BIOE60009/BIOE70030 - Physiological Fluid Mechanics

Introduction to Numerical Methods and Computational Fluid Dynamics

Dr Choon Hwai Yap
c.yap@imperial.ac.uk

Binghuan Li
binghuan.li19@imperial.ac.uk

The original version of these notes is provided courtesy of Dr Jennifer H. Tweedy.

Last Update: January 1, 2026

Contents

Accessibility Statement: More accessible versions of these notes can be provided electronically upon individual request. Adjustments may include left justification, alternative fonts, or other formatting changes as appropriate.

IMPERIAL

1 Overview

1.1 Motivation: Why Numerical Solutions?

When solving a physical problem governed by a set of differential equations, it is **not** always possible to simplify the equations to obtain an analytical solution. For example, the following scenarios may impede us from applying the assumptions to sufficiently simplify the governing equation for obtaining an explicit expression of the solution:

- Non-linear governing equations *e.g.*, 3D Navier-Stokes;
- Complex boundary conditions, *e.g.*, spatially and temporally varying boundary conditions.
- Irregular geometry domains, *e.g.*, blood flow in a bifurcated, curved vessel;

In such cases, it is often the best approach to solve the problem using computer simulations to obtain the solution numerically, *i.e.*, finding the solution subjected to the governing equation and boundary conditions at a discrete set of points (or, elements) in space and/or time, rather than using an analytical expression.

An alternative interpretation of the numerical solution is closely linked to the concept of a 'field', for example:

- In fluid dynamics, we are interested in the velocity field and the pressure field.
- In heat transfer, the temperature field is sought.
- In electromagnetics, the electric and magnetic fields must be resolved.

The numerical solution provides the entire field that describes the physical system directly, while an analytical solution yields a mathematical expression for the solution variables (yes, you can construct the field from the expression).

1.2 Overview of the Computer Simulation Procedure

In this course, we are interested in using computer simulations to solve fluid problems, and this technique is more widely known as computational fluid dynamics (CFD). Invented in the 1960s and thriving since the 1970s, CFD has been successfully applied in numerous areas of fluid mechanics. For example, CFD is consistently playing a crucial role in the study of the aerodynamics of cars and aircraft, or in the design of the turbomachinery device involving pumps. For us bioengineers, CFD offers valuable insights into the complex biological flows that are conventionally difficult to analyse *ex-vivo*: examples are blood circulation in vessels, airflow in the lungs, or complex transport phenomena such as drug delivery.

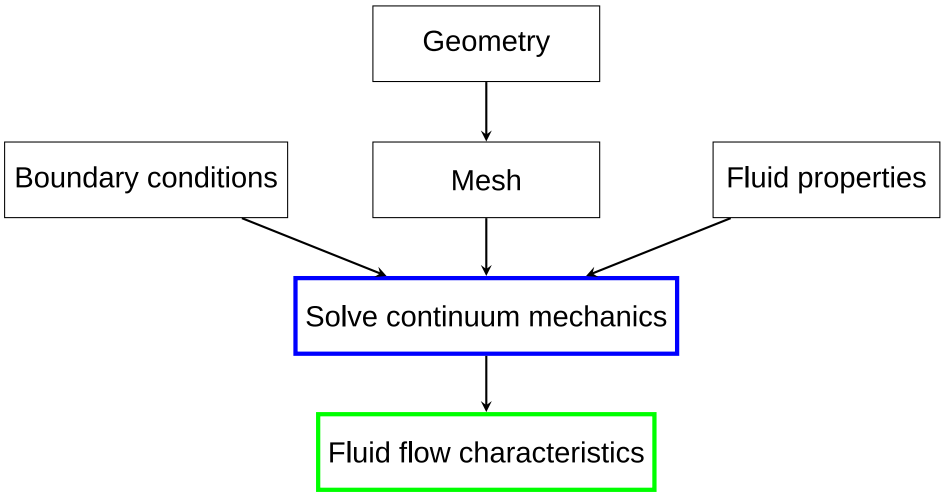
How does CFD work in action? Figure 1.1(a) presents a simplified CFD workflow, accompanied by an example of the technique applied to flow modelling in a straight pipe with a local stenosis (narrowing) in Figure 1.1(b).

1. **The geometry is discretised into a mesh** (mesh generation): this process involves identifying the fluid domain in the geometry, followed by dividing this continuous domain into a finite number of small regions using elements.

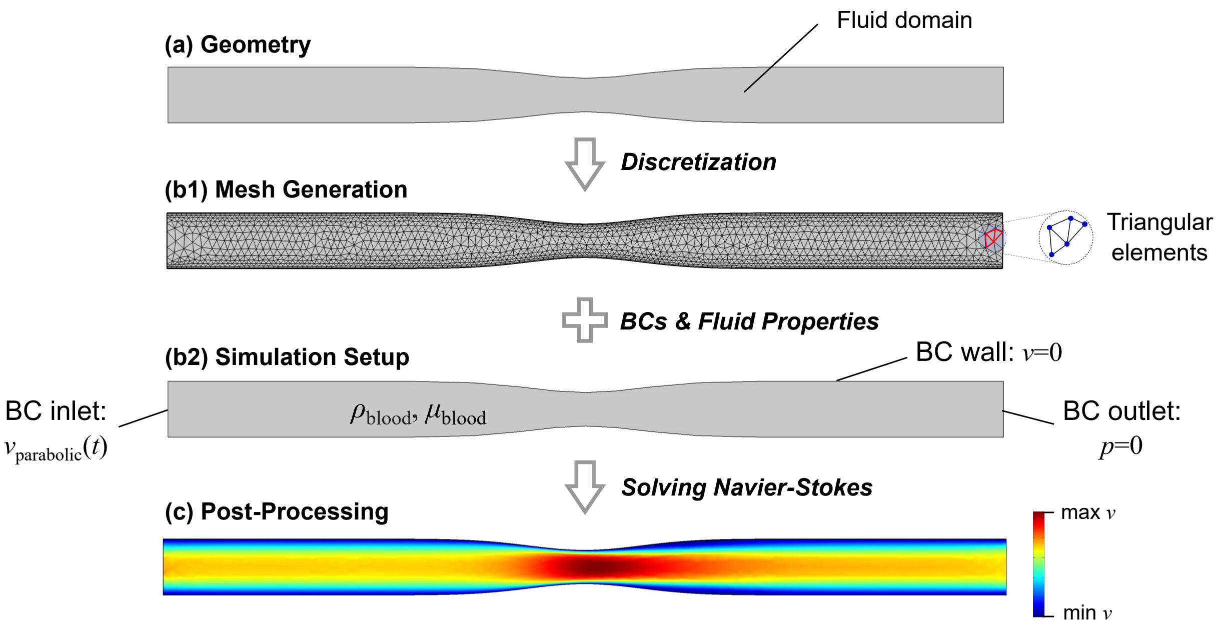
2. **Prescribing boundary conditions** on all boundaries of the fluid domain (*e.g.*, inlet, outlet, wall).

3. **Specifying fluid properties** including fluid density and viscosity.

4. **Solving** the governing equations using the numerical solver, and the resulting fluid flow characteristics (velocity and pressure) fields are obtained.



(a) Generic workflow for setting up and solving a CFD simulation, from geometry and mesh generation to solving continuum mechanics and obtaining fluid flow characteristics.



(b) Example of the CFD workflow for simulating blood flow in a 2-D pipe with a local stenosis.

Figure 1.1: Workflow of a computational fluid dynamics simulation.

Steps 1-3 listed above are commonly known as the CFD pre-processing, as these steps must be accomplished before the solution procedure and fed as the input to the CFD solver. Upon the completion of the solution procedure, the results can be post-processed – where users can generate visualisations (*e.g.*, plot vector arrows, colour maps), or quantifying desired haemodynamic metrics (*e.g.*, wall shear stress).

With the advancements in software engineering and ever-increasing computational power (as successfully predicted by Moore's Law), the CFD capability has been well encapsulated by many commercial software packages (*e.g.*, ANSYS Fluent, Siemens Star-CCM+, COMSOL Multiphysics). With the graphics user interface, the human efforts required in the rapid design, simulation, and optimisation have been significantly reduced. Furthermore, CFD software packages are equipped with the multiprocessing capability – this means that accelerating the solution procedure using CPU/GPU is possible, which makes the computation more efficient.

In this course, we will introduce the CFD capability in COMSOL Multiphysics to solve fluid mechanics problems.

1.3 Scope of the Notes

CFD should not be used as a black box! At this point, it is worth pointing out that the ease of use of modern CFD tools does not eliminate the need for a strong physical and mathematical understanding. A simulation is only as reliable as:

- the correctness of the geometry and mesh,
- the appropriateness of the chosen physical models and numerical schemes,
- the accuracy of boundary and initial conditions, and
- the rigour of post-processing and validation against experimental or theoretical results.

Failure to meet such requirements may lead to physically meaningless results (in such cases, CFD is referred to as "$\underline{\text{c}}$oloured $\underline{\text{f}}$unny $\underline{\text{d}}$iagram"), or divergence of the simulation.

Therefore, in the notes following, we will go on a rapid tour of three classical numerical methods: $\underline{\text{finite difference (FD)}}$, $\underline{\text{finite element (FE)}}$, and $\underline{\text{finite volume (FV)}}$, which are the mainstream methods used to solve fluid problems in the real world. The **aim** of these notes is

1. to introduce the fundamental numerical ideas behind CFD,
2. to build intuition about how these methods work, and,
3. to understand the pros and cons associated with these methods.

Closing Remarks The study of numerical methods is a huge topic in its own right, and it is the subject of much current research that is way beyond these notes, and beyond physiological fluid mechanics. There exist rich learning resources for the theoretical foundations of CFD, and many of the topics in these notes are covered in the following textbooks:

- F. Moukalled *et al.*, The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM$^{©}$ and Matlab.
- J. H. Ferziger *et al.*, Computational Methods for Fluid Dynamics.

The Department of Mechanical Engineering at Imperial College London offers two CFD-related modules to the MEng students at the Department of Bioengineering, these are:

- MECH60021/70028 – Computational Continuum Mechanics;
- MECH70015 – Computational Fluid Dynamics.

These courses build on our existing knowledge of fluid (and solid) mechanics, deepening the current understanding of the theoretical foundations of computational mechanics while maintaining a strong focus on practical applications.

2 Description of Common Numerical Methods

2.1 Problem Statement – Poisson Equation

To demonstrate the basic ideas of numerical methods, we consider a simplified 1-D, boundary value fluid mechanics problem where steady, viscous-dominated incompressible flow is defined on a finite domain Ω spanning the length L, *i.e.*, $\Omega = [0, L]$. With these assumptions, the Navier-Stokes equations can be reduced to 1-D model:

$$\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = \frac{1}{\mu}\frac{\mathrm{d}p}{\mathrm{d}x}, \quad x \in \Omega.$$

where $u(x)$ is the velocity field to be solved. The problem is subject to the no-slip boundary conditions at the walls:

$$u(0) = 0, \quad u(L) = 0.$$

For a constant pressure gradient, we define the R.H.S. $f = \frac{1}{\mu}\frac{\mathrm{d}p}{\mathrm{d}x}$, hence,

$$\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = f, \quad x \in \Omega, \tag{2.1}$$

which is the 1D Poisson equation for the velocity field.

2.2 Solution with Finite Difference Method

The finite difference method was first applied by Isaac Newton in 1687, and it is probably the simplest numerical method for people to perceive how to solve PDEs. In 1-D space, the domain is represented as a finite set of points. Derivatives of the functions are approximated by using a Taylor series expansion, truncated to a desired level of accuracy. We end up with a single equation at each point in the domain, and solving all of these simultaneously (a *system* of equations) gives the solution to the problem.

Recall the Taylor series expansion for a 1-D differentiable function $u(x)$ up to n^{th}-order in the vicinity of $x = x_i$:

$$\begin{aligned} u(x) &= \sum_{n=0}^{N} \frac{1}{n!}\frac{\mathrm{d}^n u}{\mathrm{d}x^n}(x - x_i)^n, \\ &= u_i + (x - x_i)\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_i + \frac{(x - x_i)^2}{2!}\left(\frac{\mathrm{d}^2 u}{\mathrm{d}x^2}\right)_i + \frac{(x - x_i)^3}{3!}\left(\frac{\mathrm{d}^3 u}{\mathrm{d}x^3}\right)_i + \ldots + \frac{(x - x_i)^N}{N!}\left(\frac{\mathrm{d}^N u}{\mathrm{d}x^N}\right)_i, \end{aligned} \tag{2.2}$$

where u_i denotes the value of u at $x = x_i$.

We can simply replace x by x_{i+1}, where x_{i+1} denotes the adjacent node next to x_i. Therefore, the first-order derivative evaluated at $x = x_i$ is

$$u_{i+1} = u_i + (x_{i+1} - x_i)\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_i + \frac{(x_{i+1} - x_i)^2}{2!}\left(\frac{\mathrm{d}^2 u}{\mathrm{d}x^2}\right)_i + \frac{(x_{i+1} - x_i)^3}{3!}\left(\frac{\mathrm{d}^3 u}{\mathrm{d}x^3}\right)_i + \ldots,$$

$$\Rightarrow \quad \left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_i = \frac{u_{i+1} - u_i}{x_{i+1} - x_i} - \frac{x_{i+1} - x_i}{2!}\left(\frac{\mathrm{d}^2 u}{\mathrm{d}x^2}\right)_i - \frac{(x_{i+1} - x_i)^2}{3!}\left(\frac{\mathrm{d}^3 u}{\mathrm{d}x^3}\right)_i + \ldots \tag{2.3}$$

With higher order terms being truncated, the only term left (blue-coded) is known as the **forward differencing** approximation of the first-order derivative.

Similarly, if we replace x by x_{i-1}, where x_{i-1} denotes the previous grid next to x_i. Therefore, the first-order derivative evaluated at $x = x_i$ is

$$\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_i = \frac{u_i - u_{i-1}}{x_i - x_{i-1}} - \frac{x_i - x_{i-1}}{2!}\left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right)_i - \frac{(x_i - x_{i-1})^2}{3!}\left(\frac{\mathrm{d}^3u}{\mathrm{d}x^3}\right)_i + \dots \tag{2.4}$$

With higher order terms being truncated, the only term left (blue-coded) is known as the **backward differencing** approximation of the first-order derivative.

If we add Equation 2.3 to Equation 2.4:

$$2\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_i = \frac{u_{i+1} - u_i}{x_{i+1} - x_i} - \frac{x_{i+1} - x_i}{2!}\left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right)_i + \dots + \frac{u_i - u_{i-1}}{x_i - x_{i-1}} - \frac{x_i - x_{i-1}}{2!}\left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right)_i + \dots$$

$$\Rightarrow \quad \left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_i = \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}} + \cancel{\frac{(x_{i+1} - x_i)^2 - (x_i - x_{i-1})^2}{2(x_{i+1} - x_{i-1})}\left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right)_i} + \cancel{\dots}$$

$$\Rightarrow \quad \left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_i = \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}} \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}. \tag{2.5}$$

Equation 2.5 is known as the **central differencing** approximation of the first-order derivative at $x = x_i$.

Comments

In the derivations above, we truncated the higher-order terms in the Taylor series expansion. These terms diminish rapidly because they are multiplied by higher powers of Δx.

For the forward and backward differencing schemes, the first neglected term involves the second derivative, giving them a **first-order accuracy**. In contrast, for the central differencing approximation, the even-order terms cancel out, and the leading neglected term involves the third derivative. Therefore, the central differencing scheme achieves a **second-order accuracy**.

The central differencing approximation of the second derivative at x_{i+1} and x_{i-1} is based on the evaluation of $\mathrm{d}u/\mathrm{d}x$ *halfway* between x_i and x_{i+1}, and x_i and x_{i-1}

$$\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_{i+\frac{1}{2}} \approx \frac{u_{i+1} - u_i}{x_{i+1} - x_i}, \qquad \left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_{i-\frac{1}{2}} \approx \frac{u_i - u_{i-1}}{x_i - x_{i-1}},$$

resulting in an expression of the second derivative as

$$\left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right)_i \approx \frac{\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_{i+\frac{1}{2}} - \left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)_{i-\frac{1}{2}}}{\frac{1}{2}(x_{i+1} - x_{i-1})} \approx \frac{u_{i+1}(x_i - x_{i-1}) + u_{i-1}(x_{i+1} - x_i) - u_i(x_{i+1} - x_{i-1})}{\frac{1}{2}(x_{i+1} - x_{i-1})(x_{i+1} - x_i)(x_i - x_{i-1})}.$$

For a constant grid space Δx, the expression above can be simplified to

$$\left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right)_i \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}. \tag{2.6}$$

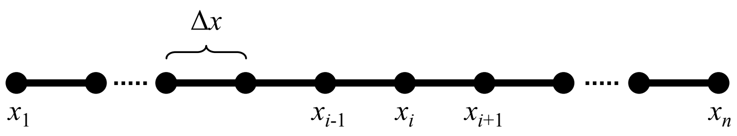


Figure 2.1: Discretisation of the 1D domain for the finite difference formulation.

This discretisation scheme is shown in Figure 2.1.

—

Therefore, the finite difference formulation of the 1D Poisson Equation becomes

$$\frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} = f_i, \quad i = 1, \ldots, N. \tag{2.7}$$

Note that the R.H.S. term f has also been discretised into elements, denoted by f_i. Arranging Equation 2.7 yields a linear system of equations

$$\underbrace{\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 & -2 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}}_{\mathbf{u}} = \Delta x^2 \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ \vdots \\ f_{N-3} \\ f_{N-2} \\ f_{N-1} \\ f_N \end{bmatrix}}_{\mathbf{b}}.$$

or, in compact matrix notation: $\mathbf{A}\mathbf{u} = \mathbf{b}$, where $\mathbf{A}$ is an $N \times N$ *tridiagonal* coefficient matrix, $\mathbf{b}$ is a $N \times 1$ vector corresponding to the R.H.S. constant terms, and $\mathbf{u}$ is a vector of unknowns, whose dimension is the same as that of $\mathbf{b}$.

Finally, to obtain the solution field $\mathbf{u}$, the solution can be formally written as:

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{b}. \tag{2.8}$$

In practice, the matrix $\mathbf{A}$ is not inverted explicitly; instead, direct or iterative solvers are used.

Comments

- The **advantage** of FDM is the simplicity. We have seen that the FD approximation of derivatives can be easily derived from the Taylor series; Further, by truncating the expansion terms, we can control the accuracy. The resultant system of equations can be easily coded and solved.
- The **disadvantage** of FDM is the difficulty of implementing for irregular geometries. Further, for the boundary layer problems, a very fine mesh is required at the boundary – hence, the space between grids may be unequal (*i.e.*, $\Delta x_{i-1} \neq \Delta x_i \neq \Delta x_{i+1}$), assembling $\mathbf{A}$ would be cumbersome.

2.3 Solution with Finite Element Method

The finite element method (FEM) is a general numerical method for PDEs; more than its capability for solving fluid problems, it has its wide applications in structural analysis, electromagnetism simulations, *etc*. To start with FEM, we shall first introduce a few new concepts.

Equation 2.1 is currently presented in the differential form over its domain Ω, which means that the equation must be satisfied pointwise for all $x \in \Omega$. In other words, the solution $u(x)$ must be continuous, so that the second derivative $\frac{d^2u}{dx^2}$ exists everywhere in Ω (twice differentiable). We say that the equation is written in its **strong form**.

The first step in FEM is to relax the requirement of the strong form. Instead of demanding that the solution u be continuously differentiable up to second order, if we multiply the governing equation by a test function, $v(x)$, and subsequently integrate w.r.t. x, yielding

$$\int_0^L \left(\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right) v \,\mathrm{d}x = \int_0^L f v \,\mathrm{d}x. \tag{2.9}$$

Comments

The test function v is chosen in a fashion that the boundary conditions are unconditionally satisfied: $v = 0$ at both $x = 0$ and $x = L$. We shall see why this is important very soon.

Applying integration by parts to the L.H.S. of 2.9, we can reduce the highest order of derivatives in the weak form:

$$\left[\frac{\mathrm{d}u}{\mathrm{d}x}v\right]_0^L - \int_0^L \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x = \int_0^L f v \,\mathrm{d}x \tag{2.10}$$

Recall the boundary conditions of the test function ($v = 0$ at $x = 0$ and $x = L$), hence, the first term vanishes, and we obtain

$$\int_0^L \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x = \int_0^L f v \,\mathrm{d}x. \tag{2.11}$$

Equation 2.11 is known as the **weak form** of the governing equation: "weak" means u and its first derivative are square-integrable. For the rest of the solution procedure, we shall work with the weak form of this governing equation.

We now introduce **shape functions**. In the 1D domain, each element is made of two nodes; we can index them with a and b. If we know the solution u on node a is U_a, and the solution on node b is U_b, what is the solution field between a and b? Here is where the shape functions come in – they are used to *interpolate* the solution field u between nodal values within each element:

$$u = U_a N_a(x) + U_b N_b(x) = \sum_{i\in\{a,b\}} N_i U_i, \tag{2.12}$$

Comments

Shape functions are defined on each node of a single element, *i.e.*, $N_a(x)$ on node a, $N_b(x)$ on node b. The following rules must be satisfied:

- Each shape function takes the value 1 at its own node and 0 at the other nodes: $N_a(x_a) = 1, N_a(x_b) = 0$ and $N_b(x_b) = 1, N_b(x_a) = 0$.
- Within each element, the shape functions sum to 1: $N_a(x) + N_b(x) = 1$.

Figure 2.2 illustrates the linear shape functions defined on the 1D element with two nodes. The linear shape functions can be derived with the analytical expression:

$$N_a(x) = \frac{-x + x_b}{x_b - x_a}, \qquad N_b(x) = \frac{x - x_a}{x_b - x_a}.$$

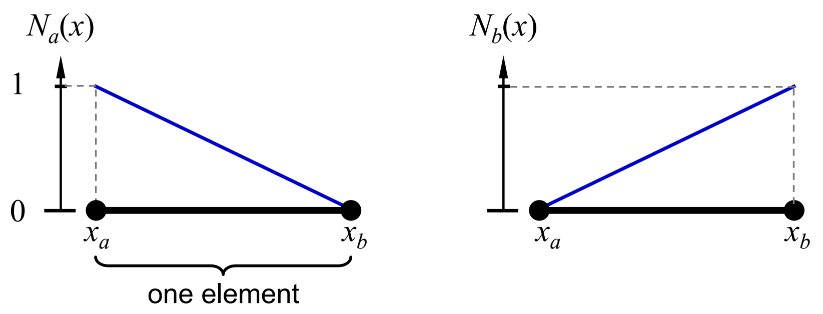


Figure 2.2: Linear shape functions defined for 1D bar elements.

Similarly, for the test function v:

$$v = V_a N_a(x) + V_b N_b(x) = \sum_{j \in \{a,b\}} N_j V_j. \tag{2.13}$$

Therefore, The L.H.S. of Equation 2.11 can be written as

$$\begin{aligned}\int_0^L \frac{\partial u}{\partial x}\frac{\partial v}{\partial x} &= \int_0^L \left(\sum_i \frac{\partial N_i}{\partial x} U_i\right)\left(\sum_j \frac{\partial N_j}{\partial x} V_j\right) \mathrm{d}x \\ &= \sum_i \sum_j U_i V_j \int_0^L \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} \mathrm{d}x \end{aligned} \tag{2.14}$$

and the R.H.S. of Equation 2.11 is

$$\int_0^L f(x)\left(\sum_i V_i N_i(x)\right) \mathrm{d}x = \sum_i V_i \int_0^L f(x) N_i(x)\, \mathrm{d}x. \tag{2.15}$$

Equating both sides:

$$\sum_i \sum_j U_i V_j \underbrace{\int_0^L \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} \mathrm{d}x}_{\mathbf{K}} = \sum_i V_i \underbrace{\int_0^L f(x) N_i(x)\, \mathrm{d}x}_{\mathbf{F}}. \tag{2.16}$$

where $\mathbf{K} = \int_0^L N_i'(x)\, N_j'(x)\, \mathrm{d}x$ is known as the **stiffness matrix**, and $\mathbf{F} = \int_0^L f(x)\, N_i(x)\, \mathrm{d}x$ is the **source**.

Since the coefficients V_i are arbitrary, for any value of V_i (test direction), the following relation must hold:

$$\sum_j K_{ij}\, U_j = F_i, \tag{2.17}$$

In compact form,

$$\mathbf{K}\,\mathbf{U} = \mathbf{F}. \tag{2.18}$$

Hence, the unknown solution vector can be easily solved by solving this linear system of equations.

Comments

- The **advantage** of FEM is its adaptability for solving equations in irregular geometries. Furthermore, FEM is a generic method with many applications in not only solving fluid or transport equations but also well-established for structural analysis, electromagnetics analysis *etc*.
- The **disadvantage** of FEM is that creating a suitable mesh requires more effort than FDM. Thus, in practice, the FE mesh generation is usually accomplished by professional pre-processing software packages (*e.g.*, gmsh, Altair Hypermesh). It is also rare to write a finite element code by hand.

2.4 Solution with Finite Volume Method

Although FEM is powerful for general PDEs and complex geometries, the finite volume method (FVM) is preferred in CFD because it guarantees strict local conservation of quantities such as mass, momentum, and energy. Unlike FEM, in FVM, the computational domain is divided into small regions called *control volumes* (CV). Each control volume is associated with a corresponding *control surface* (CS), which represents its closed boundary.

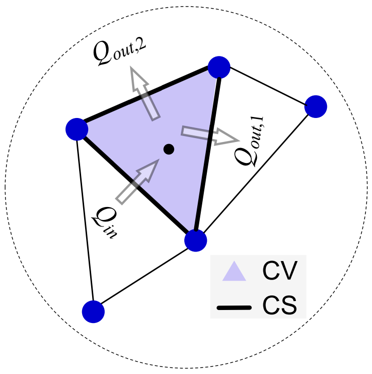


Figure 2.3: Applying the finite volume method to a simplified mesh network. The total flux within a control volume (CV) is balanced by the fluxes entering and leaving the CV through its control surfaces (CS): $Q_{\text{total}} = Q_{\text{in}} - Q_{\text{out},1} - Q_{\text{out},2}$.

The key idea of FVM is to enforce the governing transport equations in an *integral form* over each control volume. Any volume integrals that contain divergence operators are converted into surface integrals over the boundary of the control volume using the divergence theorem: the net flux of a quantity through the surface of a control volume equals the integral of the source terms inside it.

$$\iiint_{\mathrm{CV}} (\nabla \cdot \mathbf{u})\,\mathrm{d}V = \oiint_{\mathrm{CS}} (\mathbf{u} \cdot \mathbf{n})\,\mathrm{d}S, \tag{2.19}$$

where $\mathbf{n}$ is the outward unit normal on the surface.

Partition Ω into control volumes around each node. Integrate over a control volume CV:

$$\int_{\mathrm{CV}} \nabla^2 u\,\mathrm{d}V = \int_{\mathrm{CV}} f\,\mathrm{d}V. \tag{2.20}$$

Apply the divergence theorem to the L.H.S.

$$\int_{\mathrm{CV}} \nabla^2 u\,\mathrm{d}V = \int_{\mathrm{CS}} \nabla u \cdot \mathbf{n}\,\mathrm{d}S, \tag{2.21}$$

Therefore, Equation 2.20 can be expressed as

$$\int_{\mathrm{CS}} \nabla u \cdot \mathbf{n}\,\mathrm{d}S = \int_{\mathrm{CV}} f\,\mathrm{d}V. \tag{2.22}$$

Approximate fluxes across faces using gradients or neighbouring values. For the uniform 1-D mesh, Equation 2.22 reduces to a finite difference formulation:

$$\frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} = f_i. \tag{2.23}$$

Comments

- The **advantage** of FVM is that it seamlessly embeds the physical conservation laws into the numerical scheme; further, it also has the capability to adapt irregular geometries that FDM cannot handle.
- The **disadvantage** of FVM is that it is hard to scale up to achieve a higher-order numerical accuracy.

2.5 Other methods

There are many other numerical methods for solving differential equations. We shall now go through a quick tour of some of the selected methods.

Spectral Method Instead of approximating derivatives using local differences (FD) or shape functions (FE), the spectral method represents the solution as a linear combination of basis functions, *i.e.*,

$$u(x) = \sum_{k=0}^{N} \tilde{u}_k \phi_k(x),$$

where $\tilde{u}_k$ are coefficients multiplied to the basis functions $\phi_k(x)$. The basis functions can be chosen as smooth waves (*e.g.*, sine and cosine ⇒ Fourier series) or polynomials (*e.g.*, Chebyshev polynomials).

Taking derivatives of basis functions becomes fairly straightforward, since the basis functions are known analytically. For example, with the Fourier basis $\phi_k(x) = e^{ikx}$, $\mathrm{d}\phi/\mathrm{d}x = ike^{ikx}$, so the derivative of $u(x)$ is just multiplying coefficients $\tilde{u}_k$ by ik.

Reduced Order Modelling (ROM) ROM refers to a class of techniques that aim to reduce the computational cost of solving high-dimensional problems, such as CFD simulations, while retaining the essential dynamics of the system. The idea is to approximate the solution in a much smaller subspace spanned by a set of basis functions (modes), often obtained from data of high-fidelity simulations (snapshots) using proper orthogonal decomposition (mathematically analogous to the singular value decomposition).

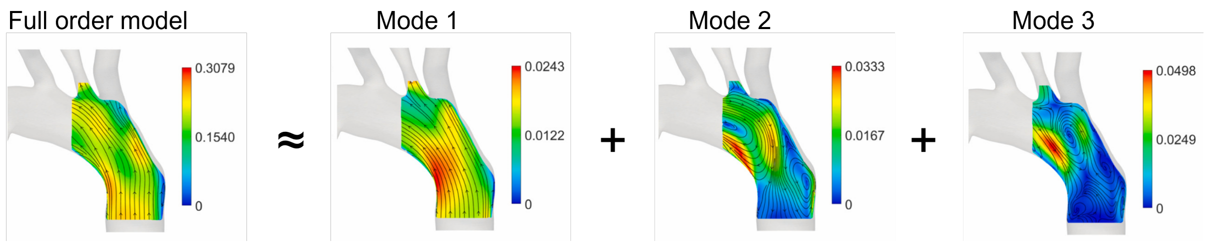


Figure 2.4: The mean velocity field of the full order model of an aorta (obtained from CFD simulation) can be approximated as the linear combination of the first three modes obtained from the proper orthogonal decomposition; the rest of the modes are truncated. (Reproduced from Chatpattanasiri *et al.*, 2023)

Physics-Informed Neural Networks (PINN) PINN is a class of mesh-free numerical methods that approximate the solution of governing equations using fully-connected neural networks. The key idea is to embed the physical laws directly into the training process by including the residual of the PDE, boundary conditions, and initial conditions in the loss function.

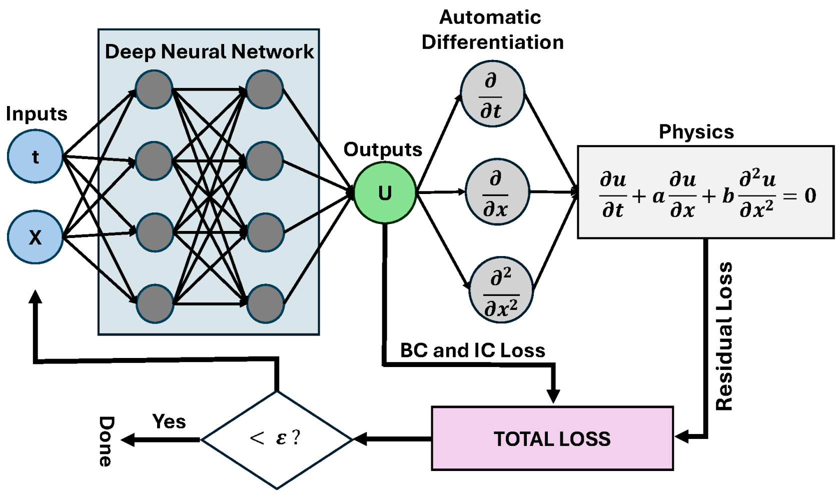

Automatic Differentiation
Deep Neural Network
Inputs
t
X
Outputs
U
$\frac{\partial}{\partial t}$
$\frac{\partial}{\partial x}$
$\frac{\partial^2}{\partial x^2}$
Physics
$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} + b\frac{\partial^2 u}{\partial x^2} = 0$
Residual Loss
BC and IC Loss
TOTAL LOSS
$< \varepsilon ?$
Yes
Done

Figure 2.5: A classic example PINN setup for the solution of an advection-diffusion equation. In this example, a and b are equation parameters, ε is a user-defined loss tolerance, x and t are the (input) independent variables (network features), and the neural network solution is given by u. (Trahan *et al.*, 2024)

3 Time Stepping for Unsteady Problems

Time discretisation is often involved in solving the unsteady partial differential equation. This step often follows space discretisation. To quickly develop the concepts, we consider a simple 1-D unsteady, viscous-dominant fluid problem:

$$\frac{\partial u}{\partial t} = \mu \frac{\partial^2 u}{\partial x^2}, \tag{3.1}$$

Here, we seek a space- and time-dependent solution $u(x, t)$, numerically.

First, we discretise the second-order partial derivative using the central differencing scheme given by Equation 2.6 (assume constant grid size Δx):

$$\frac{\partial u}{\partial t} = \mu \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2},$$

The time derivative is discretised by finding the slope of u between two successive time steps:

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{\text{new}} - u_i^{\text{old}}}{\Delta t}, \tag{3.2}$$

we use the superscript new to denote u_i at the current time step, old to denote u_i at the previous time step.

Updating Equation 3.1,

$$\frac{u_i^{\text{new}} - u_i^{\text{old}}}{\Delta t} = \mu \frac{u_{i+1}^{?} - 2u_i^{?} + u_{i-1}^{?}}{(\Delta x)^2}. \tag{A}$$

One question has emerged - for the R.H.S. of Equation A, should we take u_i, u_{i-1}, and u_{i+1} at the current time step (*i.e.* new), the previous time step (*i.e.* old), or a maybe a blending between the two (*e.g.*, half old and half new)? Answering this question reveals the heart of time discretisation:

Explicit treatment For the R.H.S. of Equation A, we can take u_i, u_{i-1}, and u_{i+1} solely from the previous time step, *i.e.*

$$\frac{u_i^{\text{new}} - u_i^{\text{old}}}{\Delta t} = \mu \frac{u_{i+1}^{\text{old}} - 2u_i^{\text{old}} + u_{i-1}^{\text{old}}}{(\Delta x)^2}. \tag{3.3}$$

This method is known as the explicit time treatment. The value of u at node i at the current time step is updated from the values of u_i, u_{i+1}, and u_{i-1} from the past step only, *i.e.*,

$$u_i^{\text{new}} = u_i^{\text{old}} + \frac{\mu \Delta t}{\Delta x^2}(u_{i+1}^{\text{old}} - 2u_i^{\text{old}} + u_{i-1}^{\text{old}}).$$

The explicit scheme is conditionally stable. This means, in order to obtain a physically meaningful solution of u, the time step Δt must be limited below a threshold[1]; The violation of this rule will lead to physically meaningless numerical oscillations in solution.

[1] In computational mechanics, the stability threshold is determined by the Courant-Friedrichs-Lewy (CFL) condition. For the diffusion-dominant problem, this condition takes the form $\mu\Delta t/\Delta x^2 \leq C$, where C is a method-dependent constant (*e.g.*, $C = 0.5$ for the explicit central-difference scheme). This condition requires the time step to be small enough such that diffusive information does not propagate faster than permitted by the spatial discretisation. Therefore, the time step chosen is bounded by the minimum mesh size.

Implicit treatment We can also take u_i, u_{i-1}, and u_{i+1} solely from the current time step for the R.H.S. of Equation A, *i.e.*

$$\frac{u_i^{\text{new}} - u_i^{\text{old}}}{\Delta t} = \mu \frac{u_{i+1}^{\text{new}} - 2u_i^{\text{new}} + u_{i-1}^{\text{new}}}{(\Delta x)^2}. \tag{3.4}$$

This method is known as the implicit time treatment. The value of u at node i at the current time step is updated from the values of u_{i+1}, and u_{i-1} at the current step only, and u_i from the previous step, *i.e.*,

$$u_i^{\text{new}} = \frac{u_i^{\text{old}} + \dfrac{\mu\Delta t}{\Delta x^2}(u_{i+1}^{\text{new}} + u_{i-1}^{\text{new}})}{1 + 2\dfrac{\mu\Delta t}{\Delta x^2}}.$$

The implicit scheme is unconditionally stable, *i.e.*, the numerical stability is irrelevant to Δt. However, properly choosing the value of Δt is still important to maintain a decent numerical resolution.

Semi-implicit treatment The semi-implicit time treatment (*a.k.a.* Crank-Nicolson method) equally "blends" the explicit and implicit treatment of the R.H.S. terms of Equation A. In other words, the explicit and implicit treatment contributes 50% to the solution u_i^{new}, respectively

$$\frac{u_i^{\text{new}} - u_i^{\text{old}}}{\Delta t} = \mu\left(\frac{1}{2}\underbrace{\frac{u_{i+1}^{\text{old}} - 2u_i^{\text{old}} + u_{i-1}^{\text{old}}}{(\Delta x)^2}}_{\text{explicit}} + \frac{1}{2}\underbrace{\frac{u_{i+1}^{\text{new}} - 2u_i^{\text{new}} + u_{i-1}^{\text{new}}}{(\Delta x)^2}}_{\text{implicit}}\right). \tag{3.5}$$

$$\Rightarrow \quad u_i^{\text{new}}\left(1 + \frac{\mu\Delta t}{\Delta x^2}\right) = \frac{\mu\Delta t}{2\Delta x^2}(u_{i+1}^{\text{old}} + u_{i+1}^{\text{new}} + u_{i-1}^{\text{old}} + u_{i-1}^{\text{new}}) + \left(1 - \frac{\mu\Delta t}{\Delta x^2}\right)u_i^{\text{old}}$$

$$\Rightarrow \quad u_i^{\text{new}} = \left[\frac{\mu\Delta t}{2\Delta x^2}(u_{i+1}^{\text{old}} + u_{i+1}^{\text{new}} + u_{i-1}^{\text{old}} + u_{i-1}^{\text{new}}) + \left(1 - \frac{\mu\Delta t}{\Delta x^2}\right)u_i^{\text{old}}\right] \bigg/ \left(1 + \frac{\mu\Delta t}{\Delta x^2}\right).$$

The semi-implicit time scheme is unconditionally stable.

Comments

Summary of three time discretisation schemes

- Explicit time treatment: u_i^{new} is determined solely from u_i^{old}, u_{i+1}^{old}, and u_{i-1}^{old};
- Implicit time treatment: u_i^{new} is determined solely from u_i^{old}, u_{i+1}^{new}, and u_{i-1}^{new};
- Semi-implicit time treatment: the explicit and implicit treatment contribute 50% to the solution u_i^{new}, respectively.

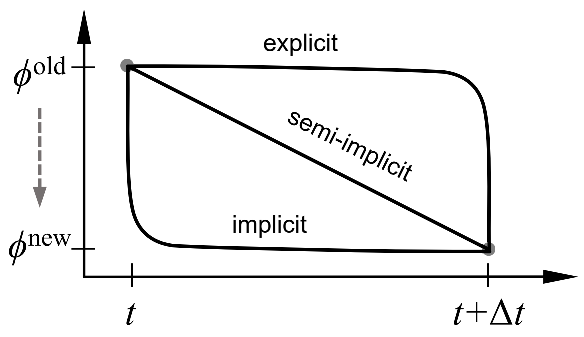


Figure 3.1: Illustration of the use of three time discretisation schemes to update u^{old} to u^{new} by the time increment.

Appendix A Sources of Error in Numerical Methods

As with all mathematical modelling, there are several potential sources of error in any solution we obtain, but the use of numerical methods brings additional complications. Errors that are common to all mathematical modelling:

- **Modelling assumptions:** For better or worse, there exist discrepancies between the mathematical models we create and the real situation. For example, we may model a fluid as Newtonian and incompressible, even though it is not (because we might not even know the true rheology); Or we may treat an artery as having a circular cross-section.
- **Parameter values:** Parameters (*e.g.* fluid viscosity, dimensions of an experiment) are usually determined experimentally, which can bring in errors. Unlike parameters in physics, for example, experimental estimates of biological parameters (*e.g.* rate of production of a chemical by a cell, or parameters for a rheological model of a viscoelastic physiological fluid) tend to have high standard deviations.

Errors specific to computational/numerical methods:

- **Machine error:** Calculations done by the computer introduce an inaccuracy, caused by the limited machine precision. In modern computers, most of the real numbers are only stored to a certain level of precision – for example, "single precision" means that only the first 32 bits of the binary representation of the number are stored. Because of this finite precision, numbers must be rounded to the nearest representable value. As a result, repeated arithmetic operations may accumulate round-off errors, and very large or very small numbers can exceed the representable range, leading to overflow or underflow.
- **Numerical/Discretisation error:** The numerical method computes an *approximation* to the solution, rather than the *exact* solution. Therefore, the difference between the exact solution and the numerical approximation to the solution brings in the **numerical error**.

  The magnitude and behaviour of the numerical error are determined by the employed numerical scheme. In particular, the **order of accuracy** of a scheme indicates how the error decreases as the grid spacing (or time step) is refined. For example, if you select "`Second Order Upwind`" for the spatial discretisation in a commercial CFD package, this means the leading truncation error term scales with $(\Delta x)^2$, (thus, we say the scheme is second-order accurate).
- **Iteration error:** When solving algebraic equation systems of the format $\mathbf{Ax} = \mathbf{b}$, rather than inverting the matrix $\mathbf{A}$ directly through factorization (*e.g.*, using Gaussian elimination), computers are prone to use iterative solvers (*e.g.*, conjugate gradient) to improve memory efficiency and speed (as $\mathbf{A}$ often has millions of DOFs). Iteration brings in errors – again, these methods are used to approximate $\mathbf{A}^{-1}$ as much as possible. If the solver is stopped before full convergence, the difference between the current approximation and the exact solution is called the **iteration error**.

We conclude that the main considerations when choosing timestep and grid size in the numerical method are a balance between the accuracy required, the time we are prepared to wait for the solution, the size of the computer's memory, and the convergence criteria we adopt to stop the solution.

Appendix B Consistency, Stability, and Convergence

B.1 Introduction

In Sections 2 and 3, we have methods for spatial and temporal discretisation, and how to construct a system of algebraic equations to be solved numerically in a computer. Although the derivations are fairly intuitive, what remains unanswered is why these methods function exactly in the way that we desire. Specifically, we seek answers for

1. Is the finite differencing construction of the PDE representing the exact PDE as the steps (*i.e.*, Δx and Δt) tend to 0?
2. Is my numerical scheme stable? Will my solution 'wiggle' (*e.g.*, unphysically oscillate) in time?
3. And eventually, how are we assured of a converged solution?

Answering these questions requires a detailed understanding of a few fundamental properties of numerical schemes, which we will discuss next.

B.2 Consistency

When deriving the finite difference approximation from the Taylor series expansions, we chose to neglect the higher order terms in the series – the difference between the *exact* solution and the numerical solution due to this truncation of higher order terms are known as the *truncation error*.

A numerical method is said to be *consistent* if the truncation errors become zero when the discretisation spacing tends to zero ($\Delta t \to 0$ and/or $\Delta x \to 0$). In other words, when truncation errors are removed, a consistent numerical method "recovers" the discretised system to the original differential equations.

B.3 Stability

A numerical scheme is said to be *stable* if any component of the initial solution is not amplified without bound, *i.e.*, the solution will not diverge.

We shall introduce the **von Neumann stability analysis** method (*a.k.a.* the Fourier method), which is the most widely used method to determine the stability of a linear numerical scheme with the periodic boundary conditions. This method is based on the fact that the general solution for such problems can be found as a linear combination of Fourier modes, and each mode corresponds to a frequency. If, for each mode, the frequency does not grow without bound, we say the numerical scheme is stable.

Example: von Neumann analysis of 1D heat equation using explicit time differencing

The discretised form of the 1-D heat equation using central differencing and explicit time differencing is given by

$$\frac{\partial u}{\partial t} = \mu \frac{\partial^2 u}{\partial x^2} \quad \Rightarrow \quad \frac{u_i^{t+1} - u_i^t}{\Delta t} = \mu \left(\frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{\Delta x^2} \right). \tag{B.1}$$

For von Neumann analysis, we first introduce the round-off error, ε, which quantifies the difference between the numerical solution, N, and the true (exact) solution, D, as

$$\varepsilon = N - D,$$

The numerical solution, N, that satisfies Equation B.1, can be re-written in terms of ε and D: $N = D + \varepsilon$, yielding

$$\frac{(D_i^{t+1} + \varepsilon_i^{t+1}) - (D_i^t + \varepsilon_i^t)}{\Delta t} = \mu \left[\frac{(D_{i+1}^t + \varepsilon_{i+1}^t) - 2(D_i^t + \varepsilon_i^t) + (D_{i-1}^t + \varepsilon_{i-1}^t)}{\Delta x^2} \right]$$

$$\Rightarrow \quad \frac{D_i^{t+1} - D_i^t}{\Delta t} - \frac{\varepsilon_i^{t+1} - \varepsilon_i^t}{\Delta t} = \mu \left(\frac{D_{i+1}^t - 2D_i^t + D_{i-1}^t}{\Delta x^2} \right) - \mu \left(\frac{\varepsilon_{i+1}^t - 2\varepsilon_i^t + \varepsilon_{i-1}^t}{\Delta x^2} \right)$$

We know that the exact solution, D, must satisfy the governing equation Equation B.1. This enforces the round-off error ε, must also satisfy Equation B.1,

$$\frac{\varepsilon_i^{t+1} - \varepsilon_i^t}{\Delta t} = \mu \left(\frac{\varepsilon_{i+1}^t - 2\varepsilon_i^t + \varepsilon_{i-1}^t}{\Delta x^2} \right).$$

We shall now decompose the error into the linear combination of frequency modes. The trick is realising the error, ε, is time and space varying, *i.e.*, $\varepsilon(x, t)$. We can express ε as a linear combination of different modes, m,

$$\varepsilon(x, t) = \sum_{m=0}^{M} e^{at} e^{jk_m x}, \qquad \text{for } m = 0, 1, 2, \ldots, M$$

where $k_m = \dfrac{m\pi}{L}$, M is the number of intervals spanning the total length of the domain, L, and $j = \sqrt{-1}$ is the imaginary unit (we use j to differentiate from the index i).

Substitute $\varepsilon(x, t)$ into Equation B.1, as

$$e^{a(t+\Delta t)} e^{jk_m x} - e^{at} e^{jk_m x} = \frac{\mu \Delta t}{\Delta x^2} \left(e^{at} e^{jk_m (x+\Delta x)} - 2e^{at} e^{jk_m x} + e^{at} e^{jk_m (x-\Delta x)} \right).$$

Divide both sides by $e^{at} e^{jk_m x}$, yielding

$$e^{a\Delta t} - 1 = \frac{\mu \Delta t}{\Delta x^2} \left(e^{jk_m \Delta x} + e^{-jk_m \Delta x} - 2 \right),$$

where $e^{a\Delta t}$ quantifies how much the error increases/decreases between two successive time steps, *i.e.* $\varepsilon_i^{t+1} = e^{a\Delta t} \varepsilon_i^t$. If the stability is reached, the error will not grow after the step increment, *i.e.*,

$$|e^{a\Delta t}| = |\varepsilon_i^{t+1} / \varepsilon_i^t| \le 1,$$

$e^{a\Delta t}$ is the amplification factor.

Realizing the trigonometry identity: $\sin^2 \dfrac{\theta}{2} = \dfrac{1 - \cos\theta}{2}$, we thus write the amplification factor:

$$e^{a\Delta t} = 1 - 4\frac{\mu \Delta t}{\Delta x^2} \sin^2 \left(\frac{k_m \Delta x}{2} \right)$$

and in order to reach the numerical stability, $|e^{a\Delta t}| \le 1$:

$$\left| 1 - 4\frac{\mu \Delta t}{\Delta x^2} \sin^2 \left(\frac{k_m \Delta x}{2} \right) \right| \le 1 \quad \Rightarrow \quad \frac{\mu \Delta t}{\Delta x^2} \le \frac{1}{2}.$$

which is the condition that must be satisfied.

B.4 Convergence

A numerical method is said to be *convergent* if the solution from the discretised system of equations tends to the *exact* solution of the governing equation. For linear initial value problems, the Lax equivalence theorem states that “if the discretisation scheme is consistent, then the stability is a necessary and sufficient condition for convergence.”

The three properties of a numerical scheme are summarised in Figure B.1: Consistency ensures that the discretisation recovers the governing equation as $\Delta x \to 0$ and $\Delta t \to 0$; stability ensures bounded solutions of the algebraic system; and convergence guarantees that the numerical solution approaches the exact solution as the mesh is refined.

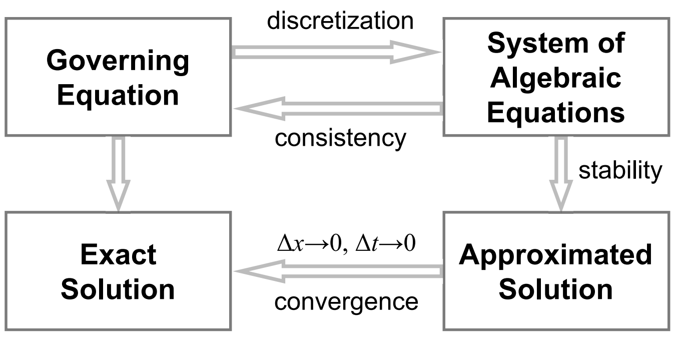


Figure B.1: Relationship between governing equations, their discretised algebraic forms, and the resulting solutions.