Computational practicum: Lecture 2

Numerical methods for partial differential equations (Part 1)

Zoë J.G. Gromotka, Artur M. Schweidtmann, Ferdinand Grozema, Tanuj Karia

With support from Giacomo Lastrucci

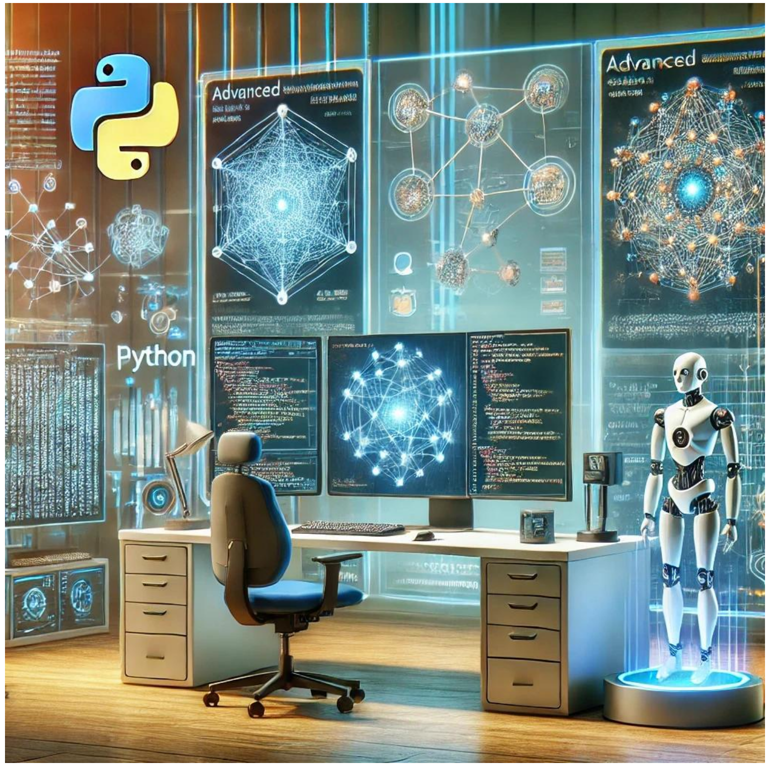
Computational Practicum
Dept. Chemical Engineering
Delft University of Technology

TU Delft   Process Intelligence RESEARCH   Delft Institute of Applied Mathematics

Wednesday, 19th November 2025

Recap last lecture

- Advanced Python programming:
  - Programming principles
  - Managing imports, packages, virtual environments
  - Managing multiple modules
  - Basic object-oriented programming (OOP)
  - Unit testing

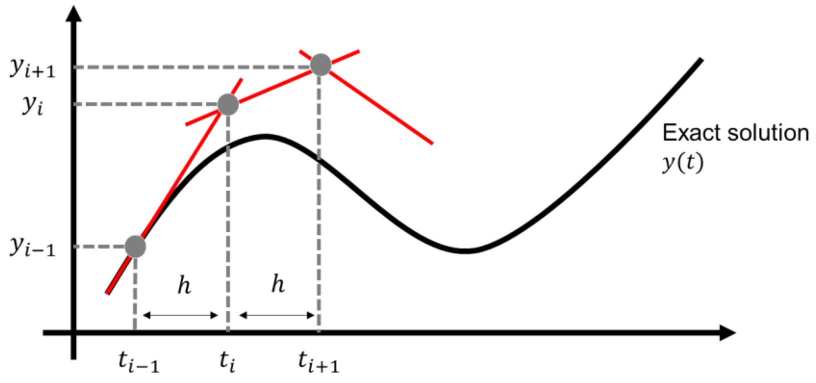


Recap from Q1: relevant concepts

Recap

Initial value problem (IVP)

- An ODE (system) with given initial conditions is called an initial value problem (IVP)
- Numerical solution methods
  - Forward Euler



  - Backward Euler

Boundary value problems (BVPs)

- BVPs have side constraints at more than one point
- Boundary conditions define side constraints, e.g.,
  - Dirichlet boundary condition $y = f$
  - Neumann boundary condition $\frac{\mathrm{d}y}{dx} = f$
  - …

Finite difference method

- Differential equation → Finite difference equations for all nodes of a discretized domain

Learning objectives

After successfully completing this lecture, you are able to…

- Explain what a partial differential equation (PDEs) is, how it can be classified, and how it is different form ordinary differential equations (ODEs).
- Give an overview of the main techniques to solve partial differential equations in (chemical) engineering
- Implement different numerical solution approaches for parabolic PDEs from scratch
- Discuss stability of numerical solution approaches for parabolic PDEs
- Use Python libraries' built-in functions to support the solution of parabolic PDEs

Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE’s (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE’s (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Partial Differential Equations: what are?

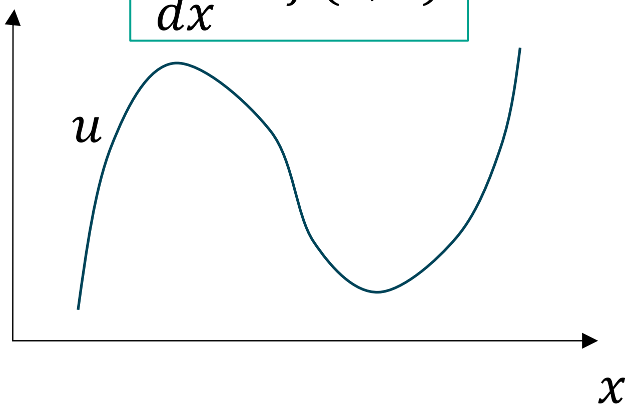
See CP Q1
Lecture 2

- Recap: Partial derivative

  Consider a function $f: D \to \mathbb{R}$, $D \subseteq \mathbb{R}^n$ with independent variables $\boldsymbol{x} = [x_1, \ldots, x_n]$. The **partial derivate** of f with respect to a variable $x_j \in \boldsymbol{x}$ is denoted by $\frac{\partial f}{\partial x_j}$.

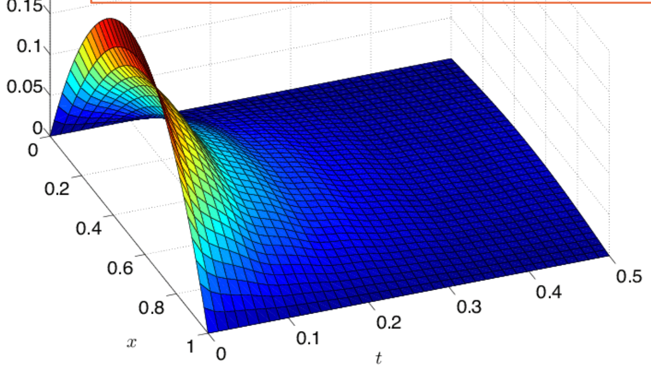
- A partial differential equation (PDE) is an equation involving partial derivatives of an unknown function with respect to more than one independent variable.

$$\boxed{\frac{du}{dx} = f(x,u)}$$



ODE vs. PDE

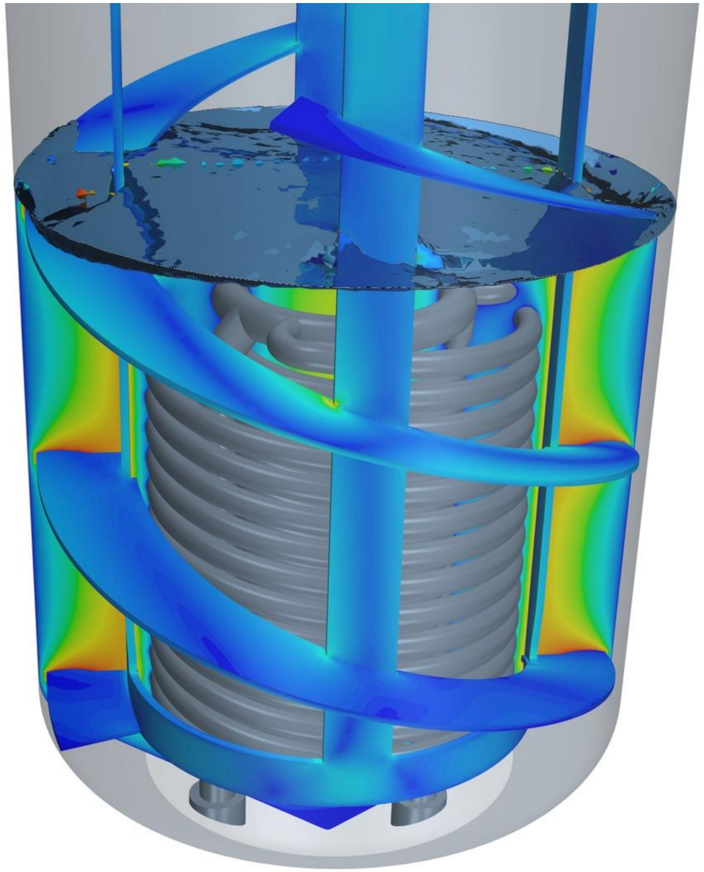
$$\boxed{\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} + f(t,x,u)}$$



Partial Differential Equation: why?

- PDEs are fundamental in modeling continuous phenomena in nature and Chemical Engineering.

- Example of applications are:
  - Unsteady processes (time and space derivatives)
  - Heat/mass transfer in reactors
  - Mass transfer in separation processes
  - Navier-Stokes equations (system of PDEs)



Partial Differential Equation: scope, aim and notation

- Solving a PDE means to search for a function $u\colon \mathbb{R}^2 \to \mathbb{R}$ (or $u\colon \mathbb{R}^3 \to \mathbb{R}$) that:
  - Satisfies the relationship prescribed by a given PDE on a specified domain
  - Meets the imposed initial and/or boundary conditions
  - In case of two independent variables, the solution is a bivariate function u and can be visualized as a surface over the 2D domain (x, t) or (x, y).
- Alternative notations you may find in books and publications:

$$\frac{\partial u}{\partial x} = u_x, \frac{\partial^2 u}{\partial x^2} = u_{xx}, \frac{\partial^2 u}{\partial x \partial y} = u_{xy} = u_{yx} = \frac{\partial^2 u}{\partial y \partial x}, \frac{\partial u}{\partial t} = u_t = \dot{u}$$

Partial Differential Equation: scope, aim and notation

- We will deal with single PDEs (systems will not be analysed)
- PDE Lecture 1
  - Two independent variables: 1D space and time *(x, t)* → Parabolic PDEs
- PDE Lecture 2:
  - Two independent variables: 2D space *(x, y)* → Elliptic PDEs

Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE’s (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Partial Differential Equation: classification

- The *order* of a PDE is determined by the highest-order partial derivative in the PDE.
- A PDE is *linear* if the dependent variable and its partial derivatives appear only linearly (only degree one, no products/nonlinear functions), otherwise it is *non-linear*
- Given an example of PDE:

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} + f(x, y) \cdot u + g(x, y, u) = 0$$

  - u is the dependent variable
  - x and y are the independent variables
  - $f(x, y) \cdot u$ is a linear function of u
  - $g(x, y, u)$ is a nonlinear function. If $g(x, y, u) = 0$, then the PDE above is linear

Partial Differential Equation: classification

A classification is available for bivariate second order PDEs:

- The general form of second order linear PDEs is:

$$a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} + d\frac{\partial u}{\partial x} + e\frac{\partial u}{\partial y} + fu + g = 0$$

- We define the *discriminant* as the quantity $\mathrm{D} = b^2 - 4ac$, then:
  - D > 0: Hyperbolic PDE
  - D = 0: Parabolic PDE
  - D < 0: Elliptic PDE
- In this course, we cover solution methods for parabolic and elliptic PDEs.

HINT

The independent variables x and y in the definition are generic. Other can be considered (e.g., x and t).

Introduction to parabolic PDE

- Parabolic PDEs describe time-dependent, dissipative physical processes, like diffusion, which evolve toward a steady state.
  - The two independent variables are time and 1D space. Generally, we refer to as 1D parabolic PDE (referring only to spatial coordinates).
- An example is the 1D heat equation:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} + S(u, x, t), \qquad 0 \leq x \leq L, \qquad t \geq 0$$

- The heat equation models the propagation of heat in a body through conduction:
  - u is the temperature (or concentration, etc.)
  - α is the thermal diffusivity $\alpha = \frac{k}{c_p \rho}$
  - $S(u, x, t)$ is the source term, potentially nonlinear

Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE's (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Initial and boundary conditions

- As you may recall from Q1, differential equations are satisfied by infinitely many solutions. Additional conditions must be specified, depending on the problem, to characterize them fully.
  - Initial conditions: quantity specified for the solution of a PDE at the beginning of the time interval.
  - Boundary conditions: quantity specified for the solution of a PDE (or its derivatives) along the boundaries of the spatial domain.

- In general, you need a condition for each order of derivative appearing in the equation
  - $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 y}{\partial x^2}$ → 1 IC, 2 BC
  - $\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2}$ → 2 IC, 2 BC

Initial and boundary conditions for parabolic PDEs

For parabolic PDEs:

- 1D parabolic PDE: $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} + S(u, x, t)$
  - First order time derivative → 1 initial condition
  - 1x second order spatial derivatives → 2 boundary conditions

Recap of different boundary conditions

- There are numerous possibilities for the boundary conditions (BCs) that must be specified on the domain boundaries:
  - Dirichlet boundary conditions (*essential* BCs): the solution u is specified.

  For instance: $u(x=0)=\bar{u}$

  - Neumann boundary conditions (*natural* BCs): one of the derivatives u_x or u_y is specified.

  For instance: $\frac{\partial u}{\partial x}|_{x=L}=0$

  - Robin boundary conditions (*mixed* boundary conditions): a combination of the previous conditions is specified.

  For instance: $\alpha u(x=0)+\beta\frac{\partial u}{\partial x}|_{x=0}=q$

- We will mainly deal with Dirichlet and Neumann boundary conditions.

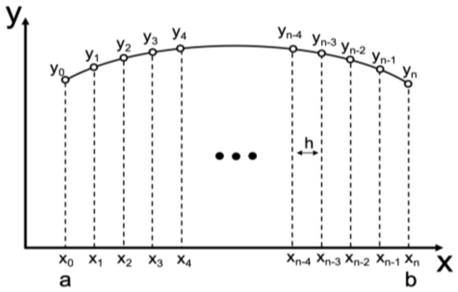
Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE's (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Numerical methods for solving PDEs

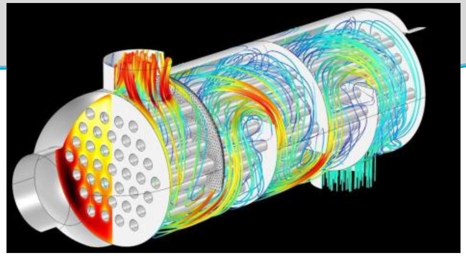
Finite Difference Method (FDM)

- The finite difference method approximates derivatives by discretizing the domain over grid points.
- Applied in mass and heat transfer problems in simple domains.



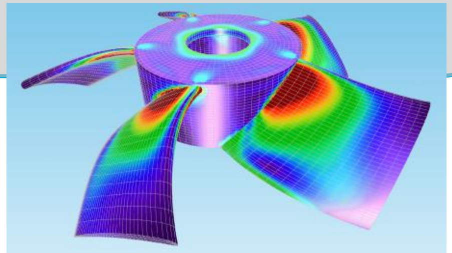
Finite Volume Method (FVM)

- The finite volume method conserves quantities by integrating over control volumes
- Widely used in computational fluid dynamics (CFD)



Finite Element Method (FEM)

- The finite element method divides the domain into elements and uses variational methods to solve.
- Ideal for structural analysis and stress distribution in materials engineering.

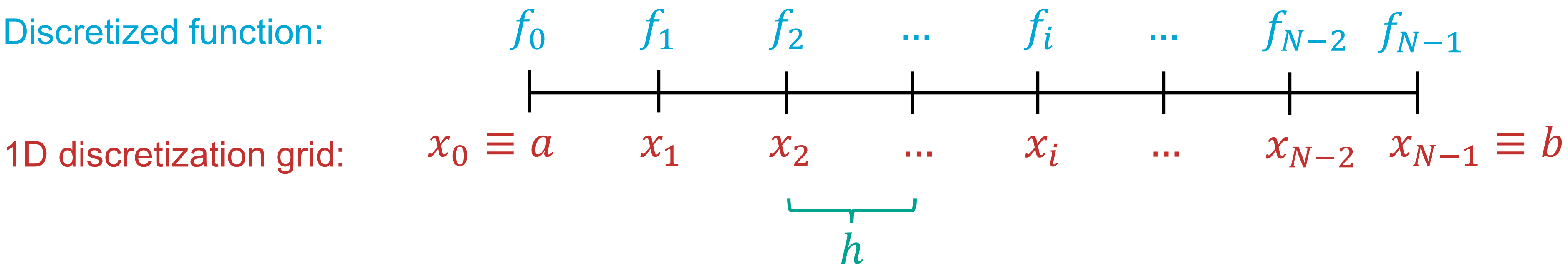


Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE’s (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

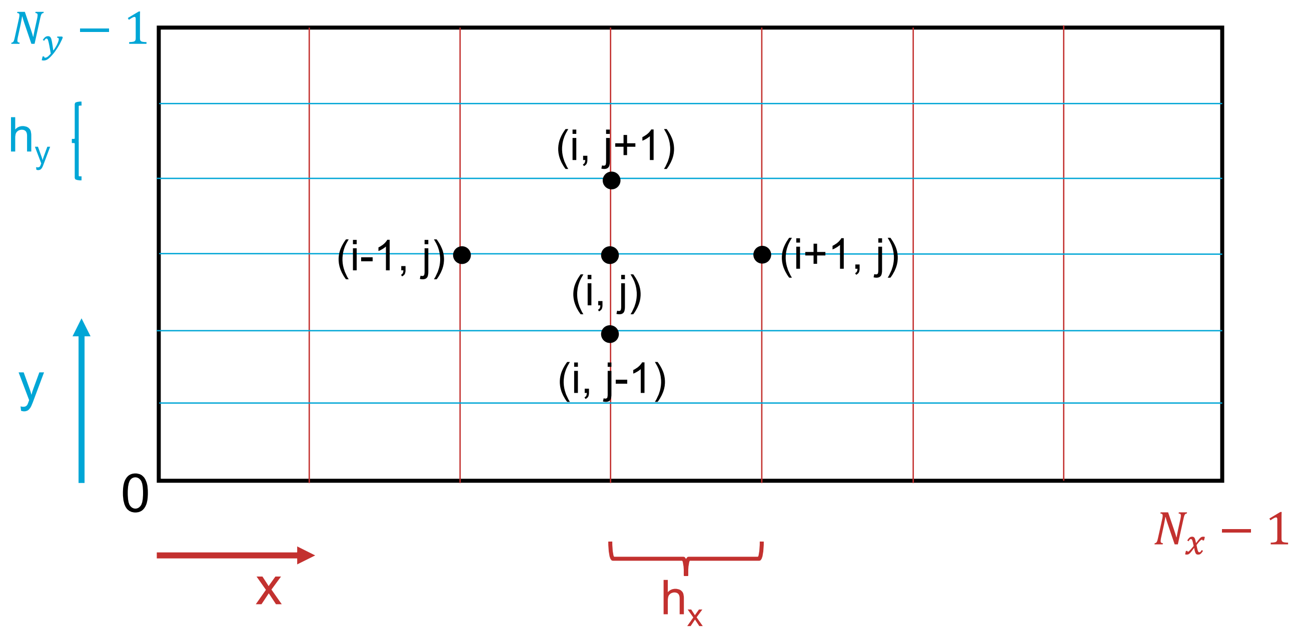
Finite difference method for PDEs: Introduction

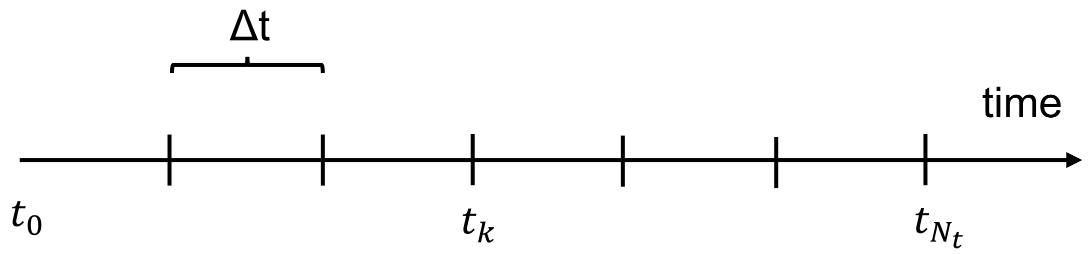
- The finite difference method (FDM) approximates derivatives in partial differential equations (PDEs) using differences between function values at discrete grid points.
  - Continuous domain → grid of discrete points
  - Partial derivatives → finite differences
  - System of algebraic equations → iterative numerical methods



Discretize multivariate domains

- By definition, PDEs involve multiple independent variables:
  - Space (x, y, z)
  - Time (t)
  - Possibly other variables
- Every variable will have its own (possibly different) discretization scheme
  - In general: $h_x \neq h_y$
  - Often: $h_x = h_y$
  - The choice of time step Δt depends on the stability of the system

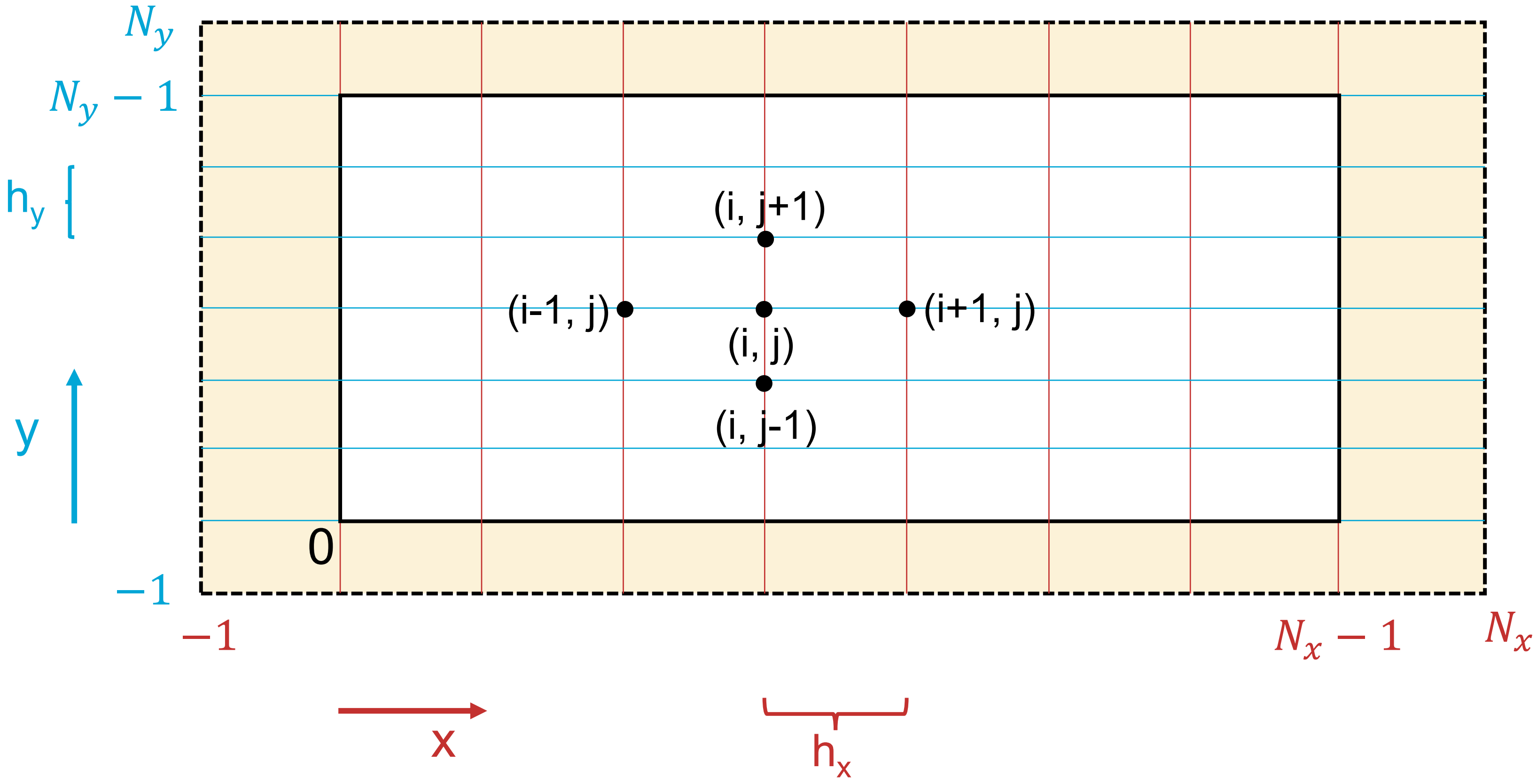




Discretization with ghost points

Recap from CP Q1L6

- Artificial (ghost) points can be added as a padding around the domain (for each coordinate: 2 points, 2 intervals)
- Useful to discretize boundary conditions with more accurate schemes



Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE’s (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Heat transport problem

- Microscopic governing equations for transport phenomena are typically modelled by partial differential equations, describing how quantities as concentration, velocity or temperature evolve in space and time.
- We consider the case of heat transport problem as a case study for PDEs.
- More specifically:
  - Unsteady heat transport in 1D (e.g., tube wall)
    → Parabolic PDE (this lecture)
  - Steady state heat transport in 2D (e.g., plate heat exchanger)
    → Elliptic PDE (next lecture)

Parabolic PDEs: chemical engineering applications

- We explore the solution method for parabolic PDEs, exemplified by the heat equation:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

The equation describes the transient temperature propagation along a single-dimensional space.

- Several other chemical engineering problems can be modelled as parabolic PDEs.
  - Mass diffusion equation

$$\frac{\partial C}{\partial t} = D \frac{\partial^2 C}{\partial x^2}$$

  - Concentration profiles in catalyst pellets

$$\frac{\partial C_A}{\partial t} = D \frac{\partial^2 C_A}{\partial r^2} - r_A(C_A)$$

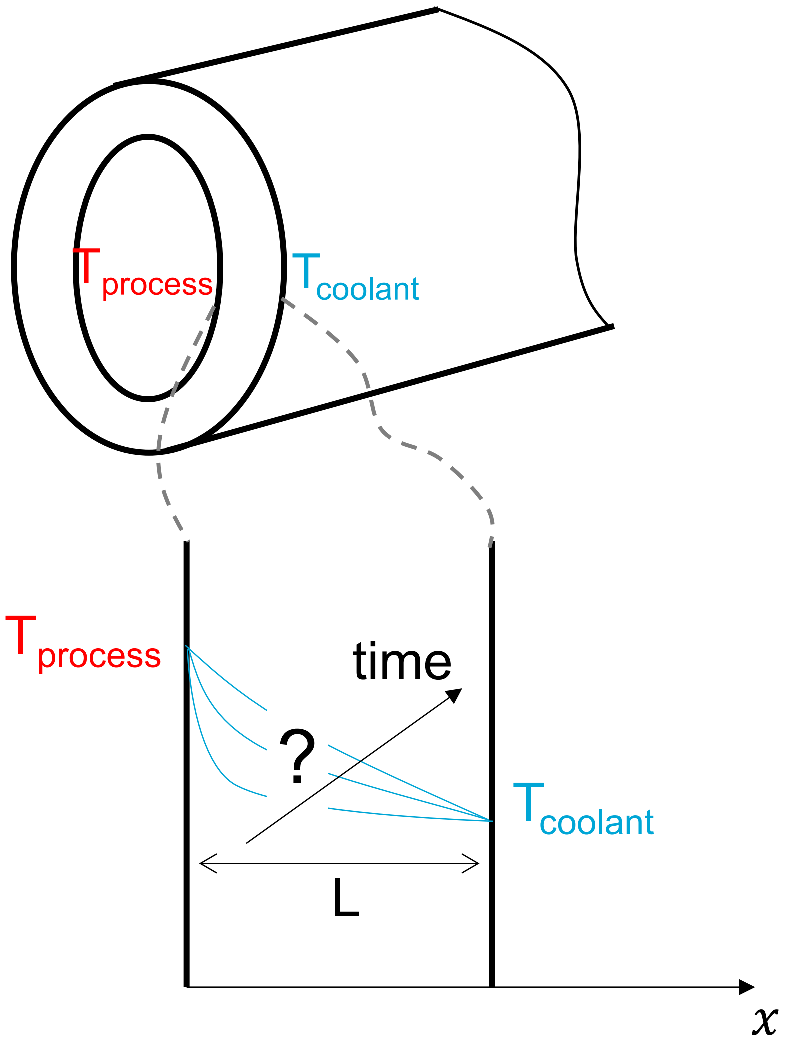
Possibly nonlinear PDE!

Case study for parabolic PDEs

- 1D Heat Equation:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

- Consider heat transfer across the wall of a tube in a shell-and-tube heat exchanger:
  - The temperatures of the process fluid (tube side) and coolant (shell side) are known.
  - Initially, the tube wall is in thermal equilibrium with the coolant.
  - How does the temperature evolve over time and across the wall?



Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE's (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Semidiscrete Method: PDE to ODE system

- A way to approximate the solution of transient PDEs is to initially discretize in space and leave the time variable continuous
- Introduce spatial mesh points x_i , $i = 0, \dots, N_x - 1$, where i is the discretization index and Δx the interval
- Discretize the second order spatial derivative by finite differences (internal points):

$$\frac{d^2T}{dx^2} \approx \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}$$

Note: $T_i = T_i(t)$ is time dependent

- The equation results in a system of ODEs:

$$\frac{dT_i}{dt} = \alpha \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}, \qquad i = 1, \dots, N_x - 2$$

- With initial and boundary condition:

$$T_i(0) = T_{coolant} \quad i = 1, \dots, N_x - 2$$
$$T_0(t) = T_{process} \quad T_{N_x - 1}(t) = T_{coolant} \quad \forall t$$

Semidiscrete Method: PDE to ODE system

- In matrix form:

$$\frac{d}{dt}\boldsymbol{T} = \frac{\alpha}{\Delta x^2} A\boldsymbol{T}_{(complete)}$$

- Here, $\frac{\alpha}{\Delta x^2}$ is a scalar constant and A is a matrix
- Note that the vector $\boldsymbol{T}$ is different from the vector $\boldsymbol{T}_{(complete)}$, this is directly arising from the indices previously shown:

boundary points included

only internal points

$$\frac{d}{dt}\begin{bmatrix} T_1 \\ \vdots \\ T_{N_x-2} \end{bmatrix} = \mathrm{A}\begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{N_x-2} \\ T_{N_x-1} \end{bmatrix}$$

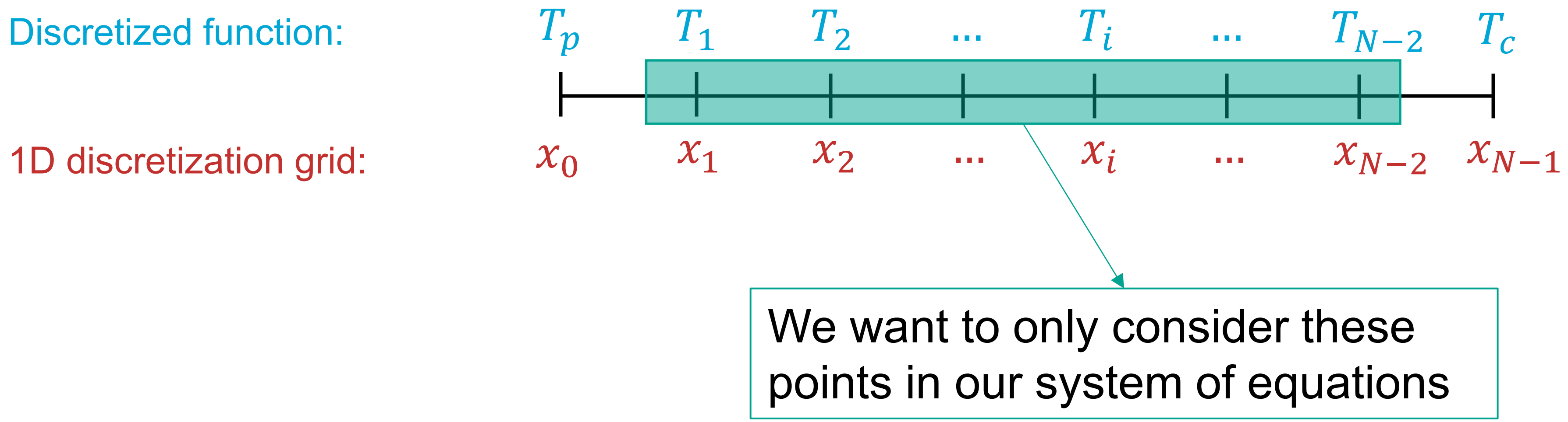
- Hence, the matrix A is rectangular ($N_x - 2 \times N_x$), thus the system is under determined.
- Two more equations are needed! → Boundary conditions!!

Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE’s (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - \- Internal points
    - \- Handling boundary conditions
  - Time derivatives
    - \- Forward Euler
    - \- Stability
    - \- Backward Euler
  - Built-in IVP solvers

Dirichlet boundary conditions

- Remember Q1 Lecture 6 – BVP with FDM:
  We want to only use the internal points of the domain grid (vector T) to discretize the boundary conditions.
- Consider Dirichlet boundary conditions as in the stated problem: $T_0 = T_p$ and $T_{N_x-1} = T_c$



Dirichlet boundary conditions

- We can develop specialized equations for T_1 and T_{N_x-2}:

$i = 1$:

$$\frac{dT_1}{dt} = \frac{\alpha}{\Delta x^2}(T_0 - 2T_1 + T_2)$$

…but $T_0(t) = T_p \quad \forall$t, then:

$$\frac{dT_1}{dt} = \frac{\alpha}{\Delta x^2}(T_p - 2T_1 + T_2)$$

$$\frac{dT_1}{dt} = \frac{\alpha}{\Delta x^2}(-2T_1 + T_2) + \frac{\alpha}{\Delta x^2}T_p$$

Dirichlet boundary conditions

- We can develop specialized equations for T_1 and T_{N_x-2}:

$i = N_x - 2$:

$$\frac{dT_{N_x-2}}{dt} = \frac{\alpha}{\Delta x^2}\left(T_{N_x-3} - 2T_{N_x-2} + T_{N_x-1}\right)$$

…but $T_{N_x-1}(t) = T_c \quad \forall t$, then:

$$\frac{dT_{N_x-2}}{dt} = \frac{\alpha}{\Delta x^2}\left(T_{N_x-3} - 2T_{N_x-2} + T_c\right)$$

$$\frac{dT_{N_x-2}}{dt} = \frac{\alpha}{\Delta x^2}\left(T_{N_x-3} - 2T_{N_x-2}\right) + \frac{\alpha}{\Delta x^2}T_c$$

System of equations with Dirichlet boundary conditions

- The final system of ODEs given Dirichlet boundary conditions is

algebraic equations

$$\begin{cases} T_0(t) = T_p \\ \dfrac{dT_i}{dt} = \dfrac{\alpha}{\Delta x^2}(T_{i-1} - 2T_i + T_{i+1}) & \forall t \text{ and } i = 1, \dots, N_x - 2 \\ T_{N_x-1}(t) = T_c \end{cases}$$

$N_x - 2$ ordinary differential equations

- The initial condition is given by

$$T_i(0) = T_c \qquad i = 1, \dots, N_{x-2}$$

Matrix form with Dirichlet boundary conditions

- The system of ODEs given Dirichlet boundary conditions can be written in matrix form, including a vector with the initial condition

$$\frac{d}{dt}\boldsymbol{T} = \frac{\alpha}{\Delta x^2}\cdot(A\boldsymbol{T} + \boldsymbol{b}), \qquad \boldsymbol{T}(0) = \boldsymbol{T}_c,$$

$$\frac{d}{dt}\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix} = \frac{\alpha}{\Delta x^2}\begin{bmatrix} -2 & 1 & 0 & \dots & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix} + \frac{\alpha}{\Delta x^2}\begin{bmatrix} \mathrm{T_p} \\ 0 \\ \vdots \\ 0 \\ \mathrm{T_c} \end{bmatrix}, \qquad \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix}(0) = \begin{bmatrix} T_c \\ T_c \\ \vdots \\ T_c \\ T_c \end{bmatrix}.$$

$$A \in \mathbb{R}^{N_x-2\times N_x-2} \qquad \boldsymbol{T},\boldsymbol{T}_c \in \mathbb{R}^{N_x-2} \qquad \boldsymbol{b} \in \mathbb{R}^{N_x-2}$$

Neumann boundary conditions

- For completeness, consider now the case of Neumann boundary conditions:

$$\left.\frac{\partial T}{\partial x}\right|_{x=0} = \gamma_1$$
$$\left.\frac{\partial T}{\partial x}\right|_{x=L} = \gamma_2$$

- We aim to derive an updating rule in matrix form, including Neumann boundary conditions for which the derivative is constant on the boundaries.
- We aim to use only internal discretization points ($i = 2, \dots, N_x - 3$)
- Discretization scheme:
  - First order derivative in $x = 0$ → <u>Forward</u> difference scheme
  - First order derivative in $x = \mathrm{L}$ → <u>Backward</u> difference scheme
  - Central difference scheme is possible with ghost points! (discuss this later…)

Neumann boundary conditions

- We can develop specialized equations for T_1 and T_{N_x-2}:

$i = 1$:

Forward difference scheme: $\frac{\partial T}{\partial x}|_{x=0} = \gamma_1 \quad \rightarrow \quad \frac{T_1 - T_0}{\Delta x} = \gamma_1 \rightarrow T_0 = T_1 - \gamma_1 \Delta x$

Updating equation:

$$\frac{dT_1}{dt} = \frac{\alpha}{\Delta x^2}(T_0 - 2T_1 + T_2)$$

Substituting the equation arising from the boundary condition:

$$\frac{dT_1}{dt} = \frac{\alpha}{\Delta x^2}(T_1 - \gamma_1 \Delta x - 2T_1 + T_2)$$

$$\frac{dT_1}{dt} = \frac{\alpha}{\Delta x^2}(-T_1 + T_2) - \frac{\alpha}{\Delta x}\gamma_1$$

Neumann boundary conditions

- We can develop specialized equations for T_1 and T_{N_x-2}:

$i = N_x - 2$:

Backward difference scheme: $\frac{\partial T}{\partial x}|_{x=L} = \gamma_2 \quad \rightarrow \quad \frac{T_{N_x-1} - T_{N_x-2}}{\Delta x} = \gamma_2 \rightarrow T_{N_x-1} = T_{N_x-2} + \gamma_2 \Delta x$

Updating equation:

$$\frac{dT_{N_x-2}}{dt} = \frac{\alpha}{\Delta x^2}\left(T_{N_x-3} - 2T_{N_x-2} + T_{N_x-1}\right)$$

Substituting the equation arising from the boundary condition:

$$\frac{dT_{N_x-2}}{dt} = \frac{\alpha}{\Delta x^2}\left(T_{N_x-3} - 2T_{N_x-2} + T_{N_x-2} + \gamma_2 \Delta x\right)$$

$$\frac{dT_{N_x-2}}{dt} = \frac{\alpha}{\Delta x^2}\left(T_{N_x-3} - T_{N_x-2}\right) + \frac{\alpha}{\Delta x}\gamma_2$$

System of ODEs with Neumann boundary conditions

- The Final system of equations given Neumann boundary conditions is

algebraic equations

$$\begin{cases} T_0(t) = T_1 - \gamma_1 \Delta x \\ \dfrac{dT_i}{dt} = \dfrac{\alpha}{\Delta x^2}\left(T_{i-1} - 2T_i + T_{i+1}\right) \\ T_{N_x-1}(t) = T_{N_x-2} + \gamma_2 \Delta x \end{cases} \qquad \forall t \text{ and } i = 1, \dots, N_x - 2$$

$N_x - 2$ ordinary differential equations

- The initial condition is given by

$$T_i(0) = T_c \qquad i = 1, \dots, N_{x-2}$$

Matrix form with Neumann boundary conditions

- The system of ODEs given Neumann boundary conditions can be written in matrix form, including a vector with the initial condition

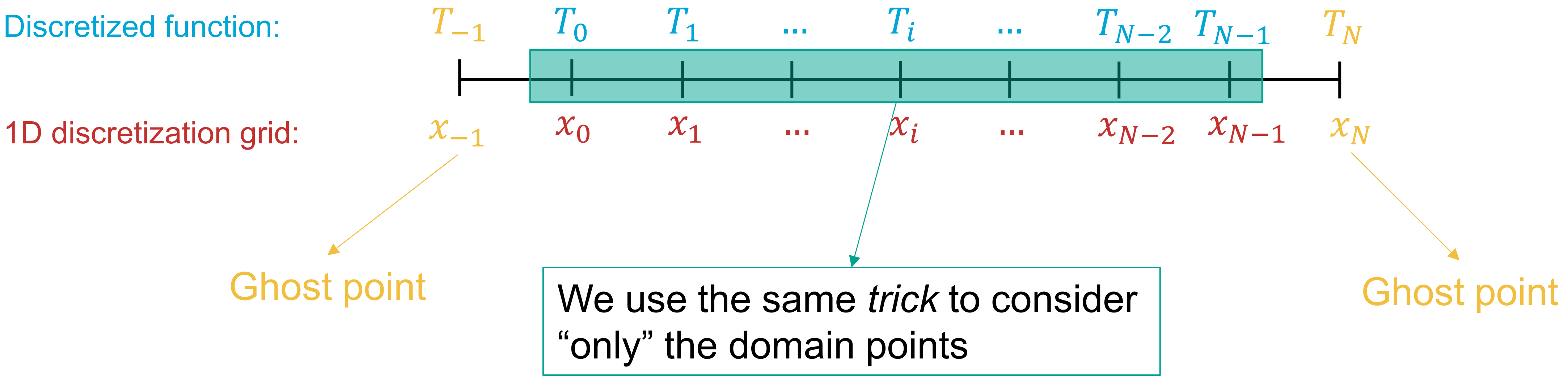
$$\frac{d}{dt}\boldsymbol{T} = \frac{\alpha}{\Delta x^2} \cdot (A\boldsymbol{T} + \boldsymbol{b}), \qquad \boldsymbol{T}(0) = \boldsymbol{T}_c,$$

$$\frac{d}{dt}\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix} = \frac{\alpha}{\Delta x^2}\begin{bmatrix} -1 & 1 & 0 & \dots & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 1 & -1 \end{bmatrix}\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix} + \frac{\alpha}{\Delta x^2}\begin{bmatrix} -\gamma_1 \Delta x \\ 0 \\ \vdots \\ 0 \\ \gamma_2 \Delta x \end{bmatrix}, \qquad \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix}(0) = \begin{bmatrix} T_c \\ T_c \\ \vdots \\ T_c \\ T_c \end{bmatrix}.$$

$$A \in \mathbb{R}^{N_x-2 \times N_x-2} \qquad \boldsymbol{T}, \boldsymbol{T}_c \in \mathbb{R}^{N_x-2} \qquad \boldsymbol{b} \in \mathbb{R}^{N_x-2}$$

Neumann boundary conditions with ghost points

- The concept of ghost points can be used to unlock more <u>accurate discretization schemes</u> for Neumann boundary conditions (i.e., central difference scheme)



Neumann boundary conditions: ghost points

- We can develop specialized equations for T_0 and T_{N_x-1}: (boundary points!)

$i = 0$:

Central difference scheme: $\frac{\partial T}{\partial x}|_{x=0} = \gamma_1 \quad \rightarrow \quad \frac{T_1 - T_{-1}}{2\Delta x} = \gamma_1 \rightarrow T_{-1} = T_1 - 2\gamma_1 \Delta x$

Updating equation:

$$\frac{dT_0}{dt} = \frac{\alpha}{\Delta x^2}(T_{-1} - 2T_0 + T_1)$$

Substituting the equation arising from the boundary condition:

$$\frac{dT_0}{dt} = \frac{\alpha}{\Delta x^2}(T_1 - 2\gamma_1 \Delta x - 2T_0 + T_1)$$

$$\frac{dT_0}{dt} = \frac{\alpha}{\Delta x^2}(-2T_0 + 2T_1) - 2\frac{\alpha}{\Delta x}\gamma_1$$

Neumann boundary conditions: ghost points

- We can develop specialized equations for T_0 and T_{N_x-1}: (boundary points!)

$i = N_x - 1$:

Central difference scheme: $\frac{\partial T}{\partial x}\big|_{x=L} = \gamma_2 \quad \rightarrow \quad \frac{T_{N_x} - T_{N_x-2}}{2\Delta x} = \gamma_2 \rightarrow T_{N_x} = T_{N_x-2} + 2\gamma_2 \Delta x$

Updating equation:

$$\frac{dT_{N_x-1}}{dt} = \frac{\alpha}{\Delta x^2}\left(T_{N_x-2} - 2T_{N_x-1} + T_{N_x}\right)$$

Substituting the equation arising from the boundary condition:

$$\frac{dT_{N_x-1}}{dt} = \frac{\alpha}{\Delta x^2}\left(T_{N_x-2} - 2T_{N_x-1} + T_{N_x-2} + 2\gamma_2 \Delta x\right)$$

$$\frac{dT_{N_x-1}}{dt} = \frac{\alpha}{\Delta x^2}\left(2T_{N_x-2} - 2T_{N_x-1}\right) + 2\frac{\alpha}{\Delta x}\gamma_2$$

System of equations with Neumann bcs

- The final system of ODEs given Neumann boundary conditions with ghost points is

algebraic equations

$$\begin{cases} T_{-1}(t) = T_1 - 2\gamma_1 \Delta x \\ \dfrac{dT_i}{dt} = \dfrac{\alpha}{\Delta x^2}(T_{i-1} - 2T_i + T_{i+1}) \\ T_{N_x}(t) = T_{N_x-2} + 2\gamma_2 \Delta x \end{cases} \qquad \forall t \text{ and } i = 0, \dots, N_x - 1$$

N_x ordinary differential equations

- The initial condition is given by

$$T_i(0) = T_c \qquad i = 0, \dots, N_{x-1}$$

Matrix form with Neumann boundary conditions

- The system of ODEs given Neumann boundary conditions with ghost points can be written in matrix form, including a vector with the initial condition

$$\frac{d}{dt}\boldsymbol{T} = \frac{\alpha}{\Delta x^2} \cdot (A\boldsymbol{T} + \boldsymbol{b}), \qquad \boldsymbol{T}(0) = \boldsymbol{T}_c,$$

$$\frac{d}{dt}\begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{N_x-2} \\ T_{N_x-1} \end{bmatrix} = \frac{\alpha}{\Delta x^2}\begin{bmatrix} -2 & 2 & 0 & \dots & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 2 & -2 \end{bmatrix}\begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{N_x-2} \\ T_{N_x-1} \end{bmatrix} + \frac{\alpha}{\Delta x^2}\begin{bmatrix} -2\gamma_1\Delta x \\ 0 \\ \vdots \\ 0 \\ 2\gamma_2\Delta x \end{bmatrix}, \qquad \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{N_x-2} \\ T_{N_x-1} \end{bmatrix}(0) = \begin{bmatrix} T_c \\ T_c \\ \vdots \\ T_c \\ T_c \end{bmatrix}.$$

$$A \in \mathbb{R}^{N_x-1\times N_x-1} \qquad \boldsymbol{T}, \boldsymbol{T}_c \in \mathbb{R}^{N_x-1} \qquad \boldsymbol{b} \in \mathbb{R}^{N_x-1}$$

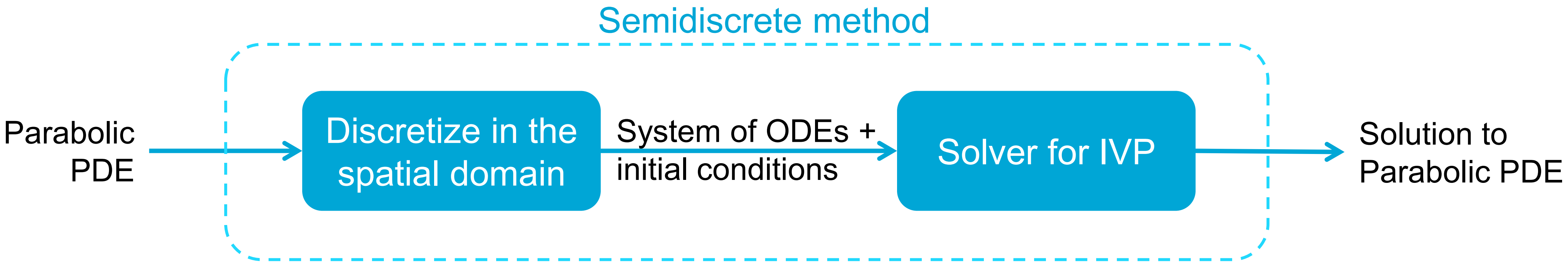
Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE's (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Semidiscrete Method: Solution methods

- The solution of the PDE is translated into solving a system of ODEs with given initial conditions.
- In Q1, you learnt how to solve Initial Value Problems (IVPs) in the context of ODEs:
  - Forward (explicit) Euler
  - Backward (implicit) Euler
  - More advanced schemes (Runge-Kutta)

See CP Q1 Lecture 5



Semidiscrete Method: Forward Euler

- The simplest choice for solving systems of ODEs is to use Forward (explicit) Euler method.
- Discretization of the first order time derivative:

$$\frac{dT_i}{dt} = \frac{T_i^{k+1} - T_i^k}{\Delta t}, \qquad t_k = k \cdot \Delta t$$

- The PDE can be written as:

$$\frac{T_i^{k+1} - T_i^k}{\Delta t} = \alpha \frac{T_{i-1}^k - 2T_i^k + T_{i+1}^k}{\Delta x^2}, \qquad i = 1, \dots, N_x - 2, \qquad k = 0, \dots, N_t - 2$$

- Thus, giving the updating closed form for every <u>internal</u> point: $(\mathrm{Fo} = \alpha \Delta t / \Delta x^2)$

$$T_i^{k+1} - T_i^k = Fo(T_{i-1}^k - 2T_i^k + T_{i+1}^k)$$

$$\boxed{T_i^{k+1} = T_i^k + Fo\left(T_{i-1}^k - 2T_i^k + T_{i+1}^k\right)}$$

Note: Boundary conditions determine final form of the system of equations

System of eqs with Dirichlet bcs: Forward Euler

- The final system of equations given Dirichlet boundary conditions is

$$\begin{cases} T_0^{k+1} = T_p \\ T_i^{k+1} = T_i^k + F_0\left(T_{i-1}^k - 2T_i^k + T_{i+1}^k\right) \\ T_{N_x-1}^{k+1} = T_c \end{cases} \qquad \forall k \text{ and } i = 1, \dots, N_x - 2$$

N_x algebraic equations

- The initial condition is given by

$$T_i^0 = T_c \qquad i = 1, \dots, N_{x-2}$$

- Double for-loop needed to calculate each T_i^k in the domain.

System of eqs with Neumann bcs (1): Forward Euler

- The final system of equations given Neumann boundary conditions is

$$\begin{cases} T_0^{k+1} = T_1 - \gamma_1 \Delta x \\ T_i^{k+1} = T_i^k + F_0\left(T_{i-1}^k - 2T_i^k + T_{i+1}^k\right) \\ T_{N_x-1}^{k+1} = T_{N_x-2} + \gamma_2 \Delta x \end{cases} \qquad \forall k \text{ and } i = 1, \dots, N_x - 2$$

N_x algebraic equations

- The initial condition is given by

$$T_i^0 = T_c \qquad i = 1, \dots, N_{x-2}$$

- Double for-loop needed to calculate each T_i^k in the domain.

System of eqs with Neumann bcs (2): Forward Euler

- The final system of equations given Neumann boundary conditions with **ghost points** is

$$\begin{cases} T_{\mathbf{-1}}^{k+1} = T_1 - 2\gamma_1 \Delta x \\ T_i^{k+1} = T_i^k + F_0\left(T_{i-1}^k - 2T_i^k + T_{i+1}^k\right) \\ T_{\boldsymbol{N_x}}^{k+1} = T_{N_x-2} + 2\gamma_2 \Delta \mathrm{x} \end{cases} \qquad \forall k \text{ and } i = 0, \dots, N_x - 1$$

N_x+2 algebraic equations

- The initial condition is given by

$$T_i^0 = T_c \qquad i = 1, \dots, N_{x-2}$$

- Double for-loop needed to calculate each T_i^k in the domain.

How do I avoid for-loops?

Semidiscrete Method (Matrix form): Forward Euler

- Again, the simplest choice for solving systems of ODEs is to use Forward Euler method.
- Discretization of the first order time derivative in **vector form**:

$$\frac{d\boldsymbol{T}}{dt} = \frac{\boldsymbol{T}^{k+1} - \boldsymbol{T}^{k}}{\Delta t}, \qquad t_k = k \cdot \Delta t$$

- The PDE can be written as:

Note: A and $\boldsymbol{b}$ depend on the dv and bcs

$$\frac{\boldsymbol{T}^{k+1} - \boldsymbol{T}^{k}}{\Delta t} = \frac{\alpha}{\Delta x^2}\left(A\boldsymbol{T}^{k} + \boldsymbol{b}\right), \qquad k = 0, \dots, N_t - 2$$

- Thus, giving the updating closed form: $(\mathrm{Fo} = \alpha \Delta t / \Delta x^2)$

$$\boldsymbol{T}^{k+1} - \boldsymbol{T}^{k} = Fo\left(A\boldsymbol{T}^{k} + \boldsymbol{b}\right)$$

$$\boxed{\boldsymbol{T}^{k+1} = \boldsymbol{T}^{k} + Fo\left(A\boldsymbol{T}^{k} + \boldsymbol{b}\right)}$$

It is preferable to work in matrix form for computational reasons (avoid for-loops!)

Matrix form with Dirichlet bcs: Forward Euler

- The equations can be composed in an **explicit** linear system in matrix form:

$$T^{k+1} = Fo \cdot A_{DF} T^k + Fo \cdot b_{DF}$$

$$\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix}^{k+1} = Fo \begin{bmatrix} \frac{1}{Fo} - 2 & 1 & 0 & \dots & \dots & 0 \\ 1 & \frac{1}{Fo} - 2 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & \frac{1}{Fo} - 2 & 1 \\ 0 & 0 & \cdots & 0 & 1 & \frac{1}{Fo} - 2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix}^{k} + Fo \begin{bmatrix} T_p \\ 0 \\ \vdots \\ 0 \\ T_c \end{bmatrix}$$

$A_{DF} \in \mathbb{R}^{N_x-2 \times N_x-2}$ $\quad$ $T^{k+1}, T^k \in \mathbb{R}^{N_x-2}$ $\quad$ $b_{DF} \in \mathbb{R}^{N_x-2}$

Matrix form with Neumann bcs (1): Forward Euler

- For Neumann boundary conditions the same Euler forward scheme holds. However, the matrix A and the vector $\boldsymbol{b}$ are replaced accordingly. Resulting in an **explicit** linear system in matrix form:

$$T^{k+1} = Fo \cdot A_{NF} T^k + Fo \cdot b_{NF}$$

$$\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix}^{k+1} = Fo \begin{bmatrix} \frac{1}{Fo} - 1 & 1 & 0 & \dots & \dots & 0 \\ 1 & \frac{1}{Fo} - 2 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & \frac{1}{Fo} - 2 & 1 \\ 0 & 0 & \cdots & 0 & 1 & \frac{1}{Fo} - 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \end{bmatrix}^{k} + Fo \begin{bmatrix} -\gamma_1 \Delta x \\ 0 \\ \vdots \\ 0 \\ \gamma_2 \Delta x \end{bmatrix}$$

$$A_{NF} \in \mathbb{R}^{N_x-2 \times N_x-2} \qquad T^{k+1}, T^k \in \mathbb{R}^{N_x-2} \qquad b_{NF} \in \mathbb{R}^{N_x-2}$$

Matrix form, Neumann bcs (2): Forward Euler

- For Neumann boundary conditions with ghost points the same Euler forward scheme holds. However, the matrix A and the vectors $\boldsymbol{b}$ and $\boldsymbol{T}$ are replaced accordingly. Resulting in an **explicit** linear system in matrix form:

$$T_G^{k+1} = Fo \cdot A_{NFG} T_G^k + Fo \cdot b_{NFG}$$

$$\begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \\ T_{N_x-1} \end{bmatrix}^{k+1} = Fo \begin{bmatrix} \frac{1}{Fo} - 2 & 2 & 0 & \dots & \dots & 0 \\ 1 & \frac{1}{Fo} - 2 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & \frac{1}{Fo} - 2 & 1 \\ 0 & 0 & \cdots & 0 & 2 & \frac{1}{Fo} - 2 \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ \vdots \\ T_{N_x-3} \\ T_{N_x-2} \\ T_{N_x-1} \end{bmatrix}^{k} + Fo \begin{bmatrix} -2\gamma_1 \Delta x \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 2\gamma_2 \Delta x \end{bmatrix}$$

extremes included → T_G

$A_{NFG} \in \mathbb{R}^{N_x \times N_x}$ $\quad T_G^{k+1}, T_G^k \in \mathbb{R}^{N_x}$ $\quad b_{NFG} \in \mathbb{R}^{N_x}$

Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE's (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Semidiscrete Method: Forward Euler

- Considerations about Explicit Euler method are the same as discussed in Q1 Lecture 5:
- PROS:
  - simple implementation
  - explicit formula
  - efficient computation
- CONS:
  - stability issues
  - required small time steps (potential memory issues)

Stability criterion for explicit methods

- The Courant-Friedrichs-Lewy (CFL) condition is a <u>necessary</u> condition for the convergence of parabolic PDEs.
- For 1D problems parabolic PDEs, the CFL condition has the following form:

$$Fo = \frac{\alpha \Delta t}{\Delta x^2} \leq \frac{1}{2}$$

- Fo is the Fourier number (dimensionless)

- CFL condition states that, given an arbitrary space step, the time step must be <u>necessarily</u> small “enough” to converge:

$$\Delta t \leq \frac{\Delta x^2}{2\alpha}$$

Are there alternatives to explicit methods to avoid instability issues?

Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE's (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Semidiscrete Method: Backward Euler

- The simplest choice for solving systems of ODEs is to use Forward (explicit) Euler method.
- Discretization of the first order time derivative:

$$\frac{\partial \boldsymbol{T}}{\partial t} = \frac{\boldsymbol{T}^{k+1} - \boldsymbol{T}^{k}}{\Delta t}, \qquad t_k = k \cdot \Delta t$$

- The PDE can be written as:

Note: A and $\boldsymbol{b}$ depend on the dv and bcs

$$\frac{\boldsymbol{T}^{k+1} - \boldsymbol{T}^{k}}{\Delta t} = \frac{\alpha}{\Delta x^2}\left(A\boldsymbol{T}^{k+1} + \boldsymbol{b}\right), \qquad k = 0, \ldots, N_t - 2$$

- Thus, giving the updating closed form: $(\mathrm{Fo} = \alpha \Delta t / \Delta x^2)$

$$\boldsymbol{T}^{k+1} - \boldsymbol{T}^{k} = Fo\left(A\boldsymbol{T}^{k+1} + \boldsymbol{b}\right)$$

$$\boxed{(I - FoA)\boldsymbol{T}^{k+1} = \boldsymbol{T}^{k} + Fo\boldsymbol{b}}$$

It is preferable to work in matrix form for computational reasons (solving systems of eq.)

Agenda

- What is a Partial Differential Equation (PDE)?
  - Classification of PDE's (Elliptic, Parabolic, Hyperbolic)
  - Recap: Initial conditions (ICs) and boundary conditions (BCs)
- Introduction to numerical methods to solve PDEs
- Finite Difference Method for PDEs
  - Discretize the multivariate domain
  - Discretize with ghost points
- Parabolic PDE: 1D unsteady heat equation
- Semidiscrete method to solve parabolic PDEs
  - Spatial derivatives
    - Internal points
    - Handling boundary conditions
  - Time derivatives
    - Forward Euler
    - Stability
    - Backward Euler
  - Built-in IVP solvers

Semidiscrete Method: ODE Solver

- Recall the method for solving parabolic PDEs:
  1. Discretize spatial derivatives with finite differences
  2. Obtain a system of ODE equations
  3. Solve the system of ODE equations
- We analysed two simple techniques for 3: Euler Forward and Backward
- More advanced schemes for ODEs can be used instead (Runge-Kutta) and built-in functions (*scipy.integrate.solve_ivp()*)

- Want to speed up your computation? Exploit the sparsity of the discretization matrices!
  - *scipy.sparse.spdiag → Define a sparse diagonal matrix*
  - *scipy.sparse.linalg.spsolve → Solve sparse linear systems efficiently*

Parabolic PDE: Live coding

- Open Colab: [Parabolic PDE - Heat Equation]

- Find more in the Github repository of the course: https://github.com/process-intelligence-research/computational_practicum_lecture_coding/tree/main

Learning objectives

After successfully completing this lecture, you are able to…

- Explain what a partial differential equation (PDEs) is, how it can be classified, and how it is different form ordinary differential equations (ODEs).
- Give an overview of the main techniques to solve partial differential equations in (chemical) engineering
- Implement different numerical solution approaches for parabolic PDEs from scratch
- Discuss stability of numerical solution approaches for parabolic PDEs
- Use Python libraries' built-in functions to support the solution of parabolic PDEs

Thank you very much for your attention!