Structured meshes

- Very simple computational domains can be discretized using boundary-fitted structured meshes (also called *grids*)
- The grid lines of a *Cartesian* mesh are parallel to one another
Structured meshes

- A general structured mesh is logically equivalent to a uniform Cartesian grid (there exists a one-to-one mapping)

- There are \( d \) families of grid lines in \( \mathbb{R}^d \), \( d = 1, 2, 3 \)

- Members of the same family do not cross each other and cross each member of the other families just once

- The grid lines of each family can be numbered consecutively

- The position of each grid point (or mesh cell) is uniquely identified by a set of \( d \) indices in \( \mathbb{R}^d \), e.g., \((x_i, y_j) \in \mathbb{R}^2\), \((x_i, y_j, z_k) \in \mathbb{R}^3\)
Structured meshes

- Each internal grid point has 4 nearest neighbors in 2D, 6 in 3D
- One of the indices of each pair of neighbor points differs by ±1
- This neighbor connectivity simplifies programming, provides fast data access and leads to efficient numerical algorithms
- The matrices of linear algebraic systems are banded and fast solvers (both direct and iterative) are readily available for such systems
Structured meshes

- The generation of a structured mesh may be difficult or impossible for computational domains of complex geometrical shape.

- Local mesh refinement in one subdomain produces unnecessarily small spacing in other parts of the domain.

- Long thin cells may adversely affect convergence of iterative solvers or result in violations of discrete maximum principles.
Block-structured meshes

- Two (or more) level subdivision of the computational domain
- Decomposition into relatively large overlapping or nonoverlapping subdomains (blocks, macroelements) on the coarse level
- Discretization of these blocks using structured meshes
Block-structured meshes

- Block-structured meshes with overlapping blocks are sometimes called *composite* or *Chimera* meshes.

- Mesh generation for complex domains and flow problems with moving objects or interfaces becomes relatively easy.
Block-structured meshes

- More flexibility than with single-block structured meshes
- Mesh spacing may be chosen individually for each block
- Local refinement in one block does not affect other blocks
- Solvers for structured grids can be applied blockwise
- Programming is more difficult than for structured meshes
- Interpolation and/or conservation errors at block interfaces
Unstructured meshes

- No restrictions on the number of neighbor elements or nodes
- Automatic mesh generation for complex domains (triangles or quadrilaterals in 2D, tetrahedra or hexaedra in 3D)
Unstructured meshes

- Highest flexibility w.r.t. mesh generation and adaptation (refinement, coarsening, redistribution of mesh points)

- The coordinates of each mesh point and numbers of nodes belonging to each element must be stored and retrieved

- High implementation effort, irregular sparsity pattern of coefficient matrices, slow data access due to indirect addressing

- Computational cost is significantly higher than that for a structured grid with the same number of mesh points/elements
Choice of the mesh

<table>
<thead>
<tr>
<th></th>
<th>structured</th>
<th>unstructured</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex geometries</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>local mesh refinement</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>automatic mesh generation</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>cost of mesh generation</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>programming effort</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>efficiency of data access</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>efficiency of algorithms</td>
<td>+</td>
<td>−</td>
</tr>
</tbody>
</table>
Finite approximations

Following the choice of the computational mesh, a discretization method for the equations of the mathematical model is selected.

Partial derivatives or integrals are commonly approximated by linear combinations of discrete function values.

The choice of the discretization method depends on the mesh:

- structured meshes: finite differences or finite volumes
- unstructured meshes: finite volumes or finite elements
Finite differences

Conservation law
\[
\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f} = q \quad \text{in } \Omega \times (0, T)
\]

Discrete unknowns
\[
u_i(t) \approx u(x_i, t), \quad i = 1, \ldots, N
\]

Approximation of derivatives
\[
\left( \frac{\partial u}{\partial x} \right)_i \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}, \quad \left( \frac{\partial^2 u}{\partial x^2} \right)_i \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}
\]
Finite differences

- This is the simplest and oldest discretization technique
- Derivation from the PDE form of the governing equation
- The discrete unknowns are solution values at the grid points
- The approximation of partial derivatives in terms of these values yields one algebraic equation per grid point
- The use of a (block-)structured mesh is usually required
Finite volumes

Integral conservation law

\[ \frac{d}{dt} \int_{V_i} u \, dx + \int_{S_i} f \cdot n \, ds = \int_{V_i} q \, dx \]

Discrete solution values

\[ u_i(t) = \frac{1}{|V_i|} \int_{V_i} u(x, t) \, dx \approx u(x_i, t) \]
Finite volumes

- Derivation from the integral form of a conservation law

- The discrete unknowns are cell averages or solution values at the centers of control volumes (mesh cells or dual mesh cells)

- Volume and surface integrals that appear in the conservation law are approximated using numerical quadrature rules

- Function values at the quadrature points are obtained using interpolation techniques (polynomial fitting)

- The computational mesh can be structured or unstructured
Finite elements

Variational problem

\[ \int_{\Omega} \left( w \frac{\partial u}{\partial t} - \nabla w \cdot \mathbf{f} - wq \right) \, dx + \int_{\Gamma} w \mathbf{f} \cdot \mathbf{n} \, ds = 0 \]

Discrete solution values

\[ u_h(x, t) = \sum_j u_j(t) \varphi_j(x), \quad u_i(t) \approx u(x_i, t) \]

Basis and test functions

\[ \varphi_i(x_i) = 1, \quad \varphi_i(x_j) = 0, \quad \forall j \neq i \]
Finite elements

- Derivation from a variational form of a PDE model

- The coefficients of the basis functions $\varphi_i$ typically represent approximate function values at interpolation points.

- Inside each cell, the approximate solution $u_h$ is a polynomial.

- Substitution of $u_h \approx u$ into the variational formulation with $w \in \{\varphi_i\}$ yields one algebraic equation per mesh node.

- The computational mesh can be structured or unstructured.
Time discretization

Discrete time levels

\[ 0 = t^0 < t^1 < \cdots < t^M = T, \quad t^n = n\Delta t \]

Approximation of the time derivative

\[ \frac{\partial u_i}{\partial t} \approx \frac{u_{i}^{n+1} - u_{i}^{n}}{\Delta t}, \quad u_{i}^{n} \approx u_i(t^n), \quad u_{i}^{0} = u_0(x_i) \]

Linear algebraic systems

\[ Au^{n+1} = b(u^n), \quad n = 0, 1, \ldots, M - 1 \]
Linear solvers

- Explicit methods: $A$ is a diagonal matrix; the equations of the linear system are decoupled and can be solved in a segregated manner.

- Implicit methods: $A$ is a **sparse** matrix; the equations of the linear system are coupled and must be solved using numerical methods.

- Iterative solvers that exploit sparsity (and structure, if any) of the coefficient matrix are usually more efficient than direct methods.
Choice of numerical tools

- There are many possibilities to discretize a given problem using finite difference, finite volume or finite element methods.

- The choice of approximations influences the accuracy of numerical solutions, programming effort, and computational cost.

- A compromise between simplicity, ease of implementation, accuracy and computational efficiency has to be made.

- Numerical algorithms must be stable to produce reasonable results.
Design criteria

- **Consistency**: local discretization errors should be proportional to positive powers of the mesh size $h$ and time step $\Delta t$

- **Stability**: rounding and iteration errors that appear in the process of numerical solution should not be magnified

- **Convergence**: numerical solutions should become exact in the limit of vanishing mesh sizes and time steps
Design criteria

- **Conservation:** numerical solutions should satisfy a discrete form of the integral conservation law (local or global)

- **Boundedness:** discrete maximum principles should hold if the exact solution satisfies a maximum principle

- **Accuracy:** at least second-order convergence for smooth data
Finite differences in 1D

- Let \( u : [a, b] \mapsto \mathbb{R} \) be a differentiable function of \( x \).

- A uniform mesh is generated using a subdivision of the domain \( \Omega = (a, b) \) into \( N \) subintervals of equal length \( \Delta x \)

  \[
  x_i = a + i\Delta x, \quad \Delta x = \frac{b - a}{N}
  \]

- The discrete unknowns are given by

  \[
  u_i \approx u(x_i), \quad i = 0, 1, \ldots, N
  \]
The first derivative of $u(x)$ at the mesh point $x_i$ is defined by

$$\frac{\partial u}{\partial x}(x_i) = \lim_{\Delta x \to 0} \frac{u(x_i + \Delta x) - u(x_i)}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{u(x_i) - u(x_i - \Delta x)}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{u(x_i + \Delta x) - u(x_i - \Delta x)}{2\Delta x}$$

and can be approximated using a finite (but small) spacing $\Delta x$
Finite differences

The first derivative of $u(x)$ at $x_i$ can be approximated by

- **forward difference**
  $\left( \frac{\partial u}{\partial x} \right)_i \approx \frac{u_{i+1} - u_i}{\Delta x}$

- **backward difference**
  $\left( \frac{\partial u}{\partial x} \right)_i \approx \frac{u_i - u_{i-1}}{\Delta x}$

- **central difference**
  $\left( \frac{\partial u}{\partial x} \right)_i \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}$

The quality of these approximations depends on $u$ and $\Delta x$
Geometric interpretation

- The first derivative of \( u \) at the mesh point \( x_i \) is the slope of the tangent to the curve \( u(x) \) at that mesh point.

- It can be approximated by the slope of a straight line passing through two nearby points on the curve.

![Diagram showing geometric interpretation of the first derivative with forward, backward, and central differences.](image-url)
Derivation from Taylor series

The Taylor series expansion of \( u \) about \( x_i \) is given by

\[
u(x) = u(x_i) + (x - x_i) \left( \frac{\partial u}{\partial x} \right)_i + \frac{(x - x_i)^2}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i + \frac{(x - x_i)^3}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \cdots + \frac{(x - x_i)^n}{n!} \left( \frac{\partial^n u}{\partial x^n} \right)_i + \cdots
\]

Using this expansion at the neighbor points, we find that

\[
u(x_{i\pm 1}) = u(x_i) \pm \Delta x \left( \frac{\partial u}{\partial x} \right)_i + \frac{(\Delta x)^2}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i \pm \frac{\Delta x^3}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \cdots
\]
Truncation errors

Forward difference

\[
\left( \frac{\partial u}{\partial x} \right)_i = \frac{u_{i+1} - u_i}{\Delta x} - \frac{\Delta x}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i - \frac{(\Delta x)^2}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \ldots
\]

Backward difference

\[
\left( \frac{\partial u}{\partial x} \right)_i = \frac{u_i - u_{i-1}}{\Delta x} + \frac{\Delta x}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i - \frac{(\Delta x)^2}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \ldots
\]

Central difference

\[
\left( \frac{\partial u}{\partial x} \right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} - \frac{(\Delta x)^2}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \ldots
\]
Truncation errors

- The **local truncation error** $\epsilon$ of a finite difference approximation is the neglected remainder of the Taylor series

\[ \epsilon = \alpha_{p+1}(\Delta x)^p + \alpha_{p+2}(\Delta x)^{p+1} + \ldots \]

- The term proportional to $(\Delta x)^p$ is the **leading truncation error** which determines the order of consistency

\[ \epsilon = O(\Delta x)^p \]

- The order of a numerical approximation indicates how fast the error is reduced when the mesh is refined
- It does not, however, provide any information about the accuracy of the approximation on a given mesh
Second derivative

- Taylor series expansion about $x_i$

\[
\begin{align*}
  u(x_{i+1}) &= u(x_i) + \Delta x \left( \frac{\partial u}{\partial x} \right)_i + \frac{\Delta x^2}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i + \frac{\Delta x^3}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \ldots \\
  u(x_{i-1}) &= u(x_i) - \Delta x \left( \frac{\partial u}{\partial x} \right)_i + \frac{\Delta x^2}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i - \frac{\Delta x^3}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \ldots
\end{align*}
\]

- Central difference approximation

\[
\left( \frac{\partial^2 u}{\partial x^2} \right)_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + \mathcal{O}(\Delta x)^2
\]
Second derivative

- **Approximation in terms of first-order divided differences**

\[
\frac{\partial^2 u}{\partial x^2} = \frac{\partial}{\partial x} \left( \frac{\partial u}{\partial x} \right) \Rightarrow \left( \frac{\partial^2 u}{\partial x^2} \right)_i \approx \frac{\left( \frac{\partial u}{\partial x} \right)_{i+1/2} - \left( \frac{\partial u}{\partial x} \right)_{i-1/2}}{\Delta x}
\]

- **Central difference approximation at the points \( x_{i \pm 1/2} \)**

\[
\left( \frac{\partial^2 u}{\partial x^2} \right)_i \approx \frac{u_{i+1} - u_i}{\Delta x} - \frac{u_i - u_{i-1}}{\Delta x} = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}
\]
Diffusive fluxes

- Approximation in terms of first-order divided differences

\[
f \left( x, \frac{\partial u}{\partial x} \right) = a(x) \frac{\partial u}{\partial x} \implies \left( \frac{\partial f}{\partial x} \right)_i \approx \frac{f_{i+1/2} - f_{i-1/2}}{\Delta x}
\]

- Central difference approximation at the points \( x_{i \pm 1/2} \)

\[
\left( \frac{\partial f}{\partial x} \right)_i \approx \frac{a_{i+1/2} \frac{u_{i+1} - u_i}{\Delta x} - a_{i-1/2} \frac{u_i - u_{i-1}}{\Delta x}}{\Delta x} = \frac{a_{i+1/2} u_{i+1} - (a_{i+1/2} + a_{i-1/2}) u_i + a_{i-1/2} u_{i-1}}{(\Delta x)^2}
\]
One-sided approximations

- No left or right neighbors at the boundary points $x_0 = a, x_N = b$

First-order forward difference

$$\left( \frac{\partial u}{\partial x} \right)_0 = \frac{u_1 - u_0}{\Delta x} + \mathcal{O}(\Delta x)$$

- How can we construct higher-order one-sided approximations?
Polynomial fitting

- Taylor expansion about the boundary point $x = 0$
  \[ u(x) = u(0) + x \left( \frac{\partial u}{\partial x} \right)_0 + \frac{x^2}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_0 + \frac{x^3}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_0 + \ldots \]

- Approximation by a polynomial and differentiation
  \[ u(x) \approx a + bx + cx^2, \quad \frac{\partial u}{\partial x} \approx b + 2cx, \quad \left( \frac{\partial u}{\partial x} \right)_0 \approx b \]

\[
\begin{cases}
  u_0 = a \\
  u_1 = a + b\Delta x + c(\Delta x)^2 \\
  u_2 = a + 2b\Delta x + 4c(\Delta x)^2
\end{cases}
\]

\[ c(\Delta x)^2 = u_1 - u_0 - b\Delta x \]

b = \frac{-3u_0 + 4u_1 - u_2}{2\Delta x}
Error analysis

Consider a one-sided difference approximation of the form

\[
\left( \frac{\partial u}{\partial x} \right)_i \approx \frac{\alpha u_i + \beta u_{i+1} + \gamma u_{i+2}}{\Delta x}, \quad \alpha, \beta, \gamma \in \mathbb{R}
\]

To prove consistency and derive the local truncation error, substitute the Taylor series expansions

\[
u_{i+1} = u_i + \Delta x \left( \frac{\partial u}{\partial x} \right)_i + \frac{(\Delta x)^2}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i + \frac{(\Delta x)^3}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \ldots
\]

\[
u_{i+2} = u_i + 2\Delta x \left( \frac{\partial u}{\partial x} \right)_i + \frac{(2\Delta x)^2}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i + \frac{(2\Delta x)^3}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \ldots
\]
Error analysis

Substitution of the Taylor series expansions yields

\[
\frac{\alpha u_i + \beta u_{i+1} + \gamma u_{i+2}}{\Delta x} = \frac{\alpha + \beta + \gamma}{\Delta x} u_i + (\beta + 2\gamma) \left( \frac{\partial u}{\partial x} \right)_i
\]

\[
+ \frac{\Delta x}{2} (\beta + 4\gamma) \left( \frac{\partial^2 u}{\partial x^2} \right)_i + \mathcal{O}(\Delta x)^2
\]

The approximation is second-order accurate if

\[
\alpha + \beta + \gamma = 0, \quad \beta + 2\gamma = 1, \quad \beta + 4\gamma = 0
\]

It follows that

\[
\left( \frac{\partial u}{\partial x} \right)_i = \frac{-3u_i + 4u_{i+1} - u_{i+2}}{2\Delta x} + \mathcal{O}(\Delta x)^2
\]
Consider a one-sided difference approximation of the form

\[
\left( \frac{\partial^2 u}{\partial x^2} \right)_i \approx \frac{\alpha u_i + \beta u_{i+1} + \gamma u_{i+2}}{(\Delta x)^2}, \quad \alpha, \beta, \gamma \in \mathbb{R}
\]

To prove consistency and derive the local truncation error, substitute the Taylor series expansions

\[
u_{i+1} = u_i + \Delta x \left( \frac{\partial u}{\partial x} \right)_i + \frac{(\Delta x)^2}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i + \frac{(\Delta x)^3}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \ldots
\]

\[
u_{i+2} = u_i + 2\Delta x \left( \frac{\partial u}{\partial x} \right)_i + \frac{(2\Delta x)^2}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i + \frac{(2\Delta x)^3}{6} \left( \frac{\partial^3 u}{\partial x^3} \right)_i + \ldots
\]
Error analysis

Substitution of the Taylor series expansions yields

\[
\frac{\alpha u_i + \beta u_{i+1} + \gamma u_{i+2}}{(\Delta x)^2} = \frac{\alpha + \beta + \gamma}{(\Delta x)^2} u_i + \frac{\beta + 2\gamma}{\Delta x} \left( \frac{\partial u}{\partial x} \right)_i + \frac{\beta + 4\gamma}{2} \left( \frac{\partial^2 u}{\partial x^2} \right)_i + O(\Delta x)
\]

The approximation is first-order accurate if

\[
\alpha + \beta + \gamma = 0, \quad \beta + 2\gamma = 0, \quad \beta + 4\gamma = 2
\]

It follows that

\[
\left( \frac{\partial^2 u}{\partial x^2} \right)_i = \frac{u_i - 2u_{i+1} + u_{i+2}}{(\Delta x)^2} + O(\Delta x)
\]
Higher-order approximations

- Third-order forward difference / 1st derivative

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

- Third-order backward difference / 1st derivative

\[
\left( \frac{\partial u}{\partial x} \right)_i = \frac{2u_{i+1} + 3u_i - 6u_{i-1} + u_{i-2}}{6\Delta x} + O(\Delta x)^3
\]
Higher-order approximations

- Fourth-order central difference / 1st derivative

\[
\left( \frac{\partial u}{\partial x} \right)_i = \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x} + O(\Delta x)^4
\]

- Fourth-order central difference / 2nd derivative

\[
\left( \frac{\partial^2 u}{\partial x^2} \right)_i = \frac{-u_{i+2} + 16u_{i+1} - 30u_{i} + 16u_{i-1} - u_{i-2}}{12(\Delta x)^2} + O(\Delta x)^4
\]
To achieve higher order accuracy, we need more grid points.

One-sided high-order approximations must be used at boundary points; the implementation becomes more involved.

The larger number of unknowns per equation requires more memory and increases the cost of solving linear systems.

Desired accuracy can be attained on coarser meshes.

Second-order approximations are usually optimal for CFD.
Poisson equation in 1D

Dirichlet problem

\[
\begin{align*}
-\frac{\partial^2 u}{\partial x^2} &= f \quad \text{in } \Omega = (0, 1) \\
u(0) &= 0, \quad u(1) = 0
\end{align*}
\]

Physical interpretation: \( u(x) \) is the displacement of a suspension bridge, 
\( f(x) \) is the load at point \( x \in \Omega \)

Maximum principle: 
\( f(x) \leq 0, \ \forall x \quad \Rightarrow \quad u(x) \leq 0, \ \forall x \)

Discretization: 
\( x_i = i \Delta x, \quad \text{where} \quad \Delta x = \frac{1}{N} \)

\( u_i \approx u(x_i), \quad f_i = f(x_i) \)
Poisson equation in 1D

- Approximation by central differences

\[
\begin{align*}
\left\{
-\frac{u_{i-1} - 2u_i + u_{i+1}}{(\Delta x)^2} &= f_i, \quad i = 1, \ldots, N - 1 \\
u_0 &= u_N = 0 \quad \text{(boundary conditions)}
\end{align*}
\]

- Linear algebraic system

\[
\begin{align*}
\left\{
\begin{array}{l}
i = 1 \\
i = 2 \\
i = 3 \\
i = N - 1
\end{array}
\begin{array}{l}
-\frac{u_0 - 2u_1 + u_2}{(\Delta x)^2} \\
-\frac{u_1 - 2u_2 + u_3}{(\Delta x)^2} \\
-\frac{u_2 - 2u_3 + u_4}{(\Delta x)^2} \\
\frac{u_{N-2} - 2u_{N-1} + u_N}{(\Delta x)^2}
\end{array}
= f_1 \\
= f_2 \\
= f_3 \\
= f_{N-1}
\end{align*}
\]
The matrix form of the linear system is given by

$$Au = f, \quad A \in \mathbb{R}^{(N-1) \times (N-1)}, \quad u, f \in \mathbb{R}^{N-1}$$

where $A$ is tridiagonal, symmetric positive-definite

$$A = \frac{1}{(\Delta x)^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 \end{pmatrix}$$

The discrete problem is well-posed (the solution $u = A^{-1}f$ exists and is unique since the matrix $A$ is invertible)

There are efficient direct solvers for tridiagonal matrices
Thomas algorithm

Tridiagonal matrix algorithm (TDMA)

\[ a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \quad i = 1, \ldots, n \]
\[ a_1 = 0, \quad c_n = 0 \]

Matrix form of the linear system

\[
\begin{pmatrix}
  b_1 & c_1 \\
  a_2 & b_2 & c_2 \\
  a_3 & b_3 & c_3 \\
  \vdots & \ddots & \ddots \\
  a_n & b_n
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  \vdots \\
  x_n
\end{pmatrix}
= 
\begin{pmatrix}
  d_1 \\
  d_2 \\
  d_3 \\
  \vdots \\
  d_n
\end{pmatrix}
\]
Thomas algorithm

Fast Gaussian elimination for tridiagonal systems:

**Forward sweep**

\[
\text{for } k = 2, \ldots, n \text{ do} \\
\quad b_k = b_k - a_k \frac{c_{k-1}}{b_{k-1}} \\
\quad d_k = d_k - a_k \frac{d_{k-1}}{b_{k-1}} \\
\text{end } k\text{-loop}
\]

**Backward sweep**

\[
\begin{align*}
\quad x_n &= \frac{d_n}{b_n} \\
\text{for } k = n - 1, \ldots, 1 \text{ do} \\
\quad x_k &= \frac{d_k - c_k x_{k+1}}{b_k} \\
\text{end } k\text{-loop}
\end{align*}
\]

Computational cost: just $O(n)$ arithmetic operations instead of $O(n^3)$