Block (1/3)

The Finite Difference Method for the Poisson in 1D and 2D

EE4375 - FEM For EE Applications

Domenico Lahaye

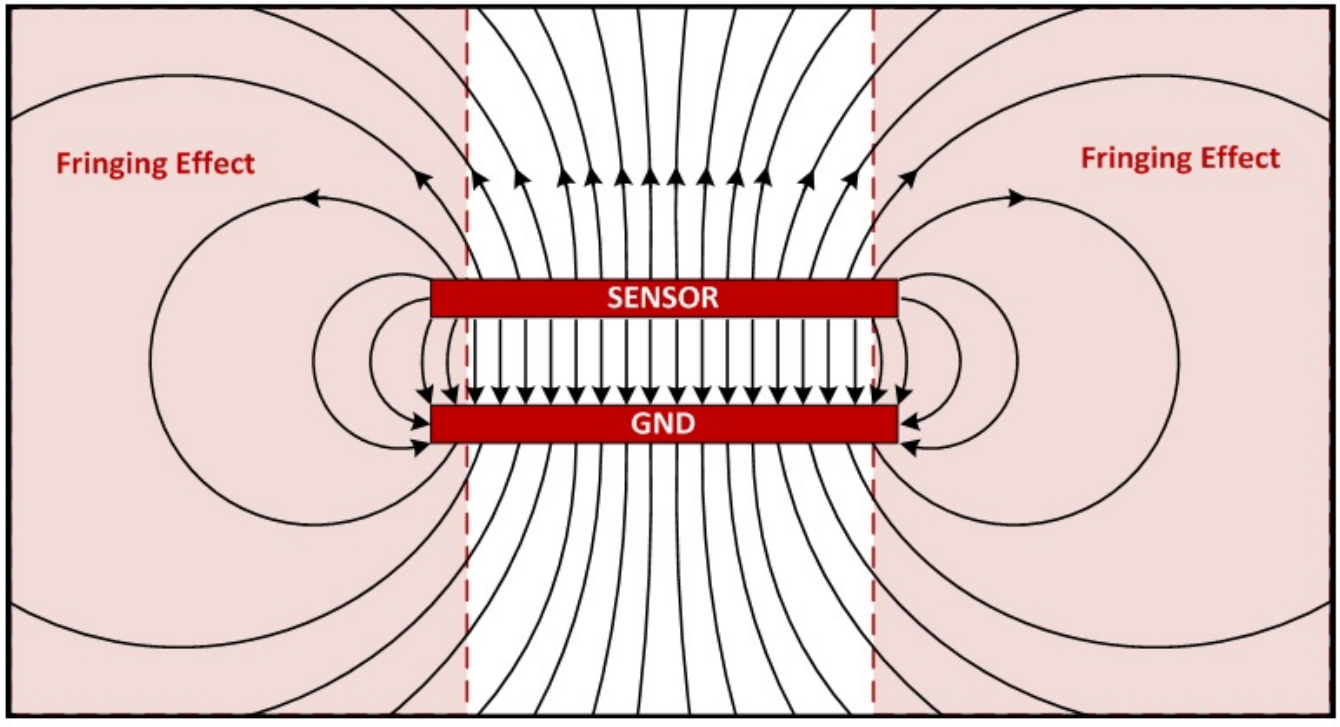
DIAM - Delft Institute of Applied Mathematics

Last updated July 17, 2025

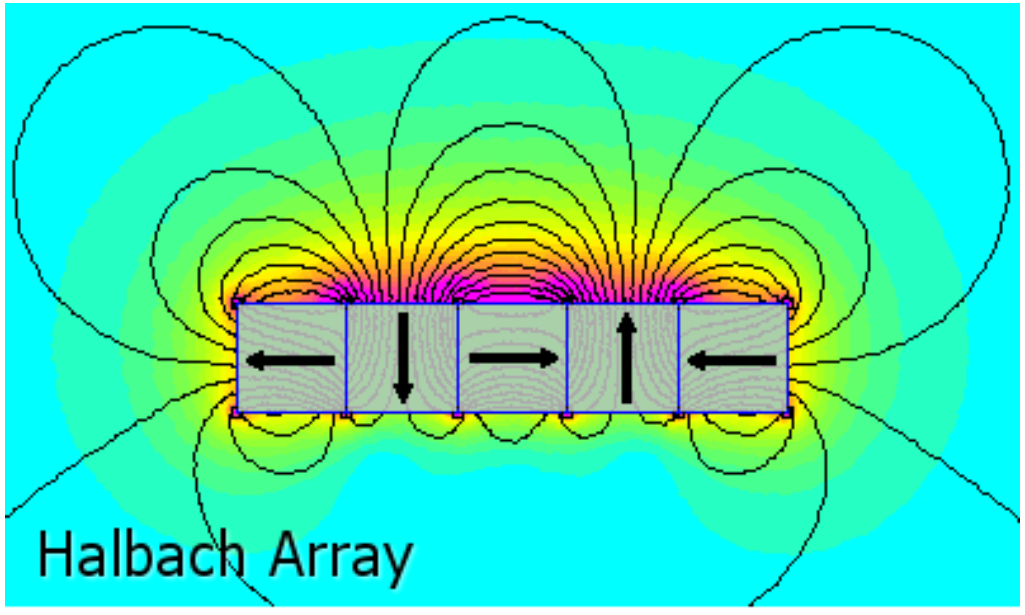
Content of This First Block (of Three)

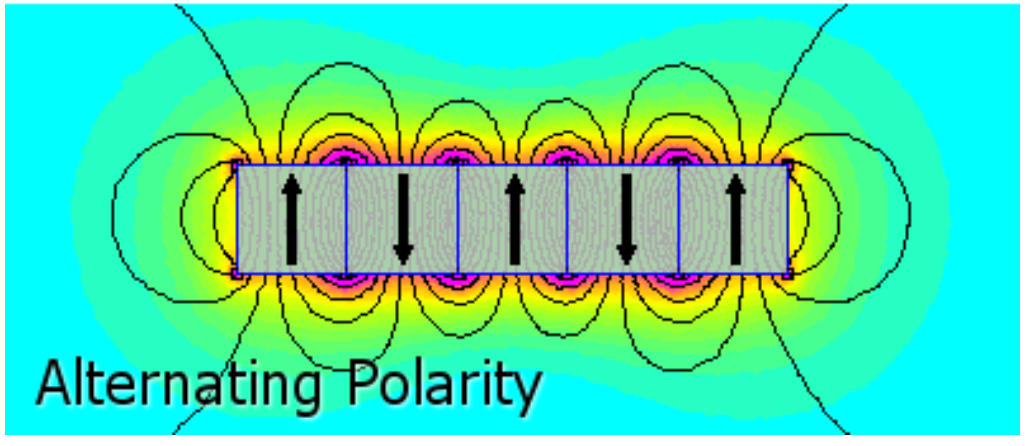
- finite difference method in one spatial dimension (1D)
- finite difference method in two spatial dimensions (2D)
- applications in electro-static and magneto-static fields

Applications in Electro-Static Fields

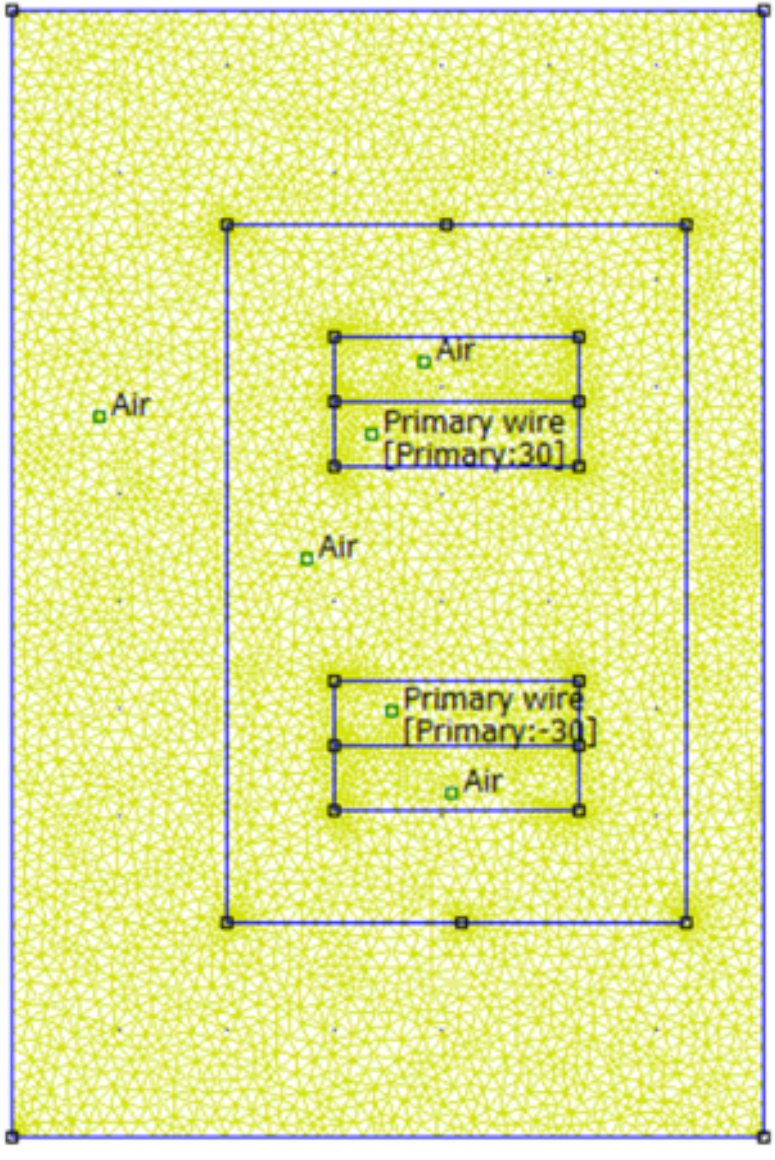


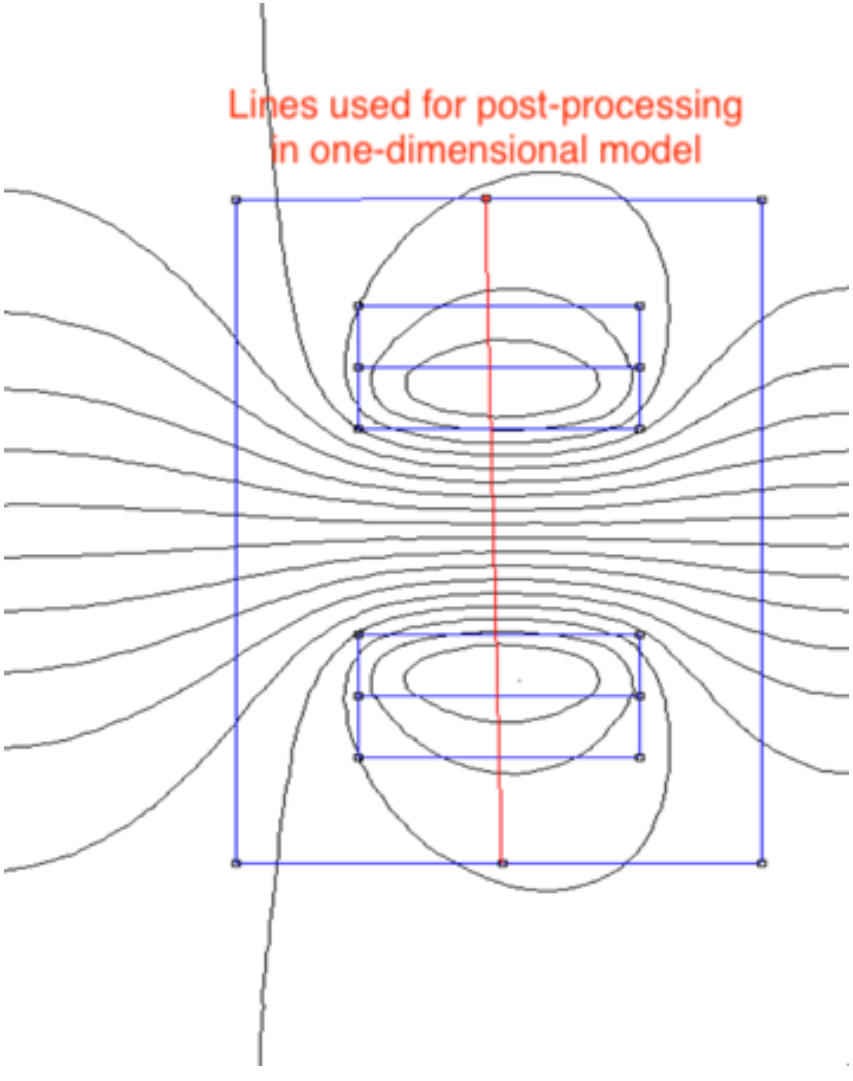
Applications in Magneto-Static Fields



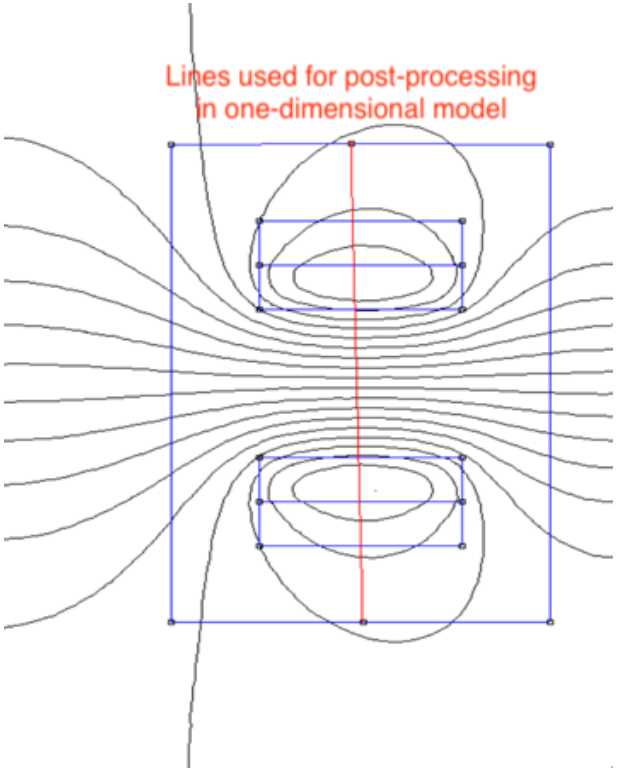


Applications in Other Course (1/2)

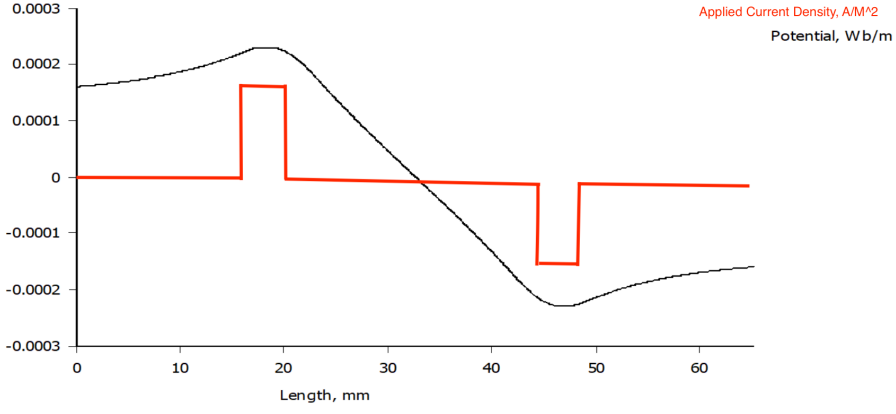




Applications in Other Course (2/2)



applied current density
computed potential



Motivation for the Finite Difference Method

- wish - desire - goal - dream - ambition:

  solve a boundary value problem for the electric or magnetic field

- pen-and-paper methods: very powerful in limited cases

- finite element method: ultimate goal in this course

- finite difference method: intermediate step

  poor man's version of the finite element method

  short-cut to the final goal

One-Dimensional Problem to Solve (1/9)

Computational Domain $x \in \Omega = (0, 1)$

- x: independent variable
- $0 < x < 1$ or $x \in \Omega$ where $\Omega = (0, 1)$
- Ω is domain of computation - open interval from $x = 0$ to $x = 1$
- $x = 0$ and $x = 1$ left and right end point or boundary of Ω
- Ω is interval, bar, wire, pipe 
- Ω is given a part of the data on the problem to solve
- in this course: Ω domain of computation

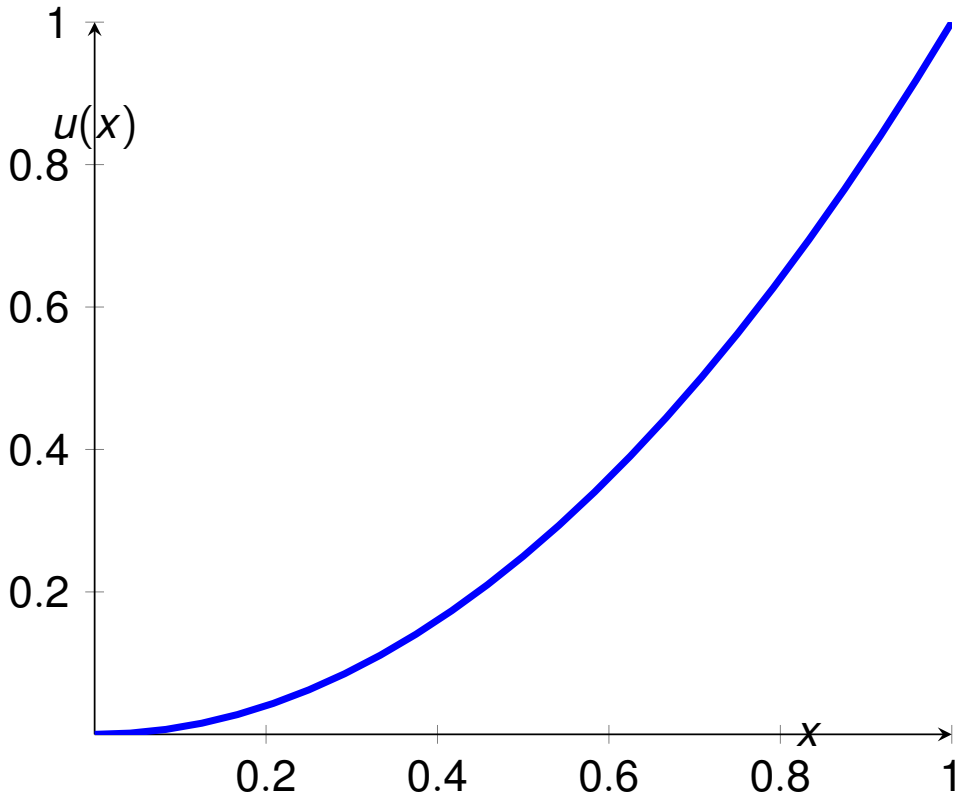
One-Dimensional Problem to Solve (2/9)

What is the unknown? What is searched for?

- $u(x)$: dependent variable and assumed unknown function
- $u(x)$: continuous function in one variable
- $u(x)$: has domain Ω or $u(x)$ lives on Ω
- $u(x)$: has scalar output
- $u(x)$: single input single (SISO) system
- in this course: electrostatics: $u(x)$ potential for electrical field
- in this course: magnetostatics: $u(x)$ potential for magnetic flux

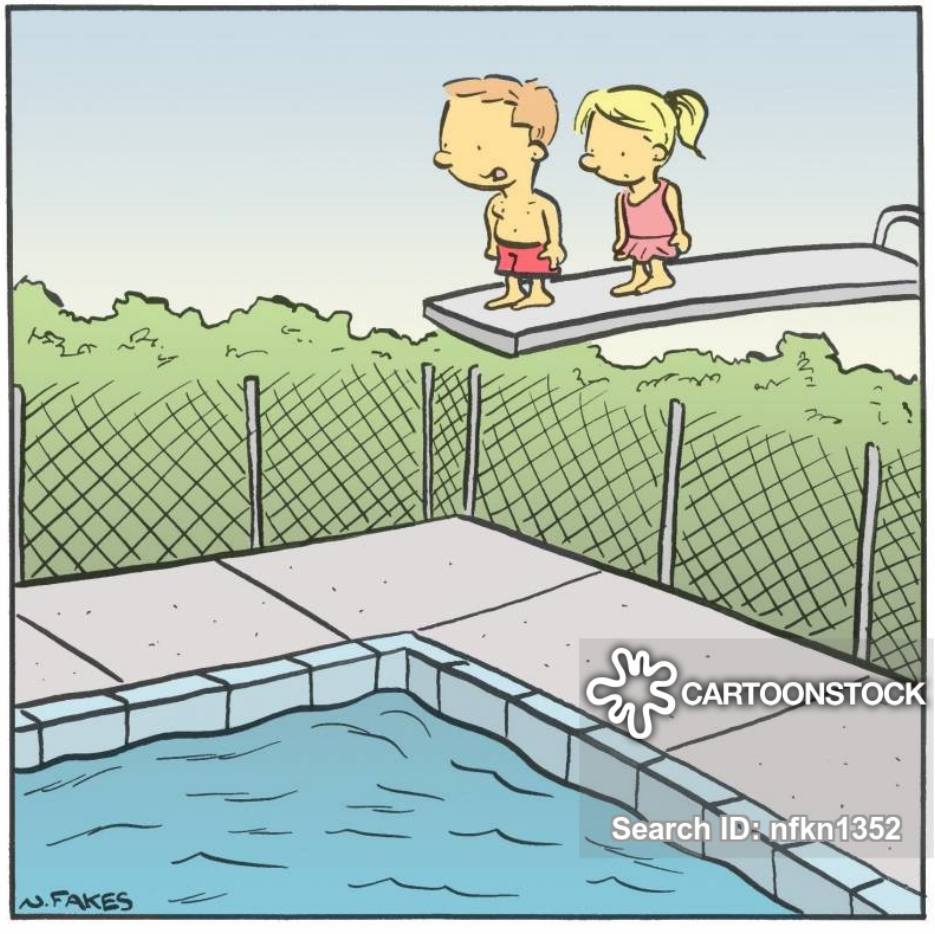
One-Dimensional Problem to Solve (3/9)

How does the graph of u(x) look like?



One-Dimensional Problem to Solve (4/9)

How to compute the unknown potential u(x)?



"This place is a dive. Let's go."

force on board

$\Downarrow$

deflection of board

(equivalent of potential)

(or $u(x)$)

One-Dimensional Problem to Solve (5/9)

Known Volumetric Data on $\Omega = (0, 1)$

- $f(x)$: dependent variable assumed known
- in deflecting beam example:

  what is force $f(x)$ at a given location x along bar?
- in electrostatic:

  what is electrical charge density [in C/m^2] at a given location?
- in magnetostatic:

  what is electrical current density [in A/m^2] at a given location?

One-Dimensional Problem to Solve (6/9)

Known Boundary Data at $x = 0$ and $x = 1$

- in deflecting beam example:

  beam clamped at pole or support

  beam free at other end

- in electrostatic or magnetostatic:

  constraint on the potential: imposing a reference - grounding

  zero derivative of the potential: imposing symmetry

  constraint on the derivative of the potential: imposing a flux

- more examples later

One-Dimensional Problem to Solve (7/9)

Ingredients Introduced Thus Far

- known data:
  $x \in \Omega$, $f(x)$ and boundary data
- unknown data: $u(x)$
- how to compute the potential $u(x)$?
- beam: Newton
- electrostatic or magnetostatic: ???

One-Dimensional Problem to Solve (8/9)

- recap - Newton's Third Law of Motion

$$F = m\,a \Leftrightarrow m\,\frac{d^2\,u(t)}{dt^2} = F(t) \Leftrightarrow \frac{d^2\,u(t)}{dt^2} = F(t)/m = f(t)$$

- replace t by x, thus
  replace initial value problem by boundary value problem
  (motivation later in course):
  introduce a minus sign

- problem to solve:

  given $x \in \Omega$, $f(x)$ and boundary data, find $u(x)$ such that

$$\boxed{-\frac{d^2\,u(x)}{dx^2} = f(x) \text{ and } u(x) \text{ satisfies boundary data}}$$

One-Dimensional Problem to Solve (9/9)

Boundary Value Problem for Second Order Differential Equations

- given $x \in \Omega = (0, 1)$ with left and right boundary point
- given: $f(x)$ given function and α given number
- find: $u(x)$ such that

$$-u''(x) = f(x) \text{ for } 0 < x < 1 \text{ (differential equation on } \Omega)$$
$$u(x = 0) = 0 \text{ (Dirichlet boundary condition in } x = 0)$$
$$\frac{du}{dx}(x = 1) = \alpha \text{ (Neumann boundary condition in } x = 1)$$

- differential equation not valid in $x = 0$ or $x = 1$

  boundary conditions are valid in $x = 0$ or $x = 1$

Example with Closed Form Solution (1/5)

- choose $f(x) = 1$
- find: $u(x)$ such that

$$-u''(x) = 1 \text{ for } 0 < x < 1 \text{ (differential equation on } \Omega)$$
$$u(x = 0) = 0 \text{ (Dirichlet boundary condition in } x = 0)$$
$$\frac{du}{dx}(x = 1) = \alpha \text{ (Neumann boundary condition in } x = 1)$$

- solve using pen and paper and plot computed solution for various values of α

Example with Closed Form Solution (2/5)

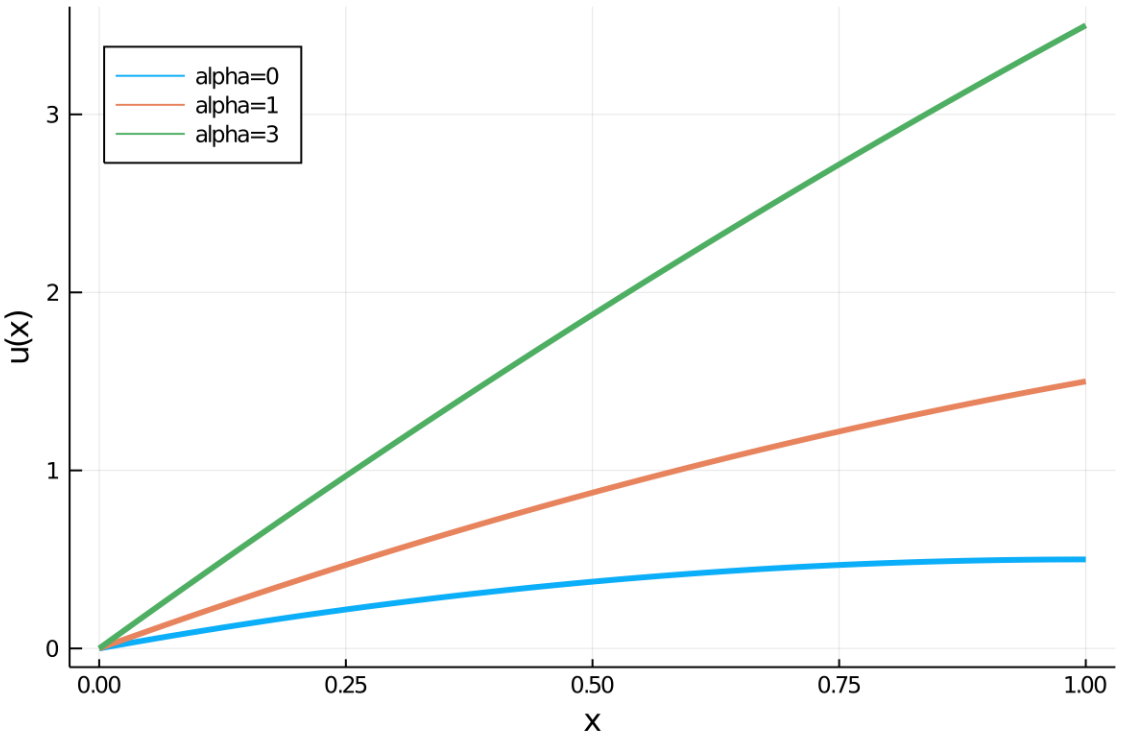
- by integration twice and using the boundary conditions

  $u(x) = -\frac{1}{2} x^2 + (1 + \alpha) x$ (typo in previous version of slides)

- solving by guessing and iteratively improving guess is allowed
- solution can be checking by checking differential equation and boundary conditions
- this examples is will guide remainder of the course

Example with Closed Form Solution (3/5)

Plot of Analytical Solution



- larger α
- larger boundary in-flux

Example with Closed Form Solution (4/5)

Coding the plot in Julia

```
using Plots
#..define vector of samples in x-direction
x = Vector(0:0.01:1);

#..define three functions u0(x), u1(x) and u2(x)
alpha0=0; u0 = -0.5*x.^2 + (1+alpha0)*x;
alpha1=1; u1 = -0.5*x.^2 + (1+alpha1)*x;
alpha2=3; u2 = -0.5*x.^2 + (1+alpha2)*x;

#..aggregate into one matrix for plotting
u = [u0, u1, u2] #..stack various functions column-wise

#..plot command
plot(x, u, label = ["alpha=0" "alpha=1" "alpha=3"], lw=3, xlabel="x",
ylabel="u(x)",legend=:topleft)
```

Example with Closed Form Solution (5/5)

Try yourself

- $f(x) = x$, $f(x) = x^2$, $f(x) = \sin(\pi x)$, $f(x) = \exp(x)$, ...
- compute analytical solution
- plot analytical solution for various values of α

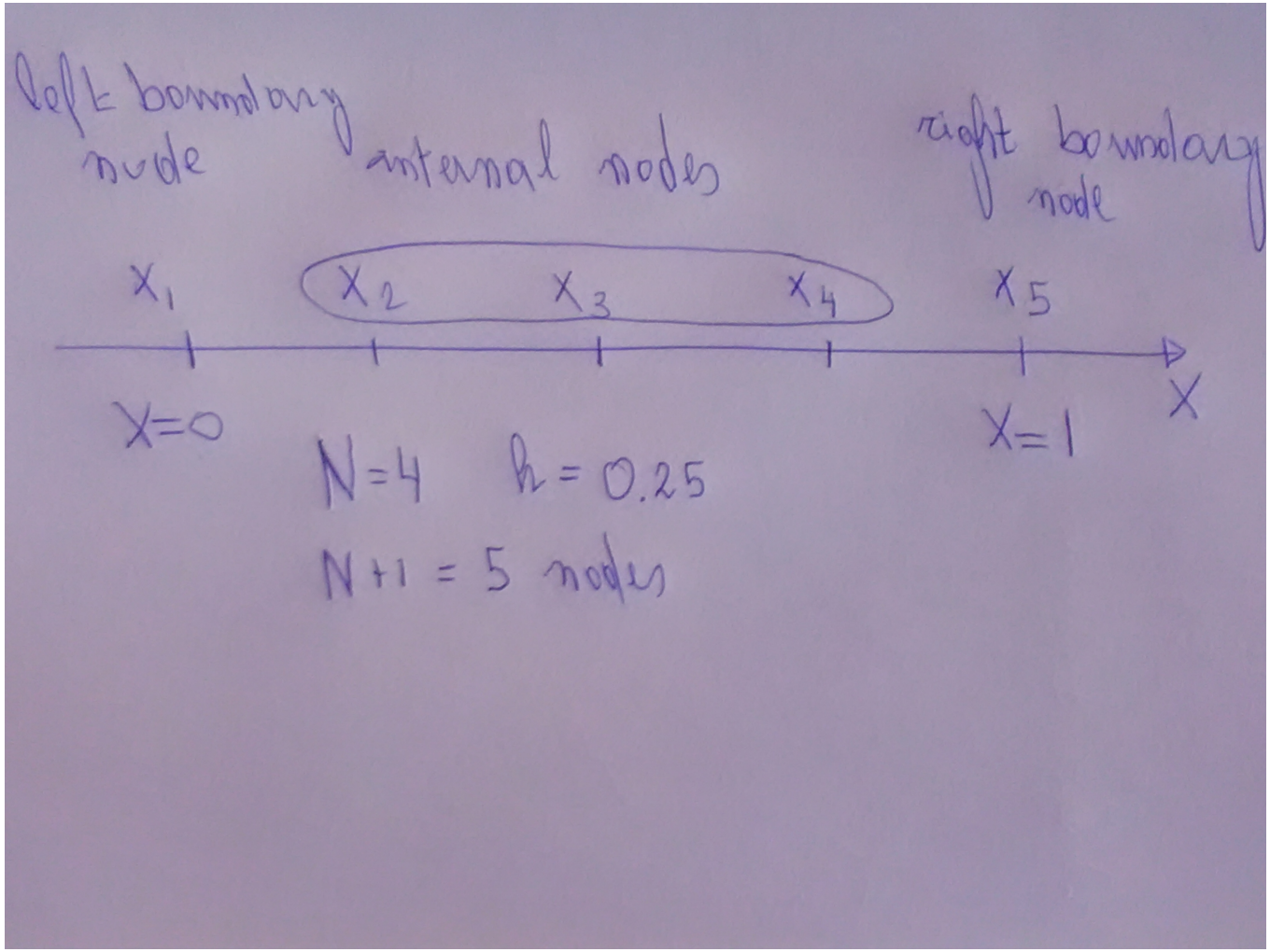
Mesh Generation (1/3)

$$\text{Mesh } \Omega^h = \{x_i \,|\, 1 \leq i \leq N+1\}$$

- create uniform or equidistant mesh on $\Omega = (0,1)$
- N: number of intervals or elements - $h = 1/N$: mesh width
- $x_i = (i-1)\,h$ for $1 \leq i \leq N+1$: nodes or mesh points
- beware: N internals and (N+1) mesh points in 1D
- set of mesh points $\Omega^h = \{x_i \,|\, 1 \leq i \leq N+1\}$
- x_i for $2 \leq i \leq N$: internal mesh points
- x_1: left-most boundary point - x_{N+1}: right-most boundary points
- in Block 2 of course: non-uniform and higher order 1D meshes

Mesh Generation (2/3)

$$\text{Mesh } \Omega^h = \{x_i \,|\, 1 \leq i \leq N+1\}$$



Mesh Generation (3/3)

$$\text{Mesh } \Omega^h = \{x_i \,|\, 1 \leq i \leq N+1\}$$

- coding mesh generation in Julia

```
N = 4; h = 1/N;
x = Vector(0:h:1);
```

- x[1]: left end point

  x[end]: right end point

  x[2:end-1]: internal nodes

- N large - h small - mesh with many nodes

Linear System (1/25)

Two Worlds and Their Connection

- analytical or pen-and-paper world

$$\boxed{-\frac{d^2}{dx^2}u(x) = f(x) \text{ and } u(x) \text{ satisfies boundary data}}$$

- numerical or Julia world

$$\boxed{A\mathbf{u} = \mathbf{f}}$$

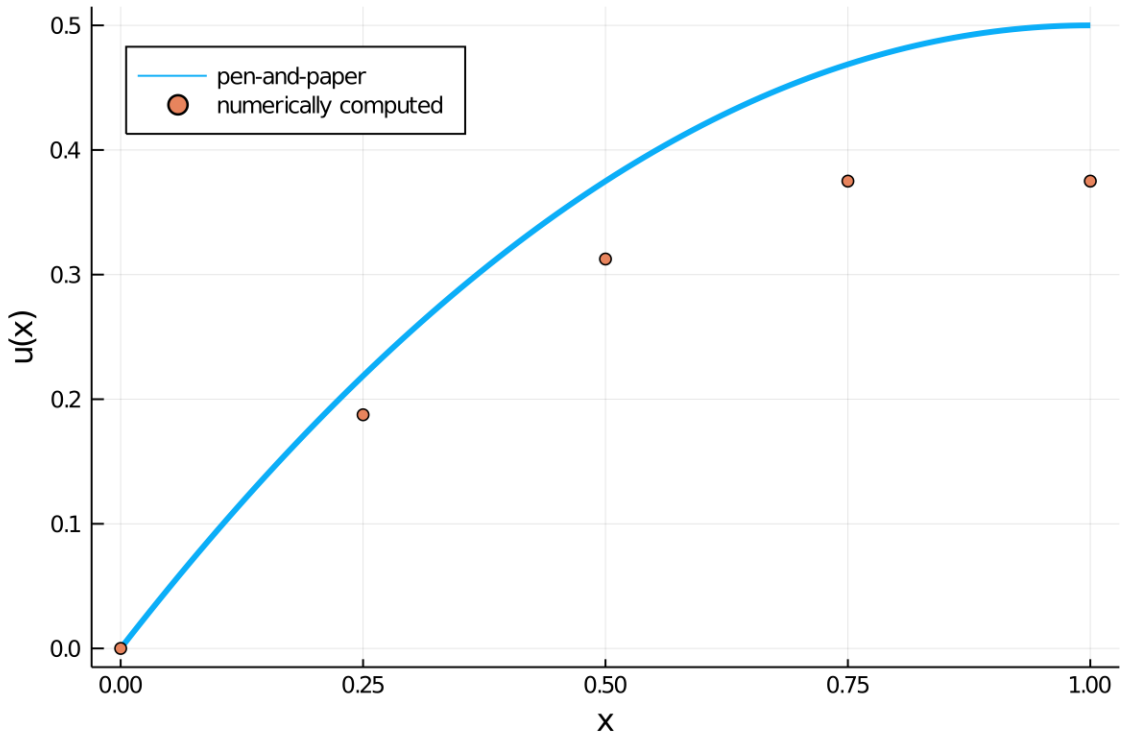
- connection: A: matrix - equivalent of $-\frac{d^2}{dx^2}$

  $\mathbf{f}$ is the equivalent of $f(x)$ (known - load - given)

  $\mathbf{u}$ is the equivalent of $u(x)$ (unknown - displacement/potential - to be computed)

Linear System (2/25)

Plot of Analytical and Numerical Solution



- $\alpha = 0$
- $h = 1/4$

Linear System (3/25)

How was previous graph generated?

- vector **f**: vector of all ones (given that $f(x) = 1$)
- matrix A: tri-diagonal matrix as in first lab session
- vector **u**: by solving the linear system $A\,\mathbf{u} = \mathbf{f}$

  in Julia using the backslash operator: $\mathbf{u} = A \setminus \mathbf{f}$

Linear Algebra Preliminaries

Calculus Preliminaries

Linear System (4/25)

- problem to solve:

  for $x \in \Omega$ or $0 < x < 1$ (continuous in x)

$$\boxed{-\frac{d^2}{dx^2}\,u(x) = f(x) \text{ and } u(x) \text{ satisfies boundary data}}$$

- suppose that x is restricted to values in the mesh Ω^h:

  $x = x_i$ for $x_i \in \Omega^h$ for $2 \leq i \leq N$ (internal nodes, discrete in x_i)

$$\boxed{-\frac{d^2}{dx^2}\,u(x = x_i) = f(x = x_i) \text{ plus stuff for boundaries}}$$

- $x = x_1$ and $x = x_{N+1}$: stuff for boundary: details soon

Linear System (5/25)

Terminology

- transition from $x \in \Omega$ to $x_i \in \Omega^h$ is called collocation
- collocation is typically for finite difference methods
- collocation is intuitive and therefore easier to understand
- finite element method does not use collocation
- finite element method less intuitive

Linear System (6/25)

Approximation of $-\frac{d^2}{dx^2}\,u(x = x_i)$

- $f(x = x_i) = -\frac{d^2}{dx^2}\,u(x = x_i) \approx \frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2}$

$$\underbrace{\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2}}_{\textit{unknown}} = \underbrace{f(x = x_i)}_{\textit{known}}$$

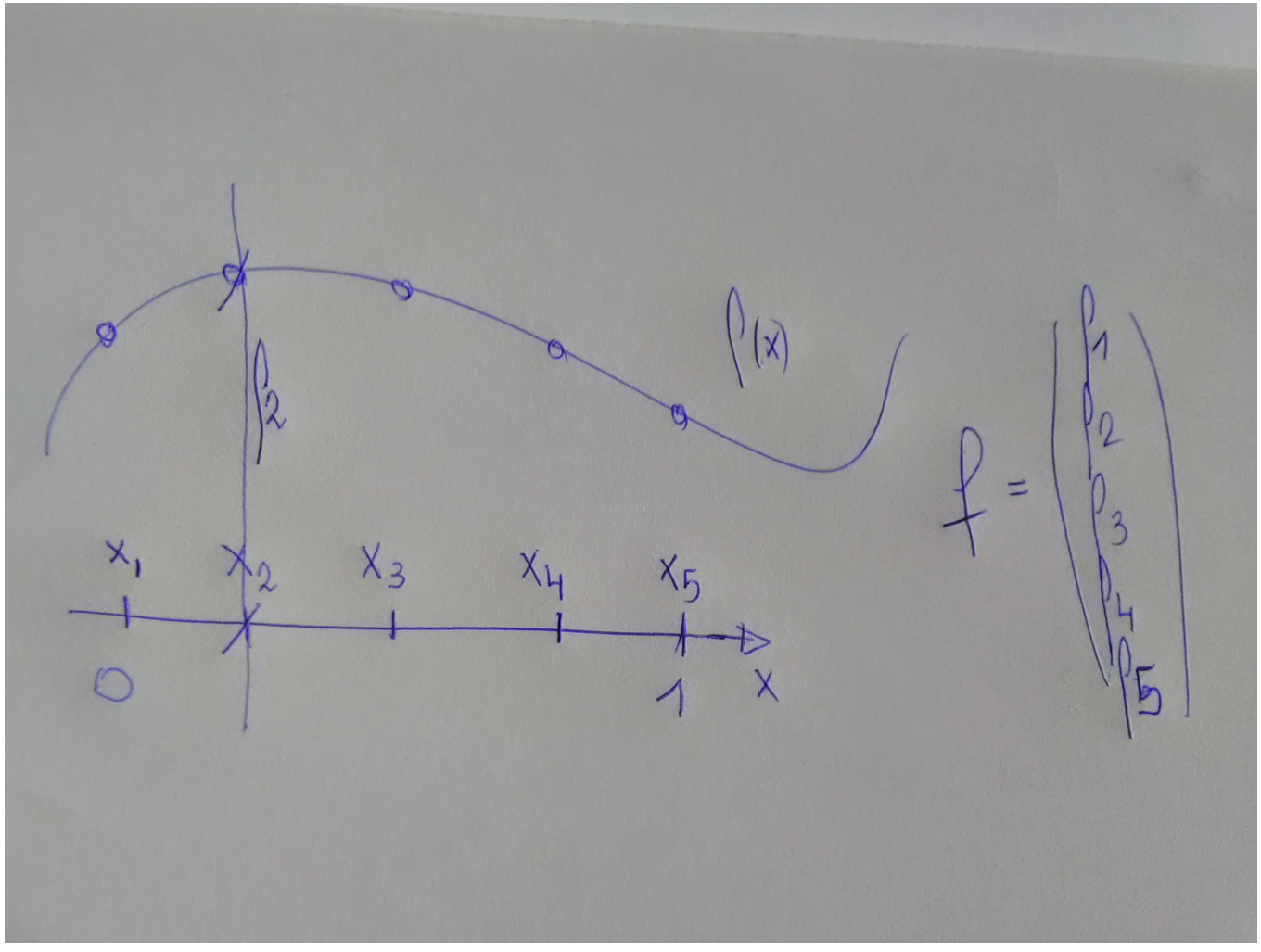
- observe the minus sign introduced:
  not essential due to superposition:
  leads to positive diagonal matrix elements

Linear System (7/25)

Notation $f_i = f(x = x_i)$

- $f(x)$ given/known function
- $x_i \in \Omega^h$ or x_i for $1 \leq i \leq N+1$ set of <u>all</u> mesh points
- $f_i = f(x = x_i)$ set of functions values obtained by evaluating $f(x)$ in each point of Ω^h
- vector $\mathbf{f} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N+1}) \end{pmatrix}$ same procedure as in e.g. plotting $f(x)$

Linear System (8/25)



Linear System (9/25)

Notation $u_i \approx u(x = x_i)$

- $u(x)$ unknown function
- $x_i \in \Omega^h$ or x_i for $1 \leq i \leq N+1$ set of all mesh points
- $u(x = x_i)$ analytical solution evaluated in $x = x_i$
- u_i: numerical approximation on mesh Ω^h

$$\text{vector } \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N+1} \end{pmatrix} \quad \text{where} \quad \underbrace{u_i}_{\textit{numerical}} \approx \underbrace{u(x = x_i)}_{\textit{analytical}}$$

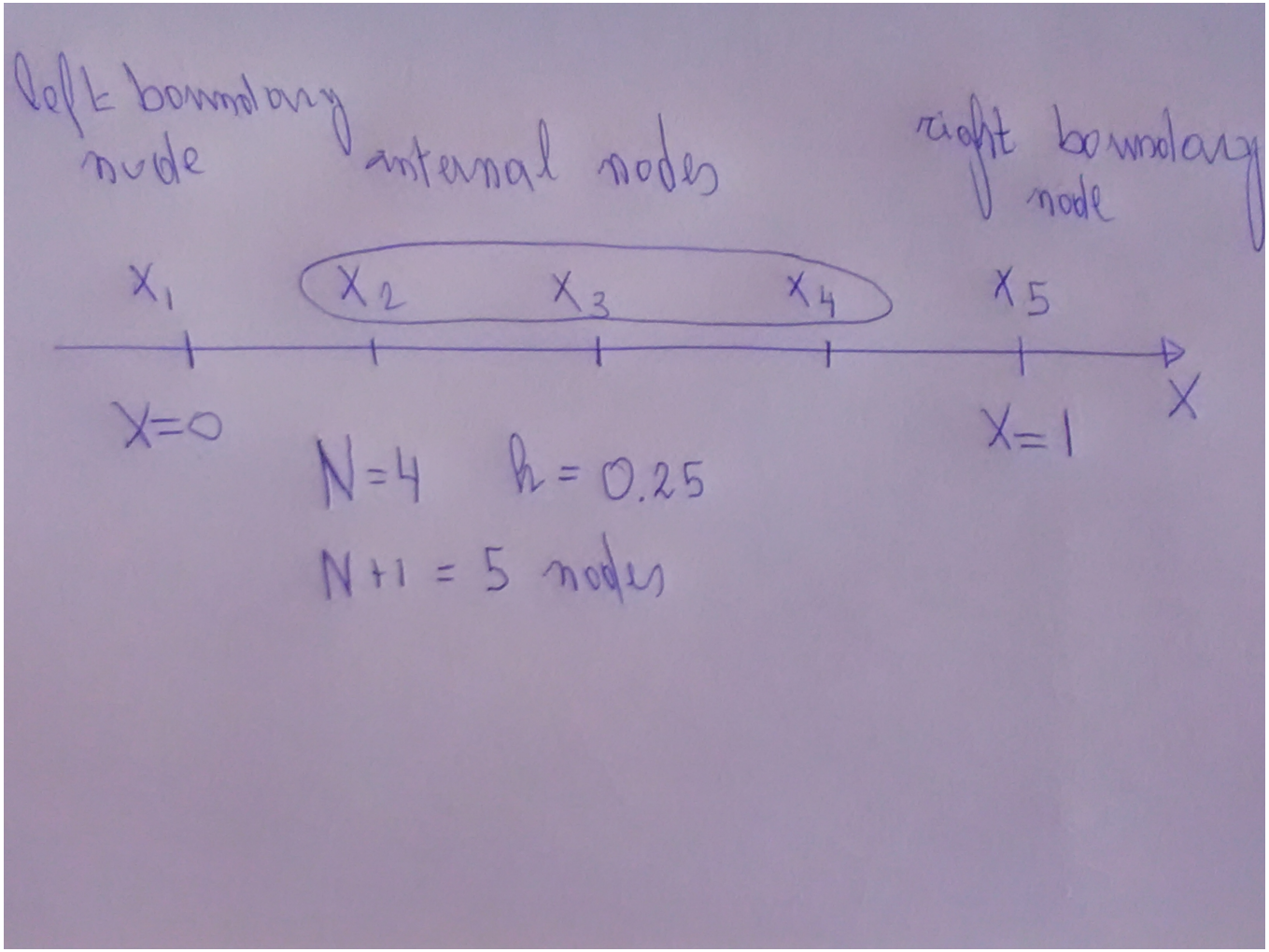
Linear System (10/25)

$$\boxed{\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f(x = x_i)}$$

- this equation holds for u_i internal nodes $2 \leq i \leq N$
- $N - 2$ equations for $N + 1$ unknowns u_i for $1 \leq i \leq N + 1$
- 2 missing equations provided by the 2 boundary conditions
- $(N + 1)$-by-$(N + 1)$ linear system $A\,\mathbf{u} = \mathbf{f}$

Linear System (11/25)

Special Case: $N = 4$ (and thus $h = 1/4$)



Linear System (12/25)

Special Case: $N = 4$ and $h = 1/4$

- at $x = x_2$:

$$\frac{1}{h^2}[-u_1 + 2u_2 - u_3] = f_2$$

- at $x = x_3$:

$$\frac{1}{h^2}[-u_2 + 2u_3 - u_4] = f_3$$

- at $x = x_4$:

$$\frac{1}{h^2}[-u_3 + 2u_4 - u_5] = f_4$$

Linear System (13/25)

Special Case: $N = 4$ and $h = 1/4$

- bundling the three previous equations

$$\frac{1}{h^2}[-1u_1 + 2u_2 - 1u_3 + 0u_4 + 0u_5] = f_2$$
$$\frac{1}{h^2}[+0u_1 - 1u_2 + 2u_3 - 1u_4 + 0u_5] = f_3$$
$$\frac{1}{h^2}[+0u_1 + 0u_2 - 1u_3 + 2u_4 - 1u_5] = f_4$$

- tri-diagonal structure emerges
- two equations missing

Linear System (14/25)

Special Case: $N = 4$ and $h = 1/4$

Handling the Dirichlet boundary condition at $x = 0$

- analytically: impose $u(x = 0) = 0$ (fix reference or grounding)
- numerically: $u_1 \approx u(x = 0)$
- beware of indices (nodes x_1 coincides with left-most boundary $x = 0$)
- suffices to impose

$$\boxed{u_1 = 0}$$

Linear System (15/25)

Special Case: $N = 4$ and $h = 1/4$

- new linear system

$$u_1 = 0$$

$$\frac{1}{h^2}[-1u_1 + 2u_2 - 1u_3 + 0u_4 + 0u_5] = f_2$$

$$\frac{1}{h^2}[+0u_1 - 1u_2 + 2u_3 - 1u_4 + 0u_5] = f_3$$

$$\frac{1}{h^2}[+0u_1 + 0u_2 - 1u_3 + 2u_4 - 1u_5] = f_4$$

- one equation missing

Linear System (16/25)

Special Case: $N = 4$ and $h = 1/4$

Handling the Neumann boundary condition at $x = 1$

- analytically: impose $u'(x = 1) = \alpha$
  (impose symmetry ($\alpha = 0$), influx ($\alpha > 0$) or outflux ($\alpha < 0$))
- numerically using backward difference approximation using stepsize h

$$\alpha = u'(x = 1) \approx \frac{u_5 - u_4}{h}$$

- beware of the indices
- equation becomes

$$\boxed{-\frac{1}{h}u_4 + \frac{1}{h}u_5 = \alpha}$$

Linear System (17/25)

Special Case: $N = 4$ and $h = 1/4$

- new linear system

$$
\begin{aligned}
u_1 &= 0 \\
\frac{1}{h^2}[-1u_1 + 2u_2 - 1u_3 + 0u_4 + 0u_5] &= f_2 \\
\frac{1}{h^2}[+0u_1 - 1u_2 + 2u_3 - 1u_4 + 0u_5] &= f_3 \\
\frac{1}{h^2}[+0u_1 + 0u_2 - 1u_3 + 2u_4 - 1u_5] &= f_4 \\
-\frac{1}{h}u_4 + \frac{1}{h}u_5 &= \alpha
\end{aligned}
$$

- 5 equations for 5 unknows $u_1, \ldots, u_5$

Linear System (18/25)

Special Case: $N = 4$ and $h = 1/4$

Entire Linear System Assuming That $f(x) = 1$

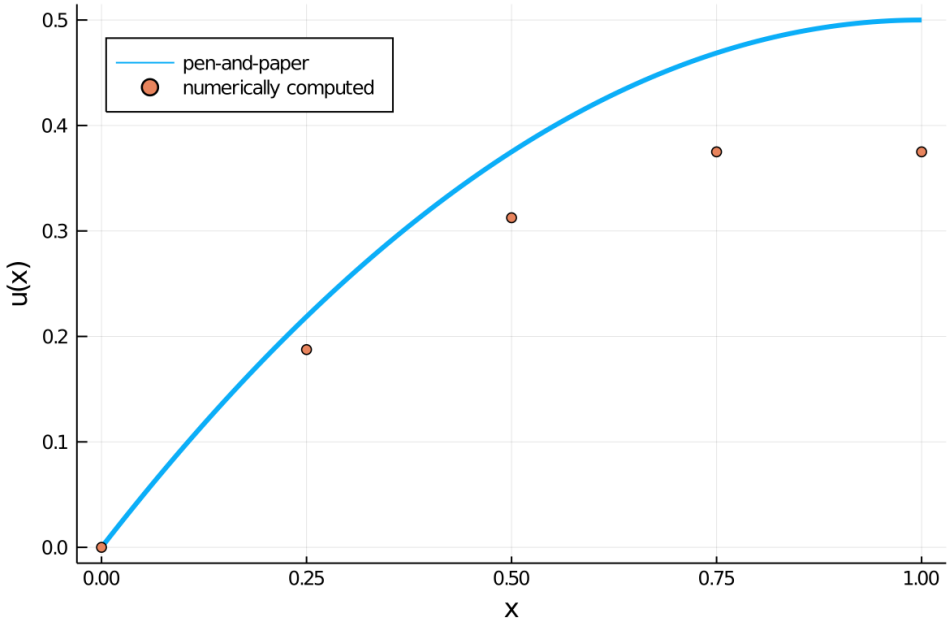
- $A = \frac{1}{h^2} \begin{pmatrix} h^2 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -h & h \end{pmatrix} \quad \mathbf{f} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ \alpha \end{pmatrix}$

- set $\alpha = 0$ and solve using Julia to find $\mathbf{u} = \begin{pmatrix} 0 \\ 0.1875 \\ 0.3125 \\ 0.375 \\ 0.375 \end{pmatrix}$

Linear System (19/25)

Special Case: $N = 4$ and $h = 1/4$

Plot of Analytical and Numerical Solution



- $\alpha = 0$
- $h = 1/4$

Now fully explained!

Obverse that numerical solution has zero derivative to the left of $x = 1$.

Linear System (20/25)

Problem to Solve: General Case

- given $x \in \Omega = (0, 1)$ with left and right boundary point
- given: $f(x)$ given function and α given number
- find: $u(x)$ such that

$$-u''(x) = f(x) \text{ for } 0 < x < 1 \text{ (differential equation on } \Omega)$$
$$u(x = 0) = 0 \text{ (Dirichlet boundary condition in } x = 0)$$
$$\frac{du}{dx}(x = 1) = \alpha \text{ (Neumann boundary condition in } x = 1)$$

Linear System (21/25)

Generate the Mesh

- N: number of intervals or elements - $h = 1/N$: mesh width
- $x_i = (i-1)\,h$ for $1 \leq i \leq N+1$: nodes or mesh points
- set of mesh points $\Omega^h = \{x_i \,|\, 1 \leq i \leq N+1\}$
- x_i for $2 \leq i \leq N$: internal mesh points
- x_1: left-most boundary point - x_{N+1}: right-most boundary points

Linear System (22/25)

Finite Difference Discretization

- internal mesh points x_i for $2 \leq i \leq N$

$$\boxed{\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f(x = x_i)}$$

- x_1: left-most boundary point

$$\boxed{u_1 = 0}$$

- x_{N+1}: right-most boundary points

$$\boxed{-\frac{1}{h}u_N + \frac{1}{h}u_{N+1} = \alpha}$$

Linear System (23/25)

Linear System Formulation $A\mathbf{u} = \mathbf{f}$

- internal mesh points x_i for $2 \leq i \leq N$: $f[i] = f(x_i)$

$$A[i,:] = \begin{pmatrix} 0 & \dots & 0 & \underbrace{-1/h^2}_{i-1} & \underbrace{2/h^2}_{i} & \underbrace{-1/h^2}_{i+1} & 0 & \dots & 0 \end{pmatrix}$$

- left-most boundary point x_1: $f[1] = 0$

$$A[1,:] = \begin{pmatrix} \underbrace{1}_{1} & 0 & \dots & 0 \end{pmatrix}$$

- right-most boundary points x_{N+1}: $f[\mathrm{N+1}] = \alpha$

$$A[N+1,:] = \begin{pmatrix} 0 & \dots & 0 & \underbrace{-1/h}_{N} & \underbrace{1/h}_{N+1} \end{pmatrix}$$

Linear System (24/25)

Linear System Solve

- $\mathbf{u} = A \setminus \mathbf{f}$
- ask details offline

Linear System (25/25)

Post Processing

- plot computed solution **u**
- done! Smile

```
#..load required packages
using LinearAlgebra
using Plots

#..(1/4): construct 1D mesh
N = 4; Np1 = N+1; h = 1/N; h2=h*h;
x = Vector(0:h:1);

#..(2/4): linear system: right-hand side vector
f = ones(size(x)); alpha = 0;

#..(2/4): linear system set-up: coefficient matrix
e = ones(Np1);
A = Tridiagonal(-e[2:end], 2*e, -e[2:end]);
A = (1/h2)*A;

#..(2/4): linear system set-up: boundary conditions
#..Dirichlet boundary conditions at left end point (x=0)
A[1,1] = 1;  A[1,2] = 0;  f[1] = 0;
#..Neumann boundary conditions at right end point (x=1)
A[end,end-1]=-1/h; A[end,end] = 1/h;  f[end] = alpha;

#..(3/4): solve the linear system for u
u = A \ f;

#..(4/4): plot computed solution
plot(x, u, label = "numerical", lw=3, xlabel="x", ylabel="u(x)",legend=:topleft)
```

Numerics (27/25)

How to enjoy this 1D Julia Code?

- run code for various choices of $f(x)$ and α
- compare analytical and numerical computation for various values of N
- try $f(x)$ to be discontinuous or otherwise weird

Numerics (28/25)

Fun Facts on Julia

- using broadcast to define the vector **f**
- using map to define the vector **f**
- using BenchMarkTools.jl to bench the code
- using SymPy.jl (using `dsolve` function with `ics` option for the boundary condition) to solve the boundary value problem symbolically

Exercises (1/1)

- exercise 1: modify previous case to implement Neumann and Dirichlet boundary conditions on the left and right end point, respectively;
- exercise 2: modify previous case to implement Dirichlet boundary conditions on both the left and right end point;
- exercise 3: modify previous case to implement $f(x) = x^2$ (or any other expression for $f(x)$)

Recap (1/2)

Where are we in EE4375 now?

- finished/done! $-u''(x) = f(x)$ + boundary conditions
  using one-dimensional finite difference method
- what is next? $-u''(x, y) = f(x, y)$ + boundary conditions
  using two-dimensional finite difference method
- what is next-next? $-u''(x) = f(x)$ + boundary conditions
  using one-dimensional finite element method

Recap (2/2)

But wait

- what is $u''(x, y)$?
- what is the domain of x and y?
- how should the 1D solver be adapted?
- ...

Two-Dimensional Problem to Solve (1/8)

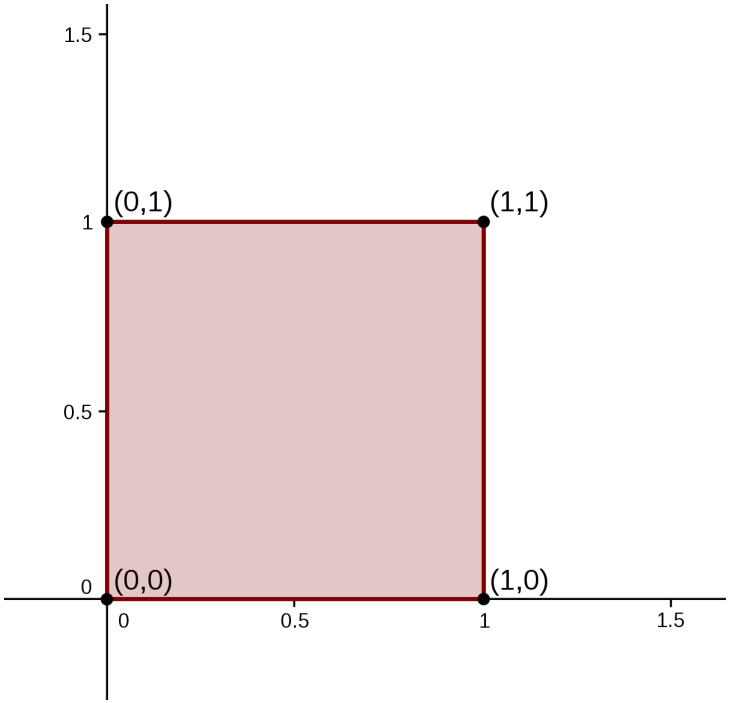
Domain of Computation

Before

Unit Interval: $\Omega = (0, 1)$

Here

Unit Square: $\Omega = (0, 1)^2$



Two-Dimensional Problem to Solve (2/8)

Domain of Computation

- Before: $u(x)$ and $f(x)$ for $x \in \Omega = (0, 1)$
- Here: $u(x, y)$ and $f(x, y)$ for $(x, y) \in \Omega = (0, 1)^2$
- Goal: formulate $u''(x, y) = f(x, y)$ + boundary conditions
- However: what $u''(x, y)$ and boundary conditions?

Two-Dimensional Problem to Solve (3/8)

What is $u''(x, y)$?

- for now:

$$u''(x, y) = \triangle u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

- forthcoming:

$$u''(x, y) = \text{div } (\text{grad } u) = \nabla \cdot (\nabla u)$$

Two-Dimensional Problem to Solve (4/8)

What are boundary conditions? (1/4)

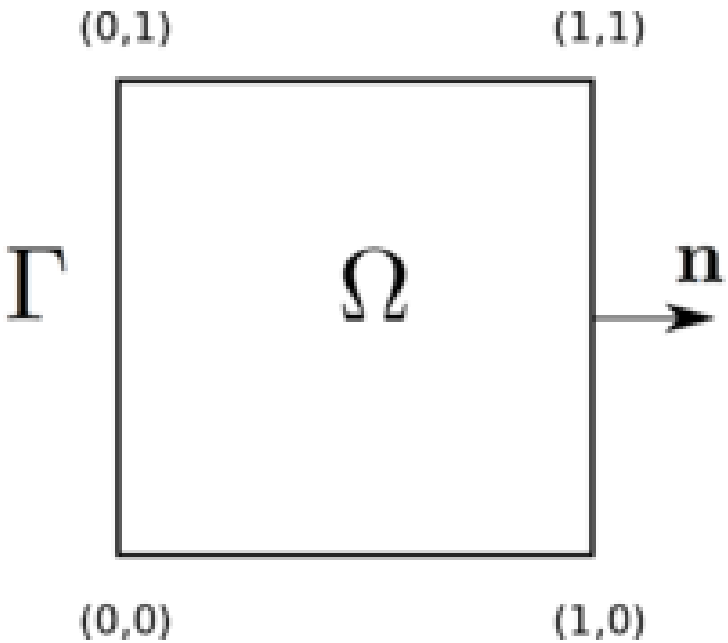
- Before: $u''(x) = f(x)$
- Now: $u''(x, y) = f(x, y)$ actually means $\triangle u = f(x, y)$
- what about the boundary conditions?
- Before: $\Omega = (0, 1)$ has boundary $x = 0$ and $x = 1$
- Now: $\Omega = (0, 1)^2$ has boundary Γ

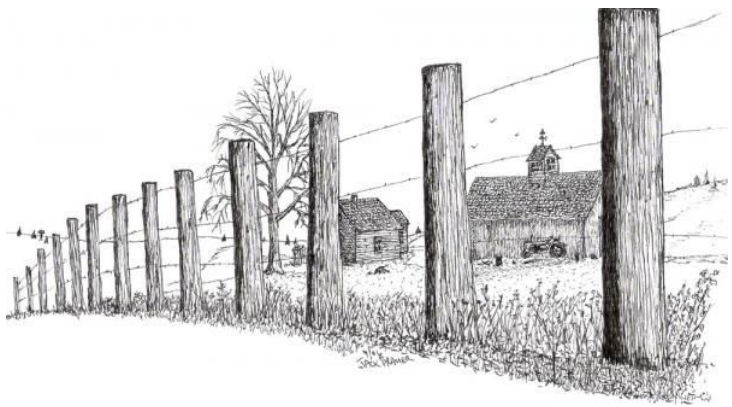
Two-Dimensional Problem to Solve (5/8)

What are boundary conditions? (2/4)

$\Omega = (0,1)^2$: Unit Square

Γ boundary of Ω - $\mathbf{n}$: outward normal on Γ





Two-Dimensional Problem to Solve (6/8)

What are boundary conditions? (3/4)

- direction or vector: $\mathbf{s} = (s_x, s_y) = s_x\,\mathbf{i} + s_y\,\mathbf{j}$
- directional derivative of $u(x, y)$ in the direction of $\mathbf{s}$

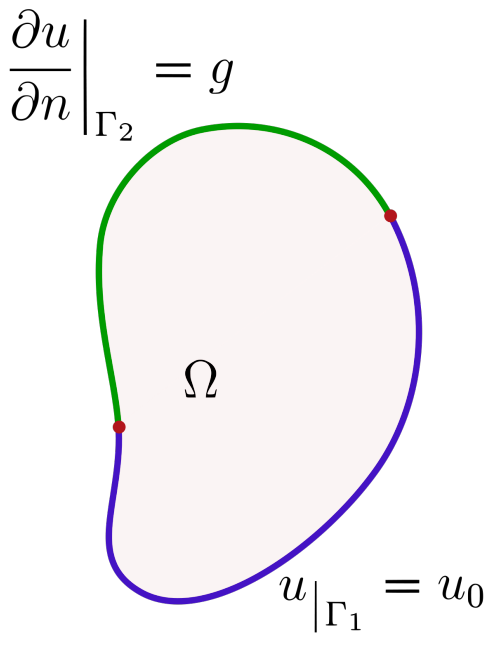
$$\frac{\partial u}{\partial s} = \underbrace{\nabla u \cdot \mathbf{s}}_{\textit{inner product}} = s_x \frac{\partial u}{\partial x} + s_y \frac{\partial u}{\partial x} \text{ (scalar)}$$

- choose $\mathbf{s} = \mathbf{n}$ or $\mathbf{s} = \mathbf{t}$: normal or tangential derivative
- example: electromagnetic force/torque computation using the Maxwell stress tensor

Two-Dimensional Problem to Solve (7/8)

What are boundary conditions? (4/4)

Two Types of Boundary Conditions: $\Gamma = \Gamma_D \cup \Gamma_N$



- Dirichlet condition on Γ_D

  fix u

  (equivalent of $x = 0$ in 1D)

- Neumann condition on Γ_N

  fix $\frac{\partial u}{\partial n}$

  (equivalent of $x = 1$ in 1D)

Two-Dimensional Problem to Solve (8/8)

Boundary Value Problem for Second Order Differential Equations

- given $(x, y) \in \Omega = (0, 1)^2$ with $\Gamma = \Gamma_D \cup \Gamma_N$ the boundary of Ω
- given: $f(x, y)$ given function and α given number
- find: $u(x, y)$ such that

$$-\triangle u(x, y) = f(x, y) \text{ for } 0 < x, y < 1 \text{ (differential equation on } \Omega)$$
$$u = 0 \text{ (Dirichlet boundary condition on } \Gamma_D)$$
$$\frac{\partial u}{\partial n} = \alpha \text{ (Neumann boundary condition on } \Gamma_N)$$

- differential equation invalid on Γ - boundary conditions valid on Γ

Example with Closed Form Solution (1/5)

Exercise: Verify Solution

- suppose given: $u(x, y) = x\,(x-1)\,y\,(y-1)$
- compute $-\triangle u(x, y)$
- evaluate $u(x, y)$ on Γ boundary of $(0, 1)^2$

Example with Closed Form Solution (2/5)

Exercise: Verify Solution

- suppose given: $u(x, y) = x\,(x-1)\,y\,(y-1)$
- then $\frac{\partial u}{\partial x} = (2\,x-1)\,y\,(y-1)$ and $\frac{\partial^2 u}{\partial x^2} = 2\,y\,(y-1)$
- thus $-\triangle u(x, y) = -2\,x\,(x-1) - 2\,y\,(y-1)$
- furthermore $u(x, y) = 0$ on Γ boundary of $(0, 1)^2$
- thus $u(x, y)$ solves

$$-\triangle u(x, y) = -2\,x\,(x-1) - 2\,y\,(y-1) \text{ on } \Omega$$
$$u = 0 \text{ on } \Gamma$$

Example with Closed Form Solution (3/5)

Exercise: Verify Solution Using Symbolic Computations

- using SymPy.jl
- `expr = x*(x-1)*y*(y-1)`
- `expr.diff(x,2) + expr.diff(y,2)`

Example with Closed Form Solution (4/5)

Construction of Analytical Reference Solution

- separation of variables $u(x, y) = X(x)\, Y(y)$
- $X(x)$ and $Y(y)$ solved separately as 1D problems
- heavily used in electrical machine design
- see References section for details
- in this course: not applied analytically
- in this course: applied as cheat-sheet to solve numerically

Example with Closed Form Solution (5/5)

Alternative Construction of Analytical Solution

- extrusion of 1D solution $u(x)$ in y-direction
- assume $u(x)$ analytical solution of $-u''(x) = f(x)$ + boundary conditions
- apply homogeneous Neumann boundary condition on south ($y = 0$) and north ($y = 1$) boundary
- put $f(x, y) = f(x)$
- then $u(x, y) = u(x)$ solution of $-u''(x, y) = f(x, y)$ + boundary conditions
- possibly valuable later

Mesh Generation (1/6)

As before - Independently in both directions

- equidistant mesh in both x and y-direction (insert figure here)
- mesh spacing: $\Delta x = \Delta y = h = 1/N$
- $x_i = (i-1)\,h$ and $y_j = (j-1)\,h$ for $1 \leq i, j \leq N+1$
- interior nodes: (x_i, y_j) for $2 \leq i, j \leq N$
- west boundary where $x = 0$: fix $i = 1$

  similar for other 3 boundaries

Mesh Generation (2/6)

grid ordering (left) and x-lexicografic ordering (right)

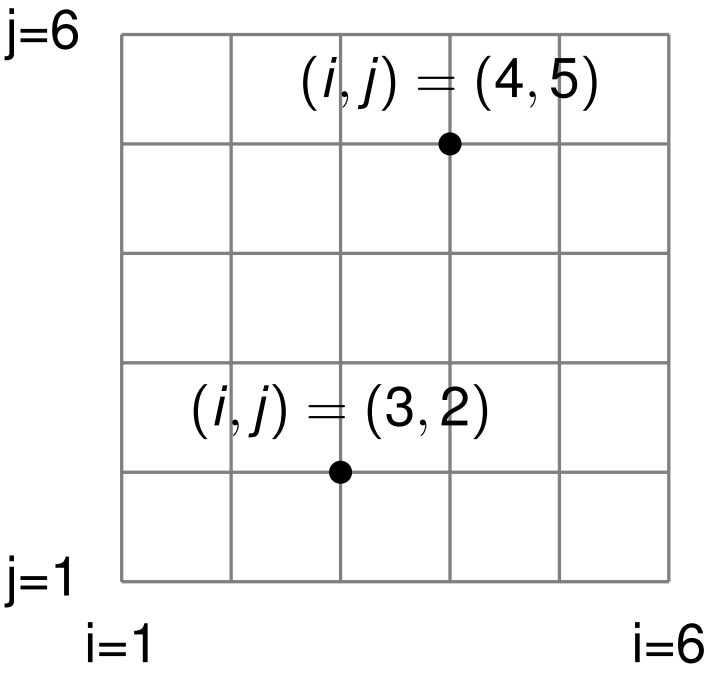


Figure: grid ordering using (i, j)

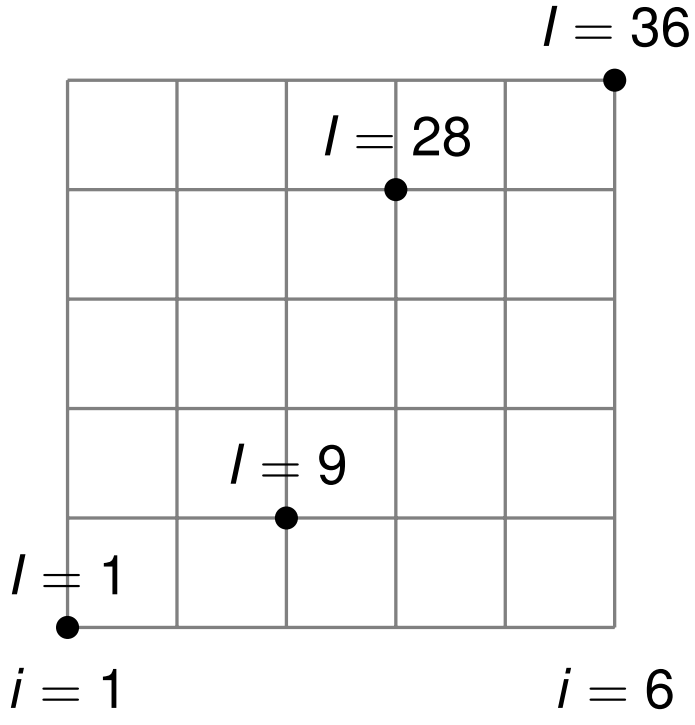


Figure: x-lexicografic using I

Mesh Generation (3/6)

Coding the Mesh in Julia

code from contour plot example in Plots.jl

```
#..define column vectors of samples in x-direction and y-direction
x = Vector(0:1/3:1);
y = Vector(0:1/3:1);

#..repeat vectors of samples to obtain matrices of 2D grid in x and y
#..reshape of x to horizontal coordinate
X = repeat(reshape(x, 1, :), length(y), 1);
Y = repeat(y, 1, length(x));
```

Mesh Generation (4/6)

Sample output for X and Y

$$X = \begin{pmatrix} 0 & 0.33 & 0.66 & 1 \\ 0 & 0.33 & 0.66 & 1 \\ 0 & 0.33 & 0.66 & 1 \\ 0 & 0.33 & 0.66 & 1 \end{pmatrix}$$ increases from left to right

reshape(x) - sampling in x-determines number of columns

$$Y = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.33 & 0.33 & 0.33 & 0.33 \\ 0.66 & 0.66 & 0.66 & 0.66 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$ increases from top to bottom

sampling in y-determines number of rows

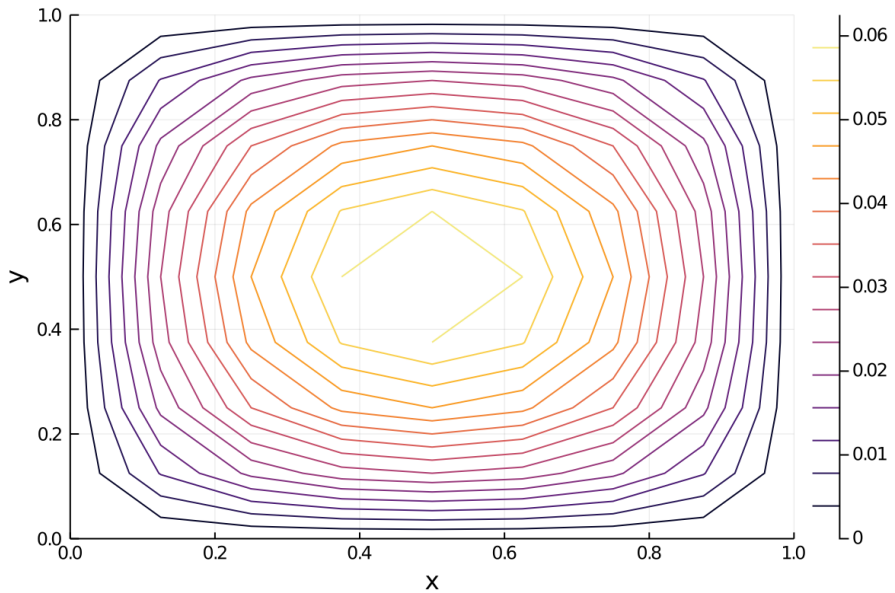
Mesh Generation (5/6)

Contour plots using Plots.jl: Three Alternatives

```
#..as before
x = Vector(0:1/100:1); y = Vector(0:1/100:1);
X = repeat(reshape(x, 1, :), length(y), 1); Y = repeat(y, 1, length(x));

#..define sourceterm as a function - can be anything else
function sourceterm(x,y)
    return x*(x-1)*y*(y-1)
end

#..alternative (1/3): compute function values using map
Z = map(sourceterm, X,Y);

#..alternative (2/3): compute function values using broadcast (dot)
Z = sourceterm.(X,Y);

#..alternative (3/3): place function inside of contour command
```

Mesh Generation (6/6)

contour(x,y,Z) of $u(x, y) = x\,(x-1)\,y\,(y-1)$

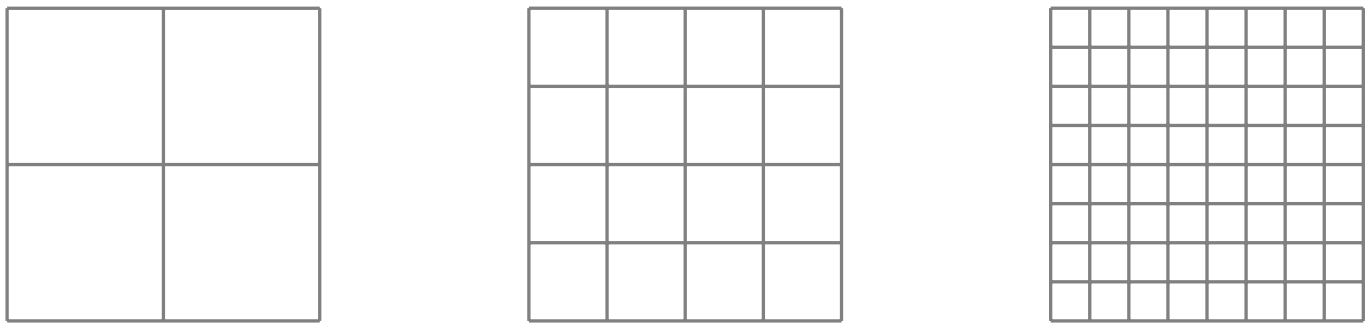


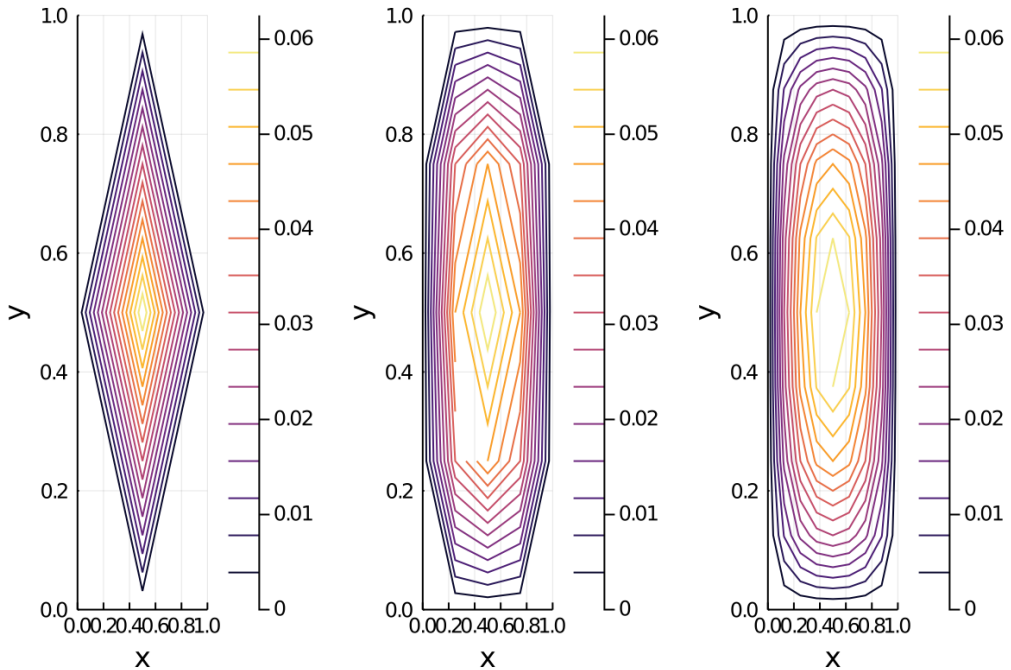
y-axis oriented upwards by explicitly providing y as argument

$u(x = .5, y = 0.5) = 1/16 = 0.0625$ reference value

Linear System (1/13)

FDM approximation of $u(x, y) = x\,(x-1)\,y\,(y-1)$





Linear System (2/13)

Questions

- numerical solution obtained from $A\,\mathbf{u} = \mathbf{f}$
- what is $\mathbf{f}$ for interior and boundary nodes?
- what is A for interior and boundary nodes?
- how to plot the numerical solution $\mathbf{u}$?

Linear System (3/13)

Discretization by collocation on mesh

- continuous problem

$$-u''(x,y) = f(x,y) + \text{boundary conditions for } (x,y) \in \Omega$$

- numerical problem

$$-u''(x_i,y_j) = f(x_i,y_j) + \text{boundary conditions for } (x_i,y_j) \in \Omega^h$$

Linear System (4/13)

Interior Nodes (x_i, y_j) for $2 \leq i, j \leq N$

- apply finite difference discretization in both x and y direction
- as before by varying x and keeping y fixed

$$\frac{\partial^2 u}{\partial x^2}(x = x_i, y = y_j) \approx \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} \quad (\leftrightarrow)$$

- likewise in y-direction and keeping x fixed

$$\frac{\partial^2 u}{\partial y^2}(x = x_i, y = y_j) \approx \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2} \quad (\updownarrow)$$

Linear System (5/13)

Interior Nodes (x_i, y_j) for $2 \leq i, j \leq N$

- add contributions in x and y-direction and multiply with -1

$$-u''(x = x_i, y = y_j) \approx \frac{-u_{i-1,j} - u_{i,j-1} + 4u_{i,j} - u_{i+1,j} - u_{i,j+1}}{h^2}$$

- $(N-1)^2$ equations for the interior nodes

$$\frac{-u_{i-1,j} - u_{i,j-1} + 4u_{i,j} - u_{i+1,j} - u_{i,j+1}}{h^2} = f(x_i, y_j)$$

Linear System (6/13)

Boundary Nodes (x_i, y_j) for $i, j \in \{1, N+1\}$

- Dirichlet boundary condition:

  force $u_{i,j}$ on Dirichlet boundary to comply with prescribed value

  for west boundary ($x = 0$) e.g.

$$u(0, y) = 0 \Rightarrow u_{1j} = 0$$

- Neumann boundary condition:

  apply finite difference approximation to the condition imposed

  for west boundary (outward normal in negative x-direction) e.g.

$$\frac{\partial u}{\partial n}(0, y) = \alpha \Leftrightarrow -\frac{\partial u}{\partial x}(0, y) = \alpha \Rightarrow -\frac{u_{2j} - u_{1j}}{h} = \alpha$$

Linear System (7/13)

- from above all entries for A and $\mathbf{f}$ can be derived

Linear System (8/13)

- construct **f** as a matrix using map on 2 matrices

  (see contour inside Plots.jl)

- reshape into vector to obtain vector for linear system solve

Linear System (9/13)

- A_{1d} matrix of the 1D problem - double derivative
- I identify matrix
- $\otimes$ Kronecker product of matrices
- $A_{xx} = A_{1d} \otimes I$: double derivative in x keeping y fixed
- $A_{yy} = I \otimes A_{1d}$: other way around
- $A = A_{xx} + A_{yy}$ is penta-diagonal (has 5 diagonals)
- details in lab session

Linear System (10/13)

- the entire code in shown in the the next three slides

Linear System (11/13)

Constructing of the mesh

```
#..load required packages
using LinearAlgebra
using Plots

#..(1/4): construct 2D mesh
#..set number of intervals (N)
N = 8;
#..compute derived numbers such as
#..the points on the boundary (Nbnd) and the mesh width (h)
Np1 = N+1; Nm1 = N-1; Np2 = Np1*Np1; Nm2 = Nm1*Nm1; Nbnd = 4*N; h = 1/N; h2=h*h;

#..construct the 2D mesh (X) starting from the 1D mesh (x)
#..observe that we make in the 2D mesh the x coordinate increase from left to right
#..the y coordinate increase from top to bottom (as expected)
x = Vector(0:h:1);
y = Vector(0:h:1);
X = repeat(reshape(x, 1, :), length(y), 1);
Y = repeat(y, 1, length(x));
```

Linear System (12/13)

Book keeping of interior and boundary nodes

```
#..(2/4): linear system set-up: boundary data
#..construct the mesh indicator matrix IG
#..this indicator matrix will allow to distinguish interior and boundary nodes
#..in this indicator matrix the boundary nodes are easy to identify
#..for interior nodes IG(i,j) = 0 and for the boundary nodes IG(i,j) = 1
#..next construct the indicator vector IGvec by reshaping the indicator matrix IG
IG = ones(Np1,Np1);
IG[2:end-1,2:end-1] = zeros(Nm1,Nm1);
IGvec = reshape(IG,Np2,1);
#..construct array with linear indices allowing to define interior and boundary node
#..interior: index array with all indices of the interior nodes
#..boundary: index array with indices of all the boundary nodes
L = LinearIndices(IGvec);
interior_cartesian = findall(x->x==0,IGvec);  interior = L[interior_cartesian];
boundary_cartesian = findall(x->x>0,IGvec); boundary = L[boundary_cartesian];

#..construct auxiliary vectors used in the 1D code as well
e = ones(Np1);              #..same as in 1D..
e_bnd = ones(Nbnd); #..used to handle the boundary nodes
```

Assembly of matrix, vector, linear solve and plotting follows

```
#..(2/4): linear system: right-hand side vector
#..define the source function
sourceterm(x,y) = - 2*x*(x-1) - 2*y*(y-1);
#..evaluate fsource on each node of the grid (Xh)
F = map(sourceterm, X,Y);
#..reshape the F 2D array into an f vector
f = reshape(F,Np2);

#..(2/4): linear system set-up: coefficient matrix
#..construct one-dimensional matrix
A1 = Tridiagonal(-e[2:end], 2*e, -e[2:end]);  A1 = (1/h2)*A1;
I    = Diagonal(e);
I_bnd = Diagonal(e_bnd);

#..construct the 2D matrix using Kronecker products
A = kron(A1,I) + kron(I,A1);

#..(2/4): linear system set-up: boundary conditions
A[boundary,boundary] = I_bnd; A[boundary,interior] = zeros(Nbnd, Nm2);
f[boundary] = zeros(Nbnd);

#..(3/4): solve the linear system for u
u = A \ f;

#..(4/4): plot computed solution
U = reshape(u,Np1,Np1);
contour(x, y, U, xlabel="x", ylabel="y")
```

Exercises

- mesh with $h = 1/3$ in x-direction and $h = 1/2$ in y-direction
- $4 * 3 = 12$ nodes in total. 2 interior nodes. 10 boundary nodes
- give the right-hand side vector **f**
- give matrix A