Computational Mathematics 2

Semester 1, 2024

Dr Elliot Carr

Tarang Janawalkar

Contents

1 Transport Problems

1.1 Transport Phenomena

Transport phenomena broadly comprises three disciplines; fluid dynamics, heat transfer, and mass transfer. **Fluid dynamics** is the study of the motion of fluids, including liquids and gases. **Heat transfer** is the study of how heat (thermal energy) is transported, generated, dissipated, and/or converted in a physical system. **Mass transfer** is the study of the movement of mass from one location to another. The mathematical equations used to describe the above phenomena involve three fundamental mechanisms of transport:

1. **Diffusion**: The gradual movement of a substance from regions of high concentration to regions of low concentration. The direction of diffusion is determined by the sign of the negative gradient of the concentration.

2. **Advection**: The transport of a substance by bulk motion of a fluid. Advection is driven by a vector field in which the substance is transported.

3. **Reaction**: The process in which substances are created or destroyed. Reaction is represented as a source or sink function.

1.2 The Transport Equation

The general form of the transport equation is given by

$$\underbrace{\frac{\partial u}{\partial t}}_{\text{unsteady term}} + \underbrace{\boldsymbol{\nabla}\cdot(\mathbf{v}u)}_{\text{advection term}} = \underbrace{\boldsymbol{\nabla}\cdot(\mathbf{D}\boldsymbol{\nabla}u)}_{\text{diffusion term}} + \underbrace{R}_{\text{reaction term}}$$

where $u(\mathbf{x},\ t)$ is the quantity being transported at position $\mathbf{x}$ and time t, $\mathbf{v}(\mathbf{x},\ t) \in \mathbb{R}^n$ is a velocity vector field which drives u, $\mathbf{D} \in \mathbb{R}^{n\times n}$ is the diffusion matrix, and $R(\mathbf{x},\ t)$ is a reaction term. n represents the dimension of the spatial domain of the problem, which can be 1, 2, or 3.
An alternative form of the transport equation combines the divergence terms

$$\underbrace{\frac{\partial u}{\partial t}}_{\text{unsteady term}} + \underbrace{\boldsymbol{\nabla}\cdot\mathbf{q}}_{\text{flux term}} = \underbrace{R}_{\text{reaction term}}$$

where $\mathbf{q} = \mathbf{v}u - \mathbf{D}\boldsymbol{\nabla}u$ is the *flux vector.* This PDE is defined on an open connected subset $\Omega \subset \mathbb{R}^n$ with the boundary $\partial\Omega$.

1.2.1 Derivation

The transport equation can be derived from the conservation of mass principle: the rate of change of the quantity u within a region D must be balanced by the net flow of u in/out of the boundary ∂D of D, and the rate of creation or destruction of u within D. Consider an arbitrarily small sub-domain D of Ω with boundary ∂D, then:

$$\begin{aligned}
\left\{\begin{array}{c}\text{Rate of change}\\ \text{of } u \text{ in } D\end{array}\right\} &= -\left\{\begin{array}{c}u \text{ leaving } D\\ \text{across } \partial D\end{array}\right\} + \left\{\begin{array}{c}\text{Generation/Destruction}\\ \text{of } u \text{ within } D\end{array}\right\}\\
\frac{\mathrm{d}}{\mathrm{d}t}\int_D u\,\mathrm{d}V &= -\int_{\partial D} \mathbf{q}\cdot\mathbf{n}\,\mathrm{d}s + \int_D R\,\mathrm{d}V\\
\int_D \frac{\partial u}{\partial t}\,\mathrm{d}V &= -\int_D \boldsymbol{\nabla}\cdot\mathbf{q}\,\mathrm{d}V + \int_D R\,\mathrm{d}V\\
\int_D\left(\frac{\partial u}{\partial t} + \boldsymbol{\nabla}\cdot\mathbf{q} - R\right)\mathrm{d}V &= 0\\
\frac{\partial u}{\partial t} + \boldsymbol{\nabla}\cdot\mathbf{q} &= R.
\end{aligned}$$

1.3 Special Cases

1.3.1 One Spatial Dimension

In one spatial dimension, the transport equation reduces to

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(vu) = \frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial x}\right) + R.$$

where $u(x,\,t)$ is a function of one spatial dimension and time, v is the velocity, and $D > 0$ is the diffusivity.

1.3.2 Two Spatial Dimensions

In two spatial dimensions, the transport equation reduces to

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(v_x u) + \frac{\partial}{\partial y}(v_y u) = \frac{\partial}{\partial x}\left(D_{xx}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(D_{yy}\frac{\partial u}{\partial y}\right) + R.$$

where $u(x,\,y,\,t)$ is a function of two spatial dimensions and time, v_x and v_y are the velocities in the x and y directions, and D_{xx} and D_{yy} are the diffusivities in the x and y directions.

1.3.3 Eliminating Terms

The transport equation is also called the *advection-diffusion-reaction equation.*

- If the *velocity term* $\mathbf{v}$ is the zero vector, the equation reduces to the *diffusion-reaction equation.*
- If the *diffusion term* $\mathbf{D}$ is the zero matrix, the equation reduces to the *advection-reaction equation.*
- If the *reaction term* R is zero, the equation reduces to the *advection-diffusion equation.*

1.4 Classification

While the terms in the transport equation may be constant or variable, certain combinations of these terms lead to different solution methods.

- The velocity vector $\mathbf{v}$ may be a constant vector or a function of space $\mathbf{x}$, time t, and/or the solution u.
- The diffusion matrix $\mathbf{D}$ may be a constant matrix or a function of space $\mathbf{x}$, time t, and/or the solution u.
- The reaction term R may be a constant or a function of space $\mathbf{x}$, time t, and/or the solution u.

When $\mathbf{v}$ and $\mathbf{D}$ are <u>not</u> functions of u, and R is a <u>linear</u> function of u, the transport equation is called *linear*. The equation is *nonlinear* otherwise. The domain Ω is called *heterogeneous* if any of the coefficients $\mathbf{v}$, $\mathbf{D}$, or R are functions of space $\mathbf{x}$, and *homogeneous* otherwise.

1.5 Dimensional Analysis

Performing a dimensional analysis on the transport equation allows us to associate physical units with the coefficients of the equation. This analysis is useful for verifying the correctness of the equation and for scaling the equation to a dimensionless form. The terms in the equation

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{v} u) = \boldsymbol{\nabla} \cdot (\mathbf{D} \boldsymbol{\nabla} u) + R$$

may only be added or subtracted if they have the same units. Therefore, given that

$$\left[\frac{\partial u}{\partial t}\right] \equiv \frac{[u]}{[t]} = \frac{[u]}{\mathsf{T}}$$

we can deduce the units of other terms in the equation.

$$\begin{aligned}
[\boldsymbol{\nabla} \cdot (\mathbf{v} u)] &\equiv \frac{[\mathbf{v}]\,[u]}{[x]} = \frac{[u]}{\mathsf{T}} \implies [\mathbf{v}] = \frac{\mathsf{L}}{\mathsf{T}} \\
[\boldsymbol{\nabla} \cdot (\mathbf{D} \boldsymbol{\nabla} u)] &\equiv \frac{[\mathbf{D}]\,[\boldsymbol{\nabla} u]}{[x]} = \frac{[\mathbf{D}]\,[u]}{[x]^2} = \frac{[u]}{\mathsf{T}} \implies [\mathbf{D}] = \frac{\mathsf{L}^2}{\mathsf{T}} \\
[R] &\equiv \frac{[u]}{[t]} \implies [R] = \frac{[u]}{\mathsf{T}}
\end{aligned}$$

1.6 Initial and Boundary Conditions

In addition to the transport equation, which describes the behaviour of u within the domain Ω, the problem must also specify how u behaves at the boundary $\partial\Omega$ with *boundary conditions.* Some common boundary conditions include:

- Specified value: $u\,(\mathbf{x},\, t) = u_b$ on $\partial\Omega$
- Specified flux: $\mathbf{q} \cdot \mathbf{n} = q_b$ on $\partial\Omega$

- Specified gradient: $\boldsymbol{\nabla} u \cdot \mathbf{n} = d_b$ on $\partial\Omega$

Here u_b, q_b, and d_b may be constants or scalar functions of $\mathbf{x}$ and/or t, and $\mathbf{n}$ is the unit normal vector to $\partial\Omega$, directed outward from Ω. We may also wish to use a Robin condition to describe a general boundary condition of the form:

$$au + b\left(\boldsymbol{\nabla} u \cdot \mathbf{n}\right) = c$$

where a, b, and c are constants or scalar functions of $\mathbf{x}$ and/or t. When $c = 0$, the condition is called *homogeneous*, and *nonhomogeneous* otherwise. In addition to these conditions, an *initial condition* is required to specify the profile of u at time $t = 0$.

1.7 Steady-State Problems

If it exists, the *steady-state solution* of the transport equation is the solution of the equation when the time-derivative of u is zero:

$$\frac{\partial u}{\partial t} = 0.$$

The steady-state solution is useful for understanding the long-term behaviour of the system, where it is assumed that the system is no longer time-dependent. The steady-state solution is expressed as $u_\infty = \lim_{t \to \infty} u\left(\mathbf{x},\, t\right)$.

Part I
Finite Volume Method

The *Finite Volume Method* (FVM) is a method for solving the transport equation at discrete points (nodes) in space: $u_i\left(t\right) \approx u\left(\mathbf{x}_i,\, t\right)$ for $i = 1,\, 2,\, \dots,\, N$. FVM is a *spatial discretisation* method that converts a spatially-continuous initial-boundary value problem to a satially-discrete initial-value problem. The FVM is used in transport phenomena because the approximate solution obeys the laws of conservation of mass and energy. This is generally not true for finite differences or finite element methods. FVM also has the advantage of being able to handle complex geometries and irregular grids.

2 Spatial Discretisation

The basic geometric structure used in the FVM is called the **mesh**. A mesh is a partitioning of the domain Ω into smaller sub-domains called **elements**. The intersection of edges in this mesh are called **vertices**. An example of a 1D mesh is shown below.

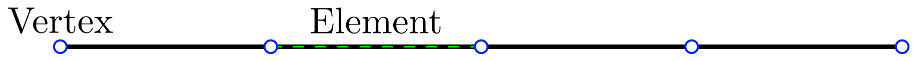
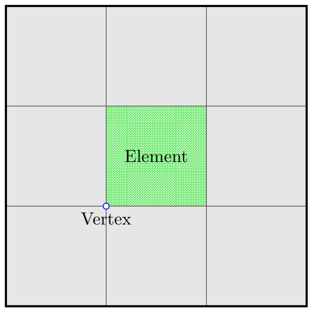
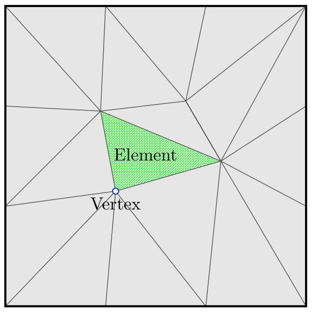


Figure 1: 1D mesh.

In two and three dimensions, the mesh may be either *structured* or *unstructured* as shown below. Unstructured meshes typically consist of triangles (or tetrahedra in 3D).



(a) 2D structured mesh.



(b) 2D unstructured mesh.

The FVM defines:

- **Nodes** x_i, at which the solution is approximated by u_i
- **Control volumes** Ω_i, over which the conservation principle is applied

where $i = 1, 2, \dots, N$ is the index of the nodes. There are several ways to define the control volumes. Two common approaches are:

- **Cell-Centred Control Volumes**: Nodes are positioned at the centroids of elements, and control volumes are defined over elements.
- **Vertex-Centred Control Volumes**: Nodes are positioned at vertices, and control volumes are constructed using the centroids of adjacent element boundaries.

This is illustrated in 1D and 2D in the following figures.

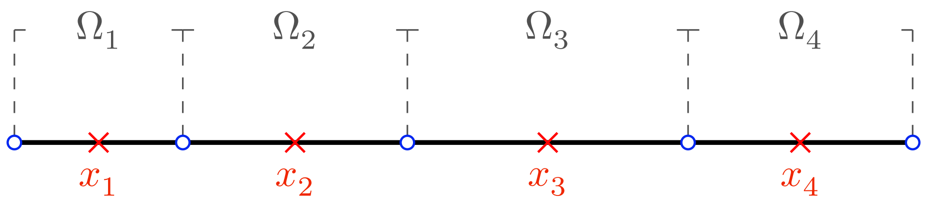


Figure 3: 1D mesh with cell-centred control volumes.

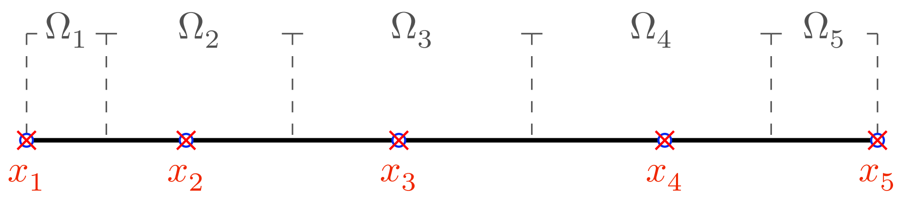
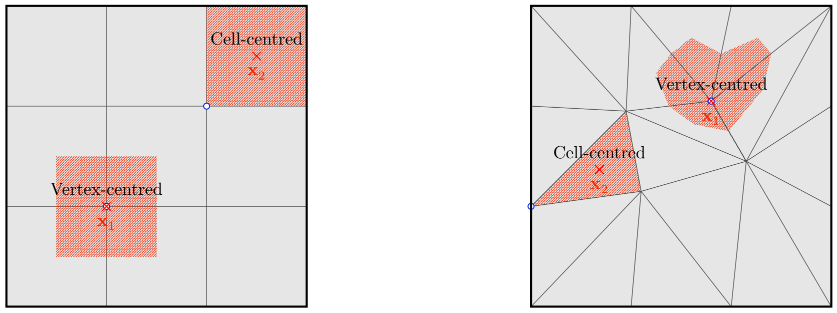


Figure 4: 1D mesh with vertex-centred control volumes.



(a) 2D structured mesh with vertex-/cell-centred control volumes. (b) 2D unstructured mesh with vertex-/cell-centred control volumes.

For problems in three dimensions using structured meshes, control volumes are cubes (or rectangular prisms). For unstructured meshes with control volumes using the cell-centred approach, elements themselves are used as control volumes. For the vertex-centred approach, control volume boundaries are defined using the centroids of adjacent element boundaries.

2.1 General Strategy

Consider a mesh with N nodes and N control volumes:

1. Label the nodes x_1, x_2, ..., x_N and the unknowns u_1, u_2, ..., u_N.
2. Identify the control volume types (interior nodes when $2 \leqslant i \leqslant N-1$, boundary nodes when $i = 1$ or $i = N$).
3. Integrate the transport equation over each control volume and apply the Divergence theorem to the flux term.
4. Incorporate the boundary conditions to approximate/discretise all remaining terms.

To discretise the transport equation, let us integrate the transport equation over each control volume Ω_i:

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{q} = R$$

$$\int_{\Omega_i} \frac{\partial u}{\partial t} \, \mathrm{d}V + \int_{\Omega_i} (\boldsymbol{\nabla} \cdot \mathbf{q}) \, \mathrm{d}V = \int_{\Omega_i} R \, \mathrm{d}V$$

Applying the divergence theorem to the flux term, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_i} u\,\mathrm{d}V + \int_{\partial\Omega_i} (\mathbf{q}\cdot\mathbf{n})\,\mathrm{d}s = \int_{\Omega_i} R\,\mathrm{d}V$$

where $\partial\Omega_i$ is the control volume boundary of Ω_i, and $\mathbf{n}$ is the outward unit vector normal to $\partial\Omega_i$, directed out of Ω_i. Consider the spatial average of u and R over Ω_i:

$$\bar{u}_i = \frac{1}{V_i}\int_{\Omega_i} u\,\mathrm{d}V, \qquad \bar{R}_i = \frac{1}{V_i}\int_{\Omega_i} R\,\mathrm{d}V$$

where V_i is the volume of Ω_i. Using this quantity, we can rewrite the transport equation as

$$\frac{\mathrm{d}\bar{u}_i}{\mathrm{d}t} + \frac{1}{V_i}\int_{\partial\Omega_i} (\mathbf{q}\cdot\mathbf{n})\,\mathrm{d}s = \bar{R}_i.$$

3 The Time-Dependent Transport Equation

Using the discrete approximations u_i and R_i, at each node x_i, we arrive at the discrete form of the transport equation:

$$\boxed{\frac{\mathrm{d}u_i}{\mathrm{d}t} = -\frac{1}{V_i}\int_{\partial\Omega_i} (\mathbf{q}\cdot\mathbf{n})\,\mathrm{d}s + R_i.} \tag{1}$$

Constructing this equation for all i gives a system of N ODEs for the unknowns u_i, in time:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{A}\mathbf{u} + \mathbf{b}(t), \qquad \mathbf{u}(0) = \mathbf{u}^{(0)}$$

where $\mathbf{u} = (u_1, \dots, u_N)^\top$ is the numerical solution vector at time t containing the solution at each node x_i, and $\mathbf{u}^{(0)} = (f(x_1), \dots, f(x_N))^\top$ is the initial solution vector. $\mathbf{A} \in \mathbb{R}^{N\times N}$ is a tridiagonal matrix which consists of coefficients obtained via the finite-difference method, and $\mathbf{b}(t)$ contains any time-dependent terms in the transport equation. Note that when the transport equation is nonlinear, it is expressed as

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{G}(t, \mathbf{u}(t)).$$

4 Time Discretisation

This time-dependent ODE can be solved using the θ-methods. Consider a sufficiently large final time T, and discretise time domain into M time steps of size $\delta t = T/M = t_{n+1} - t_n$, such that $t_n = n\,\delta t$ for $n = 0, 1, \dots, M$. Then let us split the nonhomogeneous part of this ODE into the time-independent and time-dependent parts:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{A}\mathbf{u} + \mathbf{b}_1 + \mathbf{b}_2(t), \qquad \mathbf{u}(0) = \mathbf{u}^{(0)}.$$

We can then compute $\mathbf{u}^{(n)} = \left(u_1^{(n)}, \dots, u_N^{(n)}\right)^\top$, where $u_i^{(n)} \approx u_i(t_n) = u(x_i, t_n)$, is the solution at $x = x_i$ and time $t = t_n$, using the θ-method.

Let us integrate this ODE over the time interval (t_n, t_{n+1}):

$$\begin{aligned}
\int_{t_n}^{t_{n+1}} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}\,\mathrm{d}t &= \int_{t_n}^{t_{n+1}} \left(\mathbf{A}\mathbf{u} + \mathbf{b}_1 + \mathbf{b}_2(t)\right)\mathrm{d}t \\
\mathbf{u}(t_{n+1}) - \mathbf{u}(t_n) &= \mathbf{A}\int_{t_n}^{t_{n+1}} \mathbf{u}\,\mathrm{d}t + \int_{t_n}^{t_{n+1}} \mathbf{b}_1\,\mathrm{d}t + \int_{t_n}^{t_{n+1}} \mathbf{b}_2(t)\,\mathrm{d}t \\
\mathbf{u}^{(n+1)} - \mathbf{u}^{(n)} &= \mathbf{A}\int_{t_n}^{t_{n+1}} \mathbf{u}\,\mathrm{d}t + \mathbf{b}_1(t_{n+1} - t_n) + \int_{t_n}^{t_{n+1}} \mathbf{b}_2(t)\,\mathrm{d}t.
\end{aligned}$$

To integrate the time-dependent terms, we will use a *weighted θ approximation*, where

$$\int_{t_n}^{t_{n+1}} f(t)\,\mathrm{d}t \approx \delta t\left[(1-\theta) f(t_n) + \theta f(t_{n+1})\right].$$

Then,

$$\begin{aligned}
\mathbf{u}^{(n+1)} - \mathbf{u}^{(n)} &= \delta t\mathbf{A}\left[(1-\theta_1)\mathbf{u}^{(n)} + \theta_1\mathbf{u}^{(n+1)}\right] + \mathbf{b}_1\,\delta t + \delta t\left[(1-\theta_2)\mathbf{b}_2^{(n)} + \theta_2\mathbf{b}_2^{(n+1)}\right] \\
\mathbf{u}^{(n+1)} - \mathbf{A}\theta_1\,\delta t\mathbf{u}^{(n+1)} &= \mathbf{u}^{(n)} + \delta t(1-\theta_1)\mathbf{A}\mathbf{u}^{(n)} + \delta t\mathbf{b}_1 + \delta t\left[(1-\theta_2)\mathbf{b}_2^{(n)} + \theta_2\mathbf{b}_2^{(n+1)}\right]
\end{aligned}$$

giving us the time-stepping formula

$$\boxed{(\mathbf{I} - \delta t\theta_1\mathbf{A})\mathbf{u}^{(n+1)} = \left[\mathbf{I} + \delta t(1-\theta_1)\mathbf{A}\right]\mathbf{u}^{(n)} + \delta t\left[\mathbf{b}_1 + (1-\theta_2)\mathbf{b}_2^{(n)} + \theta_2\mathbf{b}_2^{(n+1)}\right]}$$

for two choices of θ_1 and θ_2. For convenience, we will write this as

$$\tilde{\mathbf{A}}\mathbf{u}^{(n+1)} = \tilde{\mathbf{B}}\mathbf{u}^{(n)} + \tilde{\mathbf{b}}.$$

Setting $\theta_1 = \theta_2 = 0$, we obtain the Forward Euler method:

$$\mathbf{u}^{(n+1)} = (\mathbf{I} + \delta t\mathbf{A})\mathbf{u}^{(n)} + \delta t\left(\mathbf{b}_1 + \mathbf{b}_2^{(n)}\right)$$

which is an explicit method. Setting $\theta_1 = \theta_2 = 1$, yields the Backward Euler method:

$$(\mathbf{I} - \delta t\mathbf{A})\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \delta t\left(\mathbf{b}_1 + \mathbf{b}_2^{(n+1)}\right)$$

while setting $\theta_1 = \theta_2 = \frac{1}{2}$ gives the Crank-Nicolson method:

$$\left(\mathbf{I} - \frac{\delta t}{2}\mathbf{A}\right)\mathbf{u}^{(n+1)} = \left(\mathbf{I} + \frac{\delta t}{2}\mathbf{A}\right)\mathbf{u}^{(n)} + \frac{\delta t}{2}\left(2\mathbf{b}_1 + \mathbf{b}_2^{(n)} + \mathbf{b}_2^{(n+1)}\right)$$

which are both implicit methods. Using different values for θ_1 and θ_2 is also possible, and these methods are known as the IMEX (Implicit-Explicit) methods.

4.1 Dirichlet Boundary Conditions

When a problem defines Dirichlet boundary conditions rather than Neumann boundary conditions, we must replace the first row of the matrix $\tilde{\mathbf{A}}$ with $(1, 0, \ldots, 0) \in \mathbb{R}^{1\times N}$ and the first element of the RHS vector $\tilde{\mathbf{b}}$ with the Dirichlet boundary condition to ensure that the boundary condition is satisfied.

5 Stability Analysis

Consider only the interior rows and columns of the result

$$\tilde{\mathbf{A}}\mathbf{u}^{(n+1)} = \tilde{\mathbf{B}}\mathbf{u}^{(n)} + \tilde{\mathbf{b}}$$

where $\mathbf{u} \in \mathbb{R}^{N-2}$, $\tilde{\mathbf{A}} \in \mathbb{R}^{N-2\times N-2}$, $\tilde{\mathbf{B}} \in \mathbb{R}^{N-2\times N-2}$, and $\tilde{\mathbf{b}} \in \mathbb{R}^{N-2}$. Then let us assume that $\tilde{\mathbf{A}}$ is invertible, so that we can construct the recurrence relation

$$\mathbf{u}^{(n+1)} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{B}}\mathbf{u}^{(n)} + \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}.$$

If we consider the homogeneous part of this equation, and let $\mathbf{T} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{B}}$, we can analyse the stability of our time discretisation method using spectral analysis:

$$\mathbf{u}^{(n+1)} = \mathbf{T}\mathbf{u}^{(n)}.$$

For a uniform mesh, the matrix $\mathbf{A}$ is tridiagonal with constant diagonals (or *Toeplitz tridiagonal*):

$$\mathbf{A} = \begin{bmatrix} \alpha & \beta & & & \\ \gamma & \alpha & \beta & & \\ & \gamma & \alpha & \ddots & \\ & & \ddots & \ddots & \beta \\ & & & \gamma & \alpha \end{bmatrix}.$$

Such a matrix has eigenvalues

$$\lambda_i = \alpha + 2\sqrt{\beta}\sqrt{\gamma}\cos\left(\frac{i\pi}{N+1}\right), \qquad i = 1,\ 2,\ \dots,\ N-2$$

Consider an error term $\boldsymbol{\varepsilon}^{(0)}$ present in the initial solution vector $\mathbf{u}^{(0)}$ at time $t = 0$. Then, by the recurrence relation established above,

$$\begin{aligned} \mathbf{u}^{(1)} &= \mathbf{T}\mathbf{u}^{(0)} = \mathbf{T}\left(\mathbf{u}^{(0)} + \boldsymbol{\varepsilon}^{(0)}\right) = \mathbf{T}\mathbf{u}^{(0)} + \mathbf{T}\boldsymbol{\varepsilon}^{(0)} \\ \mathbf{u}^{(2)} &= \mathbf{T}\mathbf{u}^{(1)} = \mathbf{T}\left(\mathbf{T}\mathbf{u}^{(0)} + \mathbf{T}\boldsymbol{\varepsilon}^{(0)}\right) = \mathbf{T}^2\mathbf{u}^{(0)} + \mathbf{T}^2\boldsymbol{\varepsilon}^{(0)} \\ &\vdots \\ \mathbf{u}^{(n)} &= \mathbf{T}\mathbf{u}^{(n-1)} = \mathbf{T}\left(\mathbf{T}\mathbf{u}^{(n-2)} + \mathbf{T}\boldsymbol{\varepsilon}^{(n-2)}\right) = \mathbf{T}^n\mathbf{u}^{(0)} + \mathbf{T}^n\boldsymbol{\varepsilon}^{(0)}. \end{aligned}$$

Thus, we must ensure that the error term $\mathbf{T}^n\boldsymbol{\varepsilon}^{(0)}$ does not grow unbounded as $n \to M$. This is only possible when the spectral radius of the matrix $\mathbf{T}$ is smaller than 1, i.e., $\rho(\mathbf{T}) < 1$.
Given that the matrix $\mathbf{A}$ is Toeplitz tridiagonal with eigenpairs $(\lambda_i,\ \mathbf{v}_i)$, consider the eigenvalue problem for the matrix $\tilde{\mathbf{A}}$:

$$\begin{aligned} \mathbf{A}\mathbf{v}_i &= \lambda_i\mathbf{v}_i \\ (\delta t\theta_1\mathbf{A})\,\mathbf{v}_i &= (\delta t\theta_1\lambda_i)\,\mathbf{v}_i \\ \mathbf{v}_i - (\delta t\theta_1\mathbf{A})\,\mathbf{v}_i &= \mathbf{v}_i - (\delta t\theta_1\lambda_i)\,\mathbf{v}_i \\ (\mathbf{I} - \delta t\theta_1\mathbf{A})\,\mathbf{v}_i &= (1 - \delta t\theta_1\lambda_i)\,\mathbf{v}_i \\ \tilde{\mathbf{A}}\mathbf{v}_i &= \psi_i\mathbf{v}_i. \end{aligned}$$

Similarly for the matrix $\tilde{\mathbf{B}}$:

$$\begin{aligned}
\mathbf{A}\mathbf{v}_i &= \lambda_i \mathbf{v}_i \\
\left(\delta t\left(1-\theta_1\right)\mathbf{A}\right)\mathbf{v}_i &= \left(\delta t\left(1-\theta_1\right)\lambda_i\right)\mathbf{v}_i \\
\mathbf{v}_i + \left(\delta t\left(1-\theta_1\right)\mathbf{A}\right)\mathbf{v}_i &= \mathbf{v}_i + \left(\delta t\left(1-\theta_1\right)\lambda_i\right)\mathbf{v}_i \\
\left(\mathbf{I} + \delta t\left(1-\theta_1\right)\mathbf{A}\right)\mathbf{v}_i &= \left(1 + \delta t\left(1-\theta_1\right)\lambda_i\right)\mathbf{v}_i \\
\tilde{\mathbf{B}}\mathbf{v}_i &= \phi_i \mathbf{v}_i .
\end{aligned}$$

Thus, the eigenvalues of the matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are $\left(1-\delta t\theta_1\lambda_i\right)$ and $\left(1+\delta t\left(1-\theta_1\right)\lambda_i\right)$, respectively, so that the eigenvalues of the matrix $\mathbf{T}$ are given by the ratio of the eigenvalues of the matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$:

$$\begin{aligned}
\tilde{\mathbf{A}}^{-1}\tilde{\mathbf{B}}\mathbf{v}_i &= \tilde{\mathbf{A}}^{-1}\left(\phi_i \mathbf{v}_i\right) \\
\mathbf{T}\mathbf{v}_i &= \frac{1}{\psi_i}\phi_i \mathbf{v}_i \\
\mathbf{T}\mathbf{v}_i &= \frac{1+\delta t\left(1-\theta_1\right)\lambda_i}{1-\delta t\theta_1\lambda_i}\mathbf{v}_i .
\end{aligned}$$

We can now find a lower bound for the spectral radius through some algebra:

$$\begin{aligned}
\rho\left(\mathbf{T}\right) &= \max\left|\frac{1+\delta t\left(1-\theta_1\right)\lambda_i}{1-\delta t\theta_1\lambda_i}\right| \\
&= \max\left|\frac{1+\delta t\lambda_i-\delta t\theta_1\lambda_i}{1-\delta t\theta_1\lambda_i}\right| \\
&= \max\left|1+\frac{\delta t\lambda_i}{1-\delta t\theta_1\lambda_i}\right| . \\
&= \max\left|1+\frac{\delta t\left(\alpha+2\sqrt{\beta\gamma}\cos\left(\frac{i\pi}{N+1}\right)\right)}{1-\delta t\theta_1\left(\alpha+2\sqrt{\beta\gamma}\cos\left(\frac{i\pi}{N+1}\right)\right)}\right| \\
&= \max\left|1+\frac{\delta t\left(\alpha+2\sqrt{\beta\gamma}-4\sqrt{\beta\gamma}\sin^2\left(\frac{i\pi}{2(N+1)}\right)\right)}{1-\delta t\theta_1\left(\alpha+2\sqrt{\beta\gamma}-4\sqrt{\beta\gamma}\sin^2\left(\frac{i\pi}{2(N+1)}\right)\right)}\right| \\
&\geqslant \left|1+\frac{\delta t\left(\alpha+2\sqrt{\beta\gamma}\right)}{1-\delta t\theta_1\left(\alpha+2\sqrt{\beta\gamma}\right)}\right|
\end{aligned}$$

Therefore,

$$\left|1+\frac{\delta t\left(\alpha+2\sqrt{\beta\gamma}\right)}{1-\delta t\theta_1\left(\alpha+2\sqrt{\beta\gamma}\right)}\right| \leqslant \rho\left(\mathbf{T}\right) < 1 \implies -1 < 1+\frac{\delta t\left(\alpha+2\sqrt{\beta\gamma}\right)}{1-\delta t\theta_1\left(\alpha+2\sqrt{\beta\gamma}\right)} < 1.$$

We need only to consider the upper bound for δt, as the lower bound is trivially satisfied for all

$\delta t > 0$. Thus, we have

$$\begin{aligned}
-1 &< 1 + \frac{\delta t\left(\alpha + 2\sqrt{\beta\gamma}\right)}{1 - \delta t\theta_1\left(\alpha + 2\sqrt{\beta\gamma}\right)} \\
-2 + 2\,\delta t\theta_1\left(\alpha + 2\sqrt{\beta\gamma}\right) &< \delta t\left(\alpha + 2\sqrt{\beta\gamma}\right) \\
\delta t\left(\alpha + 2\sqrt{\beta\gamma}\right)(2\theta_1 - 1) &< 2 \\
\delta t &> \frac{2}{\left(\alpha + 2\sqrt{\beta\gamma}\right)(2\theta_1 - 1)}.
\end{aligned}$$

6 Monotonicity

For a numerical solution to remain physically meaningful, it must be monotonic. That is, in the absence of a reaction/source term, an increase in the solution at any neighbouring node must not cause a decrease in the solution at the current node. Failure to observe this may result in oscillations in the numerical solution. Consider the diagonal and off-diagonal parts of the LHS of the time-discretisation:

$$\begin{aligned}
\mathbf{I} - \delta t\theta_1\mathbf{A} &= \begin{bmatrix} 1-\delta t\theta_1\alpha & -\delta t\theta_1\beta & & \\ -\delta t\theta_1\gamma & 1-\delta t\theta_1\alpha & \ddots & \\ & \ddots & \ddots & -\delta t\theta_1\beta \\ & & -\delta t\theta_1\gamma & 1-\delta t\theta_1\alpha \end{bmatrix} \\
&= \begin{bmatrix} 1-\delta t\theta_1\alpha & & & \\ & 1-\delta t\theta_1\alpha & & \\ & & \ddots & \\ & & & 1-\delta t\theta_1\alpha \end{bmatrix} + \begin{bmatrix} 0 & -\delta t\theta_1\beta & & \\ -\delta t\theta_1\gamma & 0 & \ddots & \\ & \ddots & \ddots & -\delta t\theta_1\beta \\ & & -\delta t\theta_1\gamma & 0 \end{bmatrix} \\
&= (1-\delta t\theta_1\alpha)\,\mathbf{I} + \begin{bmatrix} 0 & -\delta t\theta_1\beta & & \\ -\delta t\theta_1\gamma & 0 & \ddots & \\ & \ddots & \ddots & -\delta t\theta_1\beta \\ & & -\delta t\theta_1\gamma & 0 \end{bmatrix}.
\end{aligned}$$

The off-diagonal part of the matrix can be expressed as the difference of the original matrix and a diagonal matrix:

$$\begin{aligned}
\begin{bmatrix} 0 & -\delta t\theta_1\beta & & \\ -\delta t\theta_1\gamma & 0 & \ddots & \\ & \ddots & \ddots & -\delta t\theta_1\beta \\ & & -\delta t\theta_1\gamma & 0 \end{bmatrix} &= \mathbf{I} - \delta t\theta_1\mathbf{A} - (1-\delta t\theta_1\alpha)\,\mathbf{I} \\
&= \mathbf{I} - \delta t\theta_1\mathbf{A} - \mathbf{I} + \delta t\theta_1\alpha\mathbf{I} \\
&= -\,\delta t\theta_1\left(\mathbf{A} - \alpha\mathbf{I}\right).
\end{aligned}$$

Therefore,

$$\begin{aligned}
\left(\mathbf{I} - \delta t\theta_1\mathbf{A}\right)\mathbf{u}^{(n+1)} &= \left[\mathbf{I} + \delta t\left(1-\theta_1\right)\mathbf{A}\right]\mathbf{u}^{(n)} + \tilde{\mathbf{b}} \\
\left[(1-\delta t\theta_1\alpha)\,\mathbf{I} - \delta t\theta_1\left(\mathbf{A} - \alpha\mathbf{I}\right)\right]\mathbf{u}^{(n+1)} &= \left[\mathbf{I} + \delta t\left(1-\theta_1\right)\mathbf{A}\right]\mathbf{u}^{(n)} + \tilde{\mathbf{b}} \\
(1-\delta t\theta_1\alpha)\,\mathbf{u}^{(n+1)} &= \delta t\theta_1\left(\mathbf{A} - \alpha\mathbf{I}\right)\mathbf{u}^{(n+1)} + \left[\mathbf{I} + \delta t\left(1-\theta_1\right)\mathbf{A}\right]\mathbf{u}^{(n)} + \tilde{\mathbf{b}}.
\end{aligned}$$

Let us consider the individual elements of the two matrix terms on the RHS:

$$\delta t\theta_1 (\mathbf{A} - \alpha\mathbf{I}) = \delta t\theta_1 \begin{bmatrix} 0 & \beta & & \\ \gamma & 0 & \ddots & \\ & \ddots & \ddots & \beta \\ & & \gamma & 0 \end{bmatrix},$$

$$\mathbf{I} + \delta t (1-\theta_1) \mathbf{A} = \mathbf{I} + \delta t (1-\theta_1) \begin{bmatrix} \alpha & \beta & & \\ \gamma & \alpha & \ddots & \\ & \ddots & \ddots & \beta \\ & & \gamma & \alpha \end{bmatrix}.$$

For the solution of this equation to be monotonic, all coefficients of $\mathbf{u}$ must be non-negative:

$$\begin{gathered} 1 - \delta t\theta_1 \alpha \geqslant 0, \\ \delta t\theta_1 \beta \geqslant 0, \qquad \delta t\theta_1 \gamma \geqslant 0, \\ 1 + \delta t (1-\theta_1) \alpha \geqslant 0, \quad 1 + \delta t (1-\theta_1) \beta \geqslant 0, \quad 1 + \delta t (1-\theta_1) \gamma \geqslant 0. \end{gathered}$$

From the second row, both β and γ must be non-negative. Therefore, the time step δt must satisfy the following conditions:

$$\delta t \geqslant \frac{1}{\theta_1 \alpha}, \quad \delta t \geqslant \frac{1}{(1-\theta_1)\alpha}, \quad \delta t \geqslant \frac{1}{(1-\theta_1)\beta}, \quad \delta t \geqslant \frac{1}{(1-\theta_1)\gamma}.$$

7 Finite Volume Method in 1D

Let us define the following geometric quantities (for the vertex-centred control volume strategy):

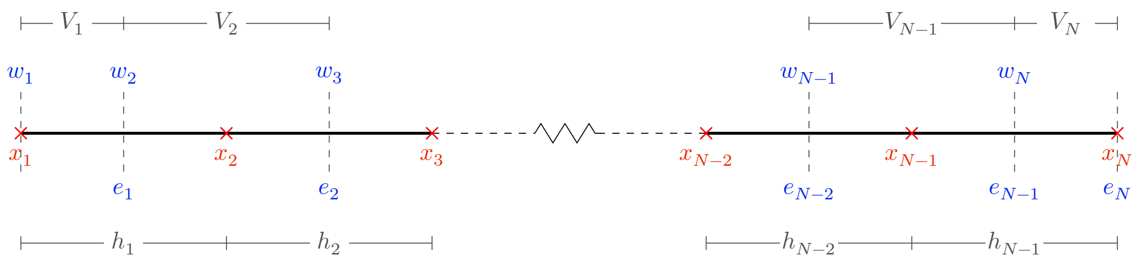


Figure 6: 1D FVM.

where

- $\Omega = (x_1, x_N)$
- $\Omega_i = (w_i, e_i)$ is the control volume for node i
- h_i is the spacing between nodes i and $i+1$

$$h_i = x_{i+1} - x_i \quad \text{for } i = 1, \ldots, N-1$$

- V_i is the volume of control volume Ω_i

$$V_i = \begin{cases} \dfrac{h_1}{2}, & i = 1 \\ \dfrac{h_{i-1} + h_i}{2}, & 2 \leqslant i \leqslant N-1 \\ \dfrac{h_{N-1}}{2}, & i = N \end{cases}$$

- w_i and e_i are the west and east boundaries of control volume Ω_i

$$w_i = \begin{cases} x_1, & i = 1 \\ \dfrac{x_{i-1} + x_i}{2}, & 2 \leqslant i \leqslant N \end{cases}$$

$$e_i = \begin{cases} \dfrac{x_i + x_{i+1}}{2}, & 1 \leqslant i \leqslant N-1 \\ x_N, & i = N \end{cases}$$

Then, if we integrate over all control volumes Ω_i in 1D, the spatial averages simplify to:

$$\bar{u}_i = \frac{1}{V_i} \int_{w_i}^{e_i} u \, \mathrm{d}x, \qquad \bar{R}_i = \frac{1}{V_i} \int_{w_i}^{e_i} R \, \mathrm{d}x$$

likewise, the flux term becomes,

$$\begin{aligned} \int_{\Omega_i} (\boldsymbol{\nabla} \cdot \mathbf{q}) \, \mathrm{d}V &= \int_{w_i}^{e_i} \frac{\partial q}{\partial x} \, \mathrm{d}x \\ &= q_{e_i} - q_{w_i} \end{aligned}$$

so that

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{1}{V_i} \left(q_{w_i} - q_{e_i} \right) + R_i.$$

7.1 Finite Differences

7.1.1 Internal Nodes

We can apply the finite-difference method at internal nodes, using:

$$u(w_i, t) = (1 - \sigma) u_{i-1} + \sigma u_i \qquad \frac{\partial u}{\partial x}(w_i, t) = \frac{u_i - u_{i-1}}{h_{i-1}} \qquad \text{(west node)}$$

$$u(e_i, t) = (1 - \sigma) u_i + \sigma u_{i+1} \qquad \frac{\partial u}{\partial x}(e_i, t) = \frac{u_{i+1} - u_i}{h_i} \qquad \text{(east node)}$$

for weights $0 \leqslant \sigma \leqslant 1$. The choice $\sigma = 1/2$ is known as *averaging*, as it produces

$$u(w_i, t) = \frac{u_{i-1} + u_i}{2}, \qquad u(e_i, t) = \frac{u_i + u_{i+1}}{2}.$$

When $v > 0$, then the flow is from left to right, and we choose $\sigma = 0$, so that

$$u(w_i,\, t) \approx u_{i-1}, \qquad u(e_i,\, t) \approx u_i.$$

When $v < 0$, then the flow is from right to left, and we choose $\sigma = 1$, so that

$$u(w_i,\, t) \approx u_i, \qquad u(e_i,\, t) \approx u_{i+1}.$$

This is known as *upwinding*. Numerical solutions obtained using upwinding exhibit *numerical diffusion* or *false diffusion* as we can rewrite the diffusivity in the upwinding method using the diffusivity from the averaging method: $D_{\text{upwinding}} = D_{\text{avg}} + c$, where c is some constant made up of the model parameters.

7.1.2 Boundary Nodes

At boundary nodes, we can use boundary conditions to approximate $\frac{\partial u}{\partial x}$ at the boundaries:

$$\begin{aligned}
&\frac{\partial u}{\partial x}(w_1,\, t) = \frac{\partial u}{\partial x}(0,\, t) && \frac{\partial u}{\partial x}(e_1,\, t) = \frac{u_2 - u_1}{h_1} && \text{(left boundary)} \\
&\frac{\partial u}{\partial x}(w_N,\, t) = \frac{u_N - u_{N-1}}{h_{N-1}} && \frac{\partial u}{\partial x}(e_N,\, t) = \frac{\partial u}{\partial x}(L,\, t) && \text{(right boundary)}
\end{aligned}$$

7.1.3 Nonlinear Diffusivity

If the diffusivity is a nonlinear function of u, then we can use the following averaging approximations:

$$\begin{aligned}
D(u(x_i,\, t)) &= D(u_i) && \text{(current node)} \\
D(u(w_i,\, t)) &= \frac{D(u_{i-1}) + D(u_i)}{2} && \text{(west node)} \\
D(u(e_i,\, t)) &= \frac{D(u_i) + D(u_{i+1})}{2} && \text{(east node)}
\end{aligned}$$

7.2 Time Discretisation

We can compute $u_i^{(n)} = u_i(t_n) = u(x_i,\, t_n)$ by integrating the above ODE over the time interval (t_n, t_{n+1}):

$$\begin{aligned}
\int_{t_n}^{t_{n+1}} \frac{\mathrm{d}u_i}{\mathrm{d}t}\,\mathrm{d}t &= \int_{t_n}^{t_{n+1}} \left(\frac{1}{V_i}\left(q_{w_i} - q_{e_i}\right) + R_i\right)\mathrm{d}t \\
u_i(t_{n+1}) - u_i(t_n) &= \frac{1}{V_i}\int_{t_n}^{t_{n+1}} q_{w_i} - q_{e_i}\,\mathrm{d}t + \int_{t_n}^{t_{n+1}} R_i\,\mathrm{d}t \\
u_i^{(n+1)} - u_i^{(n)} &= \frac{\delta t}{V_i}\left[(1-\theta_1)\left(q_{w_i}^{(n)} - q_{e_i}^{(n)}\right) + \theta_1\left(q_{w_i}^{(n+1)} - q_{e_i}^{(n+1)}\right)\right] \\
&\quad + \delta t\left[(1-\theta_2)R_i^{(n)} + \theta_2 R_i^{(n+1)}\right].
\end{aligned}$$

Part II
Newton Methods

Consider the nonlinear form of the time-dependent ODE, which arises when D or R are nonlinear functions of u:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \mathbf{G}\left(u\right).$$

If we were to use the θ-method, we find that the solution requires iterating over the solution $M-2$ times:

$$\mathbf{u}^{(n+1)} - \mathbf{u}^{(n)} = \delta t \left[\theta_1 \mathbf{G}\left(\mathbf{u}^{(n+1)}\right) + \left(1-\theta_1\right)\mathbf{G}\left(\mathbf{u}^{(n)}\right)\right].$$

Let us consider an iterative approach where we approximate the solution $\mathbf{u}^{(n+1)}$ at every time step n by finding the root of the vector function:

$$\begin{aligned} \mathbf{x} - \mathbf{u}^{(n)} - \delta t \left[\theta_1 \mathbf{G}\left(\mathbf{x}\right) + \left(1-\theta_1\right)\mathbf{G}\left(\mathbf{u}^{(n)}\right)\right] &= \mathbf{0} \\ \mathbf{F}\left(\mathbf{x}\right) &= \mathbf{0}. \end{aligned}$$

The initial guess for this algorithm will be the solution at the previous time step:

$$\mathbf{x}^{(0)} = \mathbf{u}^{(n)}$$

so that when Newton's method converges, the root after k iterations approximates the solution at the next time step:

$$\mathbf{u}^{(n+1)} = \mathbf{x}^{(k)}.$$

This is repeated until an approximate solution is obtained at each time step.

8 Newton's Method

Newton's method is an iterative method for finding the roots of a nonlinear function.

8.1 Scalar Function

Given a scalar function $f\left(x\right)$, we can approximate the root of this function by linearising the function about a guess $x^{(k)}$:

$$\begin{aligned} f\left(x\right) &\approx f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\left(x - x^{(k)}\right) \\ 0 &\approx f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\left(x - x^{(k)}\right) \\ x^{(k+1)} &= x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}. \end{aligned}$$

8.2 Vector Function

Given a system of nonlinear equations, expressed by the vector function $\mathbf{F}(\mathbf{x})$, we can use the same linearisation technique to find the root of this function:

$$\begin{aligned}
\begin{bmatrix} F_1(\mathbf{x}) \\ F_2(\mathbf{x}) \\ \vdots \\ F_n(\mathbf{x}) \end{bmatrix} &\approx \begin{bmatrix} F_1\left(\mathbf{x}^{(k)}\right) \\ F_2\left(\mathbf{x}^{(k)}\right) \\ \vdots \\ F_n\left(\mathbf{x}^{(k)}\right) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\nabla} F_1\left(\mathbf{x}^{(k)}\right)^\top \\ \boldsymbol{\nabla} F_2\left(\mathbf{x}^{(k)}\right)^\top \\ \vdots \\ \boldsymbol{\nabla} F_n\left(\mathbf{x}^{(k)}\right)^\top \end{bmatrix} \left(\mathbf{x} - \mathbf{x}^{(k)}\right) \\
\mathbf{F}(\mathbf{x}) &\approx \mathbf{F}\left(\mathbf{x}^{(k)}\right) + \mathbf{J}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x} - \mathbf{x}^{(k)}\right) \\
\mathbf{0} &\approx \mathbf{F}\left(\mathbf{x}^{(k)}\right) + \mathbf{J}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x} - \mathbf{x}^{(k)}\right) \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \mathbf{J}\left(\mathbf{x}^{(k)}\right)^{-1}\mathbf{F}\left(\mathbf{x}^{(k)}\right).
\end{aligned}$$

To avoid the matrix inverse, we can instead use the update rule

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$$

where $\boldsymbol{\delta}\mathbf{x}^{(k)}$ is the solution to the linear system

$$\mathbf{J}\left(\mathbf{x}^{(k)}\right)\left(\boldsymbol{\delta}\mathbf{x}^{(k)}\right) = -\mathbf{F}\left(\mathbf{x}^{(k)}\right).$$

The vector $\boldsymbol{\delta}\mathbf{x}^{(k)}$ is known as the **Newton correction**. We can define the stopping criterion for the Newton method as

$$\left\|\mathbf{F}\left(\mathbf{x}^{(k+1)}\right)\right\| \leqslant \text{rtol}\left\|\mathbf{F}\left(\mathbf{x}^{(0)}\right)\right\| + \text{atol}.$$

9 Modified Newton Methods

9.1 Chord Method

To avoid the computation of the Jacobian at every iteration, we can compute the Jacobian at only the first iteration:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}\left(\mathbf{x}^{(0)}\right)^{-1}\mathbf{F}\left(\mathbf{x}^{(k)}\right).$$

Using the alternate form shown in the previous section, we can write this as:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(0)}$$

where $\boldsymbol{\delta}\mathbf{x}^{(0)}$ is the solution to the linear system

$$\mathbf{J}\left(\mathbf{x}^{(0)}\right)\left(\boldsymbol{\delta}\mathbf{x}^{(0)}\right) = -\mathbf{F}\left(\mathbf{x}^{(k)}\right).$$

9.2 Shamanskii Method

Alternatively, the Jacobian matrix may be *periodically* updated every m iterations. The Shamanskii method is the most general form of Newton's method, where the original form of Newton's method is recovered when $m = 1$, and the Chord method is recovered when $m \to \infty$.

10 Convergence Theory

The sequence $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ converges with limit $\mathbf{x}^*$ if:

$$\lim_{k\to\infty}\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\|=0.$$

The following definitions and theorems will be used throughout this section to discuss the convergence of the Newton methods discussed previously.

10.1 Convergent Sequences

Definition 10.1 (Convergent sequence)**.** A sequence $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ of vectors in $\mathbb{R}^n$ is convergent with limit $\mathbf{x}^*$ if for every $\epsilon>0$, there exists an integer M such that

$$\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\|<\epsilon,$$

for all $k>M$.

Definition 10.2 (Cauchy sequence)**.** A sequence $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ of vectors in $\mathbb{R}^n$ is a **Cauchy sequence** if for every $\epsilon>0$, there exists an integer M such that

$$\left\|\mathbf{x}^{(k)}-\mathbf{x}^{(m)}\right\|<\epsilon,$$

for all $k,m>M$.

A sequence of vectors in $\mathbb{R}^n$ is Cauchy if and only if it converges.

Definition 10.3 (Rate of Convergence)**.** Let $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ be a convergent sequence of vectors in $\mathbb{R}^n$ with limit $\mathbf{x}^*$. The sequence converges

- **quadratically** if there exists a $K>0$ such that

$$\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|\leqslant K\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\|^2$$

- **superlinearly** with order $\alpha\in(1,2)$ if there exists a $K>0$ such that

$$\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|\leqslant K\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\|^\alpha$$

- **linearly** with factor $\sigma\in(0,1)$ if

$$\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|\leqslant \alpha\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\|$$

for a sufficiently large k.

Theorem 10.1. *Let $\mathbf{F}:D\to\mathbb{R}^n$ be differentiable on an open subset $D\subset\mathbb{R}^n$ and let $\mathbf{x}\in D$. Then, for any nonzero vector $\mathbf{h}$, such that $\mathbf{x}+\mathbf{h}\in D$, we have*

$$\mathbf{F}\left(\mathbf{x}+\mathbf{h}\right)=\mathbf{F}\left(\mathbf{x}\right)+\int_0^1\mathbf{J}\left(\mathbf{x}+t\mathbf{h}\right)\mathbf{h}\,\mathrm{d}t$$

where $\mathbf{J}$ is the Jacobian matrix of $\mathbf{F}$.

Proof. Consider the parameterisation $\mathbf{F}(\mathbf{x} + t\mathbf{h})$, then,

$$\frac{\partial \mathbf{F}(\mathbf{x} + t\mathbf{h})}{\partial t} = \frac{\partial \mathbf{F}(\mathbf{x} + t\mathbf{h})}{\partial (\mathbf{x} + t\mathbf{h})} \frac{\partial (\mathbf{x} + t\mathbf{h})}{\partial t} = \mathbf{J}(\mathbf{x} + t\mathbf{h})\,\mathbf{h}.$$

We can then integrate this result between 0 and 1:

$$\begin{aligned}
\int_0^1 \frac{\partial}{\partial t}\mathbf{F}(\mathbf{x} + t\mathbf{h})\,\mathrm{d}t &= \int_0^1 \mathbf{J}(\mathbf{x} + t\mathbf{h})\,\mathbf{h}\,\mathrm{d}t \\
[\mathbf{F}(\mathbf{x} + t\mathbf{h})]_0^1 &= \int_0^1 \mathbf{J}(\mathbf{x} + t\mathbf{h})\,\mathbf{h}\,\mathrm{d}t \\
\mathbf{F}(\mathbf{x} + \mathbf{h}) - \mathbf{F}(\mathbf{x}) &= \int_0^1 \mathbf{J}(\mathbf{x} + t\mathbf{h})\,\mathbf{h}\,\mathrm{d}t \\
\mathbf{F}(\mathbf{x} + \mathbf{h}) &= \mathbf{F}(\mathbf{x}) + \int_0^1 \mathbf{J}(\mathbf{x} + t\mathbf{h})\,\mathbf{h}\,\mathrm{d}t.
\end{aligned}$$

□

10.2 Lipschitz Continuity

Definition 10.4 (Lipschitz continuous)**.** Let $D \in \mathbb{R}^n$ and $\mathbf{J} : D \to \mathbb{R}^{n \times n}$. $\mathbf{J}$ is **Lipschitz continuous** on D, with **Lipschitz constant** γ if

$$\|\mathbf{J}(\mathbf{x}) - \mathbf{J}(\mathbf{y})\| \leqslant \gamma \|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in D.$$

We can denote this as $\mathbf{J} \in \mathrm{Lip}_\gamma(D)$.

Using this definition, the **Standard Assumptions of Newton's Method** can be described as follows:

1. $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ has a solution $\mathbf{x}^*$ on $D \subset \mathbb{R}^n$.
2. $\mathbf{J} \in \mathrm{Lip}_\gamma(D)$.
3. $\mathbf{J}(\mathbf{x})$ is nonsingular and $\|\mathbf{J}(\mathbf{x})\| \leqslant \beta$ for all $\mathbf{x} \in D$.

Theorem 10.2. *Let $\mathbf{F} : D \to \mathbb{R}^n$ for some open subset $D \subset \mathbb{R}^n$ and let the standard assumptions hold. Then, there exists some δ such that given $\mathbf{x}^{(0)} \in D$ with $\|\mathbf{x}^{(0)} - \mathbf{x}^*\| < \delta$, the Newton iteration*

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}\left(\mathbf{x}^{(k)}\right)^{-1} \mathbf{F}\left(\mathbf{x}^{(k)}\right)$$

satisfies $\|\mathbf{x}^{(k)} - \mathbf{x}^*\| < \delta$ for all k, and converges quadratically to $\mathbf{x}^*$.

Lemma 10.2.1. *Let $\mathbf{F} : D \to \mathbb{R}^n$ for some open subset $D \subset \mathbb{R}^n$ and let the standard assumptions hold. Then, the local linear model*

$$\mathbf{M}_k(\mathbf{x}) = \mathbf{F}\left(\mathbf{x}^{(k)}\right) + \mathbf{J}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x} - \mathbf{x}^{(k)}\right)$$

apprpoximates $\mathbf{F}(\mathbf{x})$ with error

$$\|\mathbf{F}(\mathbf{x}) - \mathbf{M}_k(\mathbf{x})\| \leqslant \frac{\gamma}{2} \|\mathbf{x} - \mathbf{x}^{(k)}\|^2.$$

Theorem 10.3. *Let $\mathbf{F} : D \to \mathbb{R}^n$ for some open subset $D \subset \mathbb{R}^n$ and let the standard assumptions hold. Then there exists some δ such that given $\mathbf{x}^{(0)} \in D$ with $\left\|\mathbf{x}^{(0)} - \mathbf{x}^*\right\| < \delta$, the Chord iteration*

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}\left(\mathbf{x}^{(0)}\right)^{-1} \mathbf{F}\left(\mathbf{x}^{(k)}\right)$$

satisfies $\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| < \delta$ for all k, and converges linearly to $\mathbf{x}^*$.

Corollary 10.3.1. *The Shamanskii method converges superlinearly to $\mathbf{x}^*$.*

11 Inexact Newton Method

The derivatives in the Jacobian matrix can be approximated using finite differences. Here, the first-order forward difference is used to approximate the jth column of the Jacobian as follows:

$$\mathbf{J}(\mathbf{x})\,\mathbf{e}_j \approx \tilde{\mathbf{J}}(\mathbf{x})\,\mathbf{e}_j = \frac{\mathbf{F}\left(\mathbf{x} + \varepsilon \mathbf{e}_j\right) - \mathbf{F}(\mathbf{x})}{\varepsilon}$$

for a suitably small value $\varepsilon > 0$, where $\mathbf{e}_j$ is the jth standard basis vector. The approximation of the Jacobian matrix is denoted $\tilde{\mathbf{J}}(\mathbf{x})$:

$$\tilde{\mathbf{J}}(\mathbf{x}) = \begin{bmatrix} | & & | \\ \tilde{\mathbf{J}}(\mathbf{x})\,\mathbf{e}_1 & \cdots & \tilde{\mathbf{J}}(\mathbf{x})\,\mathbf{e}_n \\ | & & | \end{bmatrix}.$$

The standard choice for the parameter ε is

$$\varepsilon = \begin{cases} \sqrt{\epsilon_M}\|\mathbf{x}\|_2, & \mathbf{x} \neq 0 \\ \sqrt{\epsilon_M}, & \mathbf{x} = 0 \end{cases}$$

where ϵ_M is the unit roundoff.

Theorem 11.1. *Let $\mathbf{F} : D \to \mathbb{R}^n$ for some open subset $D \subset \mathbb{R}^n$ and let $\mathbf{J} \in \operatorname{Lip}_\gamma(D)$, then*

$$\left\|\mathbf{J}(\mathbf{x})\,\mathbf{e}_j - \tilde{\mathbf{J}}(\mathbf{x})\,\mathbf{e}_j\right\| \leqslant \frac{\gamma}{2}\varepsilon, \quad 1 \leqslant j \leqslant n,$$

where $\tilde{\mathbf{J}}(\mathbf{x})$ approximates $\mathbf{J}(\mathbf{x})$ using finite differences.

Theorem 11.2. *Let $\mathbf{F} : D \to \mathbb{R}^n$ for some open subset $D \subset \mathbb{R}^n$ and let the standard assumptions hold. Then there exists some δ such that given $\mathbf{x}^{(0)} \in D$ with $\left\|\mathbf{x}^{(0)} - \mathbf{x}^*\right\| < \delta$, the inexact Newton iteration*

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \tilde{\mathbf{J}}\left(\mathbf{x}^{(k)}\right)^{-1} \mathbf{F}\left(\mathbf{x}^{(k)}\right)$$

satisfies $\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| < \delta$ for all k, and converges (near) quadratically to $\mathbf{x}^*$.

When the Jacobian matrix is tridiagonal,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & & & & \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & & & \\ & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} & & \\ & & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} & \frac{\partial f_4}{\partial x_5} & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}$$

we can use a compact storage method to store the non-zero entries of $\mathbf{J}$ in an $n \times 3$ matrix. Consider the following *shift vectors*:

$$\mathbf{s}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix}, \quad \mathbf{s}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}, \quad \mathbf{s}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix},$$

so that by multiplying $\mathbf{J}$ by each shift vector, we can pack non-overlapping columns of $\mathbf{J}$:

$$\mathbf{J}\mathbf{s}_1 = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} \\ \frac{\partial f_2}{\partial x_1} \\ \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_4} \\ \frac{\partial f_5}{\partial x_4} \\ \frac{\partial f_6}{\partial x_7} \\ \frac{\partial f_7}{\partial x_7} \\ \vdots \end{bmatrix}, \quad \mathbf{J}\mathbf{s}_2 = \begin{bmatrix} \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_2} \\ \frac{\partial f_3}{\partial x_2} \\ \frac{\partial f_4}{\partial x_5} \\ \frac{\partial f_5}{\partial x_5} \\ \frac{\partial f_6}{\partial x_5} \\ \frac{\partial f_7}{\partial x_8} \\ \vdots \end{bmatrix}, \quad \mathbf{J}\mathbf{s}_3 = \begin{bmatrix} \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_3} \\ \frac{\partial f_4}{\partial x_5} \\ \frac{\partial f_5}{\partial x_5} \\ \frac{\partial f_6}{\partial x_5} \\ \frac{\partial f_7}{\partial x_8} \\ \vdots \end{bmatrix}.$$

Therefore the first-order finite differences method becomes

$$\mathbf{J}(\mathbf{x})\,\mathbf{s}_j \approx \frac{\mathbf{F}\left(\mathbf{x} + \varepsilon_j \mathbf{s}_j\right) - \mathbf{F}(\mathbf{x})}{\varepsilon_j}$$

where

$$\varepsilon_j = \begin{cases} \dfrac{\sqrt{\epsilon_M}\|\mathbf{x}\|_2}{\|\mathbf{s}_j\|_2}, & \mathbf{x} \neq 0 \\ \dfrac{\sqrt{\epsilon_M}}{\|\mathbf{s}_j\|_2}, & \mathbf{x} = 0 \end{cases}$$

11.1 Line Searching

The methods above may not guarantee convergence for an arbitrary initial iterate $\mathbf{x}^{(0)}$. To improve upon the convergence of Newton's method, we will consider Newton methods that utilise **line searching**. When performing a Newton step, we will introduce a parameter $\lambda^{(k)}$ that limits the size of the Newton step by considering the modified Newton iteration:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \boldsymbol{\delta}\mathbf{x}^{(k)}$$

where $0 < \lambda^{(k)} \leqslant 1$. We then choose $\lambda^{(k)}$ to ensure that the nonlinear residual norm reduces during each Newton iteration:

$$\left\|\mathbf{F}\left(\mathbf{x}^{(k+1)}\right)\right\| < \left\|\mathbf{F}\left(\mathbf{x}^{(k)}\right)\right\|.$$

In some cases this may still lead to oscillations, and thus a stronger condition is used in practice:

$$\left\|\mathbf{F}\left(\mathbf{x}^{(k+1)}\right)\right\| < \left(1 - \alpha\lambda^{(k)}\right)\left\|\mathbf{F}\left(\mathbf{x}^{(k)}\right)\right\|.$$

where $\alpha = 10^{-4}$. This condition is known as the **Armijo rule**.

11.1.1 Simple Line Searching

In the simple line search, $\lambda^{(k)}$ is chosen by repeatedly reducing it by some factor $0 < \sigma < 1$ until the above criteria is met. In the simple line search, this factor is fixed for all iterations, and a common choice is 0.5:

Algorithm 1 Simple Line Searching

```
$\lambda^{(k)} = 1$
$\tilde{\mathbf{x}}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)}\boldsymbol{\delta}\mathbf{x}^{(k)}$        ▷ Next candidate for $\mathbf{x}^{(k+1)}$
while $\|\mathbf{F}(\mathbf{x}^{(k+1)})\| \geqslant (1 - \alpha\lambda^{(k)}) \|\mathbf{F}(\mathbf{x}^{(k)})\|$ do
    $\lambda^{(k)} = \sigma\lambda^{(k)}$
    $\tilde{\mathbf{x}}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)}\boldsymbol{\delta}\mathbf{x}^{(k)}$
end while
$\mathbf{x}^{(k+1)} = \tilde{\mathbf{x}}^{(k+1)}$
```

11.1.2 General Line Searching

More sophisticated variations of this algorithm allow $\lambda^{(k)}$ to reduce by a variable factor on every iteration. Here we will allow ourselves to choose the value of σ where $\sigma \in [\sigma_0, \sigma_1]$ and $0 < \sigma_0 < \sigma_1 < 1$. Typical values are $\sigma_0 = 0.1$ and $\sigma_1 = 0.5$. Consider the function:

$$g(\lambda) = \left\|\mathbf{F}\left(\mathbf{x}^{(k+1)}\right)\right\|^2 = \left\|\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)\right\|^2$$

which we will approximate by a polynomial $p(\lambda)$ to find the value of $\lambda^{(k)}$ that minimises $p(\lambda)$. Assume that we have rejected m values of $\lambda^{(k)}$ (resulting in m rejected values of $\tilde{\mathbf{x}}^{(k+1)}$). Then, we have available the values:

$$\left\|\mathbf{F}\left(\mathbf{x}^{(k)}\right)\right\|, \left\|\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda_1^{(k)}\boldsymbol{\delta}\mathbf{x}^{(k)}\right)\right\|, \dots, \left\|\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda_m^{(k)}\boldsymbol{\delta}\mathbf{x}^{(k)}\right)\right\|,$$

where $\lambda_1^{(k)}$ is the most recently rejected value for $\lambda^{(k)}$. Then, we can minimise $p(\lambda)$ using calculus. In the following sections we will consider the **two-point parabolic model** and the **three-point parabolic model**. Given,

$$g(0) = \left\|\mathbf{F}\left(\mathbf{x}^{(k)}\right)\right\|^2$$

consider the derivative of $g'(\lambda)$:

$$\begin{aligned}
g'(\lambda) &= \frac{\partial}{\partial\lambda}\left[\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)^\top \mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)\right] \\
&= \mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)^\top \frac{\partial\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)}{\partial\lambda} + \left(\frac{\partial\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)}{\partial\lambda}\right)^\top \mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right) \\
&= 2\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)^\top \frac{\partial\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)}{\partial\lambda} \\
&= 2\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)^\top \frac{\partial\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)}{\partial(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)})} \frac{\partial\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)}{\partial\lambda} \\
&= 2\mathbf{F}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)^\top \mathbf{J}\left(\mathbf{x}^{(k)} + \lambda\boldsymbol{\delta}\mathbf{x}^{(k)}\right)\boldsymbol{\delta}\mathbf{x}^{(k)}.
\end{aligned}$$

Then when $\lambda = 0$,

$$\begin{aligned}
g'(0) &= 2\mathbf{F}\left(\mathbf{x}^{(k)}\right)^\top \mathbf{J}\left(\mathbf{x}^{(k)}\right)\boldsymbol{\delta}\mathbf{x}^{(k)} \\
&= 2\mathbf{F}\left(\mathbf{x}^{(k)}\right)^\top \mathbf{J}\left(\mathbf{x}^{(k)}\right)\left[-\mathbf{J}\left(\mathbf{x}^{(k)}\right)^{-1}\mathbf{F}\left(\mathbf{x}^{(k)}\right)\right] \\
&= -2\mathbf{F}\left(\mathbf{x}^{(k)}\right)^\top \mathbf{F}\left(\mathbf{x}^{(k)}\right) \\
&= -2\left\|\mathbf{F}\left(\mathbf{x}^{(k)}\right)\right\|^2 \\
&= -2g(0).
\end{aligned}$$

11.1.3 Two-Point Parabolic Model

In the two-point parabolic model, we will use the known values $g(0)$, $g'(0)$, and $g\left(\lambda_1^{(k)}\right)$, to construct a quadratic approximation for $g(\lambda)$.

$$\begin{aligned}
p(0) &= g(0) \\
p'(0) &= g'(0) = -2g(0) \\
p\left(\lambda_1^{(k)}\right) &= g\left(\lambda_1^{(k)}\right)
\end{aligned}$$

to find

$$p(\lambda) = \frac{2g(0)\lambda_1^{(k)} - g(0) + g\left(\lambda_1^{(k)}\right)}{\lambda_1^{(k)2}}\lambda^2 - 2g(0)\lambda + g(0).$$

Using the minimum of this function, we can modify the line searching algorithm as follows:

Algorithm 2 Two-Point Parobolic Line Searching

```
$\lambda^{(k)} = 1$
$\tilde{\mathbf{x}}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)}\boldsymbol{\delta}\mathbf{x}^{(k)}$          ▷ Next candidate for $\mathbf{x}^{(k+1)}$
while $\left\|\mathbf{F}\left(\mathbf{x}^{(k+1)}\right)\right\| \geqslant \left(1 - \alpha\lambda^{(k)}\right)\left\|\mathbf{F}\left(\mathbf{x}^{(k)}\right)\right\|$ do
    $\lambda_1^{(k)} = \lambda^{(k)}$
    $\lambda^{(k)} = \arg\min_\lambda p(\lambda)$
    $\lambda^{(k)} = \max\left(0.1\lambda_1^{(k)},\ \min\left(\lambda^{(k)},\ 0.5\lambda_1^{(k)}\right)\right)$          ▷ Clamp $\lambda^{(k)}$ between $[\sigma_0\lambda_1^{(k)}, \sigma_1\lambda_1^{(k)}]$
    $\tilde{\mathbf{x}}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)}\boldsymbol{\delta}\mathbf{x}^{(k)}$
end while
$\mathbf{x}^{(k+1)} = \tilde{\mathbf{x}}^{(k+1)}$
```

The clamp function ensures the reduction factor is within $[\sigma_0, \sigma_1]$, and is alternatively defined as:

$$\max\left(0.1\lambda_1^{(k)},\ \min\left(\lambda^{(k)},\ 0.5\lambda_1^{(k)}\right)\right) = \begin{cases} \sigma_0\lambda_1^{(k)} & \lambda^{(k)} < \sigma_0\lambda_1^{(k)} \\ \sigma_1\lambda_1^{(k)} & \lambda^{(k)} > \sigma_1\lambda_1^{(k)} \\ \lambda^{(k)} & \text{otherwise.} \end{cases}$$

11.1.4 Three-Point Parabolic Model

In the three-point parabolic model, we will use the known values $g(0)$, $g'(0)$, $g\left(\lambda_1^{(k)}\right)$, and $g\left(\lambda_1^{(k)}\right)$, to construct a quadratic approximation for $g(\lambda)$.

$$\begin{aligned} p(0) &= g(0) \\ p'(0) &= g'(0) = -2g(0) \\ p\left(\lambda_1^{(k)}\right) &= g\left(\lambda_1^{(k)}\right) p\left(\lambda_2^{(k)}\right) & &= g\left(\lambda_2^{(k)}\right) \end{aligned}$$

to find

$$p(\lambda) = g(0) + \frac{\lambda}{\lambda_1^{(k)} - \lambda_2^{(k)}} \left(\frac{\left(\lambda - \lambda_2^{(k)}\right)\left(g\left(\lambda_1^{(k)}\right) - g(0)\right)}{\lambda_1^{(k)}} - \frac{\left(\lambda - \lambda_1^{(k)}\right)\left(g\left(\lambda_2^{(k)}\right) - g(0)\right)}{\lambda_2^{(k)}} \right)$$

Using the minimum of this function, we can modify the line searching algorithm as follows:

Algorithm 3 Three-Point Parobolic Line Searching

```
$\lambda^{(k)} = 1$
$\tilde{\mathbf{x}}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \boldsymbol{\delta}\mathbf{x}^{(k)}$        ▷ Next candidate for $\mathbf{x}^{(k+1)}$
while $\left\|\mathbf{F}\left(\mathbf{x}^{(k+1)}\right)\right\| \geqslant \left(1 - \alpha\lambda^{(k)}\right)\left\|\mathbf{F}\left(\mathbf{x}^{(k)}\right)\right\|$ do
    if $m = 1$ then
        $\lambda_1^{(k)} = \lambda^{(k)}$
        $\lambda^{(k)} = \lambda^{(k)}/2$
    else
        $\lambda_2^{(k)} = \lambda_1^{(k)}$
        $\lambda_1^{(k)} = \lambda^{(k)}$
        $\lambda^{(k)} = \arg\min_\lambda p(\lambda)$
        $\lambda^{(k)} = \max\left(0.1\lambda_1^{(k)},\ \min\left(\lambda^{(k)},\ 0.5\lambda_1^{(k)}\right)\right)$        ▷ Clamp $\lambda^{(k)}$ between $[\sigma_0\lambda_1^{(k)}, \sigma_1\lambda_1^{(k)}]$
    end if
    $\tilde{\mathbf{x}}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \boldsymbol{\delta}\mathbf{x}^{(k)}$
end while
$\mathbf{x}^{(k+1)} = \tilde{\mathbf{x}}^{(k+1)}$
```

Part III
Numerical Linear Algebra

Let us now consider how we might efficiently approximate the solution of the linear system:

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

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is non-symmetric and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$. In particular, we will explore the case where $\mathbf{A}$ is large and sparse, as is the case for the linear systems we have seen thus far. In the following sections, we will discuss some iterative methods that outperform traditional matrix decomposition methods, for large n.

12 Krylov Subspace

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. Krylov subpace methods seek to find an approximate solution to

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

within a Krylov subpace of $\mathbb{R}^n$.

Definition 12.1 (Krylov Subspace)**.** Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \neq \mathbf{0} \in \mathbb{R}^n$. The m-dimensional Krylov subspace is spanned by the image of $\mathbf{b}$ under the first m powers of $\mathbf{A}$:

$$\mathcal{K}_m(\mathbf{A},\, \mathbf{b}) = \operatorname{span}\left\{\mathbf{b},\, \mathbf{A}\mathbf{b},\, \mathbf{A}^2\mathbf{b},\, \ldots,\, \mathbf{A}^{m-1}\mathbf{b}\right\}.$$

12.1 Hessenberg Decomposition

Definition 12.2 (Hessenberg Matrix)**.** We define the matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ as being in Hessenberg form in one of the following ways:

- $\mathbf{H}$ is upper-Hessenberg if $h_{ij} = 0$ for all $i > j+1$
- $\mathbf{H}$ is lower-Hessenberg if $h_{ij} = 0$ for all $j > i+1$

Such matrices are almost triangular and allow the entries in the sub diagonal or super diagonal to be nonzero, for upper and lower triangular matrices respectively.

For $\mathbf{A} \in \mathbb{R}^{n\times n}$, let us consider the Hessenberg decomposition of $\mathbf{A}$ defined as:

$$\mathbf{A} = \mathbf{Q}\mathbf{H}\mathbf{Q}^\top$$

where $\mathbf{H} \in \mathbb{R}^{n\times n}$ is upper-Hessenberg and $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is orthogonal (so that $\mathbf{Q}\mathbf{Q}^\top = \mathbf{I}$). If we use block matrix notation, the factorisation $\mathbf{A}\mathbf{Q} = \mathbf{Q}\mathbf{H}$:

$$\mathbf{A}\begin{bmatrix} | & & | \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ | & & | \end{bmatrix}\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & & \\ & \ddots & \ddots & \vdots \\ & & h_{n,n-1} & h_{n,n} \end{bmatrix}$$

shows that the columns of $\mathbf{A}\mathbf{Q}$ are given by:

$$\begin{aligned} \mathbf{A}\mathbf{q}_1 &= h_{11}\mathbf{q}_1 + h_{21}\mathbf{q}_2 \\ \mathbf{A}\mathbf{q}_2 &= h_{12}\mathbf{q}_1 + h_{22}\mathbf{q}_2 + h_{32}\mathbf{q}_3 \\ &\vdots \\ \mathbf{A}\mathbf{q}_n &= h_{1n}\mathbf{q}_1 + h_{2n}\mathbf{q}_2 + \cdots + h_{n,n}\mathbf{q}_n + h_{n+1,n}\mathbf{0} \end{aligned}$$

Observe that due to the Hessenberg form of $\mathbf{H}$, we have one fewer dimensions in the span of the matrix-vector product $\mathbf{A}\mathbf{q}_n$. As n is large, let us instead consider the first $m \ll n$ columns of the factorisation $\mathbf{A}\mathbf{Q} = \mathbf{Q}\mathbf{H}$, so that the jth column of $\mathbf{A}\mathbf{Q}$ can be expressed as:

$$\begin{aligned} \mathbf{A}\mathbf{q}_1 &= h_{11}\mathbf{q}_1 + h_{21}\mathbf{q}_2 \\ \mathbf{A}\mathbf{q}_2 &= h_{12}\mathbf{q}_1 + h_{22}\mathbf{q}_2 + h_{32}\mathbf{q}_3 \\ &\vdots \\ \mathbf{A}\mathbf{q}_j &= h_{1j}\mathbf{q}_1 + h_{2j}\mathbf{q}_2 + \cdots + h_{jj}\mathbf{q}_j + h_{j+1,j}\mathbf{q}_{j+1} = \sum_{i=1}^{j+1} h_{ij}\mathbf{q}_i. \end{aligned}$$

This gives us the reduced Hessenberg factorisation:

$$\mathbf{A}\begin{bmatrix} | & & | \\ \mathbf{q}_1 & \cdots & \mathbf{q}_m \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ \mathbf{q}_1 & \cdots & \mathbf{q}_{m+1} \\ | & & | \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1m} \\ h_{21} & h_{22} & & \\ & \ddots & \ddots & \vdots \\ & & h_{m,m-1} & h_{mm} \\ & & & h_{m+1,m} \end{bmatrix}$$

$$\mathbf{A}\mathbf{Q}_m = \mathbf{Q}_{m+1}\bar{\mathbf{H}}$$

where $\mathbf{Q}_m \in \mathbb{R}^{n\times m}$, $\mathbf{Q}_{m+1} \in \mathbb{R}^{n\times m+1}$, and $\bar{\mathbf{H}} \in \mathbb{R}^{m+1\times m}$. Note that the matrix $\bar{\mathbf{H}}$ is now no longer Hessenberg. By rearranging for $\mathbf{q}_{j+1}$, we find that these vectors are the result of the Gram-Schmidt method that constructs h_{ij} and the orthonormal vectors $\mathbf{q}_j$:

$$\begin{aligned} \mathbf{q}_{j+1} &= \left(\mathbf{A}\mathbf{q}_j - h_{1j}\mathbf{q}_1 - h_{2j}\mathbf{q}_2 - \cdots - h_{jj}\mathbf{q}_j\right)/h_{j+1,j} \\ &= \frac{1}{h_{j+1,j}}\left(\mathbf{A}\mathbf{q}_j - \sum_{i=1}^{j} h_{ij}\mathbf{q}_i\right). \end{aligned}$$

12.2 Arnoldi's Method

Let us now consider Arnoldi's method for constructing the matrices $\mathbf{Q}_m$ and $\mathbf{H}_m$, using the Krylov subspace $\mathcal{K}_m(\mathbf{A},\, \mathbf{b})$, so that,

$$\operatorname{span}\{\mathbf{b},\, \mathbf{A}\mathbf{b},\, \mathbf{A}^2\mathbf{b},\, \dots,\, \mathbf{A}^{m-1}\mathbf{b}\} = \operatorname{span}\{\mathbf{q}_1,\, \mathbf{q}_2,\, \dots,\, \mathbf{q}_m\}.$$

We will do so by applying the Gram-Schmidt method on the vectors spanned by Krylov subspaces of increasing dimensions. Consider the Krylov subspace of dimension 1:

$$\mathcal{K}_1(\mathbf{A},\, \mathbf{b}) = \operatorname{span}\{\mathbf{b}\}.$$

Applying Gram-Schmidt,

$$\mathbf{v}_1 = \mathbf{b}, \qquad \mathbf{q}_1 = \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}$$

therefore,

$$\mathcal{K}_1(\mathbf{A},\, \mathbf{b}) = \operatorname{span}\{\mathbf{b}\} = \operatorname{span}\{\mathbf{q}_1\}.$$

Consider the Krylov subspace of dimension 2:

$$\mathcal{K}_2(\mathbf{A},\, \mathbf{b}) = \operatorname{span}\{\mathbf{b},\, \mathbf{A}\mathbf{b}\} = \operatorname{span}\{\mathbf{q}_1,\, \mathbf{A}\mathbf{q}_1\},$$

where we recognise that $\mathbf{q}_1$ is a linear combination of $\mathbf{b}$. Applying Gram-Schmidt using this information,

$$\mathbf{v}_2 = \mathbf{A}\mathbf{q}_1 - \left(\mathbf{q}_1^\top\mathbf{A}\mathbf{q}_1\right)\mathbf{q}_1, \qquad \mathbf{q}_2 = \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|}$$

therefore,

$$\mathcal{K}_2(\mathbf{A},\, \mathbf{b}) = \operatorname{span}\{\mathbf{b},\, \mathbf{A}\mathbf{b}\} = \operatorname{span}\{\mathbf{q}_1,\, \mathbf{q}_2\}.$$

Finally, consider the Krylov subspace of dimension 3:

$$\mathcal{K}_3(\mathbf{A},\ \mathbf{b}) = \operatorname{span}\{\mathbf{b},\ \mathbf{Ab},\ \mathbf{A}^2\mathbf{b}\} = \operatorname{span}\{\mathbf{q}_1,\ \mathbf{Aq}_1,\ \mathbf{A}^2\mathbf{q}_1\} = \operatorname{span}\{\mathbf{q}_1,\ \mathbf{q}_2,\ \mathbf{Aq}_2\},$$

where we again use the result that $\mathbf{Aq}_1$ is a linear combination of $\mathbf{q}_1$ and $\mathbf{q}_2$:

$$\mathbf{Aq}_1 = (\mathbf{q}_1^\top \mathbf{Aq}_1)\,\mathbf{q}_1 - \|\mathbf{v}_2\|\mathbf{q}_2.$$

This implies that $\mathbf{A}^2\mathbf{q}_1$ is a linear combination of $\mathbf{Aq}_1$ and $\mathbf{Aq}_2$, as:

$$\mathbf{A}^2\mathbf{q}_1 = \mathbf{A}(\mathbf{Aq}_1) = (\mathbf{q}_1^\top \mathbf{Aq}_1)\,\mathbf{Aq}_1 - \|\mathbf{v}_2\|\mathbf{Aq}_2.$$

Then, applying Gram-Schmidt shows,

$$\mathbf{v}_3 = \mathbf{Aq}_2 - (\mathbf{q}_1^\top \mathbf{Aq}_2)\,\mathbf{q}_1 - (\mathbf{q}_2^\top \mathbf{Aq}_2)\,\mathbf{q}_2, \qquad \mathbf{q}_3 = \frac{\mathbf{v}_3}{\|\mathbf{v}_3\|}$$

therefore,

$$\mathcal{K}_3(\mathbf{A},\ \mathbf{b}) = \operatorname{span}\{\mathbf{b},\ \mathbf{Ab},\ \mathbf{A}^2\mathbf{b}\} = \operatorname{span}\{\mathbf{q}_1,\ \mathbf{q}_2,\ \mathbf{q}_3\}.$$

If we continue this process, we find that for an m-dimensional Krylov subspace, the Gram-Schmidt process produces the following orthonormal vector $\mathbf{q}_{j+1}$:

$$\begin{aligned}
\|\mathbf{v}_{j+1}\|\mathbf{q}_{j+1} &= \mathbf{Aq}_j - (\mathbf{q}_1^\top \mathbf{Aq}_j)\,\mathbf{q}_1 - (\mathbf{q}_2^\top \mathbf{Aq}_j)\,\mathbf{q}_2 - \cdots - \left(\mathbf{q}_j^\top \mathbf{Aq}_j\right)\mathbf{q}_j \\
\mathbf{q}_{j+1} &= \frac{1}{\|\mathbf{v}_{j+1}\|}\left(\mathbf{Aq}_j - \sum_{i=1}^{j} (\mathbf{q}_i^\top \mathbf{Aq}_j)\,\mathbf{q}_i\right),
\end{aligned}$$

and $\mathbf{Aq}_j$ is given by:

$$\mathbf{Aq}_j = \sum_{i=1}^{j} (\mathbf{q}_i^\top \mathbf{Aq}_j)\,\mathbf{q}_i + \|\mathbf{v}_{j+1}\|\mathbf{q}_{j+1}.$$

In this form, we can see that this is precisely the Hessenberg decomposition, where we can build the matrix $\bar{\mathbf{H}}_m$ using:

$$h_{ij} = \mathbf{q}_i^\top \mathbf{Aq}_j, \quad h_{j+1,j} = \|\mathbf{v}_{j+1}\|.$$

This shows us a recurrence relationship between $\mathbf{H}_m$ and $\bar{\mathbf{H}}_m$:

$$\begin{aligned}
\bar{\mathbf{H}}_1 &= \begin{bmatrix} \mathbf{q}_1^\top \mathbf{Aq}_1 \\ \|\mathbf{v}_2\| \end{bmatrix} = [\mathbf{q}_1^\top \mathbf{Aq}_1] + \|\mathbf{v}_2\|\mathbf{e}_1^\top = \mathbf{H}_1 + \|\mathbf{v}_2\|\mathbf{e}_1^\top \\
\bar{\mathbf{H}}_2 &= \begin{bmatrix} \mathbf{q}_1^\top \mathbf{Aq}_1 & \mathbf{q}_1^\top \mathbf{Aq}_2 \\ \|\mathbf{v}_2\| & \mathbf{q}_2^\top \mathbf{Aq}_2 \\ & \|\mathbf{v}_3\| \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1^\top \mathbf{Aq}_1 & \mathbf{q}_1^\top \mathbf{Aq}_2 \\ \|\mathbf{v}_2\| & \mathbf{q}_2^\top \mathbf{Aq}_2 \end{bmatrix} + \|\mathbf{v}_3\|\mathbf{e}_2^\top = \mathbf{H}_2 + \|\mathbf{v}_3\|\mathbf{e}_2^\top \\
\bar{\mathbf{H}}_3 &= \begin{bmatrix} \mathbf{q}_1^\top \mathbf{Aq}_1 & \mathbf{q}_1^\top \mathbf{Aq}_2 & \mathbf{q}_1^\top \mathbf{Aq}_3 \\ \|\mathbf{v}_2\| & \mathbf{q}_2^\top \mathbf{Aq}_2 & \mathbf{q}_2^\top \mathbf{Aq}_3 \\ & \|\mathbf{v}_3\| & \mathbf{q}_3^\top \mathbf{Aq}_3 \\ & & \|\mathbf{v}_4\| \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1^\top \mathbf{Aq}_1 & \mathbf{q}_1^\top \mathbf{Aq}_2 & \mathbf{q}_1^\top \mathbf{Aq}_3 \\ \|\mathbf{v}_2\| & \mathbf{q}_2^\top \mathbf{Aq}_2 & \mathbf{q}_2^\top \mathbf{Aq}_3 \\ & \|\mathbf{v}_3\| & \mathbf{q}_3^\top \mathbf{Aq}_3 \end{bmatrix} + \|\mathbf{v}_4\|\mathbf{e}_3^\top = \mathbf{H}_3 + \|\mathbf{v}_4\|\mathbf{e}_3^\top \\
&\vdots
\end{aligned}$$

where

$$\bar{\mathbf{H}}_m = \mathbf{H}_m + h_{m+1,m}\mathbf{e}_m^\top$$

which we can express in block matrix notation as:

$$\bar{\mathbf{H}}_m = \begin{bmatrix} \mathbf{H}_m \\ h_{m+1,m}\mathbf{e}_m^\top. \end{bmatrix}$$

This allows us to write the Hessenberg factorisation using the Hessenberg matrix $\mathbf{H}_m$:

$$\mathbf{A}\mathbf{Q}_m = \mathbf{Q}_m\mathbf{H}_m + h_{m+1,m}\mathbf{q}_{m+1}\mathbf{e}_m^\top.$$

By orthogonality, this matrix can be decomposed in terms of $\mathbf{A}$ and $\mathbf{Q}_m$:

$$\begin{aligned} \mathbf{A}\mathbf{Q}_m &= \mathbf{Q}_m\mathbf{H}_m + h_{m+1,m}\mathbf{q}_{m+1}\mathbf{e}_m^\top \\ \mathbf{Q}_m^\top\mathbf{A}\mathbf{Q}_m &= \left(\mathbf{Q}_m^\top\mathbf{Q}_m\right)\mathbf{H}_m + h_{m+1,m}\left(\mathbf{Q}_m^\top\mathbf{q}_{m+1}\right)\mathbf{e}_m^\top \\ \mathbf{Q}_m^\top\mathbf{A}\mathbf{Q}_m &= \mathbf{H}_m. \end{aligned}$$

Using this result, we can estimate the eigenvalues and eigenvectors of $\mathbf{H}_m = \mathbf{Q}_m^\top\mathbf{A}\mathbf{Q}_m$ iteratively using the QR algorithm, to estimate m eigenvalues and eigenvectors of $\mathbf{A}$, called **Ritz values** and **Ritz vectors**, respectively. Like the Hessenberg matrix $\mathbf{H}_m$, the matrix $\bar{\mathbf{H}}_m$ can also be formed recursively:

$$\bar{\mathbf{H}}_m = \left[\begin{array}{c:c} & h_{1m} \\ \bar{\mathbf{H}}_{m-1} & \vdots \\ & h_{mm} \\ \hdashline & h_{m+1,m} \end{array}\right]$$

The Gram-Schmidt process can be performed on an m-dimensional Krylov subspace as follows:

Algorithm 4 Arnoldi's Method using the Classical Gram-Schmidt Process

```
$\mathbf{Q}_{:1} = \mathbf{b}/\|\mathbf{b}\|$
for $j = 1, \dots, m$ do
    $\mathbf{Q}_{:,j+1} = \mathbf{A}\mathbf{Q}_{:j}$
    for $i = 1, \dots, j$ do
        $\mathbf{H}_{ij} = \mathbf{Q}_{:i}^\top\mathbf{A}\mathbf{Q}_{:j}$                ▷ Construct upper triangular entries $h_{ij}$
        $\mathbf{Q}_{:,j+1} = \mathbf{Q}_{:,j+1} - \mathbf{H}_{ij}\mathbf{Q}_{:j}$
    end for
    $\mathbf{H}_{j+1,j} = \|\mathbf{Q}_{:,j+1}\|$                ▷ Construct sub-diagonal entries $h_{j+1,j}$
    if $\mathbf{H}_{j+1,j} \neq 0$ then
        $\mathbf{Q}_{:,j+1} = \mathbf{Q}_{:,j+1}/\mathbf{H}_{j+1,j}$                ▷ Normalise $\mathbf{q}_{j+1}$
    end if
end for
```

When using floating-point arithmetic, the basis vectors of $\mathbf{Q}_m$ are sometimes not orthogonal due to rounding errors. This can be corrected using the Modified Gram-Schmidt process.

Algorithm 5 Arnoldi's Method using the Modified Gram-Schmidt Process

```
$\mathbf{Q}_{:1} = \mathbf{b}/\|\mathbf{b}\|$
for $j = 1, \dots, m$ do
    $\mathbf{Q}_{:,j+1} = \mathbf{A}\mathbf{Q}_{:j}$
    for $i = 1, \dots, j$ do
        $\mathbf{H}_{ij} = \mathbf{Q}_{:i}^\top \mathbf{Q}_{:j}$                ▷ Construct upper triangular entries $h_{ij}$
        $\mathbf{Q}_{:,j+1} = \mathbf{Q}_{:,j+1} - \mathbf{H}_{ij}\mathbf{Q}_{:j}$
    end for
    $\mathbf{H}_{j+1,j} = \|\mathbf{Q}_{:,j+1}\|$                ▷ Construct sub-diagonal entries $h_{j+1,j}$
    if $\mathbf{H}_{j+1,j} \neq 0$ then
        $\mathbf{Q}_{:,j+1} = \mathbf{Q}_{:,j+1}/\mathbf{H}_{j+1,j}$                ▷ Normalise $\mathbf{q}_{j+1}$
    end if
end for
```

In this algorithm, we omit the left-multiplication by $\mathbf{A}$ on $\mathbf{Q}_{:,j}$, when computing h_{ij}.