International Master on Turbulence
Numerical Methods

J.-P. Laval, M. Marquillie

Jean-Philippe.Laval@univ-lille1.fr

Laboratoire de Mécanique de Lille (LML), CNRS
Blv Paul Langevin
59655 Vileneuve d'Ascq, FRANCE

(Office #2)

Objectives of the course

The objectives of the course are to give you:

- the basis to understand a code (home-made, Fluent, ...)
  - select the best possible algorithm when choice are required
  - be aware of the limitations
- the basis to create your own specific code (simulation or post-processing)
- **the possibility to talk with experience people developing algorithms**

You will not know all possible methods but only the most used (not necessary the best ones for your problem)

- you will need to **learn by yourself** (we can help you)
- do not (always) trust the standard way of thinking

Please ask questions when needed

Outline (1/3)

- Introduction to numerical methods
  - What is a model and why do we need one ?
  - Concept of Consistency, Stability, Convergence and Accuracy
- Interpolation methods
  - Polynomial interpolation
  - Spline interpolation
- Discretisation of Partial Derivative Equations
  - Spectral methods
  - Finite differences
  - Finite volumes
  - Finite elements (very brief introduction)

Outline (2/3)

- Solving of Linear System
  - Some definitions on matrix
  - Introduction to linear Systems
  - Well posed problems
  - Conditionning
  - Direct methods
  - Iterative methods
  - Multi-grid methods
  - Convergence
- Time integration
  - Two steps Methods
  - Predictor-Corrector Methods
  - Runge-Kutta Methods
- Consistency, stability and convergence
  - Consistency, order of accuracy
  - Stability analysis (with van Neumann methods)

Outline (3/3)

- Solving the Navier-Stokes equations
  - Incompressible flows (SIMPLE, SIMPLER, PISO, Fractional Step)
  - Compressible flows (Preserving schemes, Lax Wendrov)
- Parallelisation and numerical algorithms
  - Why using parallel algorithms ?
  - Parallelisation of the system
  - Parallelisation of the problem (domain decomposition)
- **Open questions ...**

Introduction to numerical methods

What is a model ?

The principle of the modeling is to replace a complex system by an object or an operator which represents the aspects or the behavior of the original system. Several solutions are possible:

- Series of experiments to analyze the parameters of the system and to deduce the characteristic of the model.
  - experiments may be too expensive (flying tests, expensive experimental tools,...)
  - experiments may be too dangerous (nuclear test, spatial context)
  - size of the problem too large or too small (physics of particles, astrophysics, meteorology ...)
- Work out a mathematical model which represents the physics of the original problem (system of Partial Derivative Equations)

From the modeling to the simulation

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

- Look for a mathematical model which represents the physics of the problem
- Define a computational domain and define a mesh for this domain.
- Discretisation of the equations of the physics.
- Solving the discrete equations (usually a system of equations).
- Transcription of the program.
- Numerical simulation.
- Check and validation of the results.

Stability

One can distinguish between three kinds of stability:

- Stability of the physical problem.
- Stability of the mathematical problem.
- Stability of the numerical method to solve a problem.

Stability of physical problem

- A problem is *chaotic* if a small variation of the initial data leads to a large variation of the results
- The stability is linked to the physics of the problem and is independent of the numerical methods used to solve the problem.
- Example: Turbulent flows (sensibility to the initial conditions), Earthquakes

Stability of mathematical problem

- A mathematical problem is *ill-conditioned* if a small variation of the data leads to a large variation of the results
- This notion is linked to the mathematical problem and is independent of the numerical method used to solve the problem.
- One must check that the mathematical problem is as well conditioned as possible before to try to solve it.
- It may be necessary to modify the mathematical model in order to avoid conditioning problems

Stability of numerical methods

- A numerical method is unstable if it can propagate and amplify round-off numerical errors.
- A mathematical problem can be well-conditioned with a numerical method chosen to solve the problem which is unstable.
- If the mathematical problem is ill-conditioned, no numerical method will be able to solve accurately the problem.

Consistency, Stability, Convergence and Accuracy

The resolution of partial derivative equation using discretized 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 discretized equations and to the numerical solution obtained by the numerical method.

Accuracy

One must keep in mind that CFD results are usually contaminated by many different errors:

- *Modeling errors*, which are defined as the difference between the actual flow and the exact solution of the mathematical model
- *Discretization errors*, defined as the difference between the exact solution of the conservative equations and the exact solution of the algebraic system of equation obtained by discretizing these equations
- *Iteration errors*, defined as the difference between the iterative and exact solutions of the algebraic equations systems (for iterative methods only)

Consistency

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

- The difference between the discretized equation and the exact one is called the **truncation error**.
- Even if the approximations are consistent, it does not necessarily mean that the solution of the differential equations will become the exact solution in the limit of small step size (need to be stable).

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.

- 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-linearities are present.
- Stability may require condition on the time step or under-relaxation.

Convergence

The convergence is the property which ensures 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: *"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"* (**Lax equivalence theorem**)
- For non-linear problems which are strongly influence by boundary conditions, the stability and convergence of a method are difficult to demonstrate (checked a-posteriori, repeating the calculation on a series of successively refined grids).

Conservation

Since the equations to be solved are conservation laws, the numerical scheme should also, on both local and global basis, respect the conservation

- 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 NS 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
- Non-conservative schemes can be consistent and stable and therefore lead to correct solution in the limit of very fine grids

Flow classification & Boundary Conditions

Mathematical classification of 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.

- Diffusive fluxes (second order derivative)

  $\Rightarrow$ tendency to smoothout the gradients.
- Convective fluxes (first order derivative)

  $\Rightarrow$ transport properties of the flow

Each of these contribution will influence the mathematical nature of the equations (elliptic, parabolic and hyperbolic)

Mathematical classification of flows

The distinction between hyperbolic, parabolic and elliptic types 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 therefore require two initial conditions (at the initial point of each of them). Example of hyperbolic equations: wave equation $u_{tt} - u_{xx} = 0$
- 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.

Classification : incompressible flows

Incompressible Navier Stokes equations

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

dimensionless by a reference length L, a time scale T, a velocity scale V

$$\frac{VT}{L}\frac{\partial u}{\partial t} + \rho(\vec{v}\cdot\vec{\nabla})u = -\frac{\partial p}{\partial x} + \frac{1}{Re}\Delta u \tag{2}$$

where

$$Re = \frac{\rho VL}{\mu} \tag{3}$$

Classification : incompressible flows

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

$$\frac{-V^2 T}{\nu}\frac{\partial u}{\partial t} + \Delta u = Re\frac{\partial p}{\partial x} \tag{4}$$

This equation is

- purely of an **elliptic** type in the steady state case and for a fixed pressure gradient
- **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.

Classification : incompressible flows

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 equations**

$$\frac{\partial u}{\partial t} + (\vec{v} \cdot \vec{\nabla}) u = -\frac{1}{\rho} \frac{\partial p}{\partial x} \tag{5}$$

which in a one-dimensional space takes the form

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = -\frac{1}{\rho} \frac{\partial p}{\partial x} \tag{6}$$

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

Classification : incompressible flows

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

$$\frac{V^2 T}{\nu}\frac{\partial u}{\partial t} = \Delta u \tag{7}$$

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

Classification : incompressible flows

- time-dependent Navier-Stokes equations is essentially *parabolic in space and time*
- the continuity equation has an hyperbolic structure
- therefore unsteady NS equations are considered as **parabolic-hyperbolic**
- the steady state form of the Navier-Stokes equations leads to **elliptic-hyperbolic** properties

Classification : inviscid compressible flow.

General unsteady case:

- A compressible fluid can support sound and waves
- It is not surprising that these equations have essentially **hyperbolic** character
- Most of the method to solve them are based on the idea that the equations are hyperbolic
- Example of hyperbolic equations: 1D wave equation $u_{tt} - u_{xx} = 0$

Steady cases: the character depends on the speed of the flow.

- **Supersonic flows are hyperbolic**.
- **Subsonic flows are essentially elliptic**

Boundary conditions

- The choice of BC are important in the global behavior of a numerical method
- Several conditions may be possible for a specific problem
- The accuracy and the stability of the numerical method is directly link to the boundary conditions
- The distribution of the boundary conditions must respect the classification of the equations

Boundary conditions

Example of a second order linear PDE : Laplace's equation

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = 0 \tag{8}$$

The corresponding inhomogeneous PDE

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = f(x, y) \tag{9}$$

is called the Poisson equation (ex: pressure for incompressible fluid flow)

Dirichlet Boundary conditions

In a thermal equilibrium problem it seems reasonable to expect the equilibrium temperature distribution of a planar object to be completely determined by the temperature distribution imposed on its boundary.
Ω: closed region of the plane, $\partial\Omega$: boundary of Ω.
Mathematical problem: find a function $\Phi(x, y)$ such that

$$\begin{aligned} \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} &= 0, \quad \forall (x, y) \in \Omega \\ \Phi(x, y) &= \Phi_o(x, y), \quad \forall (x, y) \in \partial\Omega \end{aligned} \tag{10}$$

- Such a PDE 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 Boundary conditions

Ex: A planar object is surrounded by material capable of transferring heat at a prescribed rate $f(x, y)$; problem: find the equilibrium temperature inside the object.
Mathematical problem: find a function $\Phi(x, y)$ such that

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

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

Neumann Boundary conditions

- If the right hand side of the Neumann conditions is zero, the conditions are called **homogeneous** Neumann boundary conditions
- In the more general case with non-zero conditions, they are called **non-homogeneous** Neumann boundary conditions

For incompressible NS equations, when solving the Poisson equation for the pressure, non-homogeneous Neumann boundary conditions can be used.

- $\partial p/\partial n$ can be computed from the pressure gradient of the momentum equation
- It is not possible to use Neumann boundary conditions on the entire boundaries of the simulation domain (can add a constant)

Robin conditions

- In some situation, it may be of interest to define mixed boundary conditions
- The Dirichlet and Neumann conditions can be mixed into a single condition at the same position

This type of conditions are called **Robin boundary conditions**:

$$a\Phi(x,y) + \frac{\partial \Phi}{\partial n}(x,y) = f(x,y) \tag{12}$$

The Robin boundary condition is a general form of the insulating boundary condition for convection-diffusion equations. the convective and diffusive fluxes at the boundary sum to zero:

$$-D\frac{\partial c(0)}{\partial x} + u_x(0)\, c(0) = 0 \tag{13}$$

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

Interpolation

Interpolation methods

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.
There are two options:

- The full data-set is used at the same time to define a single interpolation function valid on the whole domain
- Several interpolation function are defined using only a fraction of the local data-set

One could think that the polynomials are the best candidate to interpolate any function. However, using a polynomial of high order may deteriorate the quality of the approximation.

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:

$$\Pi_m(x_i) = a_m x_i^m + ... + a_1 x_i + a_0 = y_i, \quad i = 0, ..., n \tag{14}$$

where the x_i are called the *interpolation nodes*.

If $n = m$, one can prove that there is a unique polynomial Π_m such that $\Pi_m(x_i) = y_i$, for $i = 0, ..., n$.

Lagrange Polynomial

Considering the n order polynomial function l_i such as:

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}, \quad i = 0, ..., n. \tag{15}$$

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:

$$\Pi_n(x) = \sum_{j=0}^{n} b_j \; l_j(x), \tag{16}$$

and therefore:

$$\Pi_n(x_i) = \sum_{j=0}^{n} b_j \; l_j(x_i), \quad i = 0, ..., n. \tag{17}$$

Lagrange Polynomial

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:

$$\Pi_n(x) = \sum_{i=0}^{n} y_i l_i(x). \tag{18}$$

It is possible to check that

$$\Pi_n(x) = \sum_{i=0}^{n} \frac{\omega_{n+1}(x)}{(x - x_i)\omega'_{n+1}(x)} y_i, \tag{19}$$

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

$$\omega_{n+1} = \prod_{i=0}^{n} (x - x_i). \tag{20}$$

The formula (18) are called **interpolation formula of Lagrange** and the polynomial $l_i(x)$ are the **characteristic polynomial of Lagrange**.

Lagrange Polynomial

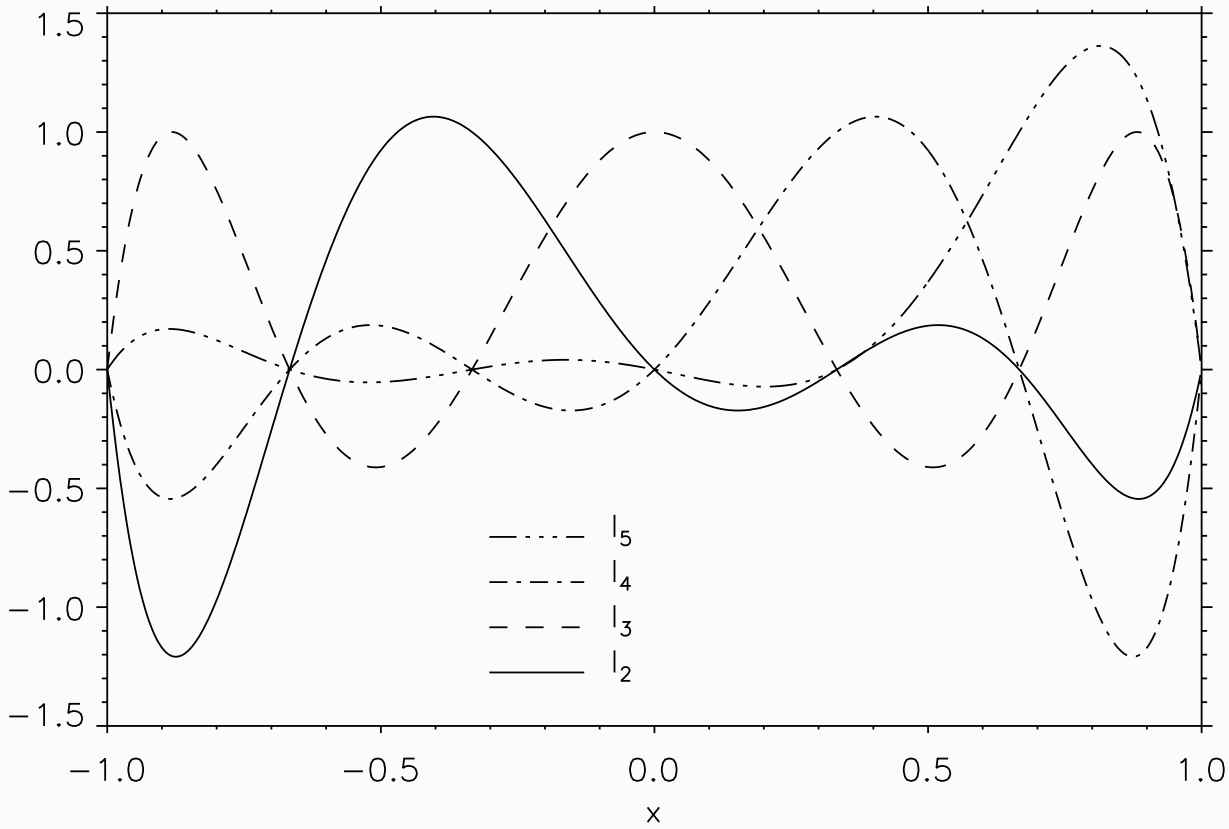


Figure: Characteristic polynomials $l_2(x)$, $l_3(x)$, $l_4(x)$ and $l_5(x)$ for $n = 6$ with homogeneous repartition of the points on the interval $[-1, 1]$

Lagrange Polynomial

Exercise:

Construct the polynomial interpolating the data

x	1	1/2	3
y	3	-10	2

by using Lagrange Polynomials.

Newton's method

There is a more efficient way to define an interpolation function which is based on an iterative method.

Given $n+1$ couples (x_i, y_i), $i = 0, ..., n$, find Π_n such as

$$\Pi_n(x) = \Pi_{n-1}(x) + q_n(x) \tag{21}$$

such as

$$\Pi_n(x_i) = y_i \quad \text{with} \quad i = 0, ..., n \tag{22}$$

$$\Pi_{n-1}(x_i) = y_i \quad \text{with} \quad i = 0, ..., n-1 \tag{23}$$

where $q_n(x)$ is polynomial of order n which depends on nodes x_i and an additional unknown coefficient

Newton's method

As $$q_n(x_i) = \Pi_n(x_i) - \Pi_{n-1}(x_i) = 0 \quad \text{for} \quad i = 0, ..., n-1 \tag{24}$$

we have

$$q_n(x) = a_n(x - x_0)...(x - x_{n-1}) = a_n\, \omega_n(x) \tag{25}$$

let's suppose that

$$y_i = f(x_i), \quad i = 0, ..., n \tag{26}$$

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. (21) that

$$a_n = \frac{f(x_n) - \Pi_{n-1} f(x_n)}{\omega_n(x_n)} \tag{27}$$

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

$$a_n = f[x_0, x_1, ..., x_n] \tag{28}$$

Newton's method

The equation (21) becomes

$$\Pi_n(x) = \Pi_{n-1}(x) + \omega_n(x)\ f[x_0, x_1, ..., x_n]. \tag{29}$$

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

$$\Pi_n(x) = \sum_{k=0}^{n} \omega_n(x)\ f[x_0, x_1, ..., x_k] \tag{30}$$

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**.

Newton's method

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. 19) and by identifying this coefficient with the one of the Newton's formula (eq. 29), we obtain

$$f[x_0, x_1, ..., x_n] = \sum_{i=0}^{n} \frac{f(x_i)}{\omega'_{n+1}(x_i)}. \tag{31}$$

We can obtain a recurrence formula to easily compute the divided differences

$$f[x_k, x_{k+1}, ..., x_n] = \frac{f[x_{k+1}, ..., x_n] - f[x_k, x_{k+1}, ..., x_{n-1}]}{x_n - x_k}, \quad n \geq 1 \tag{32}$$

Newton's method

The Newton's recurrence formula can summarized in a matrix

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

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

Newton's method: 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							

Interpolation Errors

If one wants 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 ? Equally spaced ?

Let's take the *Runge function*

$$f(x) = \frac{1}{1+x^2} \tag{33}$$

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

$$\lim_{n\to\infty} \left(\max_{x\in[-5,5]} |\Pi_n f(x) - f(x)| \right) = \infty \tag{34}$$

Interpolation Errors Runge Function

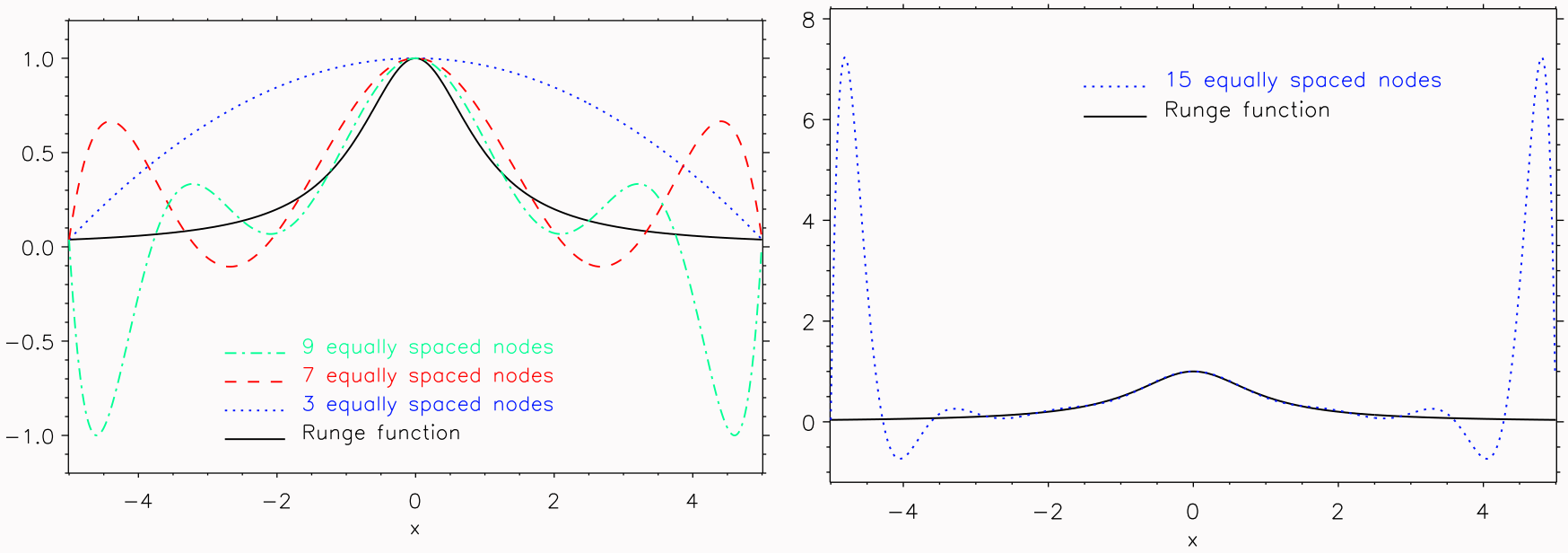


Figure: Interpolation function of the Runge function (Eq. 33) using 3,7,9 and 15 equally space nodes in the domain $[-5, 5]$.

Interpolation Errors Runge Function

A better choice is related to the **Chebyshev Polynomials**
For a closed interval of [-1,1], then interpolation nodes are defined by:

$$x_i = cos\left[\left(\frac{2i+1}{2n+2}\right)\pi\right], \quad 0 \leq i \leq n \tag{35}$$

These nodes can be seen as the projection of the nodes uniformally spaces on a semi-circle on its diameter.

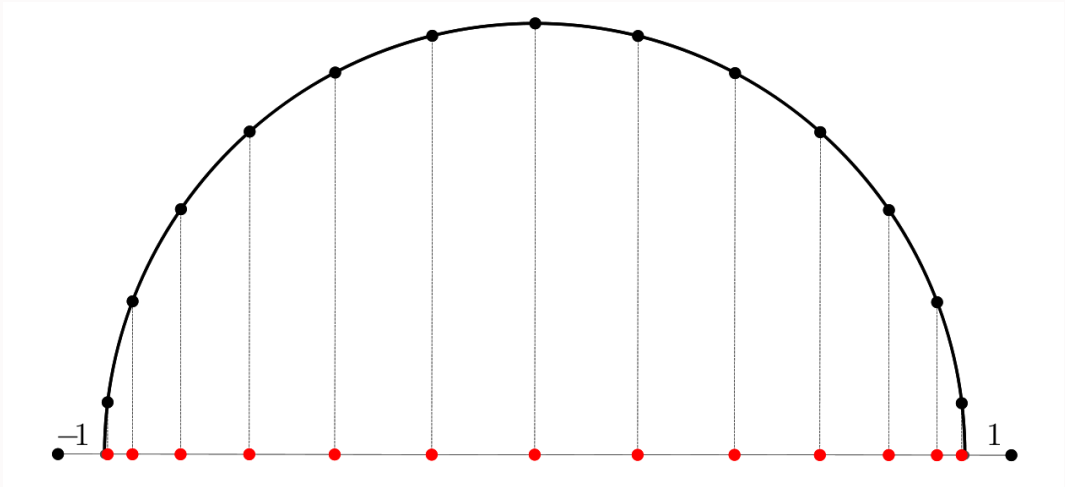


Figure: Distribution of the Chebyshev nodes on a closed interval [-1,1].

Chebyshev Polynomials

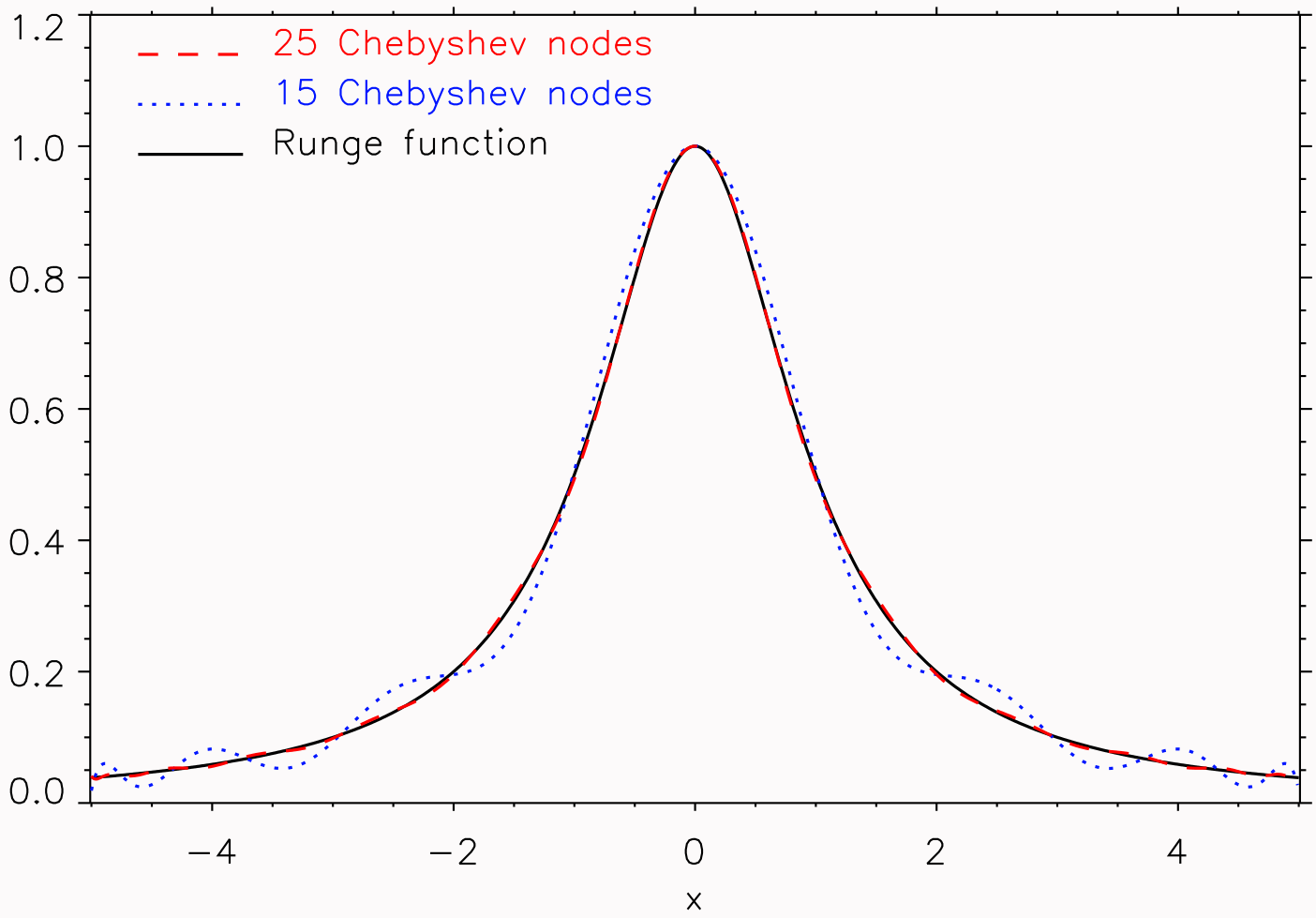


Figure: Interpolation of the Runge function using Lagrange polynomial based on Chebyshev nodes distribution for 15 and 25 nodes.

Interpolation error Theorem

Theorem (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 are some $\xi \in [a, b]$ such as:

$$E_n(x) = f(x) - \Pi_n f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\, \omega_{n+1}(x), \tag{36}$$

where $\omega_{n+1}(x)$ is the nodal polynomial defined by eq. (20).

Interpolation Errors for equally spaced nodes

Interpolation error bounding

$$\max_{x\in[a,b]} \left(\prod_{i=0}^{n} |x - x_i| \right), \tag{37}$$

where

$$x_i = a + h * i = a + \frac{(b-a)}{n} i, \quad i = 0, ..., n \tag{38}$$

and h is the node spacing.

We can assume that x is not one of the nodes and j such as x is between x_j and x_{j+1}.

$$|x - x_j||x - x_{j+1}| \leq \frac{h^2}{4} \tag{39}$$

Interpolation Errors for equally spaced nodes

We can claim

$$|x - x_i| \leq (j - i + 1)h \quad \text{for} \quad i < j \tag{40}$$

$$|x - x_i| \leq (i - j)h \quad \text{for} \quad j + 1 < i \tag{41}$$

then

$$\prod_{i=0}^{n} |x - x_i| \leq \frac{h^2}{4} \left[(j+1)!\, h^j\right] \left[(n-j)!\, h^{n-j-1}\right] \tag{42}$$

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

$$\prod_{i=0}^{n} |x - x_i| \leq \frac{h^{n+1} n!}{4}. \tag{43}$$

The interpolation theorem gives us

$$|E_n(x)| = |f(x) - \Pi_n f(x)| \leq \frac{h^{n+1}}{4(n+1)} \max_{\xi \in [a,b]} \left| f^{(n+1)}(\xi) \right|, \tag{44}$$

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

Interpolation Errors for equally spaced nodes: example

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 by piecewise Lagrange polynomials

- Not possible to guarantee the uniform convergence of the Lagrange polynomial with equally spaced nodes
- 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

$$\left\| E_h^k \right\|_\infty = \max_{[a,b]}(E_h^k) = \max_{[a,b]} \left(f - \Pi_h^k f \right) \leq C h^{k+1} \max_{\xi \in [a,b]} \left(f^{(k+1)}(\xi) \right) \tag{45}$$

where

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

Can obtain a small interpolation error using small values of k but with sufficiently low values of h.

Interpolation by piecewise Lagrange polynomials: 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	$\left\|f - \Pi_h^1\right\|_\infty$	$\left\|f - \Pi_h^2\right\|_\infty$
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	$1.7715E-05$

(46)

Spline interpolation

- Interpolation using high order polynomial on a large interval can lead to bad results at the two bound of the interval
- The approximation with piecewise first order or low order polynomial leads to a global interpolation with discontinuity of the derivative within the interval

The Spline method:

- kind of "minimization of elastic energy"
- impose the continuity of the derivative and the second order derivative in the interval

Spline interpolation

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 $\leq k$.

Spline interpolation

If the partition has $n+1$ knots the spline of degree k is defined by $n(k+1)$ parameters.

We have:

- $n+1$ constrains $s(x(i)) = y(i) \quad \text{for} \quad i = 0, 1, ..., n$
- $k(n-1)$ constrains for the continuity of $S, S', S^{(2)}, ..., S^{(k-1)}$ at the inner knots $i = 1, ..., n-1$.

$\Rightarrow k-1$ more unknown as equations (**degrees of freedom**)

For $k = 3$ (**cubic spline**) we need **2 extra constrains**:

Usual choice: $$S''(t_0) = S''(t_n) = 0 \tag{47}$$

Other choice (*not-a-knot* condition):

$$S'''(t_1^-) = S'''(t_1^+) \quad \text{and} \quad S'''(t_{n-1}^-) = S'''(t_{n-1}^+) \tag{48}$$

Spline interpolation

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 $f_i = f(x_i), i = 0, ..., n$. The objective is to find an efficient method to define the interpolating cubic spline (continuous second derivative) for these values.

Let's define

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

and $s_i(x)$, the spline on $[x_i, x_{i-1}]$ and $h_i = (x_i - x_{i-1})$ for $i = 0, ..., n$.

For cubic spline, s'' is linear

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

Spline interpolation

By integrating twice the formula, we obtain:

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

C_{i-1} and $\tilde{C}_{i-1}$ are determined by:

$$s(x_{i-1}) = f_{i-1} \quad \text{and} \quad s(x_i) = f_i \tag{52}$$

this gives

$$\tilde{C}_{i-1} = f_{i-1} - M_{i-1}\frac{h^2}{6} \tag{53}$$

$$C_{i-1} = \frac{f_i - f_{i-1}}{h_i} - \frac{h^2}{6}(M_i - M_{i-1}) \tag{54}$$

Spline interpolation

Continuity of the first derivative at x_i:

$$\begin{aligned} s'(x_i^-) &= \frac{h_i}{6} M_{i-1} + \frac{h_i}{3} M_i + \frac{f_i - f_{i-1}}{h_i} \\ &= -\frac{h_{i+1}}{3} M_i - \frac{h_{i+1}}{6} M_{i+1} + \frac{f_{i+1} - f_i}{h_{i+1}} = s'(x_i^+) \end{aligned} \quad (55)$$

where

$$s'(x_i^{\pm}) = \lim_{t \to 0} s'(x_i \pm t) \quad (56)$$

This leads to the following linear system:

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i, \quad i = 1, ..., n-1 \quad (57)$$

where we have defined:

$$\begin{aligned} \mu_i &= \frac{h_i}{h_i + h_{i+1}}, \quad \lambda_i = \frac{h_{i+1}}{h_i + h_{i+1}} \\ d_i &= \frac{6}{h_i + h_{i+1}} \left(\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \right), \quad i = 1, ..., n-1 \end{aligned} \quad (58)$$

Spline interpolation

The system (57) has $n+1$ unknowns and $n-1$ equations. 2 additional conditions need to be fixed.

Usually, these two conditions are:

$$2M_0 + \lambda_0 M_1 = d_0, \quad \mu_n M_{n-1} + 2M_n = d_n \tag{59}$$

where $\lambda_0 \geq 0$, $\mu_m \geq 1$ and d_0, d_n are given values.

- For **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 $d_0 = d_n = d_{n-1}$ (considering a and b as internal points)

Spline interpolation

The equation (57) completed by the last two conditions can be expressed in a tri-diagonal matrix form:

$$\begin{bmatrix} 2 & \lambda_0 & 0 & \dots & 0 \\ \mu_1 & 2 & \lambda_1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \mu_{n-1} & 2 & \lambda_{n-1} \\ 0 & \dots & 0 & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{60}$$

Can be solved efficiently using the *Thomas algorithm*

Introduction to Spectral Methods

Spectral methods

- spectral methods are a class of method much less used for general purpose in CFD than Finite differences and Finite Volumes.
- due to high level of accuracy, these methods can be of interest for simulations using simple geometries (especially DNS).
- 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.

Spectral methods: Fourier Transform

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

where $x_i = i\Delta x,\ i = 1, 2, ..., N$ and $k_q = 2\pi q/\Delta_x N$.

The equation (61) can be inverted by

$$\hat{f}(k_q) = \frac{1}{N} \sum_{i=1}^{i=N} f(x_i) e^{-ik_q x_i}, \tag{62}$$

Changing q from q to $q \pm lN$ (where l is integer) produces no change of $e^{\pm ik_q x_i}$ at the grid points. These propriety is know as **aliasing**.

Spectral methods: derivation

The equation (61) can be used for the interpolation of a function $f(x)$.

Differentiation of the formula to obtain the Fourier series for the derivatives:

$$\frac{df}{dx} = \sum_{q=-N/2}^{q=+N/2-1} ik_q \hat{f}(k_q) e^{ik_q x}, \tag{63}$$

Fourier coefficients of df/dx are simply $ik_q \hat{f}(k_q)$

Spectral methods: derivation

Method to derive a function:

- Given $f(x_i)$, use (62) in order to evaluate the Fourier coef. $\hat{f}(k_q)$;
- Compute the Fourier coef. of $g = df/dx$ by $\hat{g}(k_q) = ik_q\hat{f}(k_q)$;
- Evaluate the series (63) to obtain g at the grid points.

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

$$\frac{d^p f}{dx^p} = \sum_{q=-N/2}^{q=+N/2-1} (i\ k_q)^p\ \hat{f}(k_q) e^{ik_q x}, \qquad (64)$$

Spectral methods

- The cost of a derivative is in N^2 (same cost for any derivative order)
- Fast Fourier Transform reduces the cost down to $N\,log_2\,N$
- The function must be periodic and the grid points equally spaced $\Rightarrow$ main limitation of Fourier spectral method for general problems
- Other functions than complex exponential can be used for complex geometries and different boundary conditions (but important modification of the method)

Pseudospectral methods

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

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0 \tag{65}$$

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

$$\frac{\partial \hat{u}_k}{\partial t} + \widehat{\left(u\frac{\partial u}{\partial x}\right)}_k + \nu k^2 \hat{u}_k = 0 \tag{66}$$

where

$$\widehat{\left(u\frac{\partial u}{\partial x}\right)}_k \tag{67}$$

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

$$\widehat{(uv)}_k = \sum_{p+q=k} \hat{u}_p \hat{v}_q \tag{68}$$

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

Pseudospectral methods

Pseudospectral methods evaluate the nonlinear terms in physical space instead of by convolution products.

$$U_j = \sum_{k=-N/2}^{N/2+1} \hat{u}_k e^{ikx_j}$$

$$j = 0, 1, ..., N-1 \tag{69}$$

$$V_j = \sum_{k=-N/2}^{N/2+1} \hat{v}_k e^{ikx_j}$$

$$W_j = U_j V_j \quad j = 0, 1, ..., N-1 \tag{70}$$

$$\hat{W}_k = \frac{1}{N} \sum_{j=0}^{N-1} W_j e^{-ikx_j} \quad k = -\frac{N}{2}, ..., \frac{N}{2} - 1 \tag{71}$$

where

$$x_j = 2\pi j / N \tag{72}$$

Pseudospectral methods

Due to the discrete transform orthogonality relation

$$\frac{1}{N}\sum_{j=0}^{N-1} e^{-ipx_j} = \begin{cases} 1 & \text{if} \quad p = Nm, \quad m = \pm 1, \pm 2, ... \\ 0 & \text{otherwise} \end{cases} \tag{73}$$

we obtain

$$\hat{W}_k = \sum_{m+n=k} \hat{u}_m \hat{v}_n + \sum_{m+n=k\pm N} \hat{u}_m \hat{v}_n \tag{74}$$

$$= \hat{u}_k + \sum_{m+n=k\pm N} \hat{u}_m \hat{v}_n \tag{75}$$

The second term on the right hand side is the **aliasing error**.

Spectral methods: aliasing

- main drawback of the spectral method
- can be an important source of errors in the non-linear equations (such as the NS equations).
- the aliasing is not restricted to the spectral method (high order finite difference methods)
- several possibilities to suppress aliasing in spectral methods
  - Truncation (3/2 rule)
  - Phase shifting
  - ...

Spectral methods: Aliasing removal by Truncation

The simplest method (and the more used) to remove aliasing is the truncation. The key is to use discrete Fourier transform with M rather than N points with $M \geq 3N/2$

$$y_j = 2\pi j/M \tag{76}$$

$$U_j = \sum_{k=-N/2}^{N/2+1} \tilde{u}_k e^{iky_j} \quad j = 0, 1, ..., M-1 \tag{77}$$

$$V_j = \sum_{k=-N/2}^{N/2+1} \tilde{v}_k e^{iky_j} \quad j = 0, 1, ..., M-1 \tag{78}$$

$$W_j = U_j V_j \tag{79}$$

where

$$\tilde{u}_k = \begin{cases} \hat{u}_k & |k| \leq N/2 \\ 0 & \text{otherwise} \end{cases} \tag{80}$$

Spectral methods: Aliasing removal by Truncation

The $\tilde{u}_k$ coefficient are the $\hat{u}_k$ coefficients padded with zeros for the additional wavenumbers.

$$\tilde{W}_k = \frac{1}{M} \sum_{j=0}^{M-1} W_j e^{-iky_j} \quad j = -\frac{M}{2}, ..., \frac{M}{2} - 1 \tag{81}$$

then

$$\tilde{W}_k = \sum_{m+n=k} \tilde{u}_m \tilde{v}_n + \sum_{m+n=k\pm M} \tilde{u}_m \tilde{v}_n \tag{82}$$

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

$$-\frac{N}{2} - \frac{N}{2} \leq \frac{N}{2} - 1 - M \quad \text{or} \quad M \geq \frac{3N}{2} \tag{83}$$

The operation count for this transform method is $(45/4)Nlog_2(3N/2)$ which is roughly 50% larger than the aliased method.

Chebyshev Discretization

The Chebyshev polynomial are defined on $[-1, 1]$ by:

$$T_k(x) = cos(k\theta), \quad \theta = \arccos(x), k = 0, 1, 2, ... \tag{84}$$

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

$$\begin{cases} T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) \\ T_o(x) = 1, \quad T_1(x) = x \end{cases} \tag{85}$$

The Chebyshev expansion of a function f is:

$$f(x) = \sum_{k=0}^{\infty} \hat{f}_k T_k(x) \quad \hat{f}_k = \frac{1}{c_k} \int_{-1}^{1} f(x) T_k(x)(1 - x^2)^{-1/2} dx \tag{86}$$

Chebyshev Discretization

The derivative of a function f expanded in Chebyshev polynomials according to (86) can be represented formally as

$$f' = \sum_{m=0}^{\infty} \hat{f}_m^{(1)} T_m, \quad \hat{f}_m^{(1)} = \frac{2}{c_m} \sum_{\substack{p=m+1 \\ p+m \ odd}}^{\infty} p \, \hat{f}_p^{(1)} \tag{87}$$

this expression is a consequence of the relation

$$2T_k(x) = \frac{1}{k+1} T'_{k+1}(x) - \frac{1}{k-1} T'_{k-1}(x), \quad k \geq 1 \tag{88}$$

which is a consequence of a trigonometric relation. From (87) one has

$$2k\hat{f}_k = c_{k-1}\hat{f}_{k-1}^{(1)} - \hat{f}_{k+1}^{(1)}, \quad k \geq 1 \tag{89}$$

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

$$c_k \hat{f}_k^{(1)} = \hat{f}_{k+2}^{(1)} + 2(k+1)\hat{f}_{k+1} \quad 0 \geq k \geq N-1 \tag{90}$$

total number of operation to differentiate in physical space is $(5log_2 N + 10)N$ (when using FFT)

Finite Differences

Finite differences

- Finite difference method: the derivatives of the equations are approximated using Taylor expansion
- Replace the partial derivative by a combination of punctual values at a finite given number of discrete points or mesh points.
- Very simple and effective on structured grid
- Easy to obtain higher-order schemes
- Requires limited computing resources
- but ... difficult to apply to unstructured grid (complex geometries)

Finite differences

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 :

$$\begin{aligned} u(x + \Delta x, y, z, t) = \quad & u(x, y, z, t) + \Delta x \frac{\partial u}{\partial x}(x, y, z, t) \\ & + \frac{\Delta x^2}{2} \frac{\partial u^2}{\partial x^2}(x, y, z, t) + \frac{\Delta x^3}{6} \frac{\partial u^3}{\partial x^3}(x, y, z, t) + \ldots \end{aligned} \tag{91}$$

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 power of Δx by which the truncation error $O(\Delta x)$ tends to zero is called **order of the method**.

Finite differences

Let consider $u(x)$ in the interval $[0, 1]$ and a mesh composed with N+1 points x_i for i=0,...,N equally spaced with a spacing Δx ($x_i = i\Delta x$ are the mesh points).

$u_i = u(x_i)$: discrete value of $u(x)$ at point x_i

$$\left(\frac{\partial u}{\partial x}\right)_{x=x_i} = \left(\frac{\partial u}{\partial x}\right)_i$$: derivative of u at the point x_i

Finite differences

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

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

This scheme is said **upwind** as it uses the value of the function at the point $i-1$.

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

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

This last scheme is said "**forward**".

Finite differences: higher order

Higher order finite differences can be obtained by combination of Taylor expansion in the vicinity of x_i:

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

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

Subtracting the two expansions above leads to:

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

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

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x^2) \tag{95}$$

Finite differences: higher order

To obtain a higher order scheme, additional point in the neighborhood of x_i are required.

The number of points required to write a finite difference scheme are called the **stencil**

A third order scheme for the first derivative reads:

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

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

Finite differences: higher order derivative

The principle of the method is identical for higher order schemes (based on the Taylor expansion)

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

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

Must find the right combination which suppress the first derivatives

Finite differences: higher order derivative

Taking the sum of the two equations leads to:

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

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

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

Other formulations such as **upwind** of **forward** scheme can be derived for the second order derivative

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

Finite differences: remarks

- Using combination of Taylor expansion, one can derive higher order finite difference schemes for any order of the derivative
- The stencil (number of Taylor expansions) must be adapted to the order of the scheme
- Drawback of the method: can be very long (higher order schemes and schemes for higher order derivatives)

$\Rightarrow$ Can use a systematic framework using **difference operators**

Finite differences: general method

General procedures to developed in order to generate finite difference schemes to any order of accuracy (Hildebrand (1956))

Displacement operator	E :	$E\,u_i = u_{i+1}$
Forward difference operator	δ^+ :	$\delta^+\,u_i = u_{i+1} - u_i$
Backward difference operator	δ^- :	$\delta^-\,u_i = u_i - u_{i-1}$
Central difference operator	δ :	$\delta\,u_i = u_{i+1/2} - u_{i-1/2}$
Averaging operator	μ :	$\mu\,u_i = 1/2\left(u_{i+1/2} + u_{i-1/2}\right)$
Differential operator	D :	$D\,u = \frac{\partial u}{\partial x}$

Finite differences: general method

Obvious relations can be defined between these operators:

$$\delta^+ = E - 1 \quad (99)$$

$$\delta^- = 1 - E^{-1} \quad (100)$$

where

$$E^{-1} u_i = u_{i-1} \quad (101)$$

This leads to

$$\delta^- = E^{-1} \delta^+ \quad (102)$$

and

$$\delta^+ \delta^- = \delta^- \delta^+ = \delta^+ - \delta^- = \delta^2 \quad (103)$$

but also

$$\delta\mu = (E^{+1/2} - E^{-1/2}) \frac{1}{2} (E^{1/2} + E^{-1/2}) = \frac{1}{2} (E^1 - E^{-1}) \quad (104)$$

Finite differences: general method

Using the general definition, n being positive or negative:

$$E^n u_i = u_{i+n} \tag{105}$$

We also have

$$\delta = E^{1/2} - E^{-1/2} \tag{106}$$

$$\mu = \frac{1}{2}(E^{1/2} + E^{-1/2}) \tag{107}$$

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

$$\delta^{+2} = \delta^+\delta^+ = (E-1)^2 = E^2 - 2E + 1 \tag{108}$$

$$\delta^{+3} = \delta^+\delta^+\delta^+ = (E-1)^3 = E^3 - 3E^2 + 3E - 1 \tag{109}$$

$$\tag{110}$$

Finite differences: general method

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

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

which can be written in operators form:

$$Eu(x) = \left(1 + \Delta x D + \frac{(\Delta x D)^2}{2!} + \frac{(\Delta x D)^3}{3!} + ...\right) u(x) \qquad (111)$$

Finite differences: general method

Taking into account the Taylor expansion of the exponential function

$$Eu(x) = e^{\Delta x D} u(x) \tag{112}$$

or symbolically

$$E = e^{\Delta x D} \tag{113}$$

This relation can be inverted as:

$$\Delta x D = ln(E) \tag{114}$$

but also

$$(\Delta x)^n D^n = (ln(E))^n \tag{115}$$

Finite differences: forward formula

Formulas for forward differences are obtained by introducing the relation between E and the forward operator δ^+

$$\Delta x D = ln(E) = ln(1+\delta^+) \quad (116)$$

$$= \delta^+ - \frac{\delta^{+2}}{2} + \frac{\delta^{+3}}{3} - \frac{\delta^{+4}}{4} + ... \quad (117)$$

The order of the accuracy of the approximation increase with the number of terms kept in the right-and-side.

1^{st} order : truncation error equal to $\frac{\Delta x}{2}\frac{\partial^2 u}{\partial^2 x}$

2^{nd} order : truncation error equal to $\frac{\Delta x^2}{3}\frac{\partial^3 u}{\partial^3 x}$

$$\left(\frac{\partial u}{\partial x}\right)_i = D\,u_i = \frac{-3u_i + 4u_{i+1} - u_{i+2}}{2\Delta x} + \frac{\Delta x^2}{3}\left(\frac{\partial^3 u}{\partial x^3}\right)_i \quad (118)$$

Finite differences: backward formula

Similarly, backward difference formula at increasing order of accuracy can be obtained by application of the relation (114):

$$\Delta x D = ln(E) = -ln(1-\delta^-) \quad (119)$$

$$= \delta^- + \frac{\delta^{-2}}{2} + \frac{\delta^{-3}}{3} + \frac{\delta^{-4}}{4} + ... \quad (120)$$

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

$$\left(\frac{\partial u}{\partial x}\right)_i = D\, u_i = \frac{3u_i - 4u_{i-1} + u_{i-2}}{2\Delta x} + \frac{\Delta x^2}{3}\left(\frac{\partial^3 u}{\partial x^3}\right)_i \quad (121)$$

Finite differences: central formula

To obtain central finite difference formula: must link the operator E to the central difference operator δ (or to δ and μ).

$$\delta u_i = u_{i+1/2} - u_{i-1/2} = (E^{1/2} - E^{-1/2})\, u_i \tag{122}$$

and therefore

$$\delta = e^{\Delta x D/2} - e^{-\Delta x D/2} = 2\sinh\left(\frac{\Delta x D}{2}\right) \tag{123}$$

which through inversion leads to:

$$\Delta x D = 2\sinh^{-1}\left(\frac{\delta}{2}\right) \tag{124}$$

$$= 2\left[\frac{\delta}{2} - \frac{1}{2\cdot 3}\left(\frac{\delta}{2}\right)^3 + \frac{1\cdot 3}{2\cdot 4\cdot 5}\left(\frac{\delta}{2}\right)^5 - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7}\left(\frac{\delta}{2}\right)^7 + \ldots\right]$$

$$= \delta - \frac{\delta^3}{24} + \frac{3\delta^5}{640} - \frac{5\delta^7}{7168} + \ldots \tag{125}$$

Finite differences: central formula

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

$$\begin{aligned} \sinh^{-1}(x) &= x - \frac{1}{2\cdot 3}x^3 + \frac{1\cdot 3}{2\cdot 4\cdot 5}x^5 - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7}x^7 + \\ &\ldots + (-1)^n \frac{1\cdot 3\cdot 5\ldots(2n-1)}{2\cdot 4\cdot 6\ldots(2n)(2n+1)}x^{2n+1} + O(x^{2n+2}) \end{aligned} \tag{126}$$

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

$$(\Delta x D)^n = \left(\delta - \frac{\delta^3}{24} + \frac{3\delta^5}{640} - \frac{5\delta^7}{7168} + \ldots\right)^n \tag{127}$$

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:

$$\left(\frac{\partial u}{\partial x}\right)_i = Du_i = \frac{1}{\Delta x}\delta u_i + O(\Delta x^2) = \frac{u_{i+1/2} - u_{i-1/2}}{\Delta x} + O(\Delta x^2) \tag{128}$$

Finite differences: central formula

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

$$1 + \frac{\delta^2}{4} = \mu^2 \tag{129}$$

or

$$1 = \mu \left(1 + \frac{\delta^2}{4}\right)^{-1/2} = \mu \left(1 - \frac{\delta^2}{8} + \frac{3\delta^4}{128} - \frac{5\delta^6}{1024} + ...\right) \tag{130}$$

Finite differences: central formula

by using the following relation:

$$\begin{aligned}(1+x)^a &= 1 + a\,x + \frac{a(a-1)}{2!}x^2 + \frac{a(a-1)(a-2)}{3!}x^3 + \ldots \\ &+ \frac{a(a-1)(a-2)\ldots(a-n+1)}{n!}x^n + O(x^{n+1}) \quad (131)\end{aligned}$$

After multiplying the eq. (125) by eq. (130) we obtain:

$$\Delta x D = \mu\left(\delta - \frac{1}{3!}\delta^3 + \frac{1^2\cdot 2^2}{5!}\delta^5 - \ldots\right) \quad (132)$$

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

Finite differences: central formula

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

$$
\begin{aligned}
Du_i &= \frac{1}{\Delta x}\left(\mu\delta - \frac{1}{3!}\mu\delta^3\right) u_i && (133)\\
&= \frac{1}{\Delta x}\left(\frac{1}{2}(E^1 - E^{-1})(1 - \frac{1}{6}\delta^2)\right) u_i && (134)\\
&= \frac{1}{2\Delta x}\left((E^1 - E^{-1})(1 - \frac{1}{6}(E^1 + E^{-1} - 2))\right) u_i && (135)\\
&= \frac{1}{12\Delta x}\left(8(E^1 + E^{-1}) - (E^2 - E^{-2})\right) u_i && (136)\\
&= \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x} && (137)
\end{aligned}
$$

Finite differences: high order derivatives

The same techniques using operator can be applied to higher order derivative. Starting from the formula (124), the central formula of the n order derivative is obtained by:

$$
\begin{aligned}
D^n u_i &= \left(\frac{2}{\Delta x}\sinh^{-1}\left(\frac{\delta}{2}\right)\right)^n u_i && (138)\\
&= \frac{1}{(\Delta x)^n}\left[\delta - \frac{\delta^3}{24} + \frac{3\delta^5}{640} - \frac{5\delta^7}{7168} + \ldots\right]^n u_i && (139)\\
&= \frac{1}{(\Delta x)^n}\delta^n\left[1 - \frac{n}{24}\delta^2 + \frac{n}{64}\left(\frac{22+5n}{90}\right)\delta^4\right.\\
&- \left.\frac{n}{4^5}\left(\frac{5}{7} + \frac{n-1}{5} + \frac{(n-1)(n-2)}{3^5}\right)\delta^6 + \ldots\right]u_i && (140)
\end{aligned}
$$

Finite differences: high order derivatives

- 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. (129)

$$
\begin{aligned}
D^n u_i &= \frac{\mu}{[1+(\delta^2/4)]^{1/2}} \left[\frac{2}{\Delta x} \sinh^{-1}\left(\frac{\delta}{2}\right) \right]^n u_i \qquad (141) \\
&= \mu \frac{\delta^n}{(\Delta x)^n} \left[1 - \frac{n+3}{24}\delta^2 + \frac{5n^2+52n+135}{5760}\delta^4 + \ldots \right] u_i
\end{aligned}
$$

Finite differences

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	
$\Delta x^4\ u_i^{(4)}$	1	-4	6	-4	1

Finite differences

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

Finite differences

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

Finite differences

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

Finite differences

Exercise

Derive the second order backward difference fomula for the second derivative using difference operators.

Answer

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{2u_i - 5u_{i-1} + 4u_{i-2} - u_{i-3}}{\Delta x^2} - \frac{11}{12}\Delta x^2 \left(\frac{\partial^4 u}{\partial x^4}\right) \qquad (142)$$

Finite differences: non-uniform mesh

Two solutions:

1. Transformation from the physical space (x, y, z) to a Cartesian computational space (ξ, η, ζ)
   - coordinate transformation formulas $(\xi = \xi(x, y, z), \eta = \eta(x, y, z), \zeta = \zeta(x, y, z))$
   - the derivative formulas can be used in (ξ, η, ζ)
   - the transformed equations contains some metric terms which have to be discretised
2. The effect of non-uniform mesh on FD formulas can be deduced from the Taylor development.

Finite differences: non-uniform mesh

Forward and Backward FD formulas deduced from the Taylor development:

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_i}{\Delta x_{i+1}} - \frac{\Delta x_{i+1}}{2}\left(\frac{\partial^2 u}{\partial x^2}\right) \tag{143}$$

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_i - u_{i-1}}{\Delta x_i} + \frac{\Delta x_i}{2}\left(\frac{\partial^2 u}{\partial x^2}\right) \tag{144}$$

where

$$\Delta x_i = x_i - x_{i-1} \tag{145}$$

Finite differences: non-uniform mesh

To obtain the second order **central formula**: combination of the forward and backward formulas to eliminate the first order term (second order derivative)

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{1}{\Delta x_i + \Delta x_{i+1}} \left[\frac{\Delta x_i}{\Delta x_{i+1}} \left(u_{i+1} - u_i\right) + \frac{\Delta x_{i+1}}{\Delta x_i} \left(u_i - u_{i-1}\right)\right] - \frac{\Delta x_i \Delta x_{i+1}}{6} \left(\frac{\partial^3 u}{\partial x^3}\right) \tag{146}$$

Finite differences: non-uniform mesh

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

$$\left(\frac{\partial u}{\partial x}\right)_i = \left(\frac{\Delta x_{i+1}+\Delta x_{i+2}}{\Delta x_{i+2}}\cdot\frac{u_{i+1}-u_i}{\Delta x_{i+1}} - \frac{\Delta x_{i+1}}{\Delta x_{i+2}}\cdot\frac{u_{i+2}-u_i}{\Delta x_{i+1}+\Delta x_{i+2}}\right) + \frac{\Delta x_{i+1}(\Delta x_{i+1}+\Delta x_{i+2})}{6}\left(\frac{\partial^3 u}{\partial x^3}\right) \quad (147)$$

Finite differences: non-uniform mesh

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

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \left(\frac{u_{i+1}-u_i}{\Delta x_{i+1}} - \frac{u_i - u_{i-1}}{\Delta x_i}\right)\frac{2}{\Delta x_{i+1}+\Delta x_i} + \frac{1}{3}\left(\Delta x_{i+1} - \Delta x_i\right)\left(\frac{\partial^3 u}{\partial x^3}\right) - \frac{\Delta x_{i+1}^3 + \Delta x_i^3}{12(\Delta x_{i+1}+\Delta x_i)}\left(\frac{\partial^4 u}{\partial x^4}\right) \quad (148)$$

The truncation error is proportional to the difference of the consecutive mesh size $(\Delta x_{i+1} - \Delta x_i)$

$\Rightarrow$ If the size of the mesh varies abruptly, for instance $\Delta x_{i+1} \simeq 2\Delta x_i$, the formula will **only be first-order accurate**.

Finite differences: compact scheme

Implicit formula: formula where the value of the derivative at a given order is function of the function and several order derivative of the function evaluated at the neighboring points

- There 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
- Drawback: the evaluation of the derivative is not straightforward as it requires the resolution of the linear system of equation
- 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 (Thomas Algorithm).

Finite differences: compact scheme

First derivative with a centered scheme on a uniform distribution of grid points:

$$\beta(f'_{i-2} + f'_{i+2}) + \alpha(f'_{i-1} + f'_{i+1}) + f'_i = c\,\frac{f_{i+3} - f_{i-3}}{6h} + b\,\frac{f_{i+2} - f_{i-2}}{4h} + a\,\frac{f_{i+1} - f_{i-1}}{2h} \qquad (149)$$

- The elations between the coefficients a, b, 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 (149).

Finite differences: compact scheme

These constrains are:

$$a + b + c = 1 + 2(\alpha + \beta) \quad \text{(second order)} \quad (150)$$

$$a + 2^2 b + 3^2 c = 2 + \frac{3!}{2!}(\alpha + 2^2 \beta) \quad \text{(fourth order)} \quad (151)$$

$$a + 2^4 b + 3^4 c = 2 + \frac{5!}{4!}(\alpha + 2^4 \beta) \quad \text{(sixth order)} \quad (152)$$

$$a + 2^6 b + 3^6 c = 2 + \frac{7!}{6!}(\alpha + 2^6 \beta) \quad \text{(eighth order)} \quad (153)$$

$$a + 2^8 b + 3^8 c = 2 + \frac{9!}{8!}(\alpha + 2^6 \beta) \quad \text{(tenth order)} \quad (154)$$

Finite differences: compact scheme

- The equation to satisfy for a odd order of accuracy are automatically verified because the formula was already symmetrized (keeping only 5 unknowns)
- 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

Finite differences: compact scheme

- Taking not the full system of 5 equations but only the p first equations ($p < 5$) leads to a family of scheme of order $2p$ with $(5 - p)$ free parameters.
- Taking $\beta = 0$ leads to a tri-diagonal system to solve. Together with $c = 0$, one obtains a one parameter (α) family of **fourth order tri-diagonal schemes**

$$\beta = 0,\ a = \frac{2}{3}(\alpha + 2),\ b = \frac{1}{3}(4\alpha - 1),\ c = 0. \qquad (155)$$

- For $\alpha = 1/4$ the resulting scheme is the **classical Padé scheme**
- For $\alpha = 1/3$, the leading order coefficient vanishes and the scheme is formally **six order accurate.**

Finite differences: compact scheme

The formula (149) with periodic conditions

$$
\begin{bmatrix}
1 & \alpha & \beta & 0 & 0 & \dots & 0 & \beta & \alpha \\
\alpha & 1 & \alpha & \beta & 0 & \dots & \dots & 0 & \beta \\
\beta & \alpha & 1 & \alpha & \beta & 0 & \dots & \dots & 0 \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\
0 & \ddots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\
\vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\
0 & \dots & \dots & 0 & \beta & \alpha & 1 & \alpha & \beta \\
\beta & 0 & \dots & \dots & 0 & \beta & \alpha & 1 & \alpha \\
\alpha & \beta & 0 & \dots & \dots & 0 & \beta & \alpha & 1
\end{bmatrix}
\begin{bmatrix} f_1' \\ f_2' \\ f_3' \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ f_{n-2}' \\ f_{n-1}' \\ f_n' \end{bmatrix} =
$$

$$
\begin{bmatrix}
0 & a & b & c & 0 & \dots & -c & -b & -a \\
-a & 0 & a & b & c & \dots & \dots & -c & -b \\
-b & -a & 0 & a & b & c & \dots & \dots & -c \\
-c & -b & -a & 0 & a & b & c & & 0 \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & & & & & & & & \\
\vdots & & -c & -b & -a & 0 & a & b & c \\
c & \dots & \dots & -c & -b & -a & 0 & a & b \\
b & c & \dots & \dots & -c & -b & -a & 0 & a \\
a & b & c & \dots & \dots & -c & -b & -a & 0
\end{bmatrix}
\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ f_{n-2} \\ f_{n-1} \\ f_n \end{bmatrix}
\qquad (156)
$$

Finite differences: compact scheme

For non-periodic case, one needs to derive **forward** or **backward** formula. Forward formula with a 5-5 stencil:

$$\mathbf{f_i'} + \alpha\, f'_{i+1} + \beta\, f'_{i+2} + \gamma\, f'_{i+3} + \delta\, f'_{i+4} = \frac{1}{h}(a\, f_i + b\, f_{i+1} + c\, f_{i+2} + d\, f_{i+3} + e\, f_{i+5}) \qquad (157)$$

Maximum order: 8 (no free parameters)

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

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

Finite differences: compact scheme

- Compact formula can be derived for any order of the derivative
- For derivative of order n can use formula with all derivatives up to order n
  - Need to solve a larger system
  - Not necessary an improvement

Example:

$$\alpha f_{i-1}^{(2)} + f_i^{(2)} + \alpha f_{i+1}^{(2)}) + \beta f_{i-1}^{(1)} + \gamma f_i^{(1)} + \beta f_{i+1}^{(1)}) + a f_{i-1} + b f_i + a f_{i-1} = 0$$

Finite differences: discretisation error

- The accuracy of the finite difference formula are linked to their order of accuracy,
- For a given stencil, the order of accuracy is different for explicit or implicit schemes
- The order of accuracy is not the only parameter to characterize the quality of a formula
- The spectral methods are useful to analyze differently the truncation error

$$f(x_i) = \sum_{q=-N/2}^{q=+N/2-1} \hat{f}(k_q) e^{ik_q x_i}, \tag{158}$$

- The derivative can be applied term-by-term to the series

$\Rightarrow$ sufficient to consider the differentiation of $\phi(x) = e^{ikx}$ for any k

Finite differences: discretisation error

Exact derivative of $\phi(x) = e^{ikx}$: $\partial\phi/\partial x = ike^{ikx}$

Applying *first order explicit central difference operator* to this function

$$\left(\frac{\partial\phi}{\partial x}\right)_i^{FD} \simeq \frac{\phi_{i+1} - \phi_{i-1}}{x_{i+1} - x_{i-1}}, \tag{159}$$

we obtain:

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

k' is the modified wavenumber (in general complex)

$\Re(k') = k'_r$: associated with the dispersive error

$\Im(k') = k'_i$: associated with the dissipative error

For central finite defference formula, k' is **real**

Finite differences: discretisation error

Example 1: fourth order central **explicit finite difference** formula

$$\left(\frac{\partial \phi}{\partial x}\right)_i \simeq \frac{-\phi_{i+2} + 8\phi_{i+1} - 8\phi_{i-1} + \phi_{i-2}}{12\Delta x} \tag{160}$$

leads to

$$k' = \frac{sin(k\Delta x)}{3\Delta x}[4 - cos(k\Delta x)] \tag{161}$$

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

$$k' = \frac{sin(k\Delta x)}{\Delta x} \simeq k - \frac{k^3(\Delta x)^2}{6}. \tag{162}$$

Example 2: (5,5) stencil central **compact finite difference** formula:

$$k' = \frac{a\ sin(k) + b/2\ sin(2k) + c/3\ sin(3k)}{1 + 2\alpha\ cos(k) + 2\beta\ cos(2k)}. \tag{163}$$

Finite differences: discretisation error

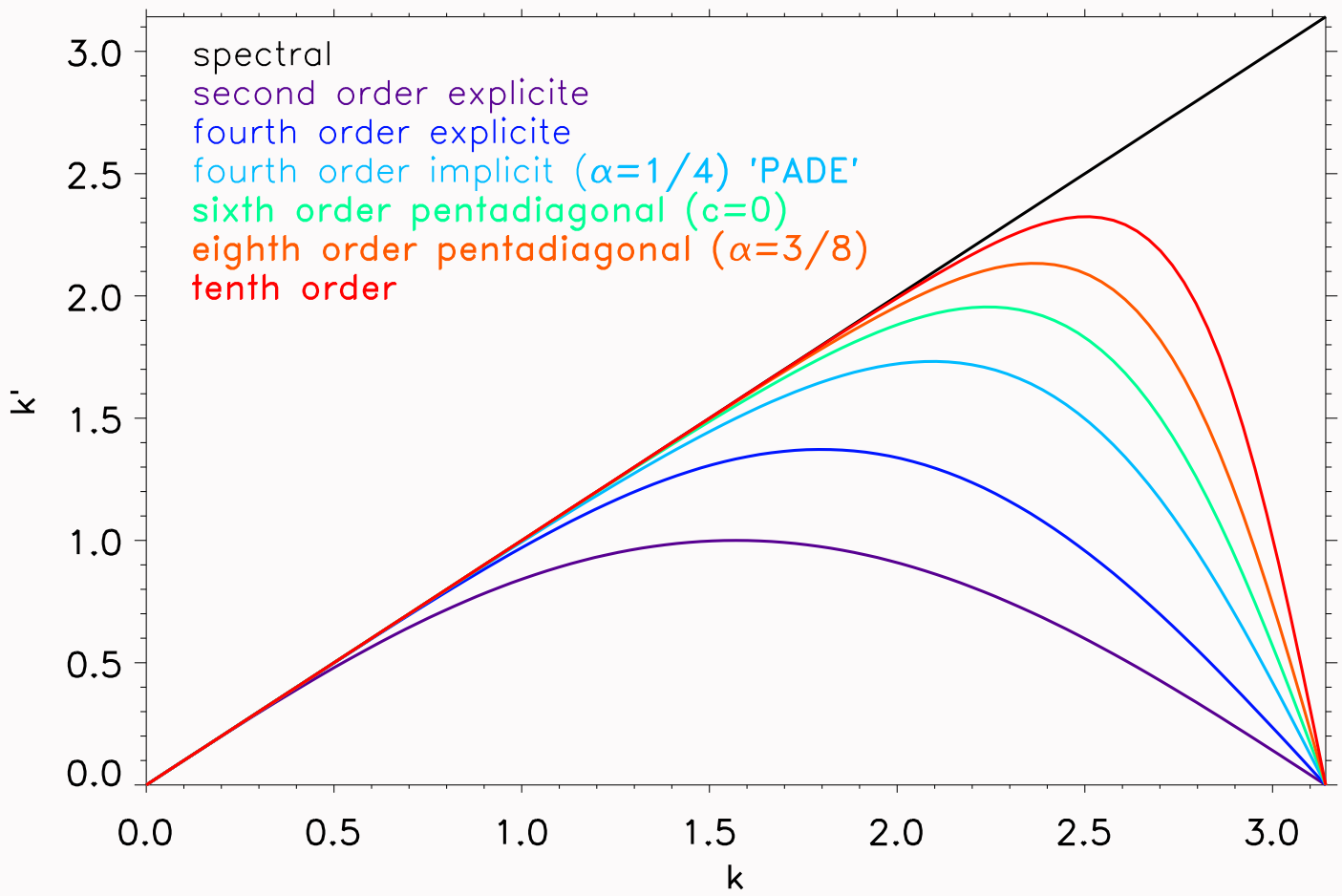


Figure: Modified wavenumbers (real part) for several explicit and implicit central finite difference formula ($k_{max} = \pi/\Delta x$).

Finite differences: discretisation error

Example 3: (5,7) stencil forward compact finite difference formula

$$f'_i + \alpha\, f'_{i+1} + \beta\, f'_{i+2} + \gamma\, f'_{i+3} + \delta\, f'_{i+4} =$$
$$\frac{1}{h}(a\, f_i + b\, f_{i+1} + c\, f_{i+2} + d\, f_{i+3} + e\, f_{i+4} + h\, f_{i+5} + g\, f_{i+6})$$

Finite differences: discretisation error

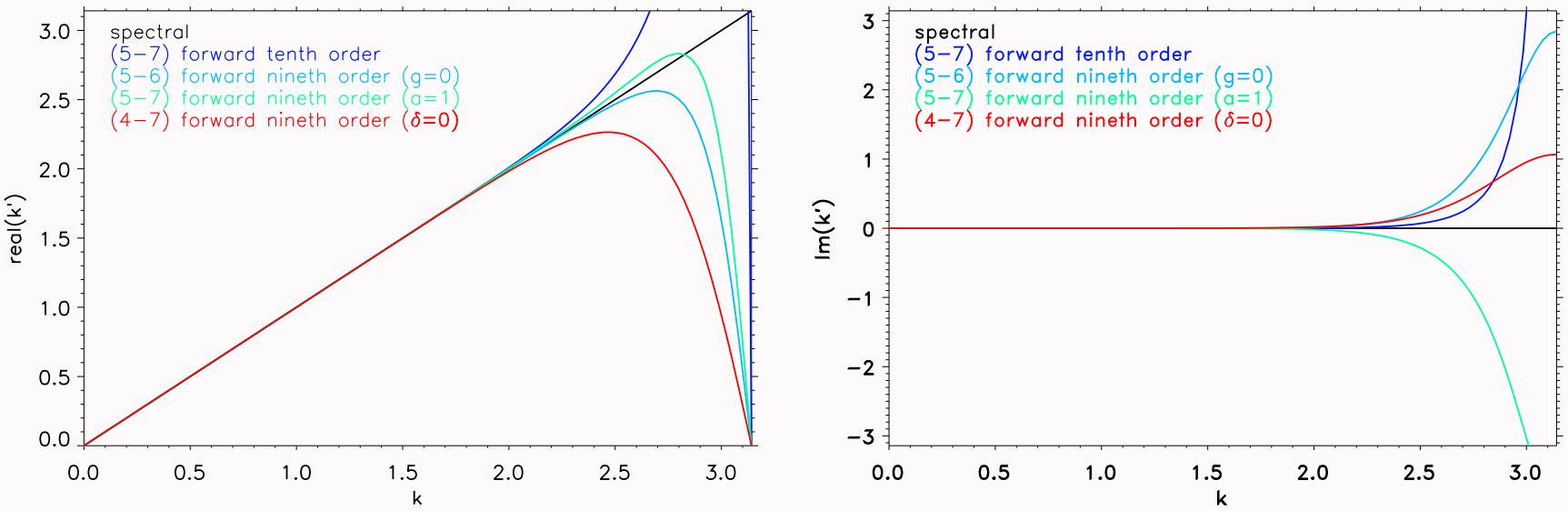


Figure: Modified wavenumbers for several **compact forward** finite difference formula.

Finite differences: discretisation error

- The spectral characteristic of the error is not the only thing to take into account
- Some schemes can have very good spectral behavior but unusable
- System must be well conditioned in order to be inverted
- $\Rightarrow$ reason why some combinations of stencil are not possible

  Ex: (5-5) stencil for forward finite difference is unusable
- Same analysis can be made with finite difference scheme for the **second derivative**
  - $\Re(k')$ associated to the dissipative error
  - $\Im(k')$ associated to the disperssive error
- Compact finite difference scheme can also be derived for irregular grid using the same technique.

  $\Rightarrow$ can lead to VERY complex formula (thousands of lines !)

  $\Rightarrow$ coefficients of the system may have to be computed numerically

Finite differences: discretisation error

- 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
- 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

Finite Volumes

Finite Volumes

The finite volume method is based on the integration of the equation written in integral form in elementary volumes.

- the method is well suited for the spatial discretisation of the conservation laws.
- the method is used intensively in fluid mechanics
- implementation is rather simple when the elementary volumes are rectangles or parallelepiped
- can be used with any shape of elementary volumes (tetrahedral, hexahedral, prismatic, pyramidal, polyhedral,...)
- method well suited for flows in complex geometries
- most of numerical codes in fluid mecanics are based on this method (FLUENT, StarCD, CFX, elSA, Code Saturn, ...)

Finite Volumes

Conservation law of a physical quantity ω inside a mesh of volume Ω and using a flux $F(\omega)$ and a source term $S(w)$.

$$\frac{\partial}{\partial t}\int_\Omega \omega \, d\Omega + \int_\Omega \text{div}[F(\omega)] \, d\Omega = \int_\Omega S(\omega) \, d\Omega \tag{164}$$

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

$$\frac{\partial}{\partial t}\int_\Omega \omega \, d\Omega + \int_\Sigma F \cdot n \, d\Sigma = \int_\Omega S(\omega) \, d\Omega \tag{165}$$

The integral $\int_\Sigma F \cdot n \, d\Sigma$ represents the sum of fluxes through each face of the mesh.

Finite Volumes

The flux is supposed to be constant on each face. As a consequence, the integral becomes a discret sum on each face of the mesh:

$$\int_{\Sigma} F \cdot n \, d\Sigma = \sum_{mesh\ face} F_{face} \cdot n_{face} \Sigma_{face} \tag{166}$$

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.

Finite Volumes

Example with the explicit Euler method for time integration:

Let's define $\Delta\omega$ the increment of the quantity ω between two successive time steps.

$$\frac{\partial}{\partial t}\int_{\Omega}\omega\ d\Omega = \Omega\left(\frac{d\omega}{dt}\right)_{mesh} = \Omega\frac{\Delta\omega}{\Delta t} \tag{167}$$

Eventually, the discretised conservation law using finite volume methods is

$$\Omega\frac{\Delta\omega}{\Delta t} + \sum_{mesh\ face} F_{face}\cdot n_{face}\Sigma_{face} = \Omega S \tag{168}$$

Finite Volumes

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 mesh using a numerical scheme
  - ⋆ computation of the source term
  - ⋆ computation of the temporal increment using a time numerical scheme
  - ⋆ application the boundary conditions

Finite Volumes: 1D case

Let's consider the 1D conservation law:

$$\frac{\partial}{\partial t}\int u\, dx + \int \frac{\partial f(u)}{\partial x} dx = 0 \tag{169}$$

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 neighbooring meshes.

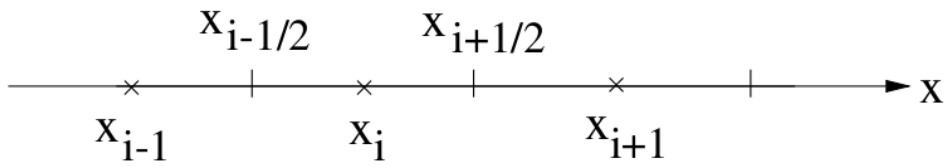


Figure: 1D Grid

Finite Volumes: 1D case

- the time is discretised into constant intervals Δ.
- u is supposed constant within each mesh and equal to an approximated value of the mean.
- u_i^n as the mean value inside the ith 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 (**Cell-Centered Finite Volume**): $u_i^n = u(x_i, t)$.

Finite Volumes: 1D case

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

$$\frac{\partial}{\partial t}\int_{maille} u\ dx + \int_{maille} \frac{\partial f(u)}{\partial x} dx = 0 \tag{170}$$

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

$$\frac{\partial}{\partial t}\int_{x_{i-1/2}}^{x_{i+1/2}} u\ dx + \int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial f(u)}{\partial x} dx = 0 \tag{171}$$

Finite Volumes: 1D case

After integration, this leads to

$$h_i \frac{\partial u_i^n}{\partial t} + \hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n = 0 \tag{172}$$

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 neighbooring mesh.

By using, the explicite Euler method for time integration:

$$h_i \frac{u_i^{n+1} - u_i^n}{\Delta t} + \hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n = 0 \tag{173}$$

Finite Volumes: Example with Dirichlet conditions

Let's consider the following differential equation:

$$\begin{cases} -u'' = f(x); x \in]0,1[\\ u(0) = \alpha; u(1) = \beta \end{cases} \tag{174}$$

- 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 ($x \in [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}], u(x) = u_i$)

Finite Volumes: Example

The spacial discretisation using finite volume consist in a cell-by-cell integration of the differential equations

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u'' = \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} f(x)dx \tag{175}$$

which becomes after integration:

$$u'(x_{i-\frac{1}{2}}) - u'(x_{i+\frac{1}{2}}) = h_i \bar{f}_i \text{ for } i = 1, ..., N \tag{176}$$

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

$$\bar{f}_i = \frac{1}{h_i} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} f(x)dx \tag{177}$$

Finite Volumes: Example

Next step: define $u'(x_{i-\frac{1}{2}})$ as a function of the unknown u_i

$\Rightarrow$ the most natural choice is to take the average of u' in $[x_{i-1}, x_i]$

$$\begin{aligned} u'(x_{i-\frac{1}{2}}) &= \frac{1}{\frac{h_{i-1}+h_i}{2}} \int_{x_{i-1}}^{x_i} u'(x)dx \\ &= \frac{u(x_i) - u(x_{i-1})}{h_{i-1/2}} \\ &= \frac{u_i - u_{i-1}}{h_{i-1/2}} \end{aligned} \tag{178}$$

with

$$h_{i-1/2} = \frac{h_{i-1} - h_i}{2} \tag{179}$$

The borders require a special treatment with a different formulation.

Finite Volumes: Example

Left condition: use average value of u' on $[x_{1/2}, x_1]$ instead of $[x_0, x_1]$

$$\begin{aligned} u'(x_{1/2}) &= \frac{2}{h_1}\int_{x_{1/2}}^{x_1} u'(x)dx \\ &= \frac{2(u_1 - u(0))}{h_1} = \frac{2(u_1 - \alpha)}{h_1} \end{aligned} \tag{180}$$

Rigth condition: use average value of u' in $[x_N, x_{N+1/2}]$ instead of $[x_N, x_{N+1}]$

$$\begin{aligned} u'(x_{N+1/2}) &= \frac{2}{h_N}\int_{x_N}^{x_{N+1/2}} u'(x)dx \\ &= \frac{2(u(1) - u_N)}{h_N} = \frac{2(\beta - u_N)}{h_N} \end{aligned} \tag{181}$$

Finite Volumes: Example

The finite volume discretisation is:

$$\begin{cases} \dfrac{u_i - u_{i-1}}{h_{i-1/2}} - \dfrac{u_{i+1} - u_i}{h_{i+1/2}} = h_i \bar{f}_i \ \text{ for } \ i = 2, ..., N-1 \\ \dfrac{2(u_1 - \alpha)}{h_1} - \dfrac{u_2 - u_1}{h_{3/2}} = h_1 \bar{f}_1 \\ \dfrac{u_N - u_{N-1}}{h_{N-1/2}} - \dfrac{2(\beta - u_N)}{h_N} = h_N \bar{f}_N \end{cases} \tag{182}$$

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

$$\begin{cases} \dfrac{2u_i - u_{i-1} - u_{i+1}}{h^2} = \bar{f}_i \ \text{ for } \ i = 2, ..., N-1 \\ \dfrac{3u_1 - u_2}{h^2} = \bar{f}_1 + 2\dfrac{\alpha}{h^2} \\ \dfrac{3u_N - u_{N-1}}{h^2} = \bar{f}_N + 2\dfrac{\beta}{h^2} \end{cases} \tag{183}$$

Finite Volumes: 2D

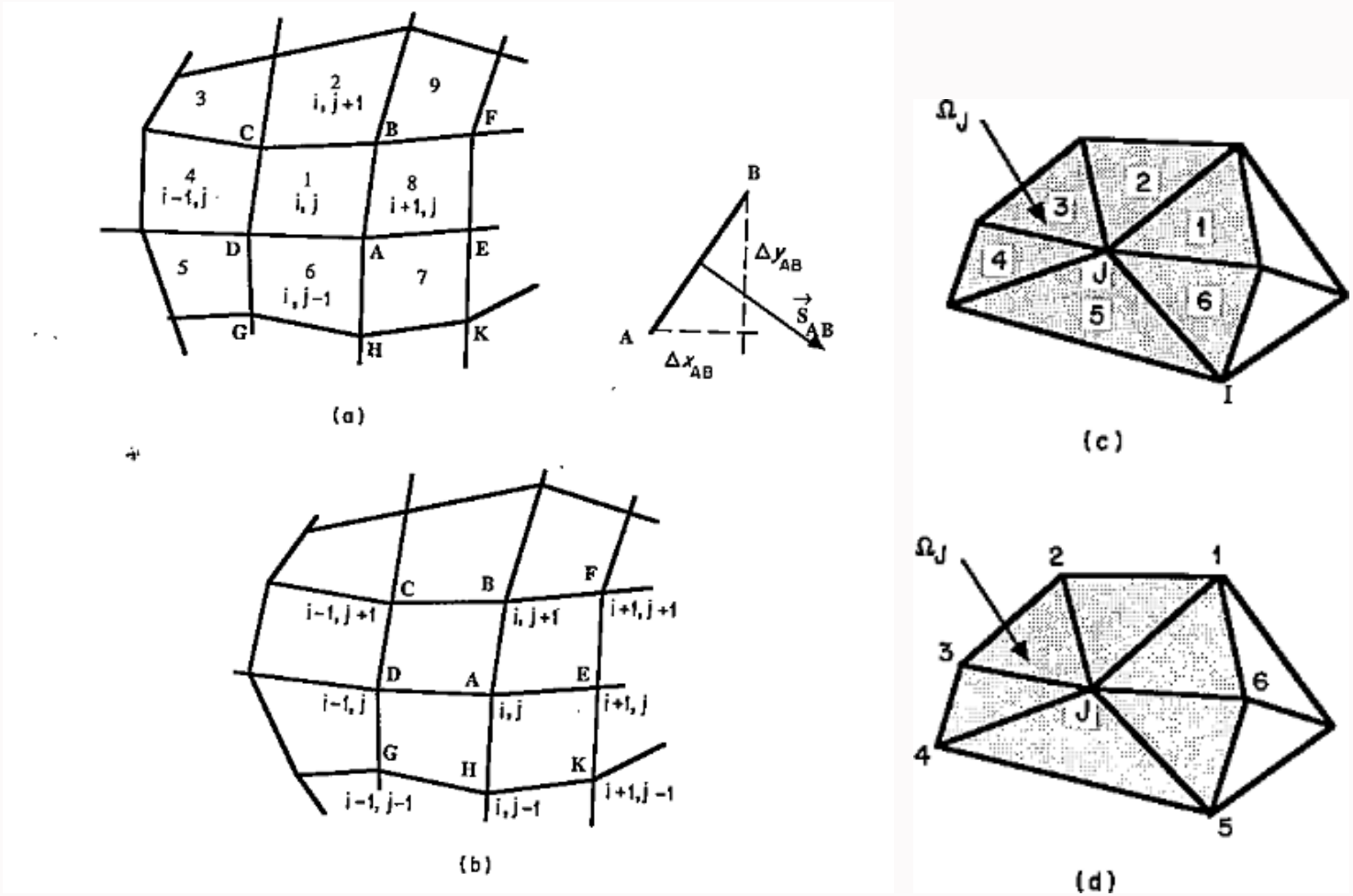


Figure: a) "cell-centered" structured finite volume mesh b) "cell vertex" structured finite volume mesh c) "cell-centered" unstructured finite volume mesh d) "cell vertex" unstructured finite volume mesh ([1])

Finite Volumes: 2D

The conservation equation for discrete volume is

$$\frac{\partial}{\partial t}\int_{\Omega} U\, d\Omega + \int_{S} \vec{F}\cdot d\vec{S} = \int_{\Omega} Q d\Omega \tag{184}$$

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

$$\frac{\partial}{\partial t}\int_{\Omega_{ij}} U\, d\Omega + \int_{ABCD} (f\, dy - g\, dx) = \int_{\Omega_{ij}} Q d\Omega \tag{185}$$

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

$$\vec{S}_{AB} = \Delta y_{AB}\, \vec{i} - \Delta x_{AB}\, \vec{j} = (y_B - y_A)\, \vec{i} - (x_A - x_B)\, \vec{j} \tag{186}$$

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

$$\frac{\partial}{\partial t}(U\Omega_{ij}) + \sum_{ABCD} [f_{AB}(y_B - y_A) - g_{AB}(x_B - x_A)] = (Q\Omega)_{ij} \tag{187}$$

Finite Volumes: 2D

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

$$\begin{aligned} \Omega_{ABCD} &= \frac{1}{2}|\vec{x}_{AC} \times \vec{x}_{BD}| \\ &= \frac{1}{2}\left[(x_c - x_A)(y_D - y_B) - (y_C - y_A)(x_D - x_B)\right] \\ &= \frac{1}{2}(\Delta x_{AC}\Delta y_{BD} - \Delta x_{BD}\Delta y_{AC}) \end{aligned} \tag{188}$$

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

Finite Volumes: 2D

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.

2 types of discretisation schemes:

- "centered" : based on a local estimation of the fluxes
- "upwind": the flux through the cell face is a function of the propagation direction of the wave component

Finite Volumes: 2D

For central scheme and cell-centered finite volume methods:

(1) Average fluxes :

$$f_{AB} = \frac{1}{2}\left(f_{ij} + f_{i+1,j}\right) \quad (189)$$

$$f_{ij} = f(U_{ij}) \quad (190)$$

This formulation is second order accurate.

(2) Other choice (not identical due to non-linearity) :

$$f_{AB} = f\left(\frac{U_{ij} + U_{i+1,j}}{2}\right) \quad (191)$$

Finite Volumes: 2D

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

$$f_{AB} = \frac{1}{2}(f_A + f_B) \tag{192}$$

where

$$f_A = f(U_A); \quad U_A = \frac{1}{4}(U_{ij} + U_{i+1,j} + U_{i+1,j-1} + U_{i,j-1}) \tag{193}$$

or the fluxes are averaged as

$$f_A = \frac{1}{4}(f_{ij} + f_{i+1,j} + f_{i+1,j-1} + f_{i,j-1}) \tag{194}$$

Finite Volumes: upwing scheme

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

$$\vec{A}(U) = \frac{\partial \vec{F}}{\partial U} = a\vec{x} + b\vec{y} \tag{195}$$

with

$$a(U) = \partial f/\partial U \quad \text{and} \quad b(U) = \partial g/\partial U. \tag{196}$$

The simplest upwind scheme takes the cell side flux equal to the flux generated in the upstream cell

$$\begin{aligned} (\vec{F}\cdot\vec{S})_{AB} &= (\vec{F}\cdot\vec{S})_{ij} \quad \text{if} \quad (\vec{A}\cdot\vec{S})_{AB} > 0 \\ (\vec{F}\cdot\vec{S})_{AB} &= (\vec{F}\cdot\vec{S})_{i+1,j} \quad \text{if} \quad (\vec{A}\cdot\vec{S})_{AB} < 0 \end{aligned} \tag{197}$$

Finite Volumes: upwing scheme

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

$$\begin{aligned} (\vec{F}\cdot\vec{S})_{AB} &= (\vec{F}\cdot\vec{S})_{CD} \quad \text{if} \quad (\vec{A}\cdot\vec{S})_{AB} > 0 \\ (\vec{F}\cdot\vec{S})_{AB} &= (\vec{F}\cdot\vec{S})_{EF} \quad \text{if} \quad (\vec{A}\cdot\vec{S})_{AB} < 0 \end{aligned} \tag{198}$$

Finite Volumes: upwing scheme

Property of upwind schemes:

- 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
- Very fine grids are required to obtain accurate solution

Finite Volumes: 2^{nd} order upwing scheme

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

$$f_{AB} = \lambda f_{i+1,j} + (1-\lambda) f_{ij} \tag{199}$$

where the linear interpolation factor λ is defined as

$$\lambda = \frac{x_{i+1/2,j} - x_{i,j}}{x_{i+1,j} - x_{i,j}} \tag{200}$$

The equation (199) is **second order accurate**

- simplest second order accurate and one of the most widely used
- as it is 2^{nd} order, it may produce oscillatory solutions

Finite Volumes: QUICK scheme

The next logical improvement is the approximation to variable profile between $x_{i,j}$ and $x_{i+1,j}$ by 2^{nd} order polynomial: need to use the data at one more point.

This point is taken on the upstream side:

- $x_{i-1,j}$ if the flow is from $x_{i,j}$ to $x_{i+1,j}$ $((\vec{A} \cdot \vec{S})_{AB} > 0)$
- $x_{i+1,j}$ if the flow is from $x_{i+1,j}$ to $x_{i,j}$ $((\vec{A} \cdot \vec{S})_{AB} < 0)$

For $(\vec{A} \cdot \vec{S})_{AB} > 0$:

$$f_{AB} = \frac{6f_{i-1,j} + 3f_{i,j} + f_{i+1,j}}{8} \tag{201}$$

The approximation is called the **Quadratic Upwind Interpolation (QUICK)** approximation and is third order accurate (exept when used with midpoint rule approximation of the surface integral $\Rightarrow$ stay 2^{nd} order

Brief Introduction to Finite Elements

Finite Elements (FE)

The method consist in approaching, in a finite dimension sub-space, a problem defined in a variational form in an infinite dimension space.

- method well suited for problem in equilibrium
- allows to deal with complex geometries
- expensive in terms of CPU time and memory requirement
- use by several commercial CFD codes (ANSYS, CATIA, ...)

Finite Elements: 1D case

Let's take a simple differential equation

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

Let's define a function $v(x) \in C^1([0,1])$, 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

$$\int_0^1 u'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx \quad \forall v \in V \tag{202}$$

with $V = v \in C^0([0,1]);\ \ v(0) = v(1) = 0,\ \ v'$ piecewise continuous, a vectorial subspace of $C^1([0,1])$.

A solution in variational form (202) is named a **weak solution** of the original differential problem.

Finite Elements: 1D case

we need to write the approximated problem in a finite dimension vectorial sub-space.

- $\tilde{V}$ a vectorial sub-space of V of finite dimension N.
- $\phi_1, \phi_2, ..., \phi_n$ be N functions linearly independent of V (create a basis of $\tilde{V}$)

Any function $\tilde{u}$ of $\tilde{V}$ can be decomposed as:

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

Finite Elements: 1D case

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

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

Finite Elements: 1D case

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

$$a_{ij} = \int_0^1 \phi_j'(x)\phi_i'(x)\, dx \quad \text{et} \quad b_i = \int_0^1 f(x)\phi_i(x)\, dx \tag{203}$$

By definition, the matrix **A** is symmetric.

If **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:

$$\mathbf{A} \cdot \mathbf{u} = \mathbf{b} \tag{204}$$

$\Rightarrow$ one must **choose the** N **functions** ϕ_i in order to lead to a simple system to solve:

Finite Elements: 1D case

One choice: $\phi_i(x)$ polynomial functions of degree 1

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

These function are named **one degree finite elements**

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

Finite Elements: 1D case

One degree finite elements leads to a tri-diagonal matrix **A**

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

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

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

Linear system to solve:

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

Finite Elements: 1D case

Comparison with to FD central second order formula

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

Rigorously identical formulation **with this choice of finite elements**

For **uniform distribution** of points the finite element discretisation becomes:

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

Finite Elements: summary

To summarize, the finite element methods consist in :

- Choosing N points between 0 and 1 and choosing the finite element 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

Solving Linear Systems

Some definitions

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

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \tag{205}$$

for $(n = m)$ the trace and determinant of $\mathbf{A}$ is define by:

$$tr(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}, \quad det(\mathbf{A}) = \begin{cases} a_{11} \text{ if } n = 1 \\ \sum_{j=1}^{n} \Delta_{ij} a_{ij} \text{ for } n > 1 \end{cases} \tag{206}$$

with

$$\Delta_{ij} = (-1)^{i+j} det(\mathbf{A}_{ij}) \tag{207}$$

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.

Some definitions

The **rank** of the matrix **A** (rg(**A**)) is the maximum number of independent column vectors of **A**.

For **A** a real or complex square matrix of order n; λ is a **eigenvalue of A** if it exists a vector $\mathbf{x} \neq 0$ such as $\mathbf{Ax} = \lambda\mathbf{x}$

The eigenvalues are solution of the following characteristic polynomial:

$$p(\lambda) = det(\mathbf{A} - \lambda I) = 0 \tag{208}$$

One can show that

$$det(\mathbf{A}) = \prod_{i=1}^{n} \lambda_i, \quad tr(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i \tag{209}$$

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

$$\rho(\mathbf{A}) = \max_{\lambda \in \sigma(\mathbf{A})} |\lambda| \tag{210}$$

Some definitions

Some properties of a matrix:

$$\begin{aligned} \text{Symmetric} &: \mathbf{A}^T = \mathbf{A} \\ \text{Orthogonal} &: \mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = 1 \\ \text{Hermitian} &: \mathbf{A} = \mathbf{A}^* \\ \text{Normal} &: \mathbf{A}\mathbf{A}^* = 1 \end{aligned}$$

Definition of some norm:

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

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

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

Some definitions

The matrix **A** is said to be diagonal dominant if:

$$\begin{cases} |a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1, .., n \ \text{(dominant by line)} \\ |a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ji}|, \quad i = 1, .., n \ \text{(dominant by column)} \end{cases} \tag{211}$$

Some definitions

Decomposition in singular values:

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

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

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 $\mathbf{A}$.

Introduction to linear system

A linear system of m equations and n unknown

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, ..., m, \tag{212}$$

can be written in matrix form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{213}$$

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

Introduction to linear system

If $(n = m)$, 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.

Introduction to linear system

The solution of the system $\mathbf{Ax} = \mathbf{b}$ is given by the **Cramer's formula**:

$$x_j = \frac{\Delta j}{det(\mathbf{A})} \tag{214}$$

where Δ_j is the determinant of the matrix obtained by replacing the jth column of $\mathbf{A}$ by $\mathbf{b}$.

However, the cost the Cramer's formula is prohibitive **(n+1)!**

Well posed problems

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

$$F(d, x) = 0 \tag{215}$$

where d is the ensemble of data from which the solution is dependent and F is the functional relation between x and d

Definition: The problem is **well posed** (or **stable**) if the solution x exist, is unique and continuously depends on the data d *(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:

$$\forall \epsilon > 0 \; \exists \; \delta(\epsilon) \text{ such as } \|\delta d\| \leq \delta \text{ then } \|\delta x\| \leq \epsilon \tag{216}$$

Well posed problems

relative conditioning propriety:

$$K(d) = sup \left\{ \frac{\|\delta x\|/\|x\|}{\|\delta d\|/\|d\|}, \delta d \neq 0, d + \delta d \in D \right\}. \tag{217}$$

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

$$K(d) = sup \left\{ \frac{\|\delta x\|}{\|\delta d\|}, \delta d \neq 0, d + \delta d \in D \right\}. \tag{218}$$

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.

Conditionning

Let's consider the linear system $\mathbf{Ax} = \mathbf{b}$ and only the data (which correspond to $\mathbf{b}$) are perturbed

The conditioning of the system can be estimated by

$$K(d) \simeq \frac{\|\mathbf{A}^{-1}\|\|\mathbf{b}\|}{\|\mathbf{A}^{-1}\mathbf{b}\|} = \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}\|\mathbf{A}^{-1}\| \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\| = K(A) \quad (219)$$

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

$$K_p(\mathbf{A}) = \|\mathbf{A}\|_p\|\mathbf{A}^{-1}\|_p \quad (220)$$

where $\|.\|_p$ is the p-norm and $K_p(\mathbf{A})$ is the conditioning of the matrix $\mathbf{A}$ under the p-norm

Conditionning

One can demonstrate several properties of the conditioning:

$$1 = \|\mathbf{A}\mathbf{A}^{-1}\| \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\| = K(\mathbf{A})$$
$$K(\mathbf{A}) = K(\mathbf{A}^{-1})$$
$$K(\alpha\mathbf{A}) = K(\mathbf{A}) \tag{221}$$
$$\tag{222}$$

For the norm $p = 2$ we have

$$K_2(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2 = \frac{\sigma_1(\mathbf{A})}{\sigma_n(\mathbf{A})} \tag{223}$$

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

Conditionning

Because of the round-off errors a numerical methods gives exact solution $(\mathbf{x} + \delta\mathbf{x})$ of the perturb system

$$(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b} \tag{224}$$

In such case we have the following relation:

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{K(\mathbf{A})}{1 - K(\mathbf{A})\|\delta\mathbf{A}\|/\|\mathbf{A}\|} \left(\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} \right) \tag{225}$$

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

$$\frac{1}{K(\mathbf{A})} \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \leq \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq K(\mathbf{A}) \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \tag{226}$$

Direct methods

For a system $\mathbf{Lx} = \mathbf{b}$ where $\mathbf{L}$ is a **lower triangular** invertible matrix

$$\begin{aligned} x_1 &= \frac{b_1}{l_{11}} \\ x_i &= \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right), \ i = 2, ..., n \end{aligned} \tag{227}$$

For a system $\mathbf{Ux} = \mathbf{b}$ where $\mathbf{U}$ is a **upper triangular** invertible matrix

$$\begin{aligned} x_n &= \frac{b_n}{u_{nn}} \\ x_i &= \frac{1}{u_{ii}} \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right), \ i = n-1, ..., 1 \end{aligned} \tag{228}$$

The total number of operations in the two cases is of the **order of** n^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.

$\mathbf{A} = \mathbf{A}^{(1)}$ and $\mathbf{b} = \mathbf{b}^{(1)}$ are the modified matrix and vector after the first step of the algorithm

$$m_{i,p} = \frac{a_{ip}^{(p)}}{a_{pp}^{(p)}} \tag{229}$$

The unknown x_1 can be eliminated from the lines $i = 2, ..., n$ by subtracting m_{i1} times the first line

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1} a_{1j}^{(1)}, \quad i, j = 2, ..., n, \tag{230}$$

$$b_i^{(2)} = b_i^{(1)} - m_{i1} b_1^{(1)}, \quad i = 2, ..., n, \tag{231}$$

Gauss Elimination

The new system $\mathbf{A}^{(2)}\mathbf{x} = \mathbf{b}^{(2)}$ (equivalent to the first one):

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \dots & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix} \tag{232}$$

Next step: eliminate x_2 of the lines $3, ..., n$ ... and so on
At the (n-1) step we obtain $\mathbf{A}^{(n)}\mathbf{x} = \mathbf{b}^{(n)}$:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & a_{2n}^{(2)} \\ \vdots & & \ddots & & \vdots \\ 0 & & & \ddots & \vdots \\ 0 & & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ \vdots \\ b_n^{(n)} \end{bmatrix} \tag{233}$$

Gauss Elimination

- The Gauss method requires that $a_{kk}^{(k)} \neq 0, 1 \leq k \leq n-1$
- $a_{ii} \neq 0$ does not prevent the appearance of $a_{kk}^{(k)=0}$.

Example:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix} \quad \Rightarrow \quad \mathbf{A}^{(2)} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & -1 \\ 0 & -6 & -12 \end{bmatrix} \tag{234}$$

The Gauss method must be stopped at the second step as $a_{22}^{(2)} = 0$

Gauss Elimination

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 $n^3/3$
  $\Rightarrow$ **rather expensive**.
- For large system that are not sparse, Gauss elimination is susceptible to accumulation errors
- The method is not easy to vectorize or parallelize

LU Decomposition

- 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 matrix $\mathbf{L}$ and $\mathbf{U}$ only depend on $\mathbf{A}$ and not on the right hand side $\mathbf{b}$
- The same factorization can be used to solve several system with the same matrix but different right hand side $\mathbf{b}$

LU Decomposition

In the Gauss procedure:

$$\mathbf{A}^{(k+1)} = \mathbf{M}_k \mathbf{A}^k \tag{235}$$

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

$$\mathbf{M}_k = \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & & 1 & 0 & & 0 \\ 0 & & -m_{k+1,k} & 1 & & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -m_{n,k} & 0 & \dots & 1 \end{bmatrix} \tag{236}$$

where the transformation matrix $\mathbf{M}_k$ is defined by:

$$(\mathbf{M}_k)_{ip} = \delta_{ip} - m_{ik}\delta_{kp} \tag{237}$$

LU Decomposition

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

$$\mathbf{M}_{n-1}\mathbf{M}_{n-2}\ldots\mathbf{M}_1\mathbf{A} = \mathbf{LU}. \tag{238}$$

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

$$\mathbf{A} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1}\ldots\mathbf{M}_{n-1}^{-1}\mathbf{U} = \mathbf{LU} \tag{239}$$

with

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \ldots & \ldots & 0 \\ m_{21} & 1 & & & \vdots \\ \vdots & m_{32} & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ m_{n1} & m_{n2} & \ldots & m_{n,n-1} & 1 \end{bmatrix} \tag{240}$$

LU Decomposition

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

$$\begin{aligned} \mathbf{Ly} &= \mathbf{b} \\ \mathbf{Ux} &= \mathbf{y} \end{aligned} \tag{241}$$

- The cost of the LU factorization is the same the the Gauss method
- Many system involving the same matrix can be solve using the same LU decomposition.

Thomas's algorithm

Let's consider a tri-diagonal matrix **A** defined as:

$$\mathbf{A} = \begin{bmatrix} a_1 & c_1 & & 0 \\ b_2 & a_2 & \ddots & \\ & \ddots & & c_{n-1} \\ 0 & & b_n & a_n \end{bmatrix} \tag{242}$$

The matrix **L** and **U** from the LU factorization of **A** are bi-diagonal:

$$\mathbf{L} = \begin{bmatrix} 1 & & & 0 \\ \beta_2 & 1 & & \\ & \ddots & \ddots & \\ 0 & & \beta_n & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \alpha_1 & c_1 & & 0 \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ 0 & & & \alpha_n \end{bmatrix} \tag{243}$$

where the coefficients α_i and β_i are defined by:

$$\alpha_1 = a_1, \beta_i = \frac{b_i}{\alpha_{i-1}}, \alpha_i = a_i - \beta_i c_{i-1}, i = 2, ..., n \tag{244}$$

Thomas's algorithm

This algorithm is known as the **Thomas algorithm**.

Using the LU factorization, the system can be solved using the following formula:

$$y_1 = b_1,\ y_i = b_i - \beta_i y_{i-1}, \quad i = 2, ..., n \tag{245}$$

$$x_n = y_n/\alpha_n,\ x_i = (y_i - c_i\ x_{i+1})/\alpha_i, \quad i = n-1, ..., 1. \tag{246}$$

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

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 operation of the order of n^2
- Iterative methods become useful if they can converge with a number of step $< n$
- 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.

Linear Iterative methods: convergence

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

$$\mathbf{x} = \lim_{k \to \infty} \mathbf{x}^{(k)} \tag{247}$$

where $\mathbf{x}$ is the solution of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$.

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.

Linear Iterative methods: convergence

Consider the matrix problem

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{248}$$

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

$$\mathbf{A}\mathbf{x}^{(k)} = \mathbf{b} - r^{(k)} \tag{249}$$

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

$$\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)} \tag{250}$$

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

$$\mathbf{A}\mathbf{e}^{(k)} = r^{(k)} \tag{251}$$

Linear Iterative methods: convergence

Let's consider an iterative scheme for a linear system which can be written as:

$$\mathbf{M}\mathbf{x}^{(k+1)} = \mathbf{N}\mathbf{x}^{(k)} + \mathbf{q} \tag{252}$$

or

$$\mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{q} \tag{253}$$

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

Since, by definition, at convergence $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} = \mathbf{x}$, we must have:

$$\mathbf{A} = \mathbf{M} - \mathbf{N} \quad \text{and} \quad \mathbf{q} = \mathbf{b} \tag{254}$$

or more generally

$$\mathbf{PA} = \mathbf{M} - \mathbf{N} \quad \text{and} \quad \mathbf{q} = \mathbf{Pb} \tag{255}$$

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

Linear Iterative methods: convergence

- Solving the system (252) must be cheap and the method must converge rapidly.
- Inexpensive iteration requires that the computation of $\mathbf{N}\,\mathbf{x}^{(k)}$ and solution of the system must be easy to perform.

  $\Rightarrow$ $\mathbf{M}$ must be easily inverted (diagonal, tri-diagonal or triangular, ...).
- For rapid convergence, $\mathbf{M}$ should be a good approximation of $\mathbf{A}$, making $\mathbf{N}\,\mathbf{x}$ "small"

Linear Iterative methods: Jacobi

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

$$x_i = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j \right], \quad i = 1, ..., n \tag{256}$$

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:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j^{(k)} \right], \quad i = 1, ..., n \tag{257}$$

Linear Iterative methods: Jacobi

This relation is equivalent to the following decomposition of the matrix **A**:

$$\mathbf{M} = \mathbf{D}, \quad \mathbf{N} = \mathbf{D} - \mathbf{A} = \mathbf{E} + \mathbf{F} \tag{258}$$

D is the diagonal matrix made of the diagonal coefficients of **A**
E is a upper triangular matrix of coefficients

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

F is a lower triangular matrix of coefficients

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

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

$$\mathbf{B}_J = \mathbf{M}^{-1}\mathbf{N} = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F}) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} \tag{259}$$

Linear Iterative methods: Jacobi Over Relaxation

A generalization of the Jacobi method is the *Jacobi over relaxation* (JOR) method where a relaxation parameter ω is introduced.

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j^{(k)} \right] + (1-\omega) x_i^{(k)}, \quad i = 1, ..., n \quad (260)$$

The corresponding iteration matrix is:

$$\mathbf{B}_{JOR} = \omega \, \mathbf{B}_J + (1-\omega) \, \mathbf{I}. \quad (261)$$

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

Linear Iterative methods: Gauss-Seidel

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:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right], \quad i = 1, ..., n \tag{262}$$

This method leads to the following decomposition of the matrix **A**:

$$\mathbf{M} = \mathbf{D} - \mathbf{E}, \quad \mathbf{N} = \mathbf{F} \tag{263}$$

and the associated iteration matrix is:

$$\mathbf{B}_{GS} = (\mathbf{D} - \mathbf{E})^{-1} \mathbf{F} \tag{264}$$

Linear Iterative methods: successive over relaxation

Starting for the Gauss-Seidel method one can define the *successive over relaxation* (SOR) method

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right] + (1-\omega) x_i^{(k)}, \quad i = 1, ..., n \tag{265}$$

The corresponding iteration matrix is

$$\mathbf{B}_{SOR}(\omega) = (\mathbf{I} - \omega\, \mathbf{D}^{-1}\mathbf{E})^{-1} \left[(1-\omega)\, \mathbf{I} + \omega\, \mathbf{D}^{-1}\, \mathbf{F} \right] \tag{266}$$

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 $\omega > 1$

Richardson stationary method

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

$$\mathbf{B} = \mathbf{I} - \mathbf{P}^{-1}\mathbf{A} \tag{267}$$

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

The iterative procedure can be generalized by introducing a **relaxation parameter** α.

This leads to the **Richardson stationary method**:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{P}^{-1}\mathbf{r}^{(k)}, \quad k \geq 0. \tag{268}$$

Richardson unstationary method

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

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{P}^{-1}\mathbf{r}^{(k)}, k \geq 0. \tag{269}$$

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

$$\mathbf{B}_{\alpha_k} = \mathbf{I} - \alpha_k \mathbf{P}^{-1}\mathbf{A} \tag{270}$$

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:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)} \tag{271}$$

and

$$\mathbf{r}^{(k+1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{A}\mathbf{z}^{(k)} \tag{272}$$

Richardson unstationary method

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 $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{A}\mathbf{z}^{(k)}$.

Richardson unstationary method

Theorem

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,*

$$\alpha_{opt} = \frac{2}{\lambda_1 + \lambda_n} \tag{273}$$

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

$$\rho_{opt} = \min_\alpha [\rho(\mathbf{B}_\alpha)] = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} \tag{274}$$

Multi-grid methods

- The rate of convergence of iterative methods depends on the eigenvalues of the iterative matrix associated with the method.
- The eigenvectors associated with the eigenvalues determines the spatial distribution of the convergence errors
- Some of iterative methods produce errors that are smooth function of the spatial coordinates
- If the error is smooth, the update can be computed on a coarse mesh (twice as coarse)
- Iterative methods converge much faster on coarser grids
- This suggest that much of the work can be done on a coarser grid
- Need to define:
  - the relationship between the two grid
  - the finite difference operator on the coarse grid
  - a method of smoothing the residual from the fine grid to the coarse grid
  - a method of interpolating the update or correction from the coarse grid to the fine one

Multi-grid methods

Algorithm of the two-grid iterative method:

- 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.

Convergence

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.
From the consistency condition, we have

$$\mathbf{e}^{(k+1)} = \mathbf{B}\mathbf{e}^{(k)} \tag{275}$$

or

$$\mathbf{e}^{(k)} = \mathbf{B}^k\mathbf{e}^{(0)} \tag{276}$$

and therefore, we have

$$\frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{e}^{(0)}\|} \leq \|\mathbf{B}^k\|. \tag{277}$$

So, $\|\mathbf{B}^k\|$ gives an estimation of the reduction factor of the error norm after k iteration.

Convergence

Typically, the iterative process is conducted until

$$\|\mathbf{e}^{(k)}\| \leq \epsilon \|\mathbf{e}^{(0)}\| \quad \text{with} \quad \epsilon < 1 \tag{278}$$

If we suppose $\rho(\mathbf{B}) < 1$, then there is a norm $\|\cdot\|$ such as $\|\mathbf{B}\| < 1$. Therefore, $\|\mathbf{B}^{(k)}\| \longrightarrow 0$ when $k \longrightarrow \infty$ and (278) can be satisfied for k large enough such as $\|\mathbf{B}^{(k)}\| < \epsilon$.

Nevertheless, as $\|\mathbf{B}^{(k)}\| < \epsilon$, the previous inequality becomes:

$$k \geq \frac{log(\epsilon)}{\left(\frac{1}{k} log \|\mathbf{B}^{(k)}\|\right)} = -\frac{log(\epsilon)}{R_k(\mathbf{B})} \tag{279}$$

where $R_k(\mathbf{B})$ is the **average convergence rate**.

Convergence

The above relation is not very useful as it is non-linear in k: can use the asymptotic rate of convergence $R(\mathbf{B})$

$$R(\mathbf{B}) = \lim_{k \to \infty} R_k(\mathbf{B}) = -log(\rho(\mathbf{B})) \tag{280}$$

to obtain the following estimation:

$$k_{min} \simeq -\frac{log(\epsilon)}{R(\mathbf{B})} \tag{281}$$

However, this estimation is rather optimistic.

Convergence: Criterion based on increments

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

$$\|\mathbf{e}^{(k+1)}\| \leq \|\mathbf{B}\|\|\mathbf{e}^{(k)}\| \tag{282}$$

As

$$x = \mathbf{e}^{(k+1)} + \mathbf{x}^{(k+1)} = \mathbf{e}^{(k)} + \mathbf{x}^{(k)} \tag{283}$$

and by using the triangular inequality, we have:

$$\|\mathbf{e}^{(k+1)}\| \leq \|\mathbf{B}\| \left(\|\mathbf{e}^{(k+1)}\| + \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \right) \tag{284}$$

and therefore,

$$\|\mathbf{e}^{(k+1)}\| \left(1 - \|\mathbf{B}\|\right) \leq \|\mathbf{B}\|\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \tag{285}$$

or

$$\|\mathbf{x} - \mathbf{x}^{(k+1)}\| \leq \frac{\|\mathbf{B}\|}{1 - \|\mathbf{B}\|}\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \tag{286}$$

Convergence: Criterion based on increments

by applying the recurrence formula (282):

$$\|\mathbf{x}^{(k+1)}-\mathbf{x}\| \leq \|\mathbf{B}\|\|\mathbf{x}^{(k)}-\mathbf{x}\| \leq \|\mathbf{B}\|^2\|\mathbf{x}^{(k-1)}-\mathbf{x}\| \leq \ldots \leq \|\mathbf{B}\|^{k+1}\|\mathbf{x}^{(0)}-\mathbf{x}\|$$

we obtain:

$$\|\mathbf{x}-\mathbf{x}^{(k+1)}\| \leq \frac{\|\mathbf{B}\|^{k+1}}{1-\|\mathbf{B}\|}\|\mathbf{x}^{(1)}-\mathbf{x}^{(0)}\| \qquad (287)$$

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

Convergence: Criterion based on increments

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

$$\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = -(\mathbf{x} - \mathbf{x}^{(k+1)}) + (\mathbf{x} - \mathbf{x}^{(k)}) = \mathbf{B}(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) \quad (288)$$

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 (286) we can have an indicator for $\|\mathbf{e}^{(k+1)}\|$

$$\epsilon^{(k+1)} = \frac{\delta_{k+1}^2}{\delta_k - \delta_{k+1}} \quad (289)$$

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

Time Integration

Time integration

The difference between space and time discretisation comes from the direction of influence

- a force at a given location will influence the flow in the whole domain (for elliptic problems)
- the forcing at a given time will influence the flow only in the future
  - unsteady flows are the equivalent of a *parabolic problem in time*
  - the time integration methods are step-by-step methods

Time integration: Two steps Methods

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

$$\frac{d\phi(t)}{dt} = f(t, \phi(t)); \quad \phi(t_0) = \phi^0 \tag{290}$$

If $t^n = n\Delta t$, the simplest integration methods can be defined by integrating the equation (290) from the time t^n to the time t^{n+1}:

$$\int_{t^n}^{t^{n+1}} \frac{d\phi(t)}{dt} dt = \phi^{n+1} - \phi^n = \int_{t^n}^{t^{n+1}} f(t, \phi(t)) \tag{291}$$

where $\phi^n = \phi(t^n)$.

$\Rightarrow$ approximated numerical formula can be used to evaluate the integral of the right-hand side

Time integration: Two steps Methods

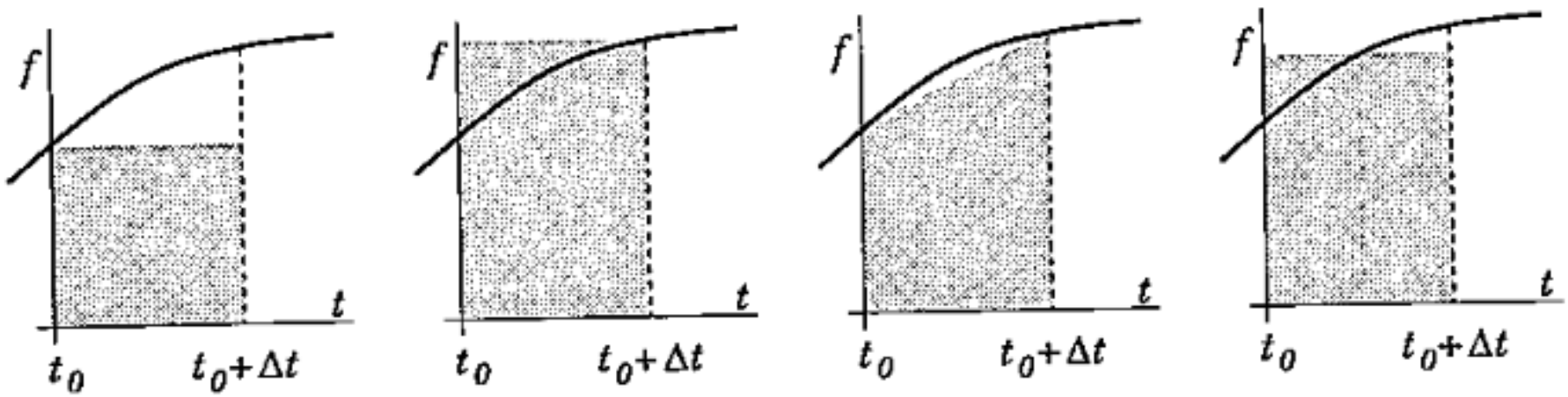


Figure: Time integration formulas (from [1])

Explicit method: ϕ^{n+1} can be computed directly from $\phi^k, k \leq n$
Implicit method: ϕ^{n+1} is only defined by a implicit relation using f.

Time integration: Explicit Euler method

The integral of the function f is estimated with the value of the function at the initial point t^n.

$$\phi^{n+1} - \phi^n = f(t^n, \phi^n)\Delta t \tag{292}$$

This is the simplest explicit method for time discretisation.

Time integration: Implicit (Backward) Euler method

The integral of the function f is estimated with the value of the function at the final point t^{n+1}.

$$\phi^{n+1} - \phi^n = f(t^{n+1}, \phi^{n+1})\Delta t \tag{293}$$

Implicit as R.H.S is function of t^{n+1}

Time integration: Leap-Frog Methods

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

$$\phi^{n+1} - \phi^n = f(t^{n+1/2}, \phi^{n+1/2})\Delta t \tag{294}$$

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

$$\phi^{n+1} - \phi^{n-1} = f(t^n, \phi^n)2\Delta t \tag{295}$$

The Leapd-Frog method is explicit and second order but:

- need to know the solution at ϕ^0 and ϕ^1 in order to used the formula
- solution of the even and odd time step number are independent (divergence of the solutions)

Time integration: Crank-Nicholson

By using the trapezoid rule to evaluate the integral of f, we obtain the **Crank-Nicholson** formula:

$$\phi^{n+1} - \phi^n = \frac{1}{2}\left[f(t^n, \phi^n) + f(t^{n+1}, \phi^{n+1})\right]\Delta t \qquad (296)$$

Time integration: Predictor Corrector methods

Aim of the *Predictor-Corrector* methods is to combine the advantages of explicit methods (simplicity, CPU resources) and implicit methods (stability for larger time step):

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

$$\phi^*_{n+1} = \phi^{n+1} + f(t^n, \phi^n)\Delta t \tag{297}$$

where * symbol indicates that this is an estimation of the solution at this time.

Corrector step: by applying the trapezoid rule using ϕ^*_{n+1}

$$\phi^{n+1} = \phi^n + \frac{1}{2}\left[f(t^n, \phi^n) + f(t^{n+1}, \phi^*_{n+1})\right] \tag{298}$$

This method is second order accurate.

Time integration: Multi-points versus Runge Kutta methods

In order to increase the order of accuracy, one must use information at more points.

The additional points can be :

- points at which the function f has already been computed $(t^n, t^{n-1}, t^{n-2}, ..., t^{n-p})$

  $\Rightarrow$ **multi-points Methods**

- points between t^n et t^{n+1} which are used only for the computation of the scheme $(t^{n+1/2}, ...)$

  $\Rightarrow$ **Runge Kutta methods**

Time integration: Adams Methods

- Adams methods are the most well known multi-points methods
- They are derived by fitting a polynomial to the derivatives at a number of points in time
- If a *Lagrange* polynomial is used to fit $f(t^{n-m}, \phi^{n-m}), f(t^{n-m+1}, \phi^{n-m+1}), ..., f(t^n, \phi^n)$ we obtain an **Adams Bashforth** (explicit) method of order $m+1$.
- If data at time t^{n+1} are included into the interpolation polynomial, **Adams Moulton** (implicit) methods are obtained

Time integration: Adams Bashforth

First order Adams Bashforth ($\Rightarrow$ corresponds to Explicit Euler)

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

Second order Adams Bashforth:

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

Third order Adams Bashforth:

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

Time integration: Adams Moulton

First order Adams Moulton ($\Rightarrow$ corresponds to Implicit Euler)

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

Second order Adams Moulton ($\Rightarrow$ corresponds to Crank-Nicholson)

$$\phi^{n+1} - \phi^n = \frac{1}{2}\left[f(t^n, \phi^n) + f(t^{n+1}, \phi^{n+1})\right]\Delta t \qquad (299)$$

Third order Adams Moulton:

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

Time integration: Adams Moulton

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 $6^t h$ 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 $6^t h$ order

Time integration: Multi-points methods

In practice, a $m-1$ order *Adams-Bashforth* method is usually used as predictor and a m order *Adams-Moulton* method as corrector.

Advantages of Multi-points methods:

- relatively easy to construct to program and to use.
- require a single evaluation of $f(t, \phi(t))$ by time step (cheap)

Drawback of Multi-points methods

- requires data from several prior points in time (need to store them)
- can not be started with the data at only the initial time

  $\Rightarrow$ *Solution: use smaller time steps using a lower order method*

Time integration: Runge Kutta methods

Runge-Kutta method: time discretisation between t^n et t^{n+1}.

The second order Runge-Kutta method consists of two steps:

1. half-step predictor based on the explicit Euler method
2. midpoint rule corrector (which make 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

Time integration: Fourth order Runge Kutta methods (RK4)

The 4 steps of the **fourth order Runge Kutta method** are:

1. 1^{st} predictor step using *explicit Euler* at $t^{n+1/2}$ $(\rightarrow \phi^*_{n+1/2})$
2. 2^{nd} predictor step using *implicit Euler* at $t^{n+1/2}$ $(\rightarrow \phi^{**}_{n+1/2})$
3. predictor step using *midpoint rule* at time t^{n+1} $(\rightarrow \phi^*_{n+1})$
4. corrector step using *Simpson formula* at t^{n+1} $(\rightarrow \phi_{n+1})$

$$
\begin{aligned}
\phi^*_{n+1/2} &= \phi^n + \frac{\Delta t}{2} f(t^n, \phi^n) \\
\phi^{**}_{n+1/2} &= \phi^n + \frac{\Delta t}{2} f(t^{n+1/2}, \phi^*_{n+1/2}) \\
\phi^*_{n+1} &= \phi^n + \Delta t\, f(t^{n+1/2}, \phi^{**}_{n+1/2}) \\
\phi^{n+1} &= \phi^n + \frac{\Delta t}{6} \Big[f(t^n, \phi^n) + 2f(t^{n+1/2}, \phi^*_{n+1/2}) \\
&\quad + 2f(t^{n+1/2}, \phi^{**}_{n+1/2}) + f(t^{n+1}, \phi^*_{n+1}) \Big] + O(\Delta t^5)
\end{aligned}
$$

Time integration: Fourth order Runge Kutta methods (RK4)

Main disavantage of Runge-Kutta methods:

- requires four evaluation of f per time step (**expensive in CPU**)
- intermediate evaluations must be stored (**expensive in memory**)

  $\Rightarrow$ **Low Storage fourth order method** which require less memory storage

However: Runge-Kutta methods of a given order are usuallt **more accurate** and **more stable** than the multi-points methods of the same order.

Stability, Consistency, Convergence

Stability, Consistency, Convergence

- **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 ($\Delta t \rightarrow 0, \Delta x \rightarrow 0$)
- **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

(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

Consistency

Example of the following conservative equation:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{300}$$

After discretisation and by using the Euler integration formula:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -\frac{a}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) \tag{301}$$

If the function is sufficiently derivable

$$u_i^{n+1} = u_i^n + \Delta t \left(\frac{\partial u}{\partial t}\right)_i^n + \frac{\Delta t^2}{2}\left(\frac{\partial^2 u}{\partial^2 t}\right) + \ldots \tag{302}$$

$$u_{i+1}^n = u_i^n + \Delta x \left(\frac{\partial u}{\partial x}\right)_i^n + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial^2 x}\right) + \frac{\Delta x^3}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right) + \tag{303}$$

$$u_{i-1}^n = u_i^n - \Delta x \left(\frac{\partial u}{\partial x}\right)_i^n + \frac{\Delta x^2}{2}\left(\frac{\partial^2 u}{\partial^2 x}\right) - \frac{\Delta x^3}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right) + \tag{304}$$

Consistency

By introducing the development into the equation (301):

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} - \left(\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x}\right)_i^n$$
$$= +\frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial^2 t}\right)_i^n + \frac{\Delta x^2}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right)_i^n + O(\Delta t^2, \Delta x^4) \quad (305)$$

As the right hand side of the equation tends to zero when Δt and Δx tend to zero, the numerical scheme is **consistent**.

- The scheme is first order in time and second order in space
- The scheme is first order if $\Delta t/\Delta x$ is kept constant
- The scheme is second order if $\Delta t/(\Delta x)^2$ is kept constant.

Consistency

Let's define: u_i^n the exact solution of the discretised equation
$\tilde{u}_i^n$ the exact solution of the numerical scheme

The equation (305) becomes :

$$\left(\frac{\partial \tilde{u}}{\partial t} + a\frac{\partial \tilde{u}}{\partial x}\right)_i^n = -\frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial^2 t}\right)_i^n - \frac{\Delta x^2}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right)_i^n + O(\Delta t^2, \Delta x^4) \tag{306}$$

- The exact solution of the finite difference equation **does not satisfy** exactly the partial differential equation
- The solution satisfy an equivalent partial differential equation (or **modified equation**)
- The difference between the two equations is the **truncation error** ϵ_T

Consistency

In this case:

$$\epsilon_T = -\frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial^2 t}\right)_i^n - \frac{\Delta x^2}{6}\left(\frac{\partial^3 u}{\partial^3 x}\right)_i^n + O(\Delta t^2, \Delta x^4) \tag{307}$$

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

$$\left(\frac{\partial u}{\partial t}\right)_i^n = -a\left(\frac{\partial u}{\partial x}\right)_i^n + O(\Delta t, \Delta x^2) \tag{308}$$

and for the second order derivative

$$\begin{aligned}\left(\frac{\partial^2 u}{\partial t^2}\right)_i^n &= -a\left(\frac{\partial^2 u}{\partial x \partial t}\right)_i^n + O(\Delta t, \Delta x^2) \\ &= +a^2\left(\frac{\partial^2 u}{\partial x^2}\right)_i + O(\Delta t, \Delta x^2)\end{aligned} \tag{309}$$

Consistency

Therefore,

$$\epsilon_T = -\frac{\Delta t}{2} a^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i^n - a\frac{\Delta x^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n + O(\Delta t^2, \Delta x^2) \qquad (310)$$

By keeping only the lowest order, the modified equation becomes

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = -\frac{\Delta t}{2} a^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + O(\Delta t^2, \Delta x^2) \qquad (311)$$

The rhs can be seen as a viscous term with a negative viscous coefficient $(-(\Delta t/2)a^2)$.

- **positive viscosity**: reduce the oscillations (gradients) $\Rightarrow$ **stable**
- **negative viscosity**: amplify the oscillations $\Rightarrow$ **unstable**

Stability

- Stability analysis restricted to the study of linear problems
- Stability analysis may be difficult for the initial value problems and problems with boundary conditions
- The **von Neumann** stability method is based on a development in frequency space.
  - one of the most used method
  - can be performed on the equations with constant coefficient and with periodic boundary conditions
  - with non-constant coefficients or non-linear terms, the information on stability becomes very limited
- linear stability is a necessary condition, but not a sufficient condition

Stability

Definition: Numerical scheme must not allow any error to grow infinitely when moving from one time step to the next one

Any error ϵ_i^n between the computed solution u and the exact solution $\tilde{u}$ must stay limited for $n \longrightarrow \infty$ at fixed Δt.

If the error is defined as

$$\epsilon_i^n = u_i^n - \tilde{u}_i^n \tag{312}$$

the stability condition can be written as

$$\lim_{n \longrightarrow \infty} |\epsilon_i^n| \leq K \qquad \text{at fixed } \Delta t \tag{313}$$

where K is independent of n.

Stability: Spectral decomposition of error

Example (300) using the explicit Euler integration scheme:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -\frac{a}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) \tag{314}$$

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 (314), 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 $\Rightarrow$ this means that the error evolves identically to the solution u_i^n.

Stability: Spectral decomposition of error

If period boundary conditions, the error ϵ_i^n can be decomposed in Fourier series for each time step n

$$\epsilon_i^n = \sum_{j=-N}^{N} E_j^n e^{I\ k_j\ i\Delta x} = \sum_{j=-N}^{N} E_j^n e^{I\ k_j\ i\ j\pi/N}$$

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.

$\Rightarrow$ Due to the linearity of the scheme **any harmonic** $E_j^n e^{li\phi}$ **of** ϵ_i^n **satisfies the scheme**.

Stability: Amplification factor

If we consider a single harmonic $E_j^n e^{I\phi}$, its temporal evolution is described by the same equation than for u_i^n (removing index j):

$$\frac{E^{n+1} - E^n}{\Delta t} e^{Ii\phi} + \frac{a}{2\Delta x}(E^n e^{I(i+1)\phi} - E^n e^{I(i-1)\phi}) = 0$$

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

$$\sigma = \frac{a\Delta t}{\Delta x} \tag{315}$$

Stability: Amplification factor

The stability condition will be satisfied if the amplitude of any error harmonic will not grow in time.

$$|G| = \left| \frac{E^{n+1}}{E^n} \right| \leq 1 \quad \forall \phi$$

where G, which is called the **amplification factor**, is function of time and space.

$$G - 1 + \frac{\sigma}{2} \cdot 2\, I\, sin(\phi) = 0$$

or

$$G = 1 - I\, \sigma\, sin(\phi)$$

Therefore, the stability condition is never verified as

$$|G|^2 = 1 + \sigma^2 (sin(\phi))^2$$

Central finite difference scheme used for the convection equation with an explicit Euler scheme **unconditionally unstable**

Stability: Example

Stability of the same partial differential equation but with an **upwind** discretisation scheme:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -\frac{a}{2\Delta x}(u_{i+1}^n - u_i^n) \tag{316}$$

By introducing $E^n e^{Ii\phi}$ in the equation, we obtain:

$$(E^{n+1} - E^n)e^{Ii\phi} + \sigma(E^n e^{Ii\phi} - E^n e^{I(i-1)\phi}) = 0$$

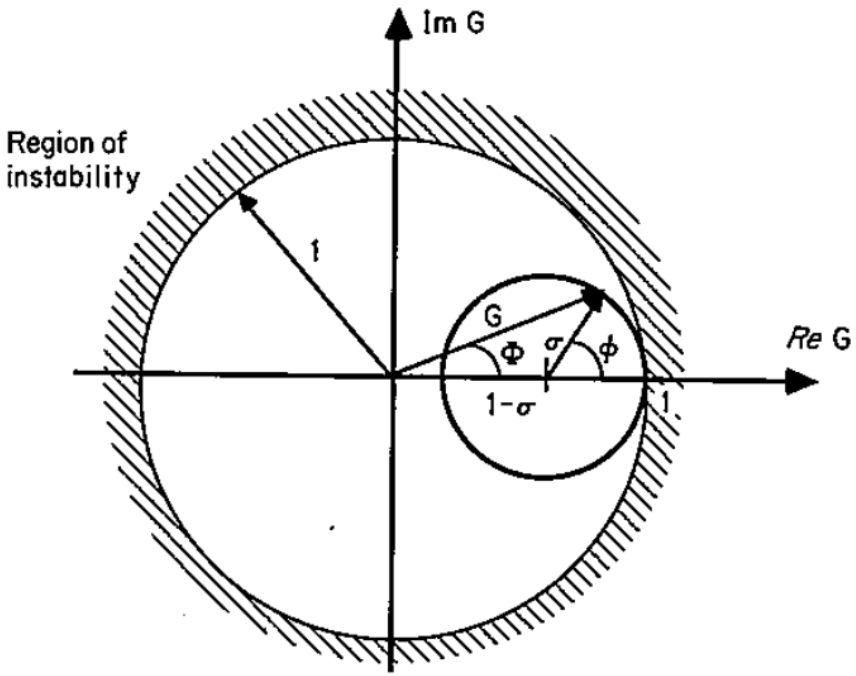
If we divide by $e^{Ii\phi}$:

$$\begin{aligned} G &= 1 - \sigma + \sigma e^{-I\phi} \\ &= 1 - 2\sigma sin(\phi/2)^2 - I\sigma sin(\phi) \end{aligned}$$

Stability

If ξ and η are the real part and the imaginary part of G respectively:

$$\begin{cases} \xi &= 1 - 2\sigma sin(\phi/2)^2 = (1-\sigma) + \sigma cos(\phi) \\ \eta &= -\sigma sin(\phi) \end{cases} \tag{317}$$



Can be seen as a parametric equation for G where ϕ is the parameter.

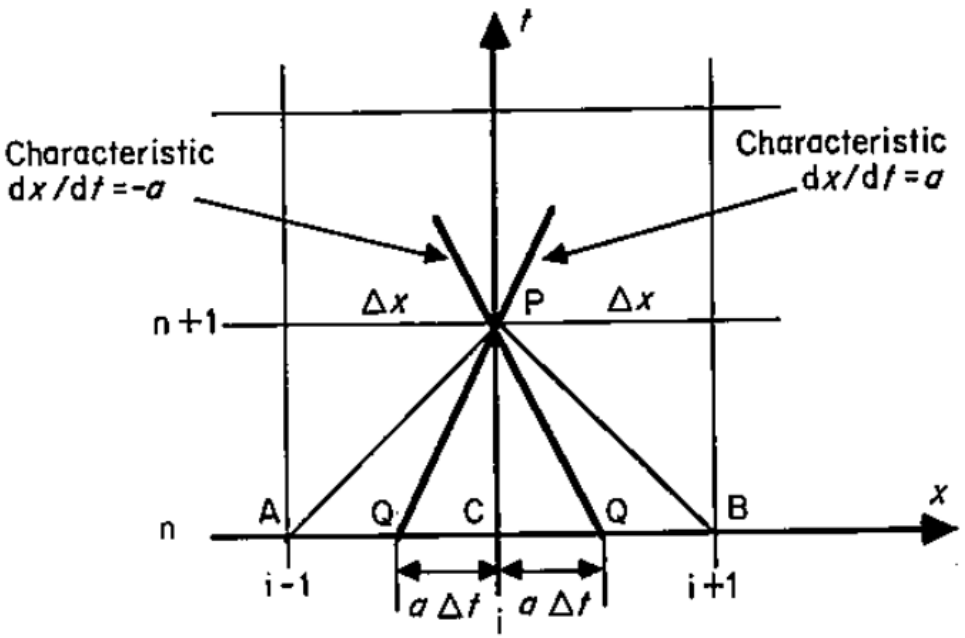
$\Rightarrow$ Parametric equation of a circle centered on the real axis ξ at $1-\sigma$ and with a radius σ.

$\Rightarrow$ Stability condition: $0 < \sigma \leq 1$ **(conditionally stable)**

Stability: CFL condition

The stability condition of most of the explicit schemes for the wave type equation or **convective equations**:

"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"



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.

Pressure Velocity Coupling

Navier Stokes equations: pressure

- difficulty to simulate the Navier Stokes equation because de pressure is not independent
- the continuity equation does not have a dominant variable for incompressible fluid
- masse conservation can be seen as a constrain on the velocity field instead of a real dynamical equation
- to overcome this difficulty, the solution is to compute pressure field which satisfy the continuity equation.
- numerical algorithm need to take into acount the strong pressure-velocity coupling

Navier Stokes equations: pressure

- Velocity component are compute from the momentum equation
- Pressure is computed from the continuity equation
- Pressure equation is obtained by combination of the two equations

By taking the divergence of the momentum equation, we obtain a **Poisson equation**:

$$div(grad(p)) = -div\left[div(\rho \mathbf{vv} - S) - \rho \mathbf{b} + \frac{\partial \rho \mathbf{v}}{\partial t}\right]$$

where **b** represent the body forces.

Navier Stokes equations: pressure

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

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial(\rho u_i u_j)}{\partial x_j}\right] \tag{318}$$

The rhs of the pressure equation is a sum of terms coming from the momentum equation $\Rightarrow$ it is important to use a discretisation which is consistent with the momentum equation.

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.
- Solving unsteady problems requires a time step adapted to the accuracy of the integration methods
- 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

Implicit Pressure-Correction Methods

Equation in a symbolic form:

$$A_P^{u_i} u_{i,P}^{n+1} + \sum_l A_l^{u_i} u_{i,l}^{n+1} = Q_{u_i}^{n+1} - \left(\frac{\delta p^{n+1}}{\delta x_i} \right)_P \tag{319}$$

- P is the arbitrary index of a velocity node
- l corresponds to the neighboring points involved in the discretised equations
- Q is the source term and its 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
- The pressure term is written in a symbolic way as the method does not depend on the discretisation which is chosen.

Implicit Pressure-Correction Methods

Because of the non-linearity in the coupling, the differential equation (319) 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
- For unsteady problems, the error at each time step may be larger as only the converged state will be of interest

- The iteration, within on time step, for which the coefficient of the source matrix will be updated is called an **external iteration**
- The iterations which may be used to solve the linear system with fixed coefficients are called **internal iteration**.

Implicit Pressure-Correction Methods

For each external iteration the equation to be solved are:

$$A_P^{u_i} u_{i,P}^{m*} + \sum_I A_I^{u_i} u_{i,I}^{m*} = Q_{u_i}^{m-1} - \left(\frac{\delta p^{m-1}}{\delta x_i} \right)_P \tag{320}$$

$n+1$ has been replaced by a index m (# of the external iteration)

u_i^{m*} is the estimation of the solution u_i^{n+1} at the current iteration m

Implicit Pressure-Correction Methods

The velocity at node P can be obtained by solving the linearized momentum equation (320):

$$u_{i,P}^{m*} = \frac{Q_{u_i}^{m-1} - \sum_l A_P^{u_i} u_{i,l}^{m*}}{A_P^{u_i}} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i} \right)_P$$

This equation does not satisfy a-priori the continuity equation as the pressure used in this equation is from the previous time step

Implicit Pressure-Correction Methods

For simplicity reason we write:

$$\tilde{u}_{i,P}^{m*} = \frac{Q_{u_i}^{m-1} - \sum_l A_P^{u_i} u_{i,l}^{m*}}{A_P^{u_i}}$$

So, the equation becomes

$$u_{i,P}^{m*} = \tilde{u}_{i,P}^{m*} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i} \right)_P$$

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.

Implicit Pressure-Correction Methods

The updated velocities and the pressure are linked by the following equation:

$$u_{i,P}^{m} = \tilde{u}_{i,P}^{m*} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^m}{\delta x_i} \right)_P \tag{321}$$

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:

$$\frac{\delta}{\delta x_i} \left[\frac{\rho}{A_P^{u_i}} \left(\frac{\delta p^m}{\delta x_i} \right) \right]_P = \left[\frac{\delta \rho \tilde{u}_i^{m*}}{\delta x_i} \right]_P \tag{322}$$

Next step: compute the updated velocities u_i^m using equation (321).

$\longrightarrow$ both the velocity field and the pressure satisfy the continuity equation but not the momentum equation (320)

$\longrightarrow$ must repeat the same procedure again and again until both the continuity equation and the momentum equations are satisfied by the velocity field.

SIMPLE Method

- 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.

SIMPLE Method

$$u_i^m = u_i^{m*} + u' \quad \text{et} \quad p^m = p^{m-1} + p' \tag{323}$$

Including the decompositions (323) into the momentum equation

$$u'_{i,P} = \tilde{u}'_{i,P} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p'}{\delta x_i} \right)_P \tag{324}$$

where $\tilde{u}'_{i,P}$ is defined by:

$$\tilde{u}'_{i,P} = -\frac{\sum_l A_l^{u_i} u'_{i,l}}{A_P^{u_i}} \tag{325}$$

SIMPLE Method

When the continuity equation is applied to the corrected velocities, and using (323):

$$\frac{\delta}{\delta x_i}\left[\frac{\rho}{A_P^{u_i}}\left(\frac{\delta p'}{\delta x_i}\right)\right]_P = \left[\frac{\delta \rho u_i^{m*}}{\delta x_i}\right]_P + \left[\frac{\delta \rho \tilde{u}_i'}{\delta x_i}\right]_P \tag{326}$$

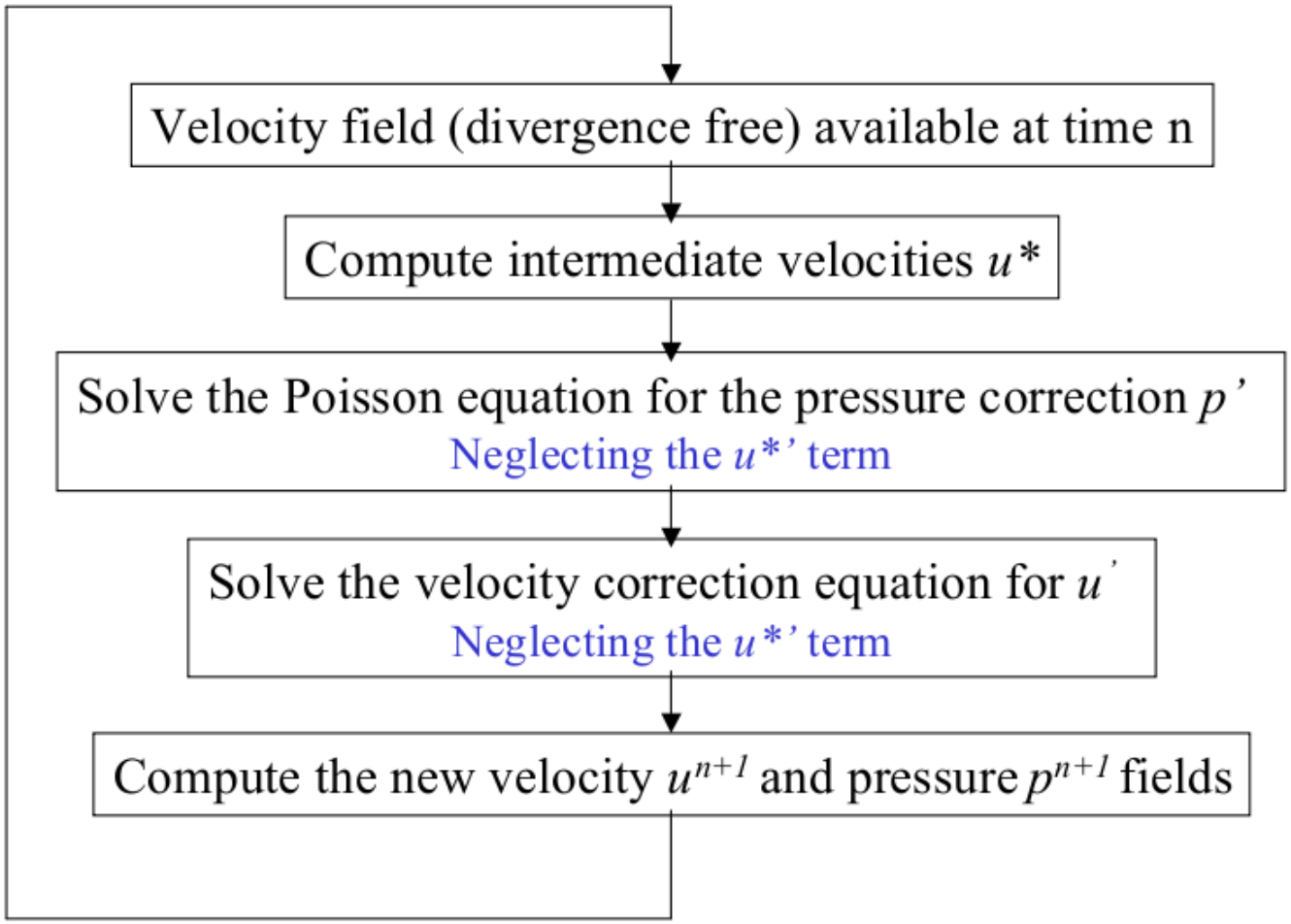
- $\longrightarrow$ the **correction of velocities** are unknown and it is usual to **neglect them**
- $\longrightarrow$ once the pressure correction is solved, the **velocities are updated** using the equations (323) and (324)

This method is known as the **SIMPLE** method *(Semi-Implicit Method for Pressure Linked Equations)*

SIMPLE Method: summary

Implicit pressure-based scheme for NS equations (SIMPLE)

SIMPLE: Semi-Implicit Method for Pressure-Linked Equations



SIMPLEC Method

Approximation of the velocity correction u'_i (instead of neglecting it) by a weighted value of the neighboring nodes.

$$u'_{i,P} \simeq \frac{\sum_l A_l^{u_i} u'_{i,l}}{\sum_l A_l^{u_i}} \tag{327}$$

This allows an approximation of $\tilde{u}'_{i,P}$ from equation (325) by:

$$\tilde{u}'_{i,P} \simeq -u'_{i,P} \frac{\sum_l A_l^{u_i}}{A_P^{u_i}}, \tag{328}$$

Once inserted into (324)

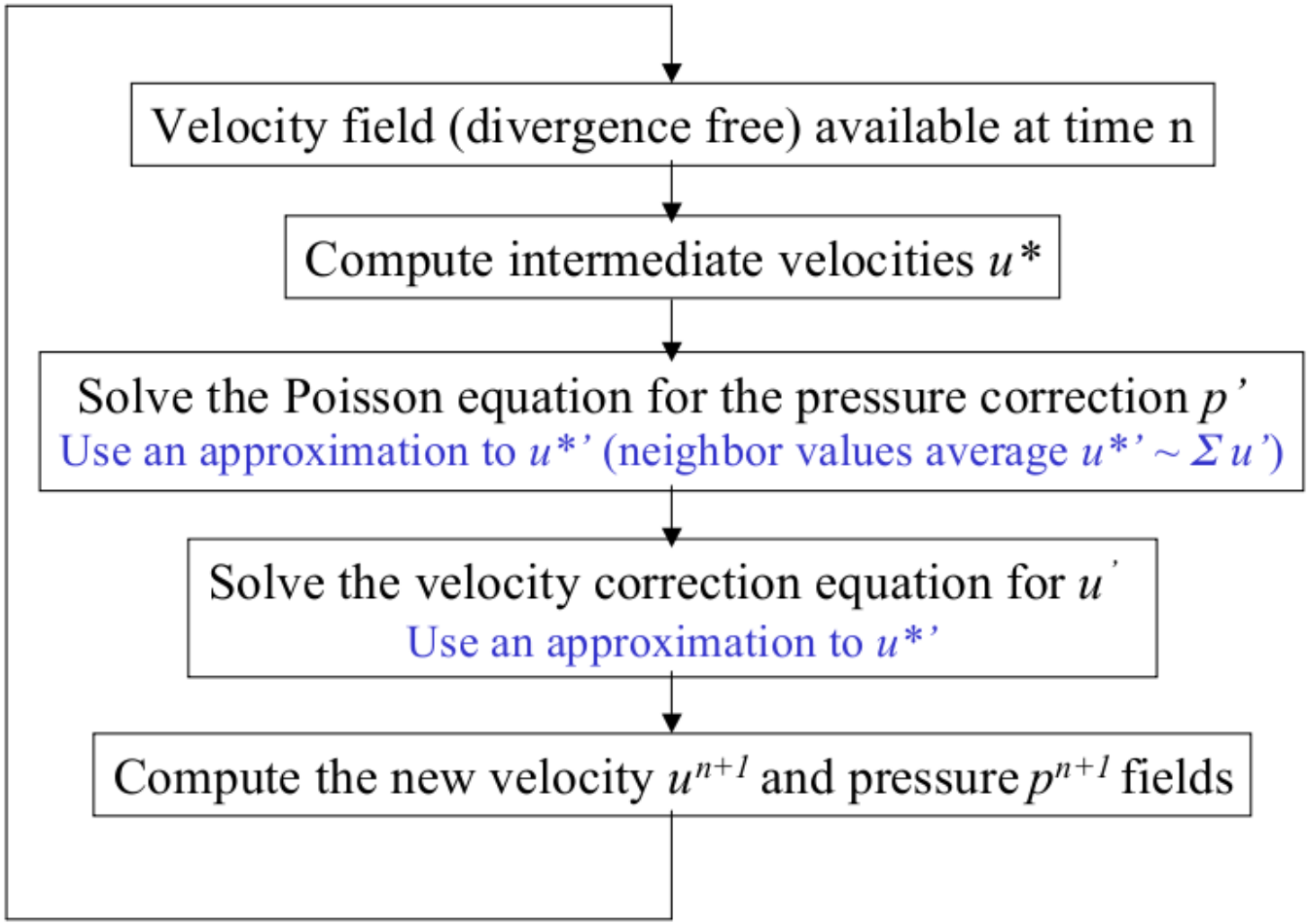
$$u'_{i,P} = -\frac{1}{A_P^{u_i} + \sum_l A_l^{u_i}} \left(\frac{\delta p'}{\delta x_i} \right)_P \tag{329}$$

This methods is known as the **SIMPLEC** method.

SIMPLEC method: summary

Implicit pressure-based scheme for NS equations (SIMPLEC)

SIMPLE: SIMPLE Corrected/Consistent



PISO Method

- neglect $\tilde{u}'_{i,P}$ in the first correction step (same as SIMPLE)
- apply a new step of velocity correction u''

$$u''_{i,P} = \tilde{u}'_{i,P} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p''}{\delta x_i} \right)_P \tag{330}$$

where

- $\tilde{u}'_i$ is computed by the equation (325)
- u'_i is computed by the equation (324) (neglecting $\tilde{u}'_i$)

The application of the discretised continuity equation to the corrected velocity leads to a second correction equation for the pressure.

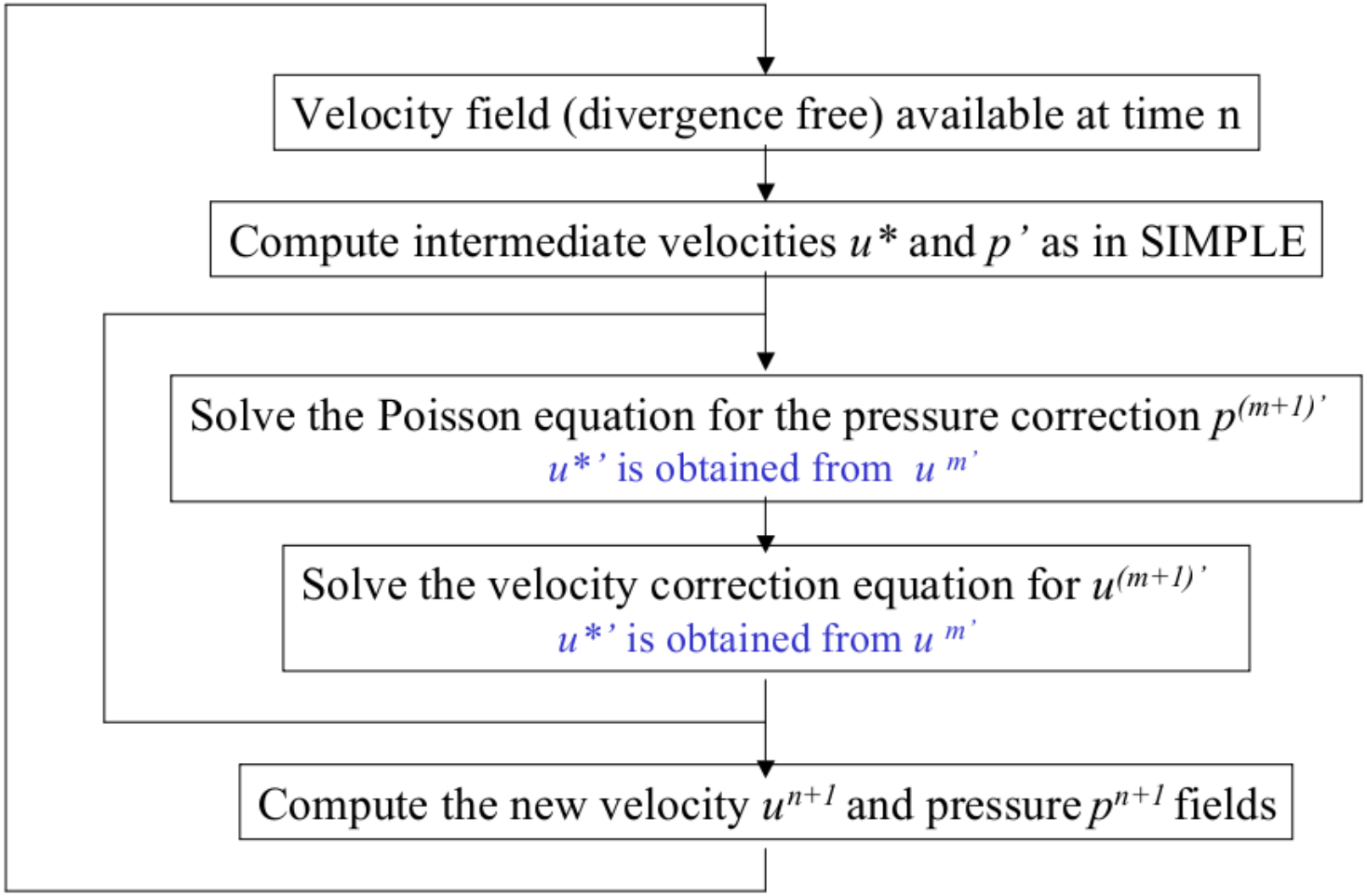
$$\frac{\delta}{\delta x_i} \left[\frac{\rho}{A_P^{u_i}} \left(\frac{\delta p''}{\delta x_i} \right) \right]_P = \left[\frac{\delta \rho \tilde{u}'_i}{\delta x_i} \right]_P \tag{331}$$

Additional correction steps can be applied (but rarely done in practice)

Implicit pressure-based scheme for NS equations: PISO

Implicit pressure-based scheme for NS equations (PISO)

PISO: Pressure Implicit with Splitting Operators



SIMPLER Method

- The equation for the pressure correction (326) is solved with the last term being neglected (as 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 (322) by replacing $\tilde{u}_i^{m*}$ par $\tilde{u}_i^m$ ($\tilde{u}_i^m$ is known)

This methods is known as the **SIMPLER** method.

Fractional Step methods

- Method developped by (Kim & Moin, 1985) [2]
- The method is more a generic approach than a particular method
- Does not use pressure in the predictor step
- For incompressible flow, the aim of the pressure is to satisfy the continuity equation (mathematical variable)

Fractional Step methods

The discretised momentum equation can be written in symbolic form:

$$\frac{(\rho u_i)^* - (\rho u_i)^n}{\Delta t} = \frac{1}{2}\left[H(u_i^n) + H(u_i^*)\right] - \frac{\delta p^n}{\delta x_i} \qquad (332)$$

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.

Fractional Step methods

In a second step, half of the old pressure gradient is removed from u_i^* leading to u_i^**:

$$\frac{(\rho u_i)^{**} - (\rho u_i)^*}{\Delta t} = \frac{1}{2}\frac{\delta p^n}{\delta x_i} \tag{333}$$

and the velocity u_i^{n+1} is estimated by:

$$\frac{(\rho u_i)^{n+1} - (\rho u_i)^{**}}{\Delta t} = -\frac{1}{2}\frac{\delta p^{n+1}}{\delta x_i} \tag{334}$$

with p^{n+1} solution of the Poisson equation:

$$\frac{\delta}{\delta x_i}\left(\frac{\delta p^{n+1}}{\delta x_i}\right) = \frac{2}{\Delta t}\frac{\delta(\rho u_i)^{**}}{\delta x_i} \tag{335}$$

Fractional Step methods

The new velocity (obtained from (334) satisfies the continuity equation and the momentum equation of the form:

$$\frac{(\rho u_i)^{n+1} - (\rho u_i)^n}{\Delta t} = \frac{1}{2}\left[H(u_i^n) + H(u_i^*)\right] - \frac{1}{2}\left(\frac{\delta p^n}{\delta x_i} + \frac{\delta p^{n+1}}{\delta x_i}\right) \quad (336)$$

- To represent the Cranck-Nicholson method correctly, $H(u_i^*)$ should be replaced by $H(u_i^{n+1})$.
- The error is of the second order in time

The equation for the pressure correction can be obtained by subtracting (332) from (336)

$$\frac{(\rho u_i)^{n+1} - (\rho u_i)^*}{\Delta t} = -\frac{1}{2}\frac{\delta p'}{\delta x_i} \quad (337)$$

Fractional Step methods / SIMPLE-type methods

Difference between fractional-step methods and SIMPLE-type methods:

- Fractional-step methods:
  - $\longrightarrow$ the pressure (or pressure-correction) equation is solved once per time step.
  - $\longrightarrow$ more suitable for unsteady flow
- SIMPLE-type methods:
  - $\longrightarrow$ both the momentum equation and the pressure-correction equations are solved several times within each time step (outer iteration)
  - $\longrightarrow$ more suitable for steady flows (continuity equation must be satisfy accurately only at convergence)
  - $\longrightarrow$ with large Δt, the pressure correction equation must be solve for each internal iteration

Artificial compressibility method

- The compressible flows are hyperbolic and the incompressible equations are parabolic-hyperbolic
- If the methods for compressible flows are to be used for incompressible flows, the character of the equations will need to be modified $\Rightarrow$ add time derivative in the continuity equation.
- As the density is constant, the best choice is to introduce a time derivative of the pressure.

$$\frac{1}{\beta}\frac{\partial p}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = 0, \tag{338}$$

where β is the artificial compressibility ($\sqrt{\beta}$ represents the speed of sound in the transformed system)

The time history of the simulation generated by this method **is not accurate** and the method should not be used to simulate unsteady flows.

Artificial compressibility method

Let's consider the implicit Euler scheme.

$$\frac{p_P^{n+1} - p_P^n}{\beta \Delta t} + \left[\frac{\delta(\rho u_i)}{\delta x_i} \right]_P^{n+1} = 0 \tag{339}$$

u_i^{n+1} is unknown. However $(\rho u_i)^{n+1}$ may be approximated as:

$$(\rho u_i)^{n+1} \simeq (\rho u_i^*)^{n+1} + \left[\frac{\partial(\rho u_i^*)}{\partial p} \right]^{n+1} (p^{n+1} - p^n) \tag{340}$$

By inserting this expression into the continuity equation (339) we obtain an equation for the new pressure p^{n+1}

Brief Introduction to Compressible Flows

Compressible flows

- Numerical methods need to be adapted to compressibles flows
- The equations for compressible flows are hyperbolic
- Sound and waves are able to travel at finite speed
- Compressible flows can include discontinuities like chocs
  - $\longrightarrow$ Important impact in the properties of the numerical schemes
  - $\longrightarrow$ Chocs are very difficult to simulate accuratly
  - $\longrightarrow$ The scheme must respect special properties (conservation, TVD)

Compressible flows

Differential form of the conservation laws inside a domain Ω

$$\frac{d}{dt}\int_\Omega \mathscr{A}\, d\Omega + \int_\Sigma a \cdot n d\Sigma = \int_\Omega A d\Omega \tag{341}$$

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

	$\mathscr{A}$	a	A
masse	ρ	0	0
momentum	ρU	$-\sigma$	ρg
energy	ρE	$q - \sigma U$	$r + \rho(gU)$

- σ is the stress tensor
- q is the heat flux by conduction
- r is the volume heat echange

Compressible flows: discontinuities

The general conservation law can be applied to a domain including a discontinuity δ.

jump relations:

$$[\![\mathscr{A}(U - S_\delta)n_\delta + an_\delta]\!] = 0 \tag{342}$$

where $[\![\Psi]\!]$ is the *jump* normal to the discontinuity n_δ.

Compressible flows: Euler equation

Let consider the 1D Euler (inviscid) equation

$$\begin{cases} \dfrac{\partial w}{\partial t} + \dfrac{\partial f(w)}{\partial x} = 0 \\ \\ w(x,0) = w_0(x) \end{cases} \tag{343}$$

with

$w = (\rho, \rho u, \rho E)$ is the vector of conservative variables

$f(w) = (\rho u, \rho u^2 + p, \rho E u + p u)$ is the flux vector

Can have solutions including some discontinuities (choc or contact discontinuity) $\Rightarrow$ need to extend the concept of "classic"solutions

Compressible flows: Euler equation

Introduce a test function $\phi(x,t)$ continuously derivable with a compact support

By integrating (343):

$$\int_0^\infty \int_{-\infty}^\infty \left[\phi \frac{\partial w}{\partial t} + \phi \frac{\partial f}{\partial x}\right] dx\ dt = 0 \qquad (344)$$

after integrating by part:

$$\int_0^\infty \int_{-\infty}^\infty \left[\frac{\partial \phi}{\partial t} w + \frac{\partial \phi}{\partial x} f\right] dx\ dt = \int_{-\infty}^\infty \phi(x,0) w(x,0) dx \qquad (345)$$

Definition: *A function $w(x,t)$ is a weak solution of the conservation law if (345) is satisfied for any function $\phi \in C_0^1$*

Compressible flows: Euler equation

If $[\![u]\!]$ is the jump of a function u through a discontinuity, the weak solution can be characterized by using the following theorem:

Theorem

A function $w(x,t)$ $\mathbb{C}^1$ by piece and which satisfy the initial condition (343) 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)*

$$[\![f(w)]\!] = s[\![w]\!] \tag{346}$$

through the discontinuity, where s id the speed of the discontinuity.

Compressible flows: Characteristics

The Jacobian associated to the system (343) is

$$\mathbf{A}(w) = \frac{df}{dw} = \begin{bmatrix} 0 & 1 & 0 \\ -\frac{\gamma-1}{2}u^2 & (3-\gamma)u & \gamma-1 \\ (\frac{\gamma-1}{2}u^2 - H)u & H-(\gamma-1)u^2 & \gamma u \end{bmatrix} \quad (347)$$

Where $H = E + p/\rho$ is the total enthalpy.

(343) is completly hyperbolic $\Rightarrow$ **A** has **3 real and distinct** eigenvalues:

$$\lambda^1(w) = u - c, \quad \lambda^2(w) = u, \quad \lambda^3(w) = u + c \quad (348)$$

where $c = \sqrt{\gamma p/\rho}$ is the **speed of sound**

The eigenvalues represents the **speed of information** inside the flow.

Characteristics: curves defined by $dx/dt = \lambda^k(w), k \in 1, 2, 3$.

Compressible flows: conservative form

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 system of equation, 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 inequallity in order to select the possible physical solution

Compressible flows: conservative form

To be in the conservative form, the scheme must satisfy

$$\frac{w_j^{n+1} - w_j^n}{\delta t} = -\frac{1}{\delta x}\left[F(w_{j-p}^n, ...w_{j+q}^n) - F(w_{j-p-1}^n, ...w_{j+q-1}^n)\right] \tag{349}$$

where $F(w_{j-p}^n, ...w_{j+q}^n)$ is the numerical flux of the scheme.

The scheme is consistant if $F(U, .., U) = f(U)$.

- Conservation property of the numerical method is fondamental for the computation of weak solutions
- Otherwise, it is possible to compute solutions of flows with chocs propagating with the wrong speed

Theorem (Lax-Wendroff, [3])

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.

Compressible flows: Lax scheme

Let consider the Euler one-dimensional equation (343) discretized with a one order central scheme:

$$\frac{w_j^{n+1} - w_j^n}{\Delta t} = -\frac{f_{j+1}^n - f_{j-1}^n}{2\Delta x} \tag{350}$$

This scheme is unstable.

The **Lax scheme** is based on the previous scheme, but w_j^n is replaced by $\frac{1}{2}(w_{j-1}^n + w_{j+1}^n)$.

$$w_j^{n+1} = \frac{1}{2}(w_{j-1}^n + w_{j+1}^n) - \frac{\Delta t}{2\Delta x}(f_{j+1}^n - f_{j-1}^n) \tag{351}$$

The scheme is still explicit and first order but is **conditionaly stable** ($CFL \leq 1$).

Compressible flows: Lax-Wendroff scheme

The basic approach of the **Lax Wendroff scheme** is to use the time serie expansion:

$$w_j^{n+1} = w_j^n = \Delta t \frac{\partial w}{\partial t} + \frac{(\Delta t)^2}{2} \frac{\partial^2 w}{\partial t^2} + \frac{(\Delta t)^3}{6} \frac{\partial^3 w}{\partial t^3} + \ldots \tag{352}$$

The term in $(\Delta t)^2$ is maintained and replaced by the space derivative term using the original equation:

$$\begin{aligned} \frac{\partial^2 w}{\partial t^2} &= -\frac{\partial}{\partial t}\left(\frac{\partial f}{\partial x}\right) = -\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial t}\right) = -\frac{\partial}{\partial x}\left(\frac{df}{dw}\frac{\partial w}{\partial t}\right) \\ &= -\frac{\partial}{\partial x}\left(A\frac{\partial w}{\partial t}\right) = \frac{\partial}{\partial x}\left(A\frac{\partial f}{\partial x}\right) \end{aligned} \tag{353}$$

where $A = \dfrac{df}{dw}$ is the Jacobian.

Compressible flows: Lax-Wendroff scheme

Then replacing this term into the Taylor expansion, one can define the one-step**non-linear** version of the Lax-Wendroff scheme:

$$\begin{aligned} w_j^{n+1} &= w_j^n - \frac{1}{2}\sigma(f_{i+1}^n - f_{i-1}^n) \\ &+ \frac{1}{2}\sigma^2\left[A_{i+1/2}^n(f_{i+1}^n - f_i^n) - A_{i-1/2}^n(f_i^n - f_{i-1}^n)\right] \end{aligned} \quad (354)$$

where $\sigma = \Delta t/\Delta x$ and

$$A_{i+1/2} = A(w_{i+1/2}) \quad (355)$$

or

$$A_{i+1/2} = \frac{1}{2}(A_i + A_{i+1}) \quad (356)$$

Compressible flows: Lax-Wendroff scheme

For the **linear form**: one more step in the evaluation of eq. (353)

$$\frac{\partial^2 w}{\partial t^2} = \frac{\partial}{\partial x}\left(A\frac{\partial f}{\partial x}\right) = \frac{\partial}{\partial x}\left(A^2\frac{\partial w}{\partial x}\right) \tag{357}$$

and the Lax-Wendroff scheme is defined by

$$\begin{aligned} w_j^{n+1} = w_j^n \quad &- \quad \frac{1}{2}\sigma(f_{i+1}^n - f_{i-1}^n) \\ &+ \quad \frac{1}{2}\sigma^2 A_j^2(w_{i+1}^n - 2w_i^n + w_{i-1}^n) \end{aligned} \tag{358}$$


C. Hirsch.
Numerical computation of internal and external flows. Volume 1: fundamentals of Numerical Discretization.
John Wiley & Sons, 1988.


J. Kim and P. Moin.
Application of a fractional-step method to incompressible Navier-Stokes equations.
J. Comp. Phys., 59:308–323, 1985.


P. D. Lax and B. Wendroff.
System of concervation laws.
Comm. Pure and Applied Mathematics, 13:217–237, 1960.