - \frac{5\delta^7}{7168} + ...\right]^n \tag{147}$$

$$= \frac{1}{(\Delta x)^n}\delta^n\left[1 - \frac{n}{24}\delta^2 + \frac{n}{64}\left(\frac{22+5n}{90}\right)\delta^4 - \frac{n}{4^5}\left(\frac{5}{7} + \frac{n-1}{5} + \frac{(n-1)(n-2)}{3^5}\right)\delta^6 + ...\right] \tag{148}$$

For n even, this equation generates difference formulas with the function values at the integer mesh point. For n uneven, the difference formulas involve points at half-integer mesh points. In order to involve only points of i for n uneven, one can use the relation of Eq. (137) to get:

$$D^n u_i = \frac{\mu}{[1+(\delta^2/4)]^{1/2}}\left(\frac{2}{\Delta x}\,\mathsf{sinh}^{-1}\left(\frac{\delta}{2}\right)\right)^n u_i \tag{149}$$

$$= \mu\frac{\delta^n}{(\Delta x)^n}\left[1 - \frac{n+3}{24}\delta^2 + \frac{5n^2+52n+135}{5760}\delta^4 + ...\right] u_i \tag{150}$$

Summary of few finite difference formulas

First order forward finite difference

	u_i	u_{i+1}	u_{i+2}	u_{i+3}	u_{i+4}
$\Delta x\ u'_i$	-1	1			
$\Delta x^2\ u''_i$	1	-2	1		
$\Delta x^3\ u'''_i$	-1	3	-3	1	
$\Delta x^4\ u^{(4)}_i$	1	-4	6	-4	1

First order backward finite difference

	u_{i-4}	u_{i-3}	u_{i-2}	u_{i-1}	u_i
$\Delta x\ u'_i$				-1	1
$\Delta x^2\ u''_i$			1	-2	1
$\Delta x^3\ u'''_i$		-1	3	-3	1
$\Delta x^4\ u^{(4)}_i$	1	-4	6	-4	1

Second order central finite difference

	u_{i-2}	u_{i-1}	u_i	u_{i+1}	u_{i+2}
$2\Delta x\ u'_i$		-1		1	
$\Delta x^2\ u''_i$		1	-2	1	
$2\Delta x^3\ u'''_i$	-1	2		-2	1
$\Delta x^4\ u^{(4)}_i$	1	-4	6	-4	1

Fourth order central finite difference formulas

	u_{i-3}	u_{i-2}	u_{i-1}	u_i	u_{i+1}	u_{i+2}	u_{i+3}
$12\Delta x\ u'_i$		1	-8		8	-1	
$12\Delta x^2\ u''_i$		-1	16	-30	16	-1	
$8\Delta x^3\ u'''_i$	-1	-8	13		-13	8	1
$6\Delta x^4\ u^{(4)}_i$	-1	12	-39	56	-39	12	-1

3.2.5 Finite differences formula on non uniform meshes

For non uniform or curvilinear mesh, the discretisation of the equations can be performed after a transformation from the physical space (x, y, z) to a Cartesian computational space (ξ, η, ζ). The relation between the two spaces can be defined through the coordinate transformation formulas such as $\xi = \xi(x, y, z)$ and similar formula for η and ζ. This is considered as a mapping from the physical space to the computational space. All the derivative formulas can be used in the (ξ, η, ζ) space with equations written in curvilinear coordinates. The transformed equations contains some metric terms which have to be discretised. These terms introduce the mesh size influence in the difference formulas.

The effect of non-uniform mesh on the finite differences formula can be deduced from the Taylor development. For a non-uniform distribution of points x_i, on can obtain the forward and backward first order derivatives:

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1}-u_i}{\Delta x_{i+1}} - \frac{\Delta x_{i+1}}{2}\left(\frac{\partial^2 u}{\partial x^2}\right) \tag{151}$$

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_i-u_{i-1}}{\Delta x_i} + \frac{\Delta x_i}{2}\left(\frac{\partial^2 u}{\partial x^2}\right) \tag{152}$$

where

$$\Delta x_i = x_i - x_{i-1} \tag{153}$$

In order to obtain the central formula, one can combine the forward and backward formula to eliminate the first order term (second order derivative). This combination leads to the following second order scheme:

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{1}{\Delta x_i + \Delta x_{i+1}}\left[\frac{\Delta x_i}{\Delta x_{i+1}}\left(u_{i+1}-u_i\right) + \frac{\Delta x_{i+1}}{\Delta x_i}\left(u_i - u_{i-1}\right)\right] - \frac{\Delta x_i \Delta x_{i+1}}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)$$

Starting from the Taylor expansions, one can also derive the forward and backward second order formula involving three consecutive mesh points.

$$\begin{aligned}\left(\frac{\partial u}{\partial x}\right)_i = & \left(\frac{\Delta x_{i+1}+\Delta x_{i+2}}{\Delta x_{i+2}}\cdot\frac{u_{i+1}-u_i}{\Delta x_{i+1}} - \frac{\Delta x_{i+1}}{\Delta x_{i+2}}\cdot\frac{u_{i+2}-u_i}{\Delta x_{i+1}+\Delta x_{i+2}}\right) \\ & + \frac{\Delta x_{i+1}(\Delta x_{i+1}+\Delta x_{i+2})}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)\end{aligned} \tag{154}$$

Similarly, starting from the Taylor expansions, one can derive a central finite difference scheme for the second derivative:

$$\begin{aligned}\left(\frac{\partial^2 u}{\partial x^2}\right)_i = & \left(\frac{u_{i+1}-u_i}{\Delta x_{i+1}} - \frac{u_i-u_{i-1}}{\Delta x_i}\right)\frac{2}{\Delta x_{i+1}+\Delta x_i} \\ & + \frac{1}{3}\left(\Delta x_{i+1}-\Delta x_i\right)\left(\frac{\partial^3 u}{\partial x^3}\right) - \frac{\Delta x_{i+1}^3 + \Delta x_i^3}{12(\Delta x_{i+1}+\Delta x_i)}\left(\frac{\partial^4 u}{\partial x^4}\right)\end{aligned} \tag{155}$$

It is important to observe that the truncation error is proportional to the difference of the consecutive mesh sizes Δx_{i+1} and Δx_i. As a consequence, if the size of the mesh varies abruptly, for instance $\Delta x_{i+1} \simeq 2\Delta x_i$, the above formula will only be first-order accurate. This is a general property of the finite difference approximations on non-uniform meshes. However if the mesh varies smoothly ($\Delta x_{i+1} \simeq \Delta x_i$), the order of the scheme is preserved.

3.2.6 Compact schemes

For explicit finite differences, the derivative at a given point is only function of the values of the function at a selected number (depending of the stencil) of points in the neighborhood of the evaluated point. In such formula, the derivative can be computed directly. However, there are other formula where the value of the derivative at a given order is function of the function and several order derivatives of the function evaluated at the neighboring points. These formula

are called *compact formula* as it allows to evaluate the derivative at a given order of accuracy with a smaller stencil than for explicit formula. However, the drawback of these methods is that the evaluation of the derivative is not straightforward as it requires the resolution of the linear system of equations. Depending of the stencil of the formula, the resolution of this system can be performed using efficient resolution methods much faster than standard methods developed for a full matrix. We are going to illustrate the compact difference methods on a example of formula for the evaluation of the first derivative with a centered scheme on a uniform distribution of grid points:

$$\beta(f'_{i-2} + f'_{i+2}) + \alpha(f'_{i-1} + f'_{i+1}) + f'_i = c\,\frac{f_{i+3} - f_{i-3}}{6h} + b\,\frac{f_{i+2} - f_{i-2}}{4h} + a\,\frac{f_{i+1} - f_{i-1}}{2h} \quad (156)$$

One can noticed that this formula is not the most general one using 5 points of the derivative (lhs) and 7 point of the function (rhs). However, as we would like to derive a centered formula, it seems logical to use the same coefficient for two points which are symmetrical with respect to x_i. As it has been done for the explicit formula, the relation between the coefficients a, b, c, α, β are derived by matching the Taylor series coefficients of various order. The first unmatched coefficient determines the formal trunction error of the approximation (156). These constrains are:

$$a + b + c = 1 + 2(\alpha + \beta) \qquad \text{(second order)} \quad (157)$$

$$a + 2^2 b + 3^2 c = 2 + \frac{3!}{2!}(\alpha + 2^2\beta) \qquad \text{(fourth order)} \quad (158)$$

$$a + 2^4 b + 3^4 c = 2 + \frac{5!}{4!}(\alpha + 2^4\beta) \qquad \text{(sixth order)} \quad (159)$$

$$a + 2^6 b + 3^6 c = 2 + \frac{7!}{6!}(\alpha + 2^6\beta) \qquad \text{(eighth order)} \quad (160)$$

$$a + 2^8 b + 3^8 c = 2 + \frac{9!}{8!}(\alpha + 2^6\beta) \qquad \text{(tenth order)} \quad (161)$$

This equation to satisfy for a odd order of accuracy are automatically verified because the formula was already symmetrized by keeping only 5 unknowns. This shows that this general formula can lead to an order of accuracy up to 10. In the case of periodic boundary conditions, the following linear system of equation can be solved without any other conditions. The non-periodic case requires additional relations appropriate for the near boundary nodes using forward and backward formula. However, taking only the constraints (157) and (158) leads to a three-parameters family of fourth order scheme and, similarly, taking one more constrains (159) leads to two-parameters family of sixth order schemes. Taking $\beta = 0$ leads to a tri-diagonal system to solve. Using this condition, together with $c = 0$, one obtains a one parameter (α) family of fourth order tri-diagonal schemes. These schemes are defined by:

$$\beta = 0,\ a = \frac{2}{3}(\alpha + 2),\ b = \frac{1}{3}(4\alpha - 1),\ c = 0. \quad (162)$$

For $\alpha = 1/4$ the resulting scheme is known as the *classical Padé scheme*. Furthermore, for $\alpha = 1/3$, the leading order coefficient vanishes and the scheme is formally six order accurate. Its coefficients are:

$$\alpha = \frac{1}{3},\ \beta = 0,\ a = \frac{14}{9},\ b = \frac{1}{9},\ c = 0 \quad (163)$$

Using the general formula (156), the computation of the first derivative on a uniform grid of spacing h with periodic boundary conditions such as $f_{i+n} = f_i$ leads to solve the matrix system:

$$\begin{bmatrix} 1 & \alpha & \beta & 0 & 0 & \dots & 0 & \beta & \alpha \\ \alpha & 1 & \alpha & \beta & 0 & \dots & \dots & 0 & \beta \\ \beta & \alpha & 1 & \alpha & \beta & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & \beta & \alpha & 1 & \alpha & \beta \\ \beta & 0 & \dots & \dots & 0 & \beta & \alpha & 1 & \alpha \\ \alpha & \beta & 0 & \dots & \dots & 0 & \beta & \alpha & 1 \end{bmatrix} \begin{bmatrix} f'_1 \\ f'_2 \\ f'_3 \\ \vdots \\ \vdots \\ \vdots \\ f'_{n-2} \\ f'_{n-1} \\ f'_n \end{bmatrix} \tag{164}$$

$$= \begin{bmatrix} 0 & a & b & c & 0 & \dots & -c & -b & -a \\ -a & 0 & a & b & c & \dots & \dots & -c & -b \\ -b & -a & 0 & a & b & c & \dots & \dots & -c \\ -c & -b & -a & 0 & a & b & c & & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & -c & -b & -a & 0 & a & b & c \\ c & \dots & \dots & -c & -b & -a & 0 & a & b \\ b & c & \dots & \dots & -c & -b & -a & 0 & a \\ a & b & c & \dots & \dots & -c & -b & -a & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ \vdots \\ f_{n-2} \\ f_{n-1} \\ f_n \end{bmatrix} \tag{165}$$

For non-periodic case, one needs to derive forward or backward formula. The first derivative at the boundary $i = 1$ can be obtained with forward formula such as:

$$f'_i + \alpha f'_{i+1} + \beta f'_{i+2} + \gamma f'_{i+3} + \delta f'_{i+4} = \frac{1}{h}(af_i + bf_{i+1} + cf_{i+2} + df_{i+3} + ef_{i+4}) \tag{166}$$

The maximum order which can be obtained using this general formula is the eighth order. In this case, there is no free parameter and the coefficient are:

$$\alpha = 16, \qquad \beta = 36 \quad \gamma = 16, \quad \delta = 1,$$

$$a = -\frac{25}{6}, \quad b = -\frac{80}{3}, \quad c = 0, \quad d = \frac{80}{3}, \quad e = \frac{25}{6}.$$

Similarly to the centered formula, families of lower order formula can be obtained. In some cases, this can be of interest to decrease the order of the scheme and keep one or several free parameters to optimize the scheme in order to reduce the dissipation and/or the dispersion. These two characteristics of the scheme will be discussed in the following section when evaluating the error.

The same techniques of compact schemes can be applied to any order of the derivative. For derivative of order n one can decide to use a general formula with all derivatives up to order n with selected stencil. For instance, the formula of the second order derivative can mix first order and second order derivative. In this case, the first and second derivatives needs to be solved at the same times by a larger matrix system which is not necessary an improvement.

3.2.7 Discretisation error

The accuracy of the finite difference formula are linked to their order of accuracy, which is linked to their stencil (the number of points used in the formula). The order of accuracy is different for explicit or implicit scheme. However, the order of accuracy is not the only parameter to characterize the quality of a formula. The spectral methods are very useful to analyze differently the truncation error. By considering a periodic functions, the series

$$f(x_i) = \sum_{q=-N/2}^{q=+N/2-1} \hat{f}(k_q) e^{ik_q x_i}, \tag{167}$$

represents the function and its derivative can be approximated by any methods. On can, in particular, used the exact spectral method below or the finite differences. Any of these method can be applied term-by-term to the series (167). Therefore, it is sufficient to consider the differentiation of $\phi(x) = e^{ikx}$ for any k. The exact result of the derivative is ike^{ikx}. However, if we apply the first order explicit central difference operator to this function

$$\left(\frac{\partial \phi}{\partial x}\right)_i^{FD} \simeq \frac{\phi_{i+1} - \phi_{i-1}}{x_{i+1} - x_{i-1}}, \tag{168}$$

we obtain:

$$\left(\frac{\partial \phi}{\partial x}\right)_i^{FD} \simeq \frac{e^{ik(x+\Delta x)} - e^{ik(x-\Delta x)}}{2\Delta x} = i\frac{sin(k\Delta x)}{\Delta x} e^{ikx} = ik' e^{ikx} \tag{169}$$

where k' is the modified wavenumber because using the finite difference approximation is equivalent to replace the exact wavenumber k by the modified wavenumber k'. The modified wavenumber k' is in general complex. The real part of k', indicated by k'_r is associated with the dispersive error and the imaginary part k'_i is associated with the dissipation error. For central difference schemes it may be shown that the differentiation of $(\partial\phi/\partial x)^{FD}(x) = ik'\partial\phi/\partial x$ where the modified wavenumber k' is real-valued. Similar expression can be derived for other finite difference formula. For instance, the fourth order central formula

$$\left(\frac{\partial \phi}{\partial x}\right)_i \simeq \frac{-\phi_{i+2} + 8\phi_{i+1} - 8\phi_{i-1} + \phi_{i-2}}{12\Delta x} \tag{170}$$

leads to

$$k' = \frac{sin(k\Delta x)}{3\Delta x}[4 - cos(k\Delta x)] \tag{171}$$

For small k the expression of the effective wavenumber can be expanded with Taylor series

$$k' = \frac{sin(k\Delta x)}{\Delta x} \simeq k - \frac{k^3(\Delta x)^2}{6}. \tag{172}$$

As the correction to the exact wavenumber k is in second power of Δx, this shows the second order nature of the approximation at small Δx. However, the highest wavenumber which can be encountered is $k_{max} = \pi/\Delta x$. The figure 5 shows the modified wavenumbers for several schemes including the second order and the fourth order explicit central finite difference formula. The two formula leads to a poor approximation for $k > k_{max}/2$. The range of wavenumbers over which the modified wavenumber k' approximates the exact differentiation within a specified error tolerance defines the set of well-resolved waves. This range is an other indicator of the accuracy of the finite

difference schemes. When the problem to solve has a solution which is not smooth, the order of the method is not anymore a good indicator of its accuracy. The results will be accurate only if several grid points are used for a length corresponding to the highest wavenumber k_{max}. This means that one must be careful not to use the formula with wavenumbers larger then $k_{max}/2$ or even lower values when a good accuracy is required. The spectral method leads to an error that decreases more rapidly than any power of the grid size as the latter goes to zero. However, this behavior is obtained only when the function is discretised with a sufficient number of points. For small number of points, spectral methods can lead to larger errors as compared to finite difference methods.

The same spectral analysis can be conducted for implicit (compact) finite difference schemes.

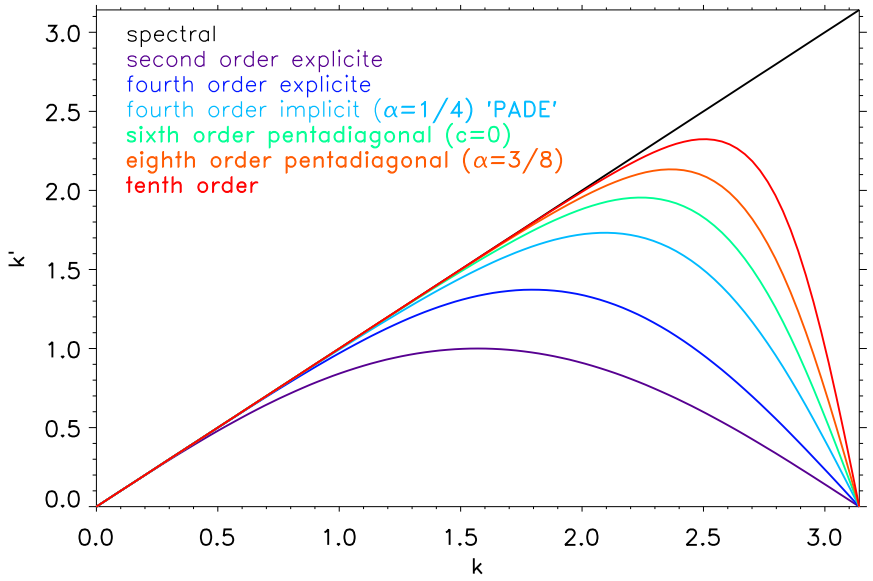

Figure 5: Real part of the modified wavenumbers (k'_r) for several explicit and implicit central finite difference formula.

For instance the modified wavenumber for the general formulation of the implicit (5,5) stencil central finite difference formula is given by:

$$k' = \frac{asin(k) + b/2sin(2k) + c/3sin(3k)}{1 + 2\alpha cos(k) + 2\beta cos(2k)}. \tag{173}$$

The modified wavenumber (which is real because of the central nature of the formula) is plotted Fig. 5 for a variety of schemes of different approximation orders explicit and implicit up to the tenth order. The range of wavenumbers which are accurately resolved is increasing with the order of the scheme. Moreover, for the same order, the compact formulation lead to a better resolution of the derivatives of large wavenumbers (see for instance the comparison of fourth order explicit and implicit scheme).

As for explicit formulation, the implicit central finite difference formula lead to no dissipative error ($k'_i = 0$). However, forward or backward implicit central finite difference formula used for boundary conditions are usually dissipative. As an example, let's compute the modified wavenumbers of the eight order forward finite difference scheme with a (5,7) stencil (5 points for

the derivative and 7 points for the function). The general formula is the following:

$$f'_i+\alpha f'_{i+1}+\beta f'_{i+2}+\gamma f'_{i+3}+\delta f'_{i+4} = \frac{1}{h}(af_i+bf_{i+1}+cf_{i+2}+df_{i+3}+ef_{i+3}+hf_{i+4}+gf_{i+5}) \quad (174)$$

The modified wavenumber computed from the above formula and the real and imaginary part is shown Fig. 6 for several schemes of this family. The highest available forward finite difference

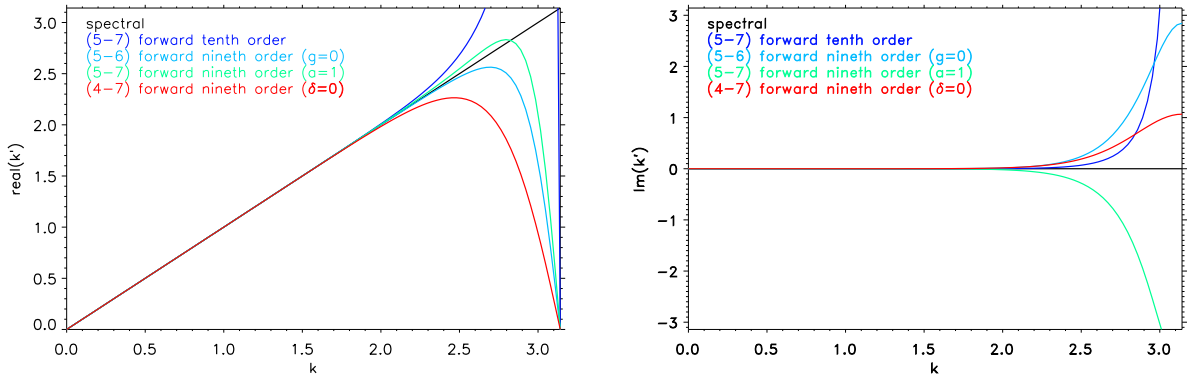

Figure 6: Modified wavenumbers for several explicit and implicit forward finite difference formula.

scheme with a (5,7) stencil is a tenth order formula. However, the spectral analysis of the error shows that the corresponding scheme has a very bad characteristics. This is evidence by the behavior of the dispersive property (real part of (k')) at the highest wavenumbers. The curves of k'_r is located above the spectral one. This behavior is not suited for a scheme as it will increase the small scales (large wavenumbers) errors. However, one can define several families of lower order schemes. By taking into account only the first 10th equations instead of the first 11th (corresponding to the number of unknown coefficients), a family of formula can be defined with one free parameter. This free parameter can be used to define specific nineth order scheme. Taking a parameter randomly will probably not improve significantly the global behavior of the corresponding scheme. This is illustrated by the characteristics of the modified wavenumber for a (5-7) formula with (a=1) which is better than the 10th order scheme but still not suitable for practical use. However, if the free parameter is used to cancel one of the coefficient either in the implicit part or in the explicit part, the corresponding schemes have better characteristics as the real part of the modified wavenumber stay below the spectral one. The dissipative characteristics are also improved but the dissipation is not totally suppressed at large wavenumbers.

One must also be careful as the spectral characteristic of the error is not the only thing to take into account. A specific scheme can have very good spectral behavior but being unusable. The derivative are computed from a system of equation and the system must be well conditioned in order to be inverted. This is the reason why some combinations of stencil are not possible. One must usually use more points for the explicit part of the formula than for the implicit part. For instance a (5-5) stencil forward finite difference scheme is unusable.

The same analysis can be made with finite difference scheme for the second derivative. In this case the real part is associated to the dissipation and the imaginary part to the dispersion. For the same stencil, the highest possible order of the schemes are lower by 1 as compared to

the equivalent schemes for the first derivative.

Compact finite difference scheme can also be derived for irregular grid using the same technique. In this case one must introduce variable steps between each points used by the stencil. This leads to much more complex formula for high order schemes. In practice, with irregular grid, the coefficients have to be computed for each points of the mesh from the system of linear equations.

The spectral characteristic of high order finite difference schemes is similar to the ones of spectral method. This means that Runge instabilities (spurious oscillations) can appear near the boundaries (similarly to what have been found for the interpolation methods). In order to overcome this prejudicial behavior of high order schemes, one must introduce a stretching of the mesh near the boundaries. This has no effect on the order of the scheme, but it may significantly reduce the error level.

3.3 Finite volumes

3.3.1 General formulation

The finite volume method is based on the integration of the equations written in integral form in elementary volumes. This makes the method particularly well suited for the spatial discretisation of the conservation laws. This is the reason why the method is used intensively in fluid mechanics.

Its implementation is rather simple when the elementary volumes (or control volumes) are rectangles or parallelepiped in 3D. However, the finite volume method can be used with any shape of elementary volumes (tetrahedral, hexahedral, prismatic, pyramidal, polyhedral,...). This makes the method well suited for flows in complex geometries (as opposed to the finite difference methods). Most of numerical simulation codes in fluid mechanics are based on this method (FLUENT, StarCD, CFX, elSA, Code Saturn, ...)

Let consider a conservation law of a physical quantity ω inside a mesh of volume Ω and using a flux $F(\omega)$ and a source term $S(w)$. Its expression in integral form is:

$$\frac{\partial}{\partial t}\int_\Omega \omega \; d\Omega + \int_\Omega \text{div}\left(F(\omega)\right) \; d\Omega = \int_\Omega S(\omega) \; d\Omega \tag{175}$$

Let's call Σ the mesh surface with its external norm n. The Ostrogradski theorem leads to:

$$\frac{\partial}{\partial t}\int_\Omega \omega \; d\Omega + \int_\Sigma F \cdot n \; d\Sigma = \int_\Omega S(\omega) \; d\Omega \tag{176}$$

The integral $\displaystyle\int_\Sigma F \cdot n \; d\Sigma$ represents the sum of fluxes through each face of the mesh. The flux is supposed to be constant on each face. As a consequence, the integral becomes a discrete sum on each face of the mesh:

$$\int_\Sigma F \cdot n \; d\Sigma = \sum_{mesh \; face} F_{face} \cdot n_{face} \Sigma_{face} \tag{177}$$

The quantity $F_{face} = F(\omega_{face})$ is an approximation of the flux F on one face of the mesh and is called the **numerical flux** onto the considered face.

The spatial discretisation leads to compute the budget of the fluxes on a elementary mesh. This budget includes the sum of all contributions evaluated on each face of the mesh. The scheme used to approximate the numerical flux as a function of the discrete unknowns determines the numerical scheme. The main characteristic of the discretisation with a finite volume method is to suppose that w **is constant inside each mesh** and is equal to the approximated value of its mean in the mesh or to its value at the center of the mesh.

The temporal derivative is evaluated by using an numerical method of partial differential equation integration (Runge-Kutta, Euler explicit, Euler implicit, ...) including a time step Δt. The time step can be constant or variable in time. As an example, the formula is given with an explicit Euler scheme for time integration. Let's define $\Delta\omega$ being the increment of the quantity ω between two successive time steps. The finite volume scheme can be written as

$$\frac{\partial}{\partial t}\int_\Omega \omega \; d\Omega = \Omega\left(\frac{d\omega}{dt}\right)_{mesh} = \Omega\frac{\Delta\omega}{\Delta t} \tag{178}$$

Eventually, the discretised conservation law using finite volume methods is

$$\Omega \frac{\Delta \omega}{\Delta t} + \sum_{mesh\ face} F_{face} \cdot n_{face} \Sigma_{face} = \Omega S \tag{179}$$

In order to summarize the method, the finite volume method can be decomposed in several steps:

- Decompose the geometry in elementary mesh
- Initialize the quantity ω in the computational domain
- Initiate the temporal integration process :
  - ⋆ computation of the fluxes budget by cell using a numerical scheme
  - ⋆ computation of the source term
  - ⋆ computation of the temporal increment using a time numerical scheme
  - ⋆ application of the boundary conditions

3.3.2 One-dimensional case

Let's consider the 1D conservation law:

$$\frac{\partial}{\partial t} \int u \, dx + \int \frac{\partial f(u)}{\partial x} dx = 0 \tag{180}$$

where u is a physical quantity function of the space variable x and of the time t and $f(u)$ is a function of u.

The computational domain is divided into N meshes centered in x_i. Each mesh is of size $h_i = x_{i+1/2} - x_{i-1/2}$. The half-integer index stands for the interface of the mesh with the neighboring meshes. The time is discretised into constant intervals Δt. The u function is supposed

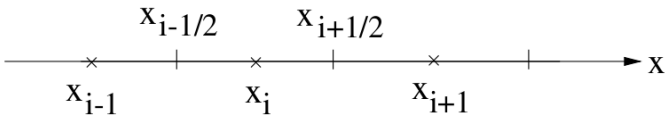

Figure 7: Mesh 1D

constant within each mesh and equal to an approximated value of the mean. Let define u_i^n as the mean value inside the i^{th} mesh centered on x_i at the time $t = n\Delta t$. The approximated value is usually defined as the value of the function u at the center of the mesh. In this case, the method is called **Cell-Centered Finite Volume** (and in this case $u_i^n = u(x_i, t)$).

The spacial discretisation is performed by a mesh-by-mesh integration of the conservation law:

$$\frac{\partial}{\partial t} \int_{cell} u dx + \int_{cell} \frac{\partial f(u)}{\partial x} dx = 0 \tag{181}$$

So, for the ith mesh centered on x_i, at time $t = n\Delta t$, one obtains

$$\frac{\partial}{\partial t}\int_{x_{i-1/2}}^{x_{i+1/2}} u dx + \int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial f(u)}{\partial x} dx = 0 \tag{182}$$

After integration, this leads to

$$h_i \frac{\partial u_i^n}{\partial t} + \hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n = 0 \tag{183}$$

where the term $\hat{f}_{i+1/2}^n$ is an approximation of the flux $f(u)$ at the interface $x_{i+1/2}$ at time $n\Delta t$. This term is called the **numerical flux** at the interface $x_{i+1/2}$ and is evaluated as a function of mean value of u in the neighboring meshes. This determines the numerical scheme. By using, for instance, the explicit Euler method for time integration, the discretised formulation of the conservation law into Finite Volume is:

$$h_i \frac{u_i^{n+1} - u_i^n}{\Delta t} + \hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n = 0 \tag{184}$$

Example with Dirichlet condition

Let's consider the following differential equation:

$$\begin{cases} -u'' = f(x); \quad x \in]0,1[\\ u(0) = \alpha; u(1) = \beta \end{cases} \tag{185}$$

with f a continuous function. The interval $]0,1[$ is discretised into N meshes of center x_i and of size $h_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. The function u is supposed to be constant inside each mesh and equal to the approximated value of the averaged over the considered cell. Let define u_i this value for the ith cell such that for $x \in [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}], u(x) = u_i$.

The spacial discretisation using finite volume consist in a cell-by-cell integration of the differential equations. This leads for the ith cell to:

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u'' = \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} f(x) dx \tag{186}$$

which becomes after integration:

$$u'(x_{i-\frac{1}{2}}) - u'(x_{i+\frac{1}{2}}) = h_i \bar{f}_i \;\text{ for }\; i = 1, ..., N \tag{187}$$

where $\bar{f}_i$ is the average value of f over the ith cell

$$\bar{f}_i = \frac{1}{h_i}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} f(x) dx \tag{188}$$

The next step is to define $u'(x_{i-\frac{1}{2}})$ as a function of the unknowns u_i The most natural choice is to take the average value of $u'(x)$ in the segment $[x_{i-1}, x_i]$ which is:

$$
\begin{aligned}
u'(x_{i-\frac{1}{2}}) &= \frac{1}{\frac{h_{i-1}+h_i}{2}} \int_{x_{i-1}}^{x_i} u'(x)dx \\
&= \frac{u(x_i) - u(x_{i-1})}{h_{i-1/2}} \\
&= \frac{u_i - u_{i-1}}{h_{i-1/2}}
\end{aligned} \tag{189}
$$

with

$$
h_{i-1/2} = \frac{h_{i-1} - h_i}{2} \tag{190}
$$

However, this last expression is not valid for the left boundary condition $x_{1/2}$ ($i = 1$) because it uses the point x_0 which is not defined. The border require a special treatment with a different formulation. One of the possibility is to define a ghost cell at the left of the interval [0,1] and to affect the average value of the function u inside this cell. A second possibility is to consider the averaged value of $u'(x_{1/2})$ on the segment $[x_{1/2}, x_1]$ instead of the segment $[x_0, x_1]$. Let's consider this last choice. We have:

$$
\begin{aligned}
u'(x_{1/2}) &= \frac{2}{h_1} \int_{x_{1/2}}^{x_1} u'(x)dx \\
&= \frac{2(u_1 - u(0))}{h_1} \\
&= \frac{2(u_1 - \alpha)}{h_1}
\end{aligned} \tag{191}
$$

The same problem is present at the right boundary condition at $x_{N+1/2} = 1$. We can proceed the same way be defining the average value of u' in the segment $[x_N, x_{N+1/2}]$ instead of the segment $[x_N, x_{N+1}]$

$$
\begin{aligned}
u'(x_{N+1/2}) &= \frac{2}{h_N} \int_{x_N}^{x_{N+1/2}} u'(x)dx \\
&= \frac{2(u(1) - u_N)}{h_N} \\
&= \frac{2(\beta - u_N)}{h_N}
\end{aligned} \tag{192}
$$

To summarize, the finite volume discretisation is:

$$
\begin{cases}
\dfrac{u_i - u_{i-1}}{h_{i-1/2}} - \dfrac{u_{i+1} - u_i}{h_{i+1/2}} = h_i \bar{f}_i \ \text{ for } \ i = 2, ..., N-1 \\
\dfrac{2(u_1 - \alpha)}{h_1} - \dfrac{u_2 - u_1}{h_{3/2}} = h_1 \bar{f}_1 \\
\dfrac{u_N - u_{N-1}}{h_{N-1/2}} - \dfrac{2(\beta - u_N)}{h_N} = h_N \bar{f}_N
\end{cases} \tag{193}
$$

In the particular case of regular mesh of size h the finite volume method becomes:

$$\begin{cases} \dfrac{2u_i - u_{i-1} - u_{i+1}}{h^2} = \bar{f}_i \ \text{ for } \ i = 2, ..., N-1 \\ \dfrac{3u_1 - u_2}{h^2} = \bar{f}_1 + 2\dfrac{\alpha}{h^2} \\ \dfrac{3u_N - u_{N-1}}{h^2} = \bar{f}_N + 2\dfrac{\beta}{h^2} \end{cases} \tag{194}$$

3.3.3 Two-dimensional case

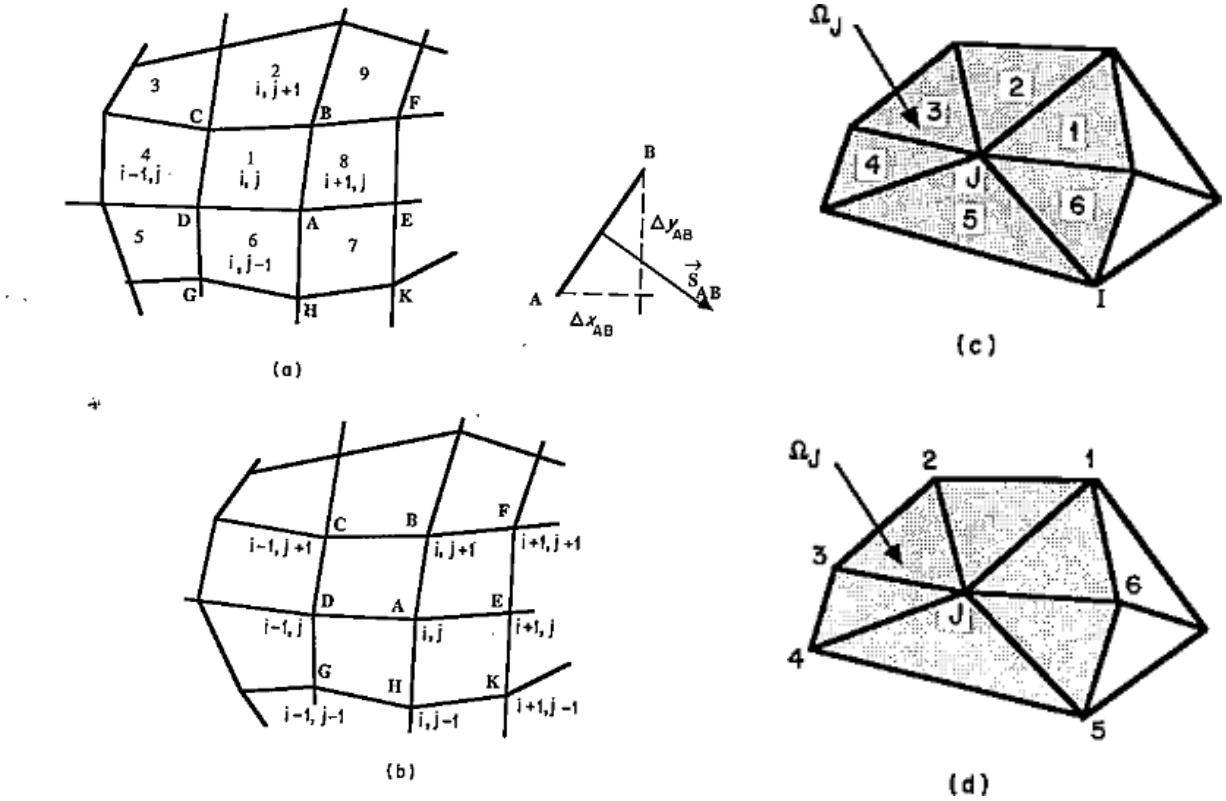

Figure 8: Two-dimensional Finite Volume meshing systems. a) "cell-centered" structured finite volume mesh b) "cell vertex" structured finite volume mesh c) "cell-centered" unstructured finite volume mesh d) "cell vertex" unstructured finite volume mesh ([4])

The conservation equation for discrete volume is

$$\frac{\partial}{\partial t}\int_\Omega U\, d\Omega + \int_S \vec{F}\cdot d\vec{S} = \int_\Omega Q d\Omega \tag{195}$$

When applied to the elementary mesh ABCD of Fig. 8, the equation becomes:

$$\frac{\partial}{\partial t}\int_{\Omega_{ij}} U\, d\Omega + \int_{ABCD}(f\, dy - g\, dx) = \int_{\Omega_{ij}} Q d\Omega \tag{196}$$

where f and g are the Cartesian components of the flux vector $\vec{F}$. The oriented surface vector for the side AB is defined by

$$\vec{S}_{AB} = \Delta y_{AB}\, \vec{i} - \Delta x_{AB}\, \vec{j} = (y_B - y_A)\, \vec{i} - (x_A - x_B)\, \vec{j} \tag{197}$$

The finite volume equation for the cell Ω_{ij} becomes:

$$\frac{\partial}{\partial t}(U\Omega_{ij}) + \sum_{ABCD} \left[f_{AB}(y_B - y_A) - g_{AB}(x_B - x_A)\right] = (Q\Omega)_{ij} \tag{198}$$

The sum $\sum_{ABCD}$ extends over the four sides of the quadrilateral ABCD. For a general quadrilateral ABCD, the area can be evaluated by the vectorial products of the diagonals. By defining $\vec{x}_{AB} = \vec{x}_A - \vec{x}_B$ (where $\vec{x}_A$ is the position vector at the point A), we have

$$\begin{aligned}
\Omega_{ABCD} &= \frac{1}{2}|\vec{x}_{AC} \times \vec{x}_{BD}| \\
&= \frac{1}{2}\left[(x_c - x_A)(y_D - y_B) - (y_C - y_A)(x_D - x_B)\right] \\
&= \frac{1}{2}(\Delta x_{AC}\Delta y_{BD} - \Delta x_{BD}\Delta y_{AC})
\end{aligned} \tag{199}$$

The right-hand side of the equation must be positive for a cell ABCD where A,B,C,D is located counterclockwise.

Evaluation of flux through cell face

The evaluation of the flux component along the sides such as f_{AB}, g_{AB} depends on the selected scheme and on the position of the variable with respect to the cell. We can distinguish between two types of discretisation schemes: "centered" and "upwind" schemes. Centered schemes are based on a local estimation of the fluxes while for upwind schemes, the flux through the cell face is a function of the propagation direction of the wave component.

For central scheme and cell-centered finite volume methods, the following option can be considered for the evaluation of the flux.

(1) Average fluxes :

$$f_{AB} = \frac{1}{2}\left(f_{ij} + f_{i+1,j}\right) \tag{200}$$

$$f_{ij} = f(U_{ij}) \tag{201}$$

This formulation is second order accurate. (2) As the fluxes are generally non-linear function of U, the following choice is not identical to the above formulation of equation (200) :

$$f_{AB} = f\left(\frac{U_{ij} + U_{i+1,j}}{2}\right) \tag{202}$$

(3) One can take for f the average of the fluxes in A and B:

$$f_{AB} = \frac{1}{2}(f_A + f_B) \tag{203}$$

where

$$U_A = \frac{1}{4}(U_{ij} + U_{i+1,j} + U_{i+1,j-1} + U_{i,j-1}) \tag{204}$$

and

$$f_A = f(U_A) \tag{205}$$

or the fluxes are averaged as

$$f_A = \frac{1}{4}(f_{ij} + f_{i+1,j} + f_{i+1,j-1} + f_{i,j-1}) \tag{206}$$

It is important to notice that the equations (202) and (205) will generally lead to schemes which require a lower number of evaluations as compared to the evaluation of fluxes using (200) or (206).

For the central scheme and the cell-vertex method, the equations (202) and (203) are straightforward approximations of the flux f_{AB}. The equation (203) corresponds to the application of the trapezium formula for the integral:

$$\int_{AB} f dy = (f_A + f_B)(y_B - y_A)/2 \tag{207}$$

By summing the contributions of these integrals over the four sides of the cell ABCD of Fig. 8b, the flux term becomes

$$\int_{ABCD} \vec{F} \cdot d\vec{S} = \frac{1}{2}[(f_A - f_C)\Delta y_{DB} + (f_B - f_D)\Delta y_{AC} - (g_A - g_C)\Delta x_{DB} - (g_B - g_D)\Delta x_{AC}] = 0 \tag{208}$$

For the **upwind scheme** and the **cell-centered** method, a convective flux is evaluated as a function of the propagation direction of associated convection speed. The latter is determined by the flux Jacobian:

$$\vec{A}(U) = \frac{\partial \vec{F}}{\partial U} = a\vec{x} + b\vec{y} \tag{209}$$

with $a(U) = \partial f/\partial U$ and $b(U) = \partial g/\partial U$.

The simplest upwind scheme takes the cell side flux equal to the flux generated in the upstream cell. The side flux of the cell is fully determined by the contributions transported in the direction of the convection velocity.

Considering the Fig. 8b, we could define:

$$\begin{array}{ll} (\vec{F} \cdot \vec{S})_{AB} = (\vec{F} \cdot \vec{S})_{ij} & \text{if} \quad (\vec{A} \cdot \vec{S})_{AB} > 0 \\ (\vec{F} \cdot \vec{S})_{AB} = (\vec{F} \cdot \vec{S})_{i+1,j} & \text{if} \quad (\vec{A} \cdot \vec{S})_{AB} < 0 \end{array} \tag{210}$$

For **upwind scheme** and a **cell-vertex** method a possibility is to define:

$$\begin{array}{ll} (\vec{F} \cdot \vec{S})_{AB} = (\vec{F} \cdot \vec{S})_{CD} & \text{if} \quad (\vec{A} \cdot \vec{S})_{AB} > 0 \\ (\vec{F} \cdot \vec{S})_{AB} = (\vec{F} \cdot \vec{S})_{EF} & \text{if} \quad (\vec{A} \cdot \vec{S})_{AB} < 0 \end{array} \tag{211}$$

The upwind schemes are the only approximations that satisfy the boundedness criteria unconditionally, *i.e.,* it never yield oscillatory solutions. However it is numerically diffusive. The numerical

diffusion is magnified in multidimensional problems if the flow is oblique to the grid. The truncation error produces diffusion in the direction normal to the flow as well as in the streamwise direction, peaks or rapid variations in the variables will be smeared out and, since the rate of error reduction is only first order, very fine grids are required to obtain accurate solutions.

Another straightforward approximation of the value at the cell volume center is a linear interpolation between the two nearest nodes. At the middle $x_{i+1/2,j}$ of segment AB on a Cartesian grid we have:

$$f_{AB} = \lambda f_{i+1,j} + (1-\lambda) f_{ij} \tag{212}$$

where the linear interpolation factor λ is defined as

$$\lambda = \frac{x_{i+1/2,j} - x_{i,j}}{x_{i+1,j} - x_{i,j}} \tag{213}$$

The equation (212) is second order accurate. This scheme is the simplest second order accurate and is the most widely used. However, as all approximations of order higher than one, the scheme may produce oscillatory solutions.

The next logical improvement is the approximation to variable profile between $x_{i,j}$ and $x_{i+1,j}$ by a parabola instead by a linear function. In order to construct a parabola, one need to use the data at one more point. In accordance with the upwind scheme, this point is taken on the upstream side, *i.e.,* $x_{i-1,j}$ if the flow is from $x_{i,j}$ to $x_{i+1,j}$ $((\vec{A}\cdot\vec{S})_{AB} > 0)$ or $x_{i+1,j}$ if the flow is from $x_{i+1,j}$ to $x_{i,j}$ $((\vec{A}\cdot\vec{S})_{AB} < 0)$.

The corresponding interpolation has a third order truncation error on both uniform an non-uniform grids. For $((\vec{A}\cdot\vec{S})_{AB} > 0)$, the scheme leads to

$$f_{AB} = \frac{6f_{i-1,j} + 3f_{i,j} + f_{i+1,j}}{8} \tag{214}$$

The approximation is called the **Quadratic Upwind Interpolation (QUICK)** approximation. Even if it is third order accurate, when this scheme is used in conjunction with the midpoint rule approximation of the surface integral (in 2D), the overall approximation is however still second order accurate.

3.4 Introduction to finite elements methods

The method consist in approaching, in a finite dimension sub-space, a problem defined in a variational form in an infinite dimension space. This variational form is equivalent to a form of minimization of the energy. In this case, the approximated solution is defined by a finite number of parameters as , for instance, the values at given grid points. This method is well suited for problems in equilibrium and allows to deal with complex geometries but is expensive in terms of CPU time and memory requirement. Several commercial code are based on the finite element methods (ANSYS, CATIA, ...). In this course we will restrict ourselves to a brief introduction to this method based on an example in one dimension. A lot of theoretical results can be demonstrated on this method.

3.4.1 Simple case in 1D

Let's take a simple differential equation

$$\begin{cases} -u''(x) = f(x), & x \in]0,1[\\ u(0) = u(1) = 0 \end{cases}$$

The method is based on the Galerkin method which allows to write the differential system into a variational form in a finite dimension space.

Let's define a function $v(x) \in \mathbb{C}^1([0,1])$, such that $v(0) = v(1) = 0$, on can write:

$$-\int_0^1 u''(x)v(x)\,dx = \int_0^1 f(x)v(x)\,dx$$

By integrating by part, we obtain

$$\int_0^1 u'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx \quad \forall v \in V \tag{215}$$

with $V = v \in \mathbb{C}^0([0,1]);\ \ v(0) = v(1) = 0,\ \ v'$ piecewise continuous, a vectorial subspace of $\mathbb{C}^1([0,1])$. A solution in variational form (215) is named a **weak solution** of the original differential problem.

Then, we need to write the approximated problem in a finite dimension vectorial sub-space. Let's define $\tilde{V}$ a vectorial sub-space of V of finite dimension N. Let $\phi_1, \phi_2, ..., \phi_n$ be N functions linearly independent of V; These functions create a basis for the sub-space $\tilde{V}$. Thus, any function $\tilde{u}$ of $\tilde{V}$ can be decomposed as:

$$\tilde{u}(x) = \sum_{j=1}^{N} u_j \phi_j(x)$$

Solving the original differential system leads to find a solution $\tilde{u} \in \tilde{V}$ such that

$$\int_0^1 \tilde{u}'(x)\tilde{v}'(x)\,dx = \int_0^1 f(x)\tilde{v}(x)\,dx \quad \forall \tilde{v} \in \tilde{V}$$

This means looking for N real $u_1, u_2, u_3, ..., u_N$ that verify:

$$\sum_{j=1}^{N} u_j \int_0^1 \phi_j'(x)\tilde{v}'(x)\,dx = \int_0^1 f(x)\tilde{v}(x)\,dx \quad \forall \tilde{v} \in \tilde{V}$$

or

$$\sum_{j=1}^{N} u_j \int_0^1 \phi_j'(x)\phi_i'(x)\,dx = \int_0^1 f(x)\phi_i(x)\,dx \quad \forall \tilde{\phi}_i \in \tilde{V}$$

Let's define A the $N \times N$ matrix of elements a_{ij} and B the N components vector defined by:

$$a_{ij} = \int_0^1 \phi_j'(x)\phi_i'(x)\,dx \text{ et } b_i = \int_0^1 f(x)\phi_i(x)\,dx \tag{216}$$

By definition, the matrix $\mathbf{A}$ is symmetric. If $\mathbf{u}$ is the vector of the N unknown $u_1, u_2, u_3, ..., u_N$, the differential problem reduces to the resolution of the linear system

$$\mathbf{A} \cdot \mathbf{u} = \mathbf{b} \tag{217}$$

One must choose the N functions ϕ_i in order to lead to a simple system to solve:

3.4.2 Choice of the finite elements ϕ_i

Let's consider an interval $]0, 1[$ divided into N points of coordinate x_i. The functions $\phi_i(x)$ are chosen as polynomial functions of degree 1 (see fig. 9) defined by:

$$\phi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & \text{if} \quad x_{i-1} \leq x \leq x_i \\ \dfrac{x - x_{i+1}}{x_i - x_{i+1}} & \text{if} \quad x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

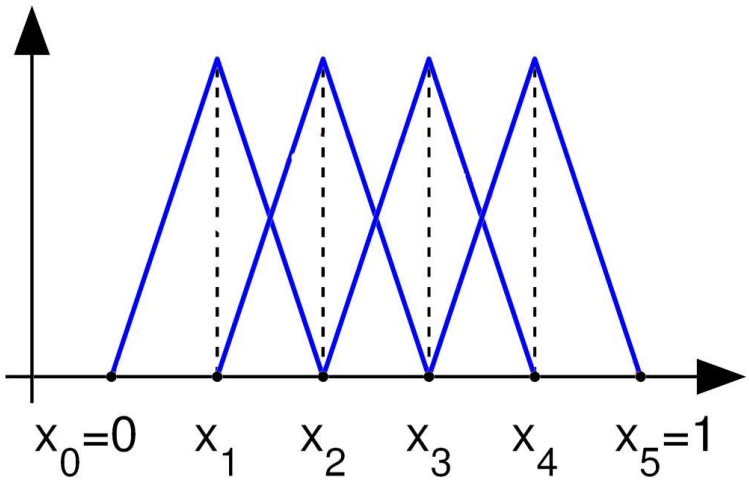

Figure 9: 1D finite elements (functions ϕ_i)

These function are named **one degree finite elements**. Using these finite elements, the matrix $\mathbf{A}$ is tri-diagonal. However, other choices with higher degree finite elements are possible. The computation of the matrix A uses derivatives $\phi_i'(x)$ that are easy to compute :

$$\phi_i'(x) = \begin{cases} \dfrac{1}{x_i - x_{i-1}} & \text{if} \quad x_{i-1} \leq x \leq x_i \\ \dfrac{1}{x_i - x_{i+1}} & \text{if} \quad x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

Let's now compute the elements of the matrix $\mathbf{A}$. The three coefficients on the diagonals are:

$$\begin{aligned} a_{ii} &= \int_0^1 \phi_i'(x)\phi_i'(x)\,dx = \frac{1}{x_i - x_{i-1}} + \frac{1}{x_{i+1} - x_i} \\ a_{i,i+1} &= \int_0^1 \phi_{i+1}'(x)\phi_i'(x)\,dx = \frac{-1}{x_{i+1} - x_i} \\ a_{i,i-1} &= \int_0^1 \phi_i'(x)\phi_{i-1}'(x)\,dx = \frac{-1}{x_i - x_{i-1}} \end{aligned}$$

The elements of the vector $\mathbf{b}$ can be evaluated using the trapezoidal formula ($\int_a^b g(x)dx = \frac{(g(a)+g(b)}{2}(b-a)$)

$$b_i = \int_0^1 f(x)\phi_i(x) = f_i\left(\frac{x_{i+1} - x_{i-1}}{2}\right)$$

Thus, the linear system to solve is:

$$\frac{u_i - u_{i-1}}{x_i - x_{i-1}} - \frac{u_{i+1} - u_i}{x_{i+1} - x_i} = \frac{x_{i+1} - x_{i-1}}{2} f_i \quad i = 1, ..., N$$

In order to compare with the finite difference method, the above formula can be compared to the central second order formula

$$\frac{u_i - u_{i-1}}{x_i - x_{i-1}} - \frac{u_{i+1} - u_i}{x_{i+1} - x_i} = \frac{x_{i+1} - x_{i-1}}{2} f_i \quad i = 1, ..., N$$

One can check that the two methods are rigorously identical with this choice of finite elements. However, this is not true anymore when the elements of $\mathbf{b}$ are evaluated with a method different from the trapezoidal one.

In the case where the N points are regularly distributed in the $]0, 1[$ interval (with a constant space increment Δx), the finite element discretisation becomes:

$$\frac{-u_{i+1} + 2u_i - u_{i-1}}{(\Delta x)^2} = f_i \quad i = 1, ..., N$$

To summarize, the finite element methods consist in :

- Choosing N points between 0 and 1 and choosing the finite elements ϕ_i
- Compute the elements of the matrix $\mathbf{A}$
- Compute the elements of the vector $\mathbf{b}$ (by choosing an integration formula)
- Solving the linear system $\mathbf{A} \cdot \mathbf{u} = \mathbf{b}$ where $\mathbf{u}$ is the vector of unknowns

4 Solving of Linear System

4.1 Some definitions on matrix

Let's define a matrix $\mathbf{A}$ with m lines and n columns

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \tag{218}$$

for a square matrix ($n = m$) the trace of $\mathbf{A}$ is define by:

$$tr(\mathbf{A}) = \sum_{i=1}^{n} a_{ii} \tag{219}$$

and the determinant is defined as:

$$det(\mathbf{A}) = \begin{cases} a_{11} \text{ if } n = 1 \\ \sum_{j=1}^{n} \Delta_{ij} a_{ij} \text{ for } n > 1 \end{cases} \tag{220}$$

with $\Delta_{ij} = (-1)^{i+j} det(\mathbf{A}_{ij})$ and $\mathbf{A}_{ij}$ is the matrix of order $n-1$ obtained from $\mathbf{A}$ by removing the i^{th} line and the j^{th} column.

The q order extracted determinant of a matrix $\mathbf{A}$ is the determinant of any matrix of order q obtained from $\mathbf{A}$ by removing $m-q$ lines and $n-q$ column.

The *rank* of the matrix $\mathbf{A}$ (rg($\mathbf{A}$)) is the maximum order of the non zero extracted determinant. It is also the maximum number of independent column vectors of $\mathbf{A}$.

For $\mathbf{A}$ a real or complex square matrix of order n, λ is a eigenvalue of $\mathbf{A}$ if it exists a vector $\mathbf{x} \neq 0$ such as $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$. The $\mathbf{x}$ vector is called the eigenvectors associated to the eigenvalue λ and $\sigma(\mathbf{A})$ is the ensemble of all eigenvalues of $\mathbf{A}$. The eigenvalues are solutions of the following characteristic polynomial:

$$p(\lambda) = det(\mathbf{A} - \lambda I) = 0 \tag{221}$$

One can show that

$$det(\mathbf{A}) = \prod_{i=1}^{n} \lambda_i, \quad tr(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i \tag{222}$$

The matrix $\mathbf{A}$ is called *singular* if it has at least one null eigenvalue. The *spectral radius* of $\mathbf{A}$ is defined by:

$$\rho(\mathbf{A}) = \max_{\lambda \in \sigma(\mathbf{A})} |\lambda| \tag{223}$$

Some properties of a matrix:

Symmetric	: $\mathbf{A}^T = \mathbf{A}$
Orthogonal	: $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = 1$
Hermitian	: $\mathbf{A} = \mathbf{A}^*$
Normal	: $\mathbf{A}\mathbf{A}^* = 1$

Definition of some norms:

$$\|\mathbf{A}\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{ij}| \tag{224}$$

$$\|\mathbf{A}\|_\infty = \max_{i=1,\dots,n} \sum_{j=1}^{n} |a_{ij}| \tag{225}$$

$$\|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A}^*\mathbf{A})} = \sigma_1(\mathbf{A}) = \text{ largest singular value of } \mathbf{A} \tag{226}$$

The matrix $\mathbf{A}$ is said to be diagonal dominant if:

$$\begin{cases} |a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1, .., n \text{ (dominant by line)} \\ \\ |a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ji}|, \quad i = 1, .., n \text{ (dominant by column)} \end{cases} \tag{227}$$

Decomposition in singular values:

Any matrix $\mathbf{A}$ can be put in diagonal form by multiplying each side by well defined unity matrix.

Let consider $\mathbf{A} \in \mathbb{C}^{n\times m}$, one can find two unity matrix $\mathbf{U} \in \mathbb{C}^{m\times m}$ and $\mathbf{V} \in \mathbb{C}^{n\times n}$ such as:

$$\mathbf{U}^*\mathbf{A}\mathbf{V} = \Sigma = diag(\sigma_1, \sigma_2, ...\sigma_p) \in \mathbb{R}^{n\times m} \text{ with } p = \min(m, n) \tag{228}$$

and $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_p \geq 0$. This relation is called *decomposition in singular values* of $\mathbf{A}$ and the scalar σ_i are called *singular values* of $\mathbf{A}$.

4.2 Introduction to linear System

A linear system of m equations and n unknown is a set of algebraic relations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, ..., m, \tag{229}$$

where x_j are the unknown and the a_{ij} are the system coefficients and b_i the component of the right-hand side. The system can be written in matrix form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{230}$$

where $\mathbf{A} = (a_{ij})$ is the coefficient matrix and $\mathbf{b} = (b_i)$ is the right-hand side vector, and $\mathbf{x}$ is the unknown vector. In this chapter, we will focus on square system of order n (such as $n = m$). In this case, the existence and the unicity of the solution is demonstrated if one of the following conditions is satisfied:

1. $\mathbf{A}$ is invertible
2. $rg(\mathbf{A}) = n$

3. the homogeneous system $\mathbf{Ax} = 0$ has only the null solution.

The solution of the system (230) is given by the **Cramer's formula**:

$$x_j = \frac{\Delta j}{det(\mathbf{A})} \tag{231}$$

where Δ_j is the determinant of the matrix obtained by replacing the jth column of $\mathbf{A}$ by $\mathbf{b}$. However, this formula is of limited interest as, if the determinant is evaluated by the recurrence formula of **Laplace**, the global computational cost of the Cramer's formula is of the order of $(n+1)$ which is prohibitive even for small dimension system (10^{66} operations for $n = 50$). This is the reason why alternative methods have been developed to solve linear system. The methods are qualified as **direct methods** if they are able to give the exact solution by a finite number of operations and they are called as **iterative methods** if they required an infinite number of steps. The choice between direct or iterative methods is function of the theoretical efficiency of the algorithm but also to the size of the system, the available computing ressources (CPU and memory).

The resolution of a linear system with a numerical method leads to the introduction of round-off errors. Only stable methods can be used to avoid to deteriorate the solution by the propagation of such round-off errors. The stability of the method with respect to the propagation of errors is able to quantify the sensibility of the result to the perturbations of $\mathbf{A}$ and $\mathbf{b}$.

4.3 Well posed problems

Let's consider the following problem: find x such as

$$F(d, x) = 0 \tag{232}$$

where d is the ensemble of data from which the solution is dependent and F is the functional relation between x and d. Depending of the problem, the variables x and d can be real numbers, vectors or functions. The problem is *well posed* (or *stable*) if the solution x exist, is unique and continuously depends on the data d. A problem which does not satisfy this condition is called *ill-posed* or *unstable*. The problem which are ill-posed must be regularized before to try to solve it. No numerical method will be able to solve correctly a problem which is ill-posed or unstable.

The propriety of linear dependence with respect of the data means that small perturbations on the data induce small modifications of the solution. Let's define δd a small perturbation of the data and δx the induce modification of the solution. The propriety of linear dependence with respect of the data can be defined by:

$$\forall \epsilon > 0, \quad \exists\ \delta(\epsilon) \text{ such as } \|\delta d\| \leq \delta \quad \text{then} \quad \|\delta x\| \leq \epsilon \tag{233}$$

where D is the ensemble of data.

In order to introduce a more quantitative definition, one can define the *relative conditioning* propriety:

$$K(d) = sup \left\{ \frac{\|\delta x\|/\|x\|}{\|\delta d\|/\|d\|}, \delta d \neq 0, d + \delta d \in D \right\}. \tag{234}$$

When $x = 0$ and $d = 0$ the *absolute conditioning* can be introduced instead:

$$K(d) = sup \left\{ \frac{\|\delta x\|}{\|\delta d\|}, \delta d \neq 0, d + \delta d \in D \right\}. \tag{235}$$

The problem is *hill conditioned* if $K(d)$ is "large" for any admissible data d. The term "large" need to be considered with respect to the problem.

4.4 Conditioning

Let's consider the linear system $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{x}$ and $\mathbf{b}$ are two vectors of $\mathbb{R}^n$ and $\mathbf{A}$ is the $n \times n$ matrix of the system coefficients. For $\mathbf{A}$ invertible, x is the solution of the unknowns $\mathbf{x}$ and d is the rhs vector $\mathbf{b}$ and the matrix $\mathbf{A}$. Let suppose that only the rhs vector are perturbed, we have $d = \mathbf{b}$, $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ and the conditioning of the system can be estimated by

$$K(d) \simeq \frac{\|\mathbf{A}^{-1}\|\|\mathbf{b}\|}{\|\mathbf{A}^{-1}\mathbf{b}\|} = \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}\|\mathbf{A}^{-1}\| \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\| = K(A) \tag{236}$$

where $K(A)$ is the conditioning of the matrix which is defined by:

$$K_p(\mathbf{A}) = \|\mathbf{A}\|_p\|\mathbf{A}^{-1}\|_p \tag{237}$$

where $\|.\|_p$ is the p-norm and $K_p(\mathbf{A})$ is the conditioning of the matrix $\mathbf{A}$ under the p-norm. The remarkable cases are for $p = 1$, $p = 2$ and $p = \infty$. The bigger is the conditioning, the more sensible is the solution of the system to the perturbations. The equation (236) indicates that the conditioning of the linear system $K(d)$ is linked to the conditioning of the matrix $K(\mathbf{A})$.

One can demonstrate several properties of the conditioning:

$$\begin{aligned} &1 = \|\mathbf{A}\mathbf{A}^{-1}\| \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\| = K(\mathbf{A}) \\ &K(\mathbf{A}) = K(\mathbf{A}^{-1}) \\ &K(\alpha\mathbf{A}) = K(\mathbf{A}) \end{aligned} \tag{238}$$

(239)

Moreover for the norm $p = 2$ we have

$$K_2(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2 = \frac{\sigma_1(\mathbf{A})}{\sigma_n(\mathbf{A})} \tag{240}$$

where $\sigma_1(\mathbf{A})$ and $\sigma_n(\mathbf{A})$ are the largest and the smallest singular values of $\mathbf{A}$ respectively.

A-priori analysis

Because of the round-off errors, a numerical method to solve the linear system $\mathbf{Ax} = \mathbf{b}$ does not provide the exact solution but only an approximated solution which satisfy a perturbed system. So, a numerical methods gives an exact solution $(\mathbf{x} + \delta\mathbf{x})$ of the perturb system:

$$(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b} \tag{241}$$

In such case we have the following relation:

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{K(\mathbf{A})}{1 - K(\mathbf{A})\|\delta \mathbf{A}\|/\|\mathbf{A}\|} \left(\frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|} \right) \tag{242}$$

In the case where $\delta \mathbf{A} = 0$, the relative error can be bounded by:

$$\frac{1}{K(\mathbf{A})} \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|} \leq \frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq K(\mathbf{A}) \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|} \tag{243}$$

4.5 Direct methods

4.5.1 Resolution of a triangular system

In the case of a system $\mathbf{Lx} = \mathbf{b}$ where $\mathbf{L}$ is a lower triangular invertible matrix of order n ($n > 2$) the direct method to solve the system is:

$$\begin{aligned} x_1 &= \frac{b_1}{l_{11}} \\ x_i &= \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right), \ i = 2, ..., n \end{aligned} \tag{244}$$

The algorithm leads to $n(n+1)/2$ multiplications and divisions and $n(n-1)/2$ additions or subtractions. The total number of operation is of the order of n^2.

The case of a system $\mathbf{Ux} = \mathbf{b}$ where $\mathbf{U}$ is a upper triangular invertible matrix can be treated similarly:

$$\begin{aligned} x_n &= \frac{b_n}{u_{nn}} \\ x_i &= \frac{1}{u_{ii}} \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right), \ i = n-1, ..., 1 \end{aligned} \tag{245}$$

The global cost of this algorithm is also of the order n^2.

4.5.2 Gauss Elimination

The Gauss elimination method is based on a transformation of the original system $\mathbf{Ax} = \mathbf{b}$ into an equivalent system $\mathbf{Ux} = \hat{\mathbf{b}}$, where $\mathbf{U}$ is a upper triangle matrix and $\hat{\mathbf{b}}$ is the modified right hand side. This last system can be solved as shown earlier. The transformation use the property such that the solution of the system is unchanged when a linear combination of several equations is added to an other equation. Let's consider an invertible matrix $\mathbf{A}$ such as $a_{11} \neq 0$. Let's indicate the first step of the algorithm by a superscript such that $\mathbf{A} = \mathbf{A}^{(1)}$ and $\mathbf{b} = \mathbf{b}^{(1)}$. We can define the multiplicative:

$$m_{i,p} = \frac{a_{ip}^{(p)}}{a_{pp}^{(p)}} \tag{246}$$

where the $a_{ip}^{(p)}$ are the elements of $\mathbf{A}^{(p)}$. The unknown x_p can be eliminated from the lines $i = p+1, ..., n$ by subtracting m_{ip} times the first line and by doing the same for the right hand side. For the first step, the elements of the new system are defined by:

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, \quad i,j = 2, ..., n, \tag{247}$$

$$b_i^{(2)} = b_i^{(1)} - m_{i1}b_1^{(1)}, \quad i = 2, ..., n, \tag{248}$$

The new system

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \dots & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix} \tag{249}$$

can be written as $\mathbf{A}^{(2)}\mathbf{x} = \mathbf{b}^{(2)}$ and is equivalent to the first one. The next step of the algorithm is to use the same procedure to eliminate the unknown x_2 of the lines $3, ..., n$. By applying the same algorithm $n-1$ times, we obtain the following upper triangular system $\mathbf{A}^{(n)}\mathbf{x} = \mathbf{b}^{(n)}$:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & a_{2n}^{(2)} \\ \vdots & & \ddots & & \vdots \\ 0 & & & \ddots & \vdots \\ 0 & & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ \vdots \\ b_n^{(n)} \end{bmatrix} \tag{250}$$

The Gauss method requires that $a_{kk}^{(k)} \neq 0, \quad 1 \leq k \leq n-1$. The fact that the diagonal coefficients of the matrix $\mathbf{A}$ are not zero does not prevent the appearance of null $a_{kk}^{(k)}$. For instance, the following matrix $\mathbf{A}$ is invertible and none of its diagonal coefficients is zero but

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix}, \quad \mathbf{A}^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & -1 \\ 0 & -6 & -12 \end{bmatrix} \tag{251}$$

The Gauss method must be stopped at the second step as $a_{22}^{(2)} = 0$. However, the Gauss method can be used without problem for the following categories of matrix:

- the matrix with a dominant diagonal by line
- the matrix with a dominant diagonal by column
- the positive defined symmetrical matrix

The global cost of the Gauss method is proportional to $n^3/3$. The bulk of this method is in the forward elimination phase as the back substitution requires only $n^2/2$ arithmetic operations. Gauss elimination is thus expensive, but, for a full matrix, it is as good as any method available. For large systems that are not sparse, Gauss elimination is susceptible to accumulation errors which make it unreliable if not modified. Moreover, the Gauss method does not vectorize or parallelize well and is rarely used without modification in CFD.

4.5.3 LU decomposition

In fact, the Gauss method is equivalent to the factorization of the matrix $\mathbf{A}$ by the product of two matrix $\mathbf{A} = \mathbf{LU}$ with $\mathbf{U} = \mathbf{A}^{(n)}$, the final step of the Gauss method. The matrix $\mathbf{L}$ and $\mathbf{U}$ only depend on $\mathbf{A}$ and not on the right hand side $\mathbf{b}$. Thus, the same factorization can be used when we want to solve several systems with the same matrix but different right hand sides $\mathbf{b}$. The number of operation is therefore significantly reduced as the main cost $2n^3/3$ comes from the Gauss elimination procedure.

In the Gauss procedure, the operation to define $\mathbf{A}^{(k+1)}$ from $\mathbf{A}^k$ can be obtained by the following multiplication:

$$\mathbf{A}^{(k+1)} = \mathbf{M}_k \mathbf{A}^k \tag{252}$$

where the k^{th} Gauss transformation matrix $\mathbf{M}_k$ is defined by:

$$\mathbf{M}_k = \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & & 1 & 0 & & 0 \\ 0 & & -m_{k+1,k} & 1 & & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -m_{n,k} & 0 & \dots & 1 \end{bmatrix} \tag{253}$$

where the components of the transformation matrix $\mathbf{M}_k$ are defined by:

$$(\mathbf{M}_k)_{ip} = \delta_{ip} - m_{ik}\delta_{kp} \tag{254}$$

The Gauss elimination procedure generates the matrix $\mathbf{M}_k$, $k = 1, ..., n-1$ and the matrix $\mathbf{U}$ satisfying the following relation:

$$\mathbf{M}_{n-1}\mathbf{M}_{n-2}\dots\mathbf{M}_1\mathbf{A} = \mathbf{U}. \tag{255}$$

Due to the properties of the matrix $\mathbf{M}_k$, we have the following relation:

$$\mathbf{A} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1}\dots\mathbf{M}_{n-1}^{-1}\mathbf{U} = \mathbf{LU} \tag{256}$$

with

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ m_{21} & 1 & & & \vdots \\ \vdots & m_{32} & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ m_{n1} & m_{n2} & \dots & m_{n,n-1} & 1 \end{bmatrix} \tag{257}$$

Once the matrix $\mathbf{L}$ and $\mathbf{U}$ are defined, solving the linear system $\mathbf{Ax} = \mathbf{b}$ is equivalent to solve successively two triangular systems:

$$\begin{aligned} \mathbf{Ly} &= \mathbf{b} \\ \mathbf{Ux} &= \mathbf{y} \end{aligned} \tag{258}$$

The cost of the LU factorization is the same the the Gauss method but many system involving the same matrix can be solve at a low price using the same LU decomposition.

4.5.4 Thomas's algorithm

Using finite difference discretisations usually leads to solve special linear systems with a corresponding matrix $\mathbf{A}$ with non-zero coefficients only in the diagonal and the diagonal immediately above and below it. Such a matrix is called *tri-diagonal* matrix and the corresponding system is pecular to solve. The matrix are usually stored into three 1D arrays. Gauss elimination are especially easy for tri-diagonal systems as only one element needs to be eliminated from each row during the forward elimination.

Let's consider a tri-diagonal matrix $\mathbf{A}$ defined as:

$$\mathbf{A} = \begin{bmatrix} a_1 & c_1 & & 0 \\ b_2 & a_2 & \ddots & \\ & \ddots & & c_{n-1} \\ 0 & & b_n & a_n \end{bmatrix} \tag{259}$$

In this case, the matrix $\mathbf{L}$ and $\mathbf{U}$ from the LU factorization of $\mathbf{A}$ are bi-diagonal matrix:

$$\mathbf{L} = \begin{bmatrix} 1 & & & 0 \\ \beta_2 & 1 & & \\ & \ddots & \ddots & \\ 0 & & \beta_n & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \alpha_1 & c_1 & & 0 \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ 0 & & & \alpha_n \end{bmatrix} \tag{260}$$

where the coefficients α_i and β_i are defined by:

$$\alpha_1 = a_1, \quad \beta_i = \frac{b_i}{\alpha_{i-1}}, \quad \alpha_i = a_i - \beta_i c_{i-1}, i = 2, ..., n \tag{261}$$

This algorithm is known as the *Thomas algorithm*. It can be used to solve a tri-diagonal system. Using the LU factorization, the system can be solved using the following formula:

$$y_1 = b_1, \quad y_i = b_i - \beta_i y_{i-1}, \quad i = 2, ..., n \tag{262}$$

$$x_n = y_n/\alpha_n, \quad x_i = (y_i - c_i x_{i+1})/\alpha_i, \quad i = n-1, ..., 1. \tag{263}$$

The Thomas algorithm requires only $8n - 7$ operations.

4.5.5 Generalization for 5 bands matrix system

4.6 Linear Iterative methods

Iterative methods give the solution $\mathbf{x}$ of a linear system after an infinite number of iterations. Each step requires a number of operations which is of the order of n^2 when the cost of a direct method is of the order of $2n^3/3$. Therefore, iterative methods become useful if they can converge with a number of steps independent of n or following a function of n lower than the linear function. Furthermore, the discretisation error is usually much higher than the accuracy of the computer arithmetic so there is no reason to solve the system that accurately. This leaves an opening for iterative methods.

4.6.1 Convergence

The basic idea of the iterative method is to construct an iterative relation of vector $\mathbf{x}^{(k)}$ such as

$$\mathbf{x} = \lim_{k \to \infty} \mathbf{x}^{(k)} \tag{264}$$

where $\mathbf{x}$ is the solution of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$. In practice, the computation must be stopped at the first iteration p such as $\|\mathbf{x}^{(p)} - \mathbf{x}\| < \epsilon$ where ϵ is a convergence parameter and $\|.\|$ is vectorial norm. However, as the exact solution is not known, one must define more practical convergence criteria.

Consider the matrix problem

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{265}$$

After k iterations we have an approximated solution $\mathbf{x}^{(k)}$ which does not satisfy this equation exactly. Instead, there is a non-zero residual $r^{(k)}$

$$\mathbf{A}\mathbf{x}^{(k)} = \mathbf{b} - r^{(k)} \tag{266}$$

By subtracting this equation to the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$, we obtain the relation between the convergence error

$$\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)} \tag{267}$$

where $\mathbf{x}$ is the converged solution and the residual

$$\mathbf{A}\mathbf{e}^{(k)} = r^{(k)} \tag{268}$$

The purpose of the iteration procedure is to drive the residual to zero; in the process, ϵ also becomes zero. To see how it can be done, consider an iterative scheme for a linear system which can be written as:

$$\mathbf{M}\mathbf{x}^{(k+1)} = \mathbf{N}\mathbf{x}^{(k)} + \mathbf{q} \tag{269}$$

or

$$\mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{q} \tag{270}$$

where $\mathbf{B}$ is the iteration matrix.

The iterative method must lead to a solution which converges to the solution of the original system (265). Since, by definition, at convergence, $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} = \mathbf{x}$, we must have:

$$\mathbf{A} = \mathbf{M} - \mathbf{N} \quad \text{and} \quad \mathbf{q} = \mathbf{b} \tag{271}$$

or more generally

$$\mathbf{P}\mathbf{A} = \mathbf{M} - \mathbf{N} \quad \text{and} \quad \mathbf{q} = \mathbf{P}\mathbf{b} \tag{272}$$

where $\mathbf{P}$ is a non-singular pre-conditioning matrix.

For an iterative method to be effective, solving the system (269) must be cheap and the method must converge rapidly. Inexpensive iteration requires that the computation of $\mathbf{N}\ \mathbf{x}^{(k)}$ and the solution of the system must be easy to perform. The second requirement implies that $\mathbf{M}$ must be easily inverted. From a practical point of view, $\mathbf{M}$ should be diagonal, tri-diagonal or triangular (or block diagonal, block tri-diagonal or block triangular). For rapid convergence, $\mathbf{M}$ should be a good approximation of $\mathbf{A}$, making $\mathbf{N}\mathbf{x}$ small in some sens.

In the following, we will presents several linear iterative methods.

4.6.2 Jacobi method

If the diagonal coefficients of the matrix $\mathbf{A}$ are not zero, the unknown x_i from the i^{th} equation can be extracted as follow:

$$x_i = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j \right], \quad i = 1, ..., n \tag{273}$$

In the *Jacobi method*, starting from an arbitrary initial value $\mathbf{x}^0$, the solution $\mathbf{x}^{k+1}$ is computed with the following recurrence formula:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} \right], \quad i = 1, ..., n \tag{274}$$

This relation is equivalent to the following decomposition of the matrix $\mathbf{A}$:

$$\mathbf{M} = \mathbf{D}, \;\; \mathbf{N} = \mathbf{D} - \mathbf{A} = \mathbf{E} + \mathbf{F} \tag{275}$$

where $\mathbf{D}$ is the diagonal matrix made of the diagonal coefficients of $\mathbf{A}$, $\mathbf{E}$ is a upper triangular matrix of coefficients

$$\begin{array}{rcrl} e_{ij} & = & -a_{ij} & \text{if} \quad i > j \\ e_{ij} & = & 0 & \text{if} \quad i \leq j \end{array}$$

and $\mathbf{F}$ is a lower triangular matrix of coefficients

$$\begin{array}{rcrl} f_{ij} & = & -a_{ij} & \text{if} \quad j > i \\ f_{ij} & = & 0 & \text{if} \quad j \leq i \end{array}$$

The iteration matrix $\mathbf{B}_J$ (such as $\mathbf{x}^{(k+1)} = \mathbf{B}_J \mathbf{x}^{(k+1)} + \mathbf{D}^{-1}\mathbf{b}$) for the Jacobi method is given by:

$$\mathbf{B}_J = \mathbf{M}^{-1}\mathbf{N} = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F}) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} \tag{276}$$

A generalization of the Jacobi method is the *Jacobi over relaxation* (JOR) method where a relaxation parameter ω is introduced. The new recurrence formula is:

$$x_i^{k+1} = \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^k \right] + (1 - \omega) x_i^{(k)}, \quad i = 1, ..., n \tag{277}$$

The corresponding iteration matrix is:

$$\mathbf{B}_{JOR}(\omega) = \omega \mathbf{B}_J + (1 - \omega)\mathbf{I}. \tag{278}$$

For $\omega = 1$, the method is equivalent to the standard Jacobi method.

4.6.3 Gauss-Seidel methods

The *Gauss Seidel method* differs from the Jacobi method by the fact that, for the $(k+1)^{th}$ iteration the values of $x_i^{(k+1)}$ already computed are used to update the solution for the remaining values:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right], \quad i = 1, ..., n \tag{279}$$

This method leads to the following decomposition of the matrix $\mathbf{A}$:

$$\mathbf{M} = \mathbf{D} - \mathbf{E}, \quad \mathbf{N} = \mathbf{F} \tag{280}$$

and the associated iteration matrix is:

$$\mathbf{B}_J = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{F} \tag{281}$$

Starting for the Gauss-Seidel method and by analogy with the introduction of a relaxation parameter for the Jacobi method, one can define the *successive over relaxation* (SOR) method

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right] + (1-\omega)x_i^{(k)}, \quad i = 1, ..., n \tag{282}$$

The corresponding iteration matrix is

$$\mathbf{B}_{SOR}(\omega) = (\mathbf{I} - \omega\mathbf{D}^{-1}\mathbf{E})^{-1}[(1-\omega)\mathbf{I} + \omega\mathbf{D}^{-1}\mathbf{F}] \tag{283}$$

For $\omega = 1$, the method is equivalent to the Gauss-Seidel method. The method is named *under-relaxation* for $\omega \in]0, 1[$, and *over-relaxation* for $\omega > 1$

4.7 stationary and unstationary iterative methods

Let's consider the linear system $\mathbf{Ax} = \mathbf{b}$ and

$$\mathbf{B} = \mathbf{I} - \mathbf{P}^{-1}\mathbf{A} \tag{284}$$

the iteration matrix associated to iterative method $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{P}^{-1}\mathbf{r}^{(k)}$.

By acting as for the relaxation methods, the iterative procedure can be generalized by introducing a **relaxation parameter** α. This leads to the **Richardson stationary method**.

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{P}^{-1}\mathbf{r}^{(k)}, \quad k \geq 0. \tag{285}$$

More generally, α may depends on the iteration. This leads to the **unstationary Richardson method** or **semi-iterative** method:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{P}^{-1}\mathbf{r}^{(k)}, \quad k \geq 0. \tag{286}$$

The associated iteration matrix at the k^{th} step is:

$$\mathbf{B}_{\alpha_k} = \mathbf{I} - \alpha_k\mathbf{P}^{-1}\mathbf{A} \tag{287}$$

If $\mathbf{P} = \mathbf{I}$, the method is not preconditioned. The Jacobi or Gauss-Seidel iterations can be seen as stationary Richardson methods with $\alpha = 1$ and $\mathbf{P} = \mathbf{D}$ and $\mathbf{P} = \mathbf{D} - \mathbf{E}$ respectively.

By defining the preconditioned residual $\mathbf{z}^{(k)} = \mathbf{P}^{-1}\mathbf{r}^{(k)}$, we obtain:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{z}^{(k)} \tag{288}$$

and

$$\mathbf{r}^{(k+1)} = \mathbf{b} - \mathbf{Ax}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k\mathbf{Az}^{(k)} \tag{289}$$

To summarize, the unstationary Richardson method at the step $k + 1$ consists in:

- solve the linear system $\mathbf{P}\mathbf{z}^{(k)} = \mathbf{r}^{(k)}$
- compute the acceleration parameter α_k
- update the solution $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)}$
- update the residual $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{A}\mathbf{z}^{(k)}$.

Theorem 4.1. *Let suppose the matrix* $\mathbf{P}$ *invertible and the eigenvalues of* $\mathbf{P}^{-1}\mathbf{A}$ *strictly positive and such that* $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n > 0$. *Thus, the stationary Richardson method is convergent if and only if* $0 < \alpha < 2/\lambda_1$. *Moreover,*

$$\alpha_{opt} = \frac{2}{\lambda_1 + \lambda_n} \tag{290}$$

The spectral radius of the iteration matrix $\mathbf{B}_\alpha$ *is minimal if* $\alpha = \alpha_{opt}$ *with*

$$\rho_{opt} = \min_\alpha [\rho(\mathbf{B}_\alpha)] = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} \tag{291}$$

4.8 Multi-grid methods: basic concept

The basis of the multi-grid method is an observation about iterative methods. Their rate of convergence depends on the eigenvalues of the iterative matrix associated with the method. In particular the eigenvalues with largest magnitude (the spectral radius of the matrix). The eigenvectors associated with the eigenvalues determines the spatial distribution of the convergence errors and varies considerably from method to method. Some of these methods produce errors that are smooth functions of the spatial coordinates. If the error is smooth, the update can be computed on a coarse mesh. On a grid twice as coarse as the original on in three dimensions, the cost is $1/8$ the fine grid cost. Furthermore, iterative methods converge much faster on coarser grids (Gauss Seidel methods may converge four times as fast on a grid twice as coarse).

This suggests that much of the work can be done on a coarser grid. In order to do this, we need to define: the relationship between the two grids, the finite difference operators on the coarse grid, a method of smoothing the residual from the fine grid to the coarse grid and a method of interpolating the update or correction from the coarse grid to the fine one. Several choices are possible for each of them. The simplest interpolation method from the coarse grid to the fine one is a linear interpolation.

We will not discuss the detail of the method, but a two grids iterative method can be summarized as follows:

- One the fine grid, perform iterations with a method that gives smooth errors
- Compute the residual on the fine grid
- Restrict the residual to the coarse grid
- Perform iterations of the correction equations on the coarse grid

- Interpolate the correction to the fine grid
- Update the solution on the fine grid
- Repeat the entire procedure until the residual is reduced to the desired level

Multi-grid is more a strategy than a particular method. Within the framework just described there are many parameters that can be selected more or less arbitrary: the coarse grid structure, the interpolation method, the number of iterations on each grid, the order of which the various grid are visited. The rate of convergence depends on the choices made. In two and three dimensional problems with about 100 nodes in each direction, the multi-grid methods may converge in one-tenth to one-hundredth of the time required by the basic methods.

4.9 Convergence

When using iterative methods, we must estimate the error in order to define a criterion to stop the iterative process. One need to evaluate the number of iteration k_{min} which is necessary for the norm of the error divided by the norm of the initial error to be lower than a given value ϵ. In practice, an a-priori estimation of k_{min} can be obtained from the iteration formula (270) which gives the convergence speed of $\|e^{(k)}\| \to 0$ when $k \to \infty$.

Criterion based on the iterative matrix

From the consistency condition, we have

$$\mathbf{e}^{(k+1)} = \mathbf{B}\mathbf{e}^{(k)} \tag{292}$$

or

$$\mathbf{e}^{(k)} = \mathbf{B}^k\mathbf{e}^{(0)} \tag{293}$$

and therefore, we have

$$\frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{e}^{(0)}\|} \leq \|\mathbf{B}^k\|. \tag{294}$$

So, $\|\mathbf{B}^k\|$ gives an estimation of the reduction factor of the error norm after k iterations. Typically, the iterative process is conducted until

$$\|\mathbf{e}^{(k)}\| \leq \epsilon\|\mathbf{e}^{(0)}\| \quad \text{with} \quad \epsilon < 1 \tag{295}$$

If we suppose $\rho(\mathbf{B}) < 1$, then there is a norm $\|\cdot\|$ such as $\|\mathbf{B}\| < 1$. Therefore, $\|\mathbf{B}^{(k)}\| \to 0$ when $k \to \infty$ and (295) can be satisfied for k large enough such as $\|\mathbf{B}^{(k)}\| < \epsilon$. Nevertheless, as $\|\mathbf{B}^{(k)}\| < \epsilon$, the previous inequality becomes:

$$k \geq \frac{log(\epsilon)}{\left(\frac{1}{k}log\|\mathbf{B}^{(k)}\|\right)} = -\frac{log(\epsilon)}{R_k(\mathbf{B})} \tag{296}$$

where $R_k(\mathbf{B})$ is the **average convergence rate**. However, the above relation is not very useful as it is non-linear in k. In practice, we can use the asymptotic rate of convergence $R(\mathbf{B})$ defined as

$$R(\mathbf{B}) = \lim_{k\to\infty} R_k(\mathbf{B}) = -log(\rho(\mathbf{B})) \tag{297}$$

to obtain the following estimation:

$$k_{min} \simeq -\frac{log(\epsilon)}{R(\mathbf{B})} \tag{298}$$

However, one must have in mind that this estimation is rather optimistic.

Criterion based on increments

From the recurrence formula $\mathbf{e}^{(k+1)} = \mathbf{B}\mathbf{e}^{(k)}$ we have:

$$\|\mathbf{e}^{(k+1)}\| \leq \|\mathbf{B}\|\|\mathbf{e}^{(k)}\| \tag{299}$$

by replacing $\mathbf{e}^{(k)}$ from the previous equation by $\mathbf{e}^{(k+1)} = \mathbf{e}^{(k)} - (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$ and using the triangular inequality, we have:

$$\|\mathbf{e}^{(k+1)}\| \leq \|\mathbf{B}\| \left(\|\mathbf{e}^{(k+1)}\| + \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|\right) \tag{300}$$

and therefore by factorizing by $\|\mathbf{e}^{(k+1)}\|$,

$$\|\mathbf{x} - \mathbf{x}^{(k+1)}\| \leq \frac{\|\mathbf{B}\|}{1 - \|\mathbf{B}\|}\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \tag{301}$$

by applying the recurrence formula (299) we obtain:

$$\begin{aligned}
&\|\mathbf{x} - \mathbf{x}^{(1)}\| \leq \|\mathbf{B}\|^1\|\mathbf{x} - \mathbf{x}^{(0)}\| \\
&\|\mathbf{x} - \mathbf{x}^{(2)}\| \leq \|\mathbf{B}\|^1\|\mathbf{x} - \mathbf{x}^{(1)}\| \leq \|\mathbf{B}\|^2\|\mathbf{x} - \mathbf{x}^{(0)}\| \\
&\|\mathbf{x} - \mathbf{x}^{(3)}\| \leq \|\mathbf{B}\|^1\|\mathbf{x} - \mathbf{x}^{(2)}\| \leq \|\mathbf{B}\|^2\|\mathbf{x} - \mathbf{x}^{(1)}\| \leq \|\mathbf{B}\|^3\|\mathbf{x} - \mathbf{x}^{(0)}\| \\
&\qquad \vdots \\
&\|\mathbf{x} - \mathbf{x}^{(k+1)}\| \leq ... \leq \|\mathbf{B}\|^k\|\mathbf{x} - \mathbf{x}^{(1)}\| \leq \|\mathbf{B}\|^{k+1}\|\mathbf{x} - \mathbf{x}^{(0)}\|
\end{aligned} \tag{302}$$

and then, by using $k = 0$ in (301)

$$\|\mathbf{x} - \mathbf{x}^{(1)}\| \leq \frac{\|\mathbf{B}\|^1}{1 - \|\mathbf{B}\|}\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \tag{303}$$

we obtain the following formula:

$$\|\mathbf{x} - \mathbf{x}^{(k+1)}\| \leq \frac{\|\mathbf{B}\|^{k+1}}{1 - \|\mathbf{B}\|}\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \tag{304}$$

which can be used to estimate the number of iteration necessary to satisfy $\|\mathbf{e}^{(k+1)}\| \leq \epsilon$.

Practically, an estimation of $\|\mathbf{B}\|$ can be formulated as:

$$\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = -(\mathbf{x} - \mathbf{x}^{(k+1)}) + (\mathbf{x} - \mathbf{x}^{(k)}) = \mathbf{B}(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) \tag{305}$$

A maximum of $\|\mathbf{B}\|$ can be estimated by $c = \delta_{k+1}/\delta_k$ where $\delta_{k+1} = \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|$. By replacing $\|\mathbf{B}\|$ by c in (301) we can have an indicator for $\|\mathbf{e}^{(k+1)}\|$

$$\epsilon^{(k+1)} = \frac{\delta_{k+1}^2}{\delta_k - \delta_{k+1}} \tag{306}$$

The approximation used for $\|\mathbf{B}\|$ is such that $\epsilon^{(k+1)}$ can not be used as overestimation for $\|\mathbf{e}^{(k+1)}\|$.

5 Time integration

5.1 Introduction

For unsteady flows, there are 4 dimensions of discretisation: 3 in space and one in time. The difference between the two types of discretisation comes from the direction of influence: a force at a given location will influence the flow in the whole domain (for elliptic problems), however, the forcing at a given time will influence the flow only in the future. Therefore, the unsteady flows are the equivalent of a *parabolic problem in time*. In order to stay compatible with the natural evolution of the flow, the time integration methods are step-by-step methods. These methods are very similar to the methods used for the Ordinary Differential Equations.

5.2 Two steps Methods

Let's consider a first order ordinary differential equation with initial conditions:

$$\frac{d\phi(t)}{dt} = f(t, \phi(t)); \;\; \phi(t_0) = \phi^0 \tag{307}$$

If t^n is the time at the nth time step ($t^n = n\Delta t$), the simplest integration methods can be defined by integrating the equation (307) from the time t^n to the time t^{n+1}:

$$\int_{t^n}^{t^{n+1}} \frac{d\phi(t)}{dt} dt = \phi^{n+1} - \phi^n = \int_{t^n}^{t^{n+1}} f(t, \phi(t)) \tag{308}$$

where $\phi^n = \phi(t^n)$. As f is unknown, an approximated numerical formula can be used to evaluate the integral of the right-hand side. Several methods are summarized in the figure 10.

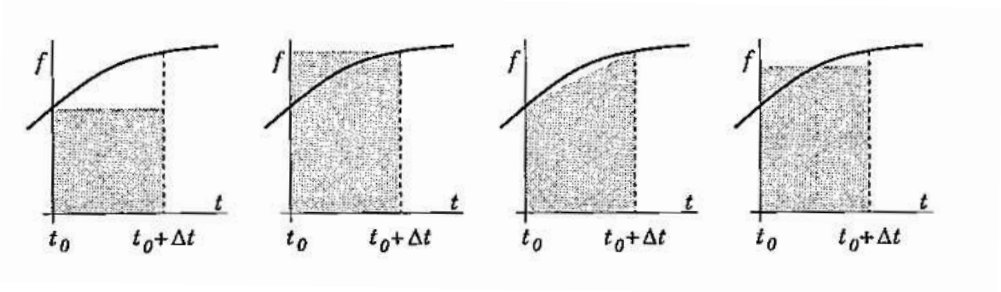

Figure 10: Time integration formulas (Figure from [4])

A method is *explicit* if the value ϕ^{n+1} can be computed directly from the previous values $\phi^k, k < n$ (or only part of them). A method is *implicit* if ϕ^{n+1} is only defined by a implicit relation using the function f.

Explicit (or Forward) Euler Method:

The integral of the function f in (308) is estimated with the value of the function at the initial point t^n. This leads to the following formula:

$$\phi^{n+1} - \phi^n = f(t^n, \phi^n)\Delta t \tag{309}$$

This method is one of the simplest explicit method for time discretisation.

Implicit (or Backward) Euler Method:

The integral of the function f in (308) is estimated with the value of the function at the final point t^{n+1}. This leads to the following formula:

$$\phi^{n+1} - \phi^n = f(t^{n+1}, \phi^{n+1})\Delta t \tag{310}$$

Leap-frog Method:

By considering the midpoint ($t^{n+1/2}$) for the evaluation of the function, we obtain

$$\phi^{n+1} - \phi^n = f(t^{n+1/2}, \phi^{n+1/2})\Delta t \tag{311}$$

By considering a time step which is double, this formulation is equivalent to the Leap-frog formula:

$$\phi^{n+1} - \phi^{n-1} = f(t^n, \phi^n)2\Delta t \tag{312}$$

This method is explicit and second order. However, there are important drawbacks. The first one is that one need to know the solution at two successive time steps (ϕ^0 and ϕ^1) in order to used the formula. The second one is that the solution of the even and odd time step number are independent. This is usually not suitable as the solution of the odd and even time step can diverge one from each others.

Crank-Nicholson Method:

By using the trapezoid rule to evaluate the integral in (308), we obtain the Crank-Nicholson formula:

$$\phi^{n+1} - \phi^n = \frac{1}{2}\left[f(t^n, \phi^n) + f(t^{n+1}, \phi^{n+1})\right]\Delta t \tag{313}$$

These methods are called *two level methods* (except the leapfrog method) because they involved the unknown values at only two points.

5.3 Predictor-Corrector methods

The explicit methods are usually easy to code and use a limited amount of memory and CPU time. However, the explicit method are unstable for too large time steps. On the other hand, in order to get the solution at the new time step, the implicit methods require complex (iterative) methods to solve a system. These scheme are much more difficult to implement and require much more memory and CPU. However, the implicit methods are more stable for large time steps (some of them are unconditionally stable). The aim of the *Predictor-Corrector* methods is to combine the

advantages of these two types of methods. A wide variety of predictor-corrector methods have been developed. We will describe the most well known method of this type (sometimes called the *predictor-corrector* method).

In this method, the solution a the new time step ϕ^{n+1} is predicted by using the Euler formula:

$$\phi^*_{n+1} = \phi^{n+1} + f(t^n, \phi^n)\Delta t \tag{314}$$

where the * symbol indicates that this is not a final solution at time t^{n+1} but only an estimation of the solution at this time. Then, the solution is corrected by applying the trapezoid rule using ϕ^*_{n+1} to evaluate the integral

$$\phi^{n+1} = \phi^n + \frac{1}{2}\left[f(t^n, \phi^n) + f(t^{n+1}, \phi^*_{n+1})\right]\Delta t \tag{315}$$

One can show that this method is second order accurate and have roughly the same stability condition than the explicit Euler scheme. This predictor-corrector method is a two levels method for which the highest possible accuracy is second order. For higher order approximations one must use the information at more points. The additional points can be ones at which the function f has already been computed or points between t^n et t^{n+1} which are used only for the computation of the scheme. The first type of methods is called *multi-points* methods and the latter *Runge-Kutta* methods.

5.3.1 Adams Methods

The best known multi-points method, the *Adams* methods, are derived by fitting a polynomial to the derivatives at a number of points in time. If a *Lagrange* polynomial is used to fit $f(t^{n-m}, \phi^{n-m})$, $f(t^{n-m+1}, \phi^{n-m+1})$, ..., $f(t^n, \phi^n)$ and the result is used to compute the integral (308), we obtain an explicit method of order $m+1$. Such methods are called *Adams Bashforth* methods. The first order method is simply the explicit Euler method and the second and third order methods are

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{2}\left[3f(t^n, \phi^n) - f(t^{n-1}, \phi^{n-1})\right]$$

and

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{12}\left[23f(t^n, \phi^n) - 16f(t^{n-1}, \phi^{n-1}) + 5f(t^{n-2}, \phi^{n-2})\right]$$

If the data at time t^{n+1} are included into the interpolation polynomial, implicit methods know as *Adams Moulton* methods are obtained. The first order corresponds to the implicit Euler method, the second order corresponds to the trapezoid method and the third order is

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{12}\left[5f(t^{n+1}, \phi^{n+1}) + 8f(t^n, \phi^n) - f(t^{n-1}, \phi^{n-1})\right]$$

In practice, a $m-1$ order *Adams-Bashforth* method is usually used as predictor and a m order *Adams-Moulton* method as corrector. Thus predictor-corrector methods of any order can be obtained.

The multi-point methods has the advantage that it is relatively easy to construct to program and to use. Moreover, they require a single evaluation of $f(t, \phi(t))$ by time step which make

them relatively cheap in computational time. One drawback of the method is that, because it requires data from several prior points in time, they can not be started with the data at only the initial time. One has to use an other method to get the calculation started. One solution can be to use smaller time steps using a lower order method (for instance, Euler method followed by a Leap-Frog method).

The *Adams Bashforth* and the *Adams Moulton* formula are given up to the six order in the following tables:

order	n	n-1	n-2	n-3	n-4	n-5
1	1					
2	3/2	-1/2				
3	23/12	-16/12	5/12			
4	55/24	-59/24	37/24	-9/24		
5	1901/720	-2274/720	2616/720	-1274/720	251/720	
6	4277/1440	-7923/1440	9982/1440	-7298/1440	2877/1440	-475/1440

Adams-Bashforth up to the 6^{th} order

order	n+1	n	n-1	n-2	n-3	n-4
1	1					
2	1/2	1/2				
3	5/12	8/12	-1/12			
4	9/24	19/24	-5/24	1/24		
5	251/720	646/720	-264/720	106/720	-19/720	
6	475/1440	1427/1440	-798/1440	482/1440	-173/1440	27/1440

Adams-Moulton up to the 6^{th} order

5.3.2 Runge-Kutta Methods

For the Runge-Kutta method, the time discretisation is done between t^n et t^{n+1} rather than earlier points. The second order Runge-Kutta method consists of two steps. The first one can be seen as a half-step predictor based on the explicit Euler method. It is followed by a midpoint rule corrector which make the complete method second order:

$$\begin{aligned} \phi^*_{n+1/2} &= \phi^n + \frac{\Delta t}{2} f(t^n, \phi^n) \\ \phi^{n+1} &= \phi^n + \Delta t \; f(t^{n+1/2}, \phi^*_{n+1/2}) \end{aligned}$$

The Runge-Kutta methods are easy to use as we can start the integration from the initial value only. The methods are in fact very similar to predictor-corrector methods. The only difference is that the predictor step is done with the midpoint $t^{n+1/2}$ using a half-time step.

Higher order Runge-Kutta methods have been derived. The most popular one is of fourth order (RK4). The first two steps use the explicit Euler methods as predictor steps followed by the implicit Euler method as corrector step. The last two steps corresponds to the midpoint rule as

predictor at time t^{n+1} followed by the Simpson formula for the last corrector step. The complete method is:

$$\begin{aligned}
\phi^*_{n+1/2} &= \phi^n + \frac{\Delta t}{2}\, f(t^n, \phi^n) \\
\phi^{**}_{n+1/2} &= \phi^n + \frac{\Delta t}{2}\, f(t^{n+1/2}, \phi^*_{n+1/2}) \\
\phi^*_{n+1} &= \phi^n + \Delta t\, f(t^{n+1/2}, \phi^{**}_{n+1/2}) \\
\phi^{n+1} &= \phi^n + \frac{\Delta t}{6}\, \big[f(t^n, \phi^n) + 2f(t^{n+1/2}, \phi^*_{n+1/2}) \\
&\quad + 2f(t^{n+1/2}, \phi^{**}_{n+1/2}) + f(t^{n+1}, \phi^*_{n+1})\big]
\end{aligned}$$

One of the main disadvantage of this method is that it requires up to four evaluations of f per time step. Moreover, the intermediate evaluation have to be stored which make this method relatively expensive in memory. It is interesting to know that there is a fourth order method which require less memory (low storage). It is very difficult to develop method of very high order as the n^{th} order Runge-Kutta method require that the derivative be evaluated n times per time step making these methods more expensive than multi-point methods of the same order. However, the Runge-Kutta methods of a given order are more accurate and more stable than the multi-points methods of the same order.

6 Consistency, Stability and Convergence

6.1 Definitions

The resolution of partial derivative equation using discretised equations should have certain properties. the three major properties are: consistency, stability and convergence. These three properties allow to link the exact solution of the continuous equations to the exact solution of the discretised equation and to the numerical solution obtained by the numerical method. The definition of each property and the link between them were already discussed in section1.2.

- **The consistency** is the property which ensure that the exact solution of the discretised equations tend to the exact solution of the continuous equation when the time and space discretisations tend to zero ($\Delta t \to 0, \Delta x \to 0$)

- **The stability** is the property which ensure that the difference between the numerical solution and the exact solution remains bounded.

- **The convergence** is the property which ensure that the numerical solution tends to the (or one) exact solution of the continuous equation when the grid spacing tends to zero.

Theorem 6.1. (Lax Theorem) *Given a properly posed linear initial value problem and a finite difference approximation to it that satisfies the consistency condition, stability is a necessary and sufficient condition for convergence*

6.2 Consistency

In order to check the consistency the term u_i^m which appear in the discretised equations are developed using Taylor expansion in the vicinity of u_i^n . The high order terms are kept and their Taylor development are substituted into the equations. One can take the example of the following conservative equation:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{316}$$

After discretisation and by using the Euler integration formula, the following scheme is obtained

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -\frac{a}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) \tag{317}$$

If the function is sufficiently derivable, one can introduce the following Taylor expansion:

$$u_i^{n+1} = u_i^n + \Delta t \left(\frac{\partial u}{\partial t}\right)_i^n + \frac{\Delta t^2}{2}\left(\frac{\partial^2 u}{\partial^2 t}\right) + ... \tag{318}$$

$$u_{i+1}^n = u_i^n + \Delta x \left(\frac{\partial u}{\partial x}\right)_i^n + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial^2 x}\right) + \frac{\Delta x^3}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right) + ... \tag{319}$$

$$u_{i-1}^n = u_i^n - \Delta x \left(\frac{\partial u}{\partial x}\right)_i^n + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial^2 x}\right) - \frac{\Delta x^3}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right) + ... \tag{320}$$

By introducing this development into the equation (317):

$$\begin{aligned} &\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} - \left(\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x}\right)_i^n \\ &= +\frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial^2 t}\right)_i^n + a\frac{\Delta x^2}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right)_i^n + O(\Delta t^2, \Delta x^4) \end{aligned} \tag{321}$$

Obviously, the right hand side of the equation tends to zero when Δt and Δx tend to zero. Therefore, the numerical scheme is consistent. As expected the scheme is first order in time and second order in space as the rhs of the equation tends to zero as the first power of Δt and the second power of Δx. However, it is important to notice that if the ratio $\Delta t/\Delta x$ is kept constant, the scheme is first order, but it becomes second order if $\Delta t/\Delta x^2$ is kept constant.

The consistency equation can be interpreted as follows. If the values u_i^n are considered as the exact solution of the discretised equation, and if $\tilde{u}_i^n$ is the exact solution of the numerical scheme, the equation (321) becomes :

$$\left(\frac{\partial \tilde{u}}{\partial t} + a\frac{\partial \tilde{u}}{\partial x}\right)_i^n = -\frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial^2 t}\right)_i^n - a\frac{\Delta x^2}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right)_i^n + O(\Delta t^2, \Delta x^4) \tag{322}$$

This shows that, for finite values of Δt et Δx, the exact solution of the finite difference equation does not satisfy exactly the partial differential equation. However, the solution satisfies an equivalent partial differential equation (also named **modified equation**) which differs from the original equation by the **truncation error** ϵ_T which is in this case :

$$\epsilon_T = -\frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial^2 t}\right)_i^n - a\frac{\Delta x^2}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right)_i^n + O(\Delta t^2, \Delta x^4) \tag{323}$$

which can be written differently by applying the differential equation to eliminate the time derivative

$$\left(\frac{\partial u}{\partial t}\right)_i^n = -a\left(\frac{\partial u}{\partial x}\right)_i^n + O(\Delta t, \Delta x^2) \tag{324}$$

and for the second order derivative

$$\begin{aligned}\left(\frac{\partial^2 u}{\partial t^2}\right)_i^n &= -a\left(\frac{\partial^2 u}{\partial x \partial t}\right)_i^n + O(\Delta t, \Delta x^2) \\ &= +a^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + O(\Delta t, \Delta x^2)\end{aligned} \tag{325}$$

Therefore, the truncation error can be written as

$$\epsilon_T = -\frac{\Delta t}{2}a^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n - a\frac{\Delta x^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n + O(\Delta t^2, \Delta x^2) \tag{326}$$

By keeping only the lowest order, the partial differential equation becomes:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = -\frac{\Delta t}{2}a^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + O(\Delta t^2, \Delta x^2) \tag{327}$$

The right hand side can be seen as a viscous term with a negative viscous coefficient ($-(\Delta t/2)a^2$). The effect of a positive viscosity is to reduce the oscillation and the sharp gradient when a negative viscosity will amplify the errors. Therefore, the numerical scheme which satisfies the above equation is unstable. The determination of the equivalent equation and the truncation error may provide important information on the properties of the numerical scheme which usually leads to necessary condition for stability.

6.3 Stability

There are several methods to study the stability of a numerical scheme but almost all of them are restricted to the study of linear problems. However, even for linear problems, the stability analysis may be difficult for the initial value problems and problems with boundary conditions. The stability analysis for linear problem with constant coefficient is relatively well known when the influence of the boundary condition can be removed. The **von Neumann** stability method is based on a development in frequency space. This is the most used method for stability analysis which allows a complete analysis of the behavior of the initial condition errors. The stability analysis can be performed on the equations with constant coefficient and with periodic boundary conditions. When dealing with non-constant coefficients or non-linear terms, the information on stability becomes very limited. In such a case, one must come back to a linear formulation by freezing the non-linear terms and non-constant coefficients. In all cases, the linear stability is a necessary condition, but not a sufficient condition.

6.3.1 Definition

Numerical schemes must not allow any errors to grow infinitely when moving from one time step to the next one. A stability condition can be defined as: Any error ϵ_i^n between the computed

solution u and the exact solution $\tilde{u}$ must stay limited for $n \to \infty$ at fixed Δt. If the error is defined as the difference between the computed solution and the exact solution:

$$\epsilon_i^n = u_i^n - \tilde{u}_i^n \tag{328}$$

The stability condition can be written as

$$\lim_{n\to\infty} |\epsilon_i^n| \leq K \qquad \text{at fixed } \Delta t \tag{329}$$

where K is independent of n. The stability condition must be respected for any types of errors. However, this condition do not insure that the error can not become too large to be acceptable for intermediate times $t^n = n\Delta t$.

6.3.2 Spectral decomposition of the error

If $\tilde{u}_i^n$ is the exact solution of the partial differential equation and u_i^n is the computed solution, the difference between the two quantities can be due to the round-off errors and the error introduced in initial conditions.

$$u_i^n - \tilde{u}_i^n = \epsilon_i^n$$

where ϵ_i^n is the error a time n and at point i. The linear numerical scheme satisfied by u_i^n is also exactly satisfied by $\tilde{u}_i^n$. Therefore, the error ϵ_i^n is also solution of the same discretised equation.

Let's take the previous example (316) using the explicit Euler integration scheme:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -\frac{a}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) \tag{330}$$

we have :

$$\frac{\tilde{u}_i^{n+1} - \tilde{u}_i^n}{\Delta t} + \frac{\epsilon_i^{n+1} - \epsilon_i^n}{\Delta t} = -\frac{a}{2\Delta x}(\tilde{u}_{i+1}^n - \tilde{u}_{i-1}^n) - \frac{a}{2\Delta x}(\epsilon_{i+1}^n - \epsilon_{i-1}^n)$$

As $\tilde{u}_i^n$ satisfies the equation (330), the equation for the error ϵ_i^n is :

$$\frac{\epsilon_i^{n+1} - \epsilon_i^n}{\Delta t} = -\frac{a}{2\Delta x}(\epsilon_{i+1}^n - \epsilon_{i-1}^n)$$

This formulation is identical to the initial scheme. This means that the error evolves identically to the solution u_i^n.

If we consider period boundary conditions, the error ϵ_i^n can be decomposed in Fourier series in space for each time step n. We can use the discrete Fourier representation and sum over a finite number of harmonics. In a 1D domain $[-L, L]$ discretised with 2N points uniformly distributed, the fundamental frequency corresponds to the maximal wavelength $\lambda_{max} = 2L$. The associated wavenumber $k = 2\pi/\lambda$ reaches a minimum value $k_{min} = \pi/L$. At the opposite, the maximum value of the wavelength is associated to the lowest wavenumber on a mesh $\Delta x = L/N$ (see fig. 11). This minimum wavelength is $\lambda_{min} = 2\Delta x$, which corresponds to $k_{max} = \pi/\Delta x$. All the harmonics represented on the finite mesh are given by:

$$k_j = jk_{min} = j\frac{2\pi}{2L} = j\frac{\pi}{N\Delta x} \quad j = 0, 1, 2, ..., N \tag{331}$$

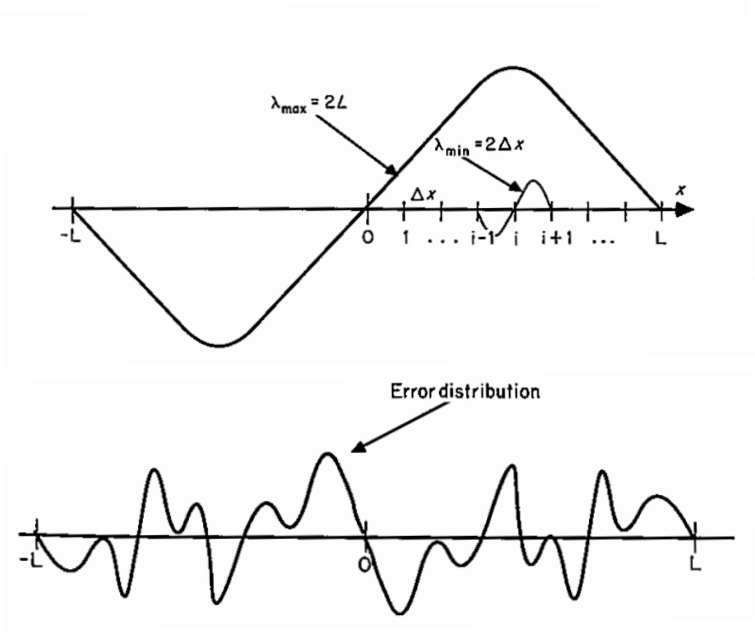

Figure 11: Fourier representation of the error (Figure from [4])

Any function on a the grid defined above as the error ϵ_i^n or the solution u_i^n can be decomposed in Fourier series:

$$\epsilon_i^n = \sum_{j=-N}^{N} E_j^n e^{Ik_j \cdot i\Delta x} = \sum_{j=-N}^{N} E_j^n e^{Ik_j \cdot ij\pi/N}$$

where $I = \sqrt{-1}$ and E_j^n is the amplitude of the j^{th} harmonic. The product $k_j \cdot \Delta x$ can be represented as a phase:

$$\phi \equiv k_j \cdot \Delta x = \frac{j\pi}{N}$$

and varies in the domain $[-\pi, \pi]$ by steps of π/N. The domain near $\phi = 0$ corresponds to the smallest frequencies when $\phi = \pi$ corresponds to the highest frequency of the spectrum. Given the linearity of the scheme (330) which is satisfied for ϵ_i^n, any harmonic $E_j^n e^{Ii\phi}$ of ϵ_i^n satisfies the scheme as well.

6.3.3 Amplification factor

If we consider a single harmonic $E_j^n e^{Ii\phi}$, its temporal evolution is described by the same equation than for u_i^n. We can write (by removing the index j for clarity):

$$\frac{E^{n+1} - E^n}{\Delta t} e^{Ii\phi} + \frac{a}{2\Delta x}(E^n e^{I(i+1)\phi} - E^n e^{I(i-1)\phi}) = 0$$

or, dividing by $e^{Ii\phi}$

$$E^{n+1} - E^n + \frac{\sigma}{2}(E^n e^{I\phi} - E^n e^{-I\phi}) = 0$$

where the following parameter has been introduced

$$\sigma = \frac{a\Delta t}{\Delta x} \tag{332}$$

The stability condition will be satisfied if the amplitude of any error harmonic will not grow in time. This can be expressed by the following relation:

$$|G| = |\frac{E^{n+1}}{E^n}| \leq 1 \quad \forall \phi$$

where G, which is called the **amplification factor**, is function of time and space. In our case, we have

$$G - 1 + \frac{\sigma}{2} \cdot 2Isin(\phi) = 0$$

or

$$G = 1 - I\sigma sin(\phi)$$

Therefore, the stability condition is never verified as

$$|G|^2 = 1 + \sigma^2 (sin(\phi))^2$$

This shows that the central finite difference scheme used for the convection equation associated with an explicit Euler scheme for time integration is unconditionally unstable (always unstable)

6.3.4 Example of scheme conditionally stable

We will now study the stability of the same partial differential equation but with an **upwind** discretisation scheme in space:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -\frac{a}{\Delta x}(u_i^n - u_{i-1}^n) \tag{333}$$

By introducing the harmonic $E^n e^{Ii\phi}$ of the error in the equation, we obtain:

$$(E^{n+1} - E^n)e^{Ii\phi} + \sigma(E^n e^{Ii\phi} - E^n e^{I(i-1)\phi}) = 0$$

If we divide by $e^{Ii\phi}$ we can deduce the amplification factor:

$$\begin{aligned} G &= 1 - \sigma + \sigma e^{-I\phi} \\ &= 1 - 2\sigma sin^2(\phi/2) - I\sigma sin(\phi) \end{aligned}$$

In order to analyze the stability of the scheme, which corresponds to the time-space regions where the amplification factor G is lower than 1, G can be represented in the complex plan (fig. 12). If ξ and η are the real part and the imaginary part of G respectively, we have:

$$\begin{aligned} \xi &= 1 - 2\sigma sin^2(\phi/2) = (1 - \sigma) + \sigma cos(\phi) \\ \eta &= -\sigma sin(\phi) \end{aligned} \tag{334}$$

This can be seen as a parametric equation for G where ϕ is the parameter. On can recognize the parametric equation of a circle centered on the real axis ξ at $1 - \sigma$ and with a radius σ.

The stability condition can be expressed as follow: the parametric curve of G for any values of $\phi = k\Delta x$ must stay included in the unity circle. This leads to following the stability condition:

$$0 < \sigma \leq 1 \tag{335}$$

Therefore, the scheme is conditionally stable and the condition is called the **Courant-Friedrichs-Lewy** or **CFL** condition.

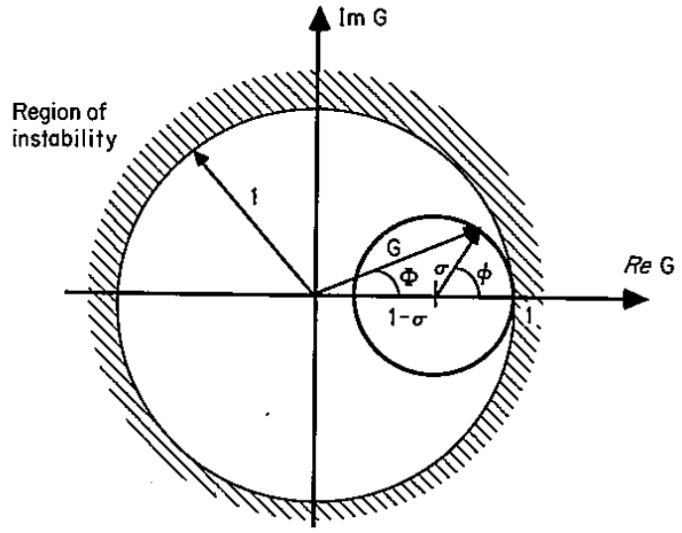

Figure 12: Amplification factor G for the upwind scheme (eq. 333) (Figure from [4])

6.3.5 The CFL condition

The stability condition of most of the explicit schemes for the *"wave type"* equation or convective equations can be formulated this way: *the distance covered during the time Δt by the disturbances propagating with a speed a should be lower than the minimum distance between two mesh points*. In the figure 13 this condition is equivalent to say that the lines PQ, which are the characteristic $dx/dt = \pm a$, must stay in the domain of dependence of the point P which means in the triangle PAC. The stability condition expresses that the mesh ratio $\Delta t/\Delta x$ has to chosen in such way

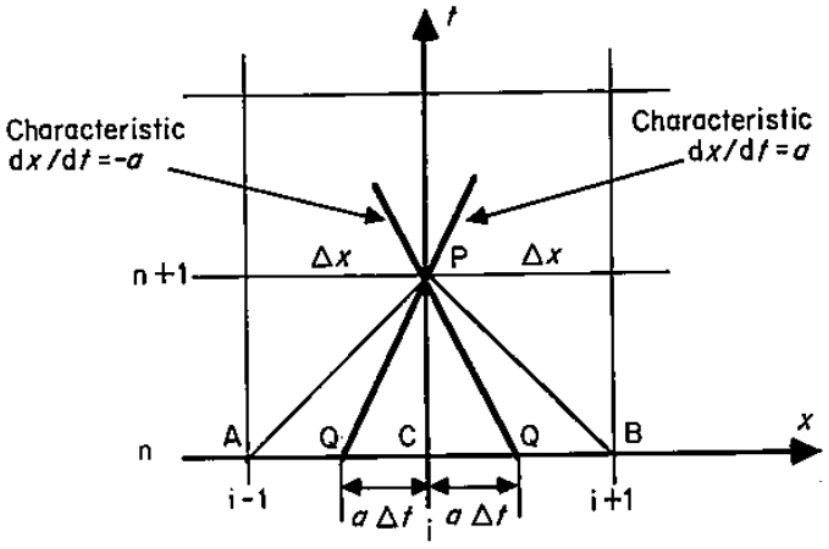

Figure 13: Geometrical interpretation of the CFL condition $\sigma \leq 1$ (Figure from [4])

that the domain of dependence of the differential equation should be contained in the domain of

dependence of the discretised equations.

6.3.6 Exercise

Analyze the stability conditions for the partial differential equation (316) using a central second order finite difference formula and an implicit Euler scheme for time integration ?

7 Solving the Navier-Stokes equations

7.1 Incompressible flows

The numerical simulation of the Navier Stokes equations are difficult because of the equation for the pressure which is not independent as the pressure gradient is involved in the moment equation. Moreover, the continuity equation does not have dominant variables in the equations for incompressible fluid. The conservation of mass can be seen as a constrain on the velocity field instead of a real dynamical equation. The solution to overcome this difficulty is to compute a pressure field to satisfy the continuity equation. It is important to notice that the pressure is involved in the momentum incompressible equations only by its gradient. For compressible equations, the continuity equation can be used define the density and the pressure with a state equation. In the following, we will focus on some methods of pressure-velocity coupling.

7.1.1 Pressure equation

The momentum equations give the velocity components when the continuity equation will be used to compute the pressure. One of the simplest method to do that is to combine the two equations. By taking the divergence of the momentum equation ($\mathbf{b}$ represents the body forces), we obtain the Poisson equation for the pressure:

$$div(grad(p)) = -div\left[div(\rho \mathbf{v}\mathbf{v} - S) - \rho \mathbf{b} + \frac{\partial \rho \mathbf{v}}{\partial t}\right]$$

In a Cartesian system of coordinate, the equation becomes

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial}{\partial x_j}(\rho u_i u_j - \tau_{ij})\right] + \frac{\partial(\rho b_i)}{\partial x_i} - \frac{\partial^2 \rho}{\partial t^2}$$

For constant viscosity and constant density, the equation reduces to

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial(\rho u_i u_j)}{\partial x_j}\right] \tag{336}$$

The pressure equation can be solved by any numerical method for elliptic equations. However, it is important to notice that the right-hand side of the pressure equation is a sum of terms coming from the momentum equation. As a consequence, it is important to use a discretisation which is consistent with the momentum equation.

7.1.2 Implicit Pressure-Correction Methods

Many methods used to solve the steady problems can be seen as methods to solve unsteady problem until a convergence state is reached. The main differences is that, solving unsteady problems requires a time step adapted to the accuracy of the integration methods. However, for steady problems, the time step is chosen in order to reach the convergence as fast as possible. The implicit methods are often used to solve steady flows or slowly evolving flows as these methods required less constrained time steps.

When an implicit method is used to integrate the momentum equation, the discretised equations for the velocity at the new time step are non-linear. When the pressure term is not included into the source term, we can write the equation in a symbolic way as

$$A_P^{u_i} u_{i,P}^{n+1} + \sum_l A_l^{u_i} u_{i,l}^{n+1} = Q_{u_i}^{n+1} - \left(\frac{\delta p^{n+1}}{\delta x_i}\right)_P \tag{337}$$

where P is the arbitrary index of a velocity node, l corresponds to the neighboring points involved in the discretised equations. The source term Q contains all the terms which can be computed explicitly as a function of u_i^n as well as the forcing terms and the linearized terms which may involve u_i^{n+1} or other variable at time $n+1$ like the temperature for instance. The pressure term is written in a symbolic way as the method does not depend on the discretisation which is chosen.

Because of the non-linearity in the coupling, the differential equation (337) can not be solve directly as the coefficient A and the source term Q depend on the solution u_i^{n+1}. The only solution is to use an iterative method. For unsteady problems, solving the coupled system with a good accuracy is important. However, for unsteady problems, the error at each time step may be larger as only the converged state will be of interest. The iteration, within one time step, for which the coefficient of the source matrix will be updated is called an *external iteration* and the iterations which may be used to solve the linear system with fixed coefficients are called *internal iteration*. For each external iteration the equation to be solved are

$$A_P^{u_i} u_{i,P}^{m*} + \sum_l A_l^{u_i} u_{i,l}^{m*} = Q_{u_i}^{m-1} - \left(\frac{\delta p^{m-1}}{\delta x_i}\right)_P \tag{338}$$

where the index $n+1$ has been replaced by a index m which is linked to the identification of the external iteration. u_i^{m*} is the estimation of the solution u_i^{n+1} at the current iteration m.

The velocity at node P, obtained by solving the linearized momentum equation (338) can be formally expressed as:

$$u_{i,P}^{m*} = \frac{Q_{u_i}^{m-1} - \sum_l A_P^{u_i} u_{i,l}^{m*}}{A_P^{u_i}} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i}\right)_P$$

This equation does not satisfy a-priori the continuity equation as the pressure used in this equation is from the previous time step. Therefore, u_i^{m*} is not the final value of the velocity at the iteration m, but only an estimated value (written with a "$*$ symbol"). The final corrected value must satisfy the continuity equation. For simplicity reason, the first term of the right-hand side of the previous equation will be written with $\tilde{u}_{i,P}^{m*}$:

$$u_{i,P}^{m*} = \tilde{u}_{i,P}^{m*} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i}\right)_P \tag{339}$$

In the next step, the velocities must be corrected in order to satisfy the continuity equation:

$$\frac{\delta(\rho u_i^m)}{\delta x_i} = 0$$

This can be achieved by correcting the pressure field. The updated velocities and the pressure are linked by the following equation:

$$u_{i,P}^{m} = \tilde{u}_{i,P}^{m*} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m}}{\delta x_i}\right)_P \tag{340}$$

The continuity is forced by plugging the expression for u_i^m into the continuity equation in order to obtain a discretised equation for the pressure:

$$\frac{\delta}{\delta x_i}\left[\frac{\rho}{A_P^{u_i}}\left(\frac{\delta p^m}{\delta x_i}\right)\right]_P = \left[\frac{\delta \rho \tilde{u}_i^{m*}}{\delta x_i}\right]_P \tag{341}$$

After having solved the pressure equation below, one must compute the updated velocities u_i^m using the equation (340). At this step of the resolution, both the velocity field and the pressure satisfy the continuity equation but not the momentum equation (338). One must repeat the same procedure again and again until both the continuity equation and the momentum equations are satisfied by the velocity field.

SIMPLE algorithm

Such methods based on the construction of a velocity field which does not satisfy the continuity equation but corrected by subtracting the pressure gradient are named the *projection methods*. In most of the cases a pressure correction is used instead of the pressure itself. The velocity computed from the linearized momentum equation and the pressure p^{m-1} are used as predictor values and a correction term must be added to it.

$$u_i^m = u_i^{m*} + u' \quad \text{et} \quad p^m = p^{m-1} + p' \tag{342}$$

Including the decompositions (342) into the momentum equation we obtain a relation between the velocity correction and the pressure correction:

$$u'_{i,P} = \tilde{u}'_{i,P} - \frac{1}{A_P^{u_i}}\left(\frac{\delta p'}{\delta x_i}\right)_P \tag{343}$$

where $\tilde{u}'_{i,P}$ is defined by:

$$\tilde{u}'_{i,P} = -\frac{\sum_l A_l^{u_i} u'_{i,l}}{A_P^{u_i}} \tag{344}$$

When the continuity equation is applied to the corrected velocities, and using the expression (342), the equation for the pressure correction can be written as:

$$\frac{\delta}{\delta x_i}\left[\frac{\rho}{A_P^{u_i}}\left(\frac{\delta p'}{\delta x_i}\right)\right]_P = \left[\frac{\delta \rho u_i^{m*}}{\delta x_i}\right]_P + \left[\frac{\delta \rho \tilde{u}'_i}{\delta x_i}\right]_P \tag{345}$$

At this level, the correction of velocities are unknown and it is usual to neglect them. This explains why this method does not converge very quickly. In this method, once the pressure correction is solved, the velocities are updated using the equations (342) and (343). This method is known as the SIMPLE method *(Semi-Implicit Method for Pressure Linked Equations)*.

SIMPLEC algorithm

A less crude method to treat the last term of the equation of the pressure correction is to make an approximation of it instead of neglecting it. It is possible to approximate the velocity correction u'_i at any node by a weighted value of the neighboring nodes. For instance, one can use:

$$u'_{i,P} \simeq \frac{\sum_l A_l^{u_i} u'_{i,l}}{\sum_l A_l^{u_i}} \tag{346}$$

This allows an approximation of $\tilde{u}'_{i,P}$ from equation (344) by:

$$\tilde{u}'_{i,P} \simeq -u'_{i,P}\frac{\sum_l A_l^{u_i}}{A_P^{u_i}}, \tag{347}$$

Once this approximation is inserted into the equation (343), this leads to the following approximation relation between u'_i et p':

$$u'_{i,P} = -\frac{1}{A_P^{u_i} + \sum_l A_l^{u_i}} \left(\frac{\delta p'}{\delta x_i}\right)_P \tag{348}$$

This methods is known as the **SIMPLEC** method.

PISO algorithm

An other general method can be derived by neglecting $\tilde{u}'_{i,P}$ in a first step of the pressure correction (as for the SIMPLE algorithm), but this first step is followed by an new correction step. The second correction of the velocity u'' is defined by:

$$u''_{i,P} = \tilde{u}'_{i,P} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p''}{\delta x_i}\right)_P \tag{349}$$

where $\tilde{u}'_i$ is computed by the equation (344) where u'_i is computed by the equation (343) with $\tilde{u}'_i$ being neglected. The application of the discretised continuity equation to the corrected velocity leads to a second correction equation for the pressure.

$$\frac{\delta}{\delta x_i}\left[\frac{\rho}{A_P^{u_i}}\left(\frac{\delta p''}{\delta x_i}\right)\right]_P = \left[\frac{\delta(\rho \tilde{u}'_i)}{\delta x_i}\right]_P \tag{350}$$

One can notice that the right coefficient is the same the for the equation (345). This can be used for the computation. At this level additional correction steps can be applied but this is rarely done in practice. This methods is know as the PISO method [6].

SIMPLER algorithm

An other method of the same type was proposed by Patankar [11]. In this new method, the equation for the pressure correction (345) is solved with the last term being neglected as in the SIMPLE algorithm. The pressure correction obtained from this step is only used to correct the velocity field u_i^m in order to satisfy the continuity equation. The new pressure field is computed from the pressure equation (341) by replacing $\tilde{u}_i^{m*}$ par $\tilde{u}_i^m$. This is now possible as $\tilde{u}_i^m$ is known. This methods is known as the **SIMPLER** method.

7.1.3 Fractional Step methods

In the previous method, the pressure is used to enforce the continuity. It is also used in computing the velocity field in the first step of the method; in this step the pressure is treated explicitly. The fractional step method of Kim and Moin (1985) provides an approach that does not use pressure in the predictor step. The fractional step method is more a generic approach than a particular

method.

In the first step, the velocity is advanced using pressure from the previous time step; convective terms, viscous terms and body forces are represented by an equal blend of old and new values (Cranck-Nicolson method in this particular case). The discretised momentum equation for this first step can be written in symbolic form:

$$\frac{(\rho u_i)^* - (\rho u_i)^n}{\Delta t} = \frac{1}{2}\left[H(u_i^n) + H(u_i^*)\right] - \frac{\delta p^n}{\delta x_i} \tag{351}$$

where $H(u_i)$ is an operator which represents the convective terms , the diffusive terms and the discretised source terms. The equation for u_i^* can be solve using any method. For large time step, an iterative method will be needed because of the non-linearity of H.

In a second step, half of the old pressure gradient is removed from u_i^* leading to u_i^**:

$$\frac{(\rho u_i)^{**} - (\rho u_i)^*}{\Delta t} = \frac{1}{2}\frac{\delta p^n}{\delta x_i} \tag{352}$$

and the velocity u_i^{n+1} is estimated by:

$$\frac{(\rho u_i)^{n+1} - (\rho u_i)^{**}}{\Delta t} = -\frac{1}{2}\frac{\delta p^{n+1}}{\delta x_i} \tag{353}$$

with p^{n+1} solution of the Poisson equation:

$$\frac{\delta}{\delta x_i}\left(\frac{\delta p^{n+1}}{\delta x_i}\right) = \frac{2}{\Delta t}\frac{\delta (\rho u_i)^{**}}{\delta x_i} \tag{354}$$

The new velocity is obtained from eq. (353). It satisfies the continuity equation and the momentum equation of the form:

$$\frac{(\rho u_i)^{n+1} - (\rho u_i)^n}{\Delta t} = \frac{1}{2}\left[H(u_i^n) + H(u_i^*)\right] - \frac{1}{2}\left(\frac{\delta p^n}{\delta x_i} + \frac{\delta p^{n+1}}{\delta x_i}\right) \tag{355}$$

For this equation to represent the Cranck-Nicholson method correctly, $H(u_i^*)$ should be replaced by $H(u_i^{n+1})$. However, from equation (352) and (353), one can easily show that the error is of the second order in time and thus consistent with other errors.

$$u_i^{n+1} - u_i^* = -\frac{\Delta t}{2\rho}\frac{\delta (p^{n+1} - p^n)}{\delta x_i} \simeq \frac{(\Delta t)^2}{2\rho}\frac{\delta}{\delta x_i}\left(\frac{\delta p}{\delta t}\right) \tag{356}$$

The equation for the pressure correction can be obtained by subtracting eq. (351) from eq. (355)

$$\frac{(\rho u_i)^{n+1} - (\rho u_i)^*}{\Delta t} = -\frac{1}{2}\frac{\delta p'}{\delta x_i} \tag{357}$$

The major difference between the fractional-step method and the pressure correction methods of the SIMPLE-type is that in the former, the pressure (or pressure-correction) equation is solved once per time step while in the latter, both the momentum equation and the pressure-correction equations are solved several times within each time step (outer iteration). The SIMPLE-type method are more used for steady flows and the fractional-step method for unsteady flows. As the time step used for SIMPLE-type methods are usually much larger, the pressure correction equation must be solve for each internal iteration. For steady flows, the continuity equation must be satisfy accurately only at convergence.

7.1.4 Artificial compressibility method

Simulations of compressible flows are widely used and a deal of attention is focused on development of methods for the numerical simulation of compressible flows. An interesting question is whether these methods can be used for incompressible flows. The major difference between compressible and incompressible flows is their mathematical character. The compressible flows are hyperbolic and the incompressible equations have mixed parabolic hyperbolic character. Therefore, if the methods for compressible flows are to be used for incompressible flows, the character of the equations will need to be modified. The difference between the two equations is that time derivative are present in the continuity equations for compressible flows. So, the most straight-forward means of giving the incompressible equations hyperbolic character is to introduce a time derivative in the continuity equation. As the density is constant, adding $\partial\rho/\partial t$ using the compressible equation is not possible. The best choice is to introduce a time derivative of the pressure.

$$\frac{1}{\beta}\frac{\partial p}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0, \tag{358}$$

where β is the artificial compressibility parameter whose value is a key to the performance of the method. The larger the value of β, the more "incompressible" the equations. $\sqrt{\beta}$ represents the speed of sound in the transformed system.

As the equations are modified, we are no longer solving the "true incompressible" equations. This means that the time history of the simulation generated by this method is not accurate and the method should not be used to simulate unsteady flows. The meaning of the equations is only recovered at steady state, where the divergence free condition is satisfied. For solving the equations many methods are available. However, since the time development of pressure is not important and we are interested in obtaining the steady state solution as quickly as possible, implicit methods should be favored.

Let's consider the implicit Euler scheme. The equations can be written in symbolic form:

$$\frac{p_P^{n+1} - p_P^n}{\beta \Delta t} + \left[\frac{\delta(\rho u_i)^{n+1}}{\delta x_i}\right]_P = 0 \tag{359}$$

The problem is that the velocity field at the new time step $n+1$ is unknown. However, the unknown quantity $(\rho u_i)^{n+1}$ may be approximated as:

$$(\rho u_i)^{n+1} \simeq (\rho u_i^*)^{n+1} + \left[\frac{\partial(\rho u_i^*)}{\partial p}\right]^{n+1} (p^{n+1} - p^n) \tag{360}$$

By inserting this expression into the continuity equation (359) we obtain an equation for the new pressure p^{n+1}.

7.2 Compressible flows

7.2.1 Conservation

Numerical methods need to be adapted to compressibles flows. The main difference between compressible and incompressible flows comes from the nature of the equation. The equations for compressible flows are hyperbolic as sound and waves are able to travel at finite speed as

compared to the incompressible flow where the speed of sound is infinite. An other important difference, is that compressible flows can include discontinuities like chocs. This has a important impact in the properties of the numerical schemes to solve such problems. Chocs are very difficult to simulate accuratly and the scheme must respect special properties like the *conservation* or some other suitable characteristics such as TVD (Total Variation Diminishing) property.

The differential form of the conservation laws inside a domain Ω can be written in a symbolic form:

$$\frac{d}{dt}\int_\Omega \mathscr{A} d\Omega + \int_\Sigma a \cdot n d\Sigma = \int_\Omega A d\Omega \tag{361}$$

Where

- $\mathscr{A}$ is the vector of the considered quantities,
- A is the production of $\mathscr{A}$ in Ω
- a is the echange through the surface Σ

They are defined by:

	$\mathscr{A}$	a	A
masse	ρ	0	0
momentum	ρU	$-\sigma$	ρg
energy	ρE	$q - \sigma U$	$r + \rho(gU)$

where σ is the stress tensor, q is the heat flux by conduction and r is the volume heat echange.

The general conservation law can be applied to a domain including a discontinuity δ. This leads to the following relation for the discontiuity:

$$[\![\mathscr{A}(U - S_\delta)n_\delta + a n_\delta]\!] = 0 \tag{362}$$

where $[\![\Psi]\!]$ is the *jump* normal to the discontinuity n_δ. The eq. (362) are the *jump relations*.

Let's consider the 1D Euler (inviscid) equation

$$\frac{\partial w}{\partial t} + \frac{\partial f(w)}{\partial x} = 0 \tag{363}$$

with $w = (\rho, \rho u, \rho E)$ is the vector of conservative variables and $f(w) = (\rho u, \rho u^2 + p, \rho E u + p u)$ is the flux vector. Initial conditions must be added to the system

$$w(x, 0) = w_0(x) \tag{364}$$

The system above can have solution including some discontinuities (chocs or contact discontinuities). It is necessary to extend the concept of "classic"solutions (i.e. $\mathbb{C}^0$ and $\mathbb{C}^1$ by piece) in order to define a type of equation which require less continuity. A test function $\phi(x, t)$ continuously derivable with a compact support is used. By integrating (363) we obtain:

$$\int_0^\infty \int_{-\infty}^\infty \left[\phi \frac{\partial w}{\partial t} + \phi \frac{\partial f}{\partial x}\right] dx\, dt = 0 \tag{365}$$

or, after integrating by part:

$$\int_0^\infty \int_{-\infty}^\infty \left[\frac{\partial \phi}{\partial t} w + \frac{\partial \phi}{\partial x} f\right] dx\, dt = \int_{-\infty}^\infty \phi(x,0) w(x,0) dx \tag{366}$$

Definition: A function $w(x,t)$ is a weak solution of the conservation law if (366) is satisfied for any function $\phi \in C_0^1$

If $[\![u]\!]$ is the jump of a function u through a discontinuity, the weak solution can be characterized by using the following theorem:

Theorem 7.1. *A function $w(x,t)$, $\mathbb{C}^1$ by piece, and which satisfies the initial condition (364) is a weak solution if and only if:*

- *w is a "classic" solution in the domain where it is $\mathbb{C}^1$*
- *w satisfy the jump relation (***Rankine-Hugoniot** *relations)*

$$[\![f(w)]\!] = s[\![w]\!] \tag{367}$$

through the discontinuity, where s is the speed of the discontinuity.

The Jacobian associated to the system (363) is

$$A(w) = \frac{df}{dw} = \begin{bmatrix} 0 & 1 & 0 \\ -\frac{\gamma-1}{2}u^2 & (3-\gamma)u & \gamma - 1 \\ (\frac{\gamma-1}{2}u^2 - H)u & H - (\gamma-1)u^2 & \gamma u \end{bmatrix} \tag{368}$$

Where $H = E + p/\rho$ is the total enthalpy. The system (363) is completly hyperbolic. This implies that A has three real and distinct eigenvalues $(\lambda^1(w), \lambda^2(w), \lambda^3(w))$ which are

$$\lambda^1 = u - c, \lambda^2 = u, \lambda^3 = u + c \tag{369}$$

where c is the speed of sound defined as $c^2 = \gamma p/\rho$. These eigenvalues, which represents the speed of the information inside the flow, are the speed of the fluid and the speed of sound relative to the speed of the fluid in the two directions $-x$ and $+x$. In an incompressible fluid, these two last velocities have desappeared as the speed of sound is infinite and therefore, the corresponding information propagates instantaneously to the whole domain. The characteristics are the curves defined by $dx/dt = \lambda^k(w), k \in 1, 2, 3$.

In some situations, when the characteristics are convergent in time, a **choc wave** is generated due to the convergence of the characteristics.

For the computation of the weak solutions of hyperbolic systems of equations, the numerical schemes must have the following properties:

- Being in the conservative form
- Being non-oscilatory for the computation of discontinuities
- Must verify an *entropy inequality* in order to select the possible physical solution

In order to be in the conservative form, the scheme must satisfy

$$\frac{w_j^{n+1} - w_j^n}{\delta t} = -\frac{1}{\delta x}\left(F(w_{j-p}^n, ...w_{j+q}^n) - F(w_{j-p-1}^n, ...w_{j+q-1}^n)\right) \tag{370}$$

where $F(w_{j-p}^n, ...w_{j+q}^n)$ is the numerical flux of the scheme. The scheme is consistant if $F(U, .., U) = f(U)$.

The property of conservation of the numerical method is very fondamental for the computation of weak solutions, *i.e.,* solutions that can include discontinuities which must satify the Range-Hugoniot relations. Otherwise, it is possible to compute solutions of flows with chocs propagating with the wrong speed.

Theorem 7.2 (Lax-Wendroff, [10]). *If $w(x,t)$ is the discret solution of a conservative scheme and if $w \to u$ when the space and time grid goes to zero ($\delta x \to 0, \delta t \to 0$), therefore, u is a weak solution (which may include chocs) of the exact problem.*

Let consider the one-dimensional equation (363). One can define a simple one order central scheme to solve this equation

$$\frac{w_j^{n+1} - w_j^n}{\Delta t} = -\frac{f_{j+1}^n - f_{j-1}^n}{2\Delta x} \tag{371}$$

However this scheme is unstable.

The **Lax scheme** is based on the previous scheme, but w_j^n is replaced by $\frac{1}{2}(w_{j-1}^n + w_{j+1}^n)$. This leads to the following scheme:

$$w_j^{n+1} = \frac{1}{2}(w_{j-1}^n + w_{j+1}^n) - \frac{\Delta t}{2\Delta x}(f_{j+1}^n - f_{j-1}^n) \tag{372}$$

The scheme is still explicit and first order but it is conditionaly stable ($CFL \leq 1$) **(check as exercise)**.

7.2.2 Lax-Wendroff scheme

The basic approach of the Lax Wendroff scheme is to use the time serie expansion:

$$w_j^{n+1} = w_j^n + \Delta t\frac{\partial w}{\partial t} + \frac{(\Delta t)^2}{2}\frac{\partial^2 w}{\partial t^2} + \frac{(\Delta t)^3}{6}\frac{\partial^3 w}{\partial t^3} + ... \tag{373}$$

The term in $(\Delta t)^2$ is maintained and replaced by the space derivative term using the original equation:

$$\begin{aligned}\frac{\partial^2 w}{\partial t^2} &= -\frac{\partial}{\partial t}\left(\frac{\partial f}{\partial x}\right) = -\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial t}\right) = -\frac{\partial}{\partial x}\left(\frac{df}{dw}\frac{\partial w}{\partial t}\right) \\ &= -\frac{\partial}{\partial x}\left(A\frac{\partial w}{\partial t}\right) = \frac{\partial}{\partial x}\left(A\frac{\partial f}{\partial x}\right)\end{aligned} \tag{374}$$

where $A = \frac{df}{dw}$ is the Jacobian. Then replacing this term into the Taylor expansion, one can define the one-step no-linear version of the Lax-Wendroff scheme:

$$w_j^{n+1} = w_j^n - \frac{1}{2}\sigma(f_{i+1}^n - f_{i-1}^n) + \frac{1}{2}\sigma^2\left[A_{i+1/2}^n(f_{i+1}^n - f_i^n) - A_{i-1/2}^n(f_i^n - f_{i-1}^n)\right] \tag{375}$$

where $\sigma = \Delta t/\Delta x$ and

$$A_{i+1/2} = A(w_{i+1/2}) \tag{376}$$

or

$$A_{i+1/2} = \frac{1}{2}(A_i + A_{i+1}) \tag{377}$$

For the linear form, one can go one more step in the evaluation of eq. (374)

$$\frac{\partial^2 w}{\partial t^2} = \frac{\partial}{\partial x}\left(A\frac{\partial f}{\partial x}\right) = \frac{\partial}{\partial x}\left(A^2\frac{\partial w}{\partial x}\right) \tag{378}$$

and the Lax-Wendroff scheme is defined by

$$w_j^{n+1} = w_j^n - \frac{1}{2}\sigma(f_{i+1}^n - f_{i-1}^n) + \frac{1}{2}\sigma^2 A_j^2(w_{i+1}^n - 2w_i^n + w_{i-1}^n) \tag{379}$$

7.2.3 Preserving schemes (Total Variation Diminishing)

The concept of bounded total variation finds its origin in an important property of a scalar connservation law

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = 0 \tag{380}$$

The total variation of any physically admissible solution u

$$TV = \int \left\| \frac{\partial u}{\partial x} \right\| \tag{381}$$

does not increase in time.

The total variation in x of a discrete solution of a scalar conservation law is defined by

$$TV(w) = \sum_i |w_{i+1} - w_i| \tag{382}$$

A numerical solution is said to be **bounded total variation** or **total variation stable** is the total variation is uniformaly bounded in t and δx. A numerical scheme is said to be **total variation diminishing (TVD)** if

$$TV(w^{n+1}) \leq TV(w^n) \tag{383}$$

If the scheme is TVD, monotone profiles are preserved during the time evolution of the discrete solutions. This means that spurious oscillations (or overshoot) can not be created.

Definition: A numerical scheme can be put into incremental form if it can be decomposed as follows

$$w_j^{n+1} = w_j^n + C_{j+1/2}(w_{j+1}^n - w_j^n) + D_{j-1/2}(w_j^n - w_{j-1}^n) \tag{384}$$

Theorem 7.3. *A incremental scheme defined as (384) is TVD if*

$$C_{j+1/2} \geq 0, \quad D_{j+1/2} \geq 0, \quad \textit{and} \quad (1 - C_{j+1/2} - D_{j+1/2}) \geq 0, \quad \text{J} = 1, ..., n \tag{385}$$

References

[1] M. Y. Canuto, C. aand Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, 1987.

[2] J. H. Ferziger and M. Perìc. *Computational methods for fluid dynamics*. Springer-Verlag Berlin, 1999.

[3] F. B. Hildebrand. *Introduction to numerical analysis*. McGraw Hill, New York, 2nd ed., 1956.

[4] C. Hirsch. *Numerical computation of internal and external flows. Volume 1: fundamentals of Numerical Discretization*. John Wiley & Sons, 1988.

[5] C. Hirsch. *Numerical computation of internal and external flows. Volume 2: Computational Methods for Inviscid and Viscous Flows*. John Wiley & Sons, 1988.

[6] R. I. Issa. Solution of implicitly discretized fluid flow equations by operator-splitting. *J. Comp. Phys.*, 62:40–65, 1986.

[7] P. D. Lax and B. Wendroff. System of concervation laws. *Comm. Pure and Applied Mathematics*, 13:217–237, 1960.

[8] S. V Patankar. *Numerical heat transfer and fluid flow*. McGraw-Hill, New York, 1980.

[9] A. Quarteroni, R. Sacco, and F. Saleri. *Méthodes Numériques, Algorithmes, analyse et applications*. Springer Verlag, 2007.

[10] R.D. Richtmyer and K.W. Morton. *Difference methods for initial value problems*. Wiley, New York, 1967.