

10.34: Numerical Methods Applied to Chemical Engineering

Finite Volume Methods
Constructing Simulations of PDEs

Recap

- von Neumann stability analysis
- Finite volume methods

Finite Volume Method

- Generally used for conservation equations of the form:

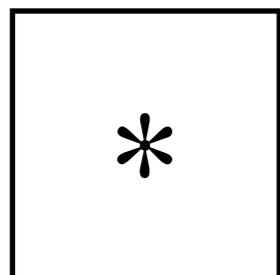
$$\frac{\partial b}{\partial t} = -\nabla \cdot \mathbf{j} + r(\mathbf{x}, t)$$

- $b(\mathbf{x}, t)$ is the density of a conserved quantity
- $\mathbf{j}(\mathbf{x}, t)$ is the flux density of a conserved quantity

- The integral version of such an equation is:

$$\frac{d}{dt} \int_{V^*} b(\mathbf{x}, t) dV = \int_{S^*} \mathbf{n} \cdot \mathbf{j}(\mathbf{x}, t) dS + \int_{V^*} r(\mathbf{x}, t) dV$$

or



$$\frac{d}{dt} B^*(t) = F^*(t) + R^*(t)$$

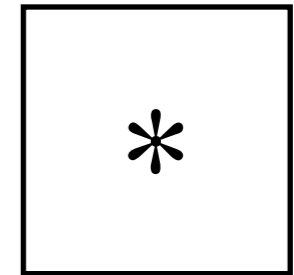
ACC IN/OUT GEN/CON

Finite Volume Method

- Conservation within a finite volume:

$$\frac{d}{dt} B^*(t) = F^*(t) + R^*(t)$$

ACC IN/OUT GEN/CON



- What are each of these terms?

- $B^*(t) = V^* \bar{b}^*(t)$

- $R^*(t) = V^* \bar{r}^*(t)$

- $F^*(t) = \sum_{k \in \text{faces}^*} F_k(t) = \sum_{k \in \text{faces}^*} A_k^* (\overline{\mathbf{n}_k \cdot \mathbf{j}})(t)$

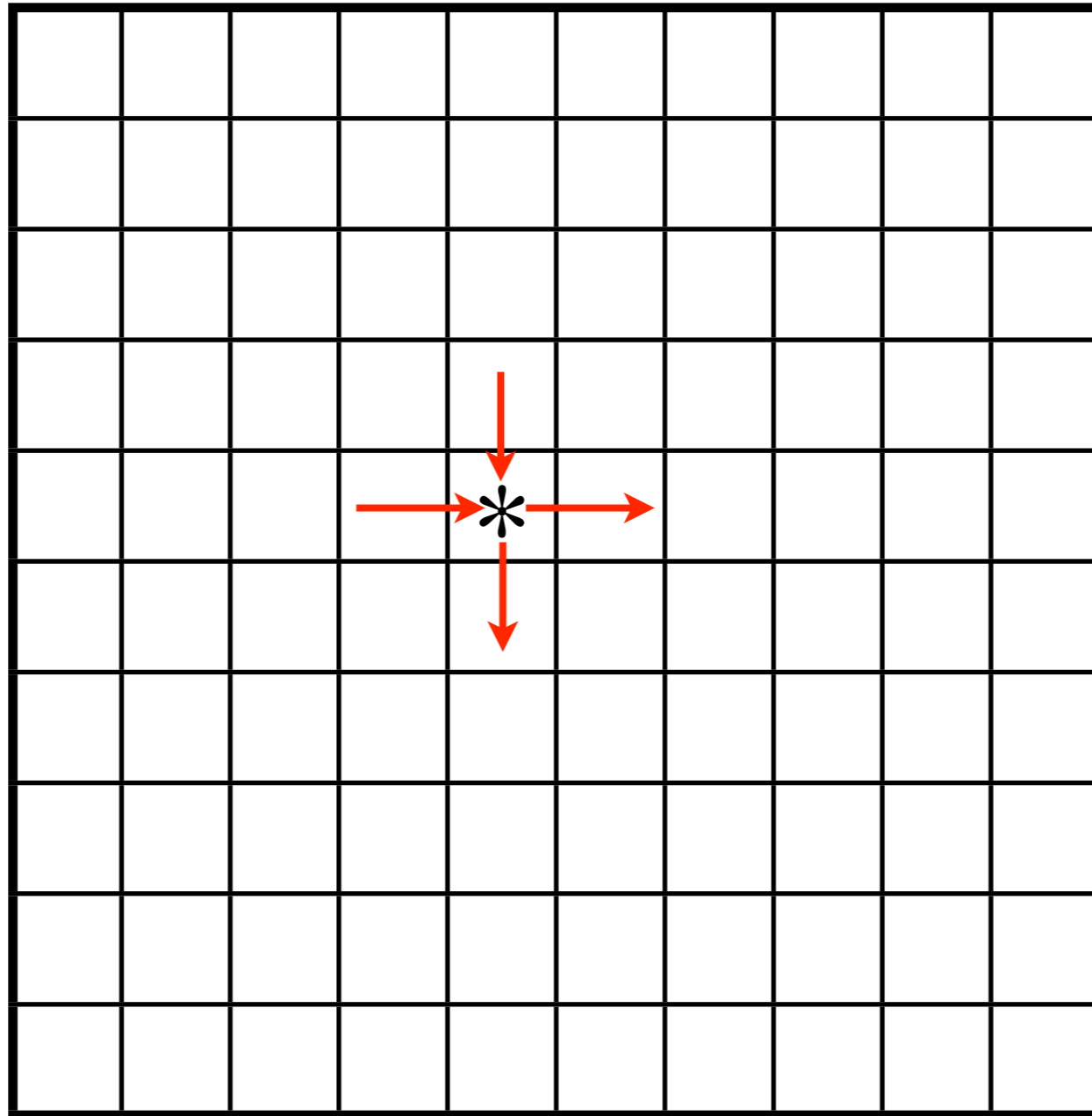
- the sum of fluxes through each face of the volume *

$$V^* \frac{d\bar{b}^*}{dt} = \sum_{k \in \text{faces}^*} F_k(t) + V^* \bar{r}^*(t)$$

- We want to solve for $\bar{b}(t)$ by approximating the reaction and flux terms. Let's construct low order approximations physically.

Finite Volume Method

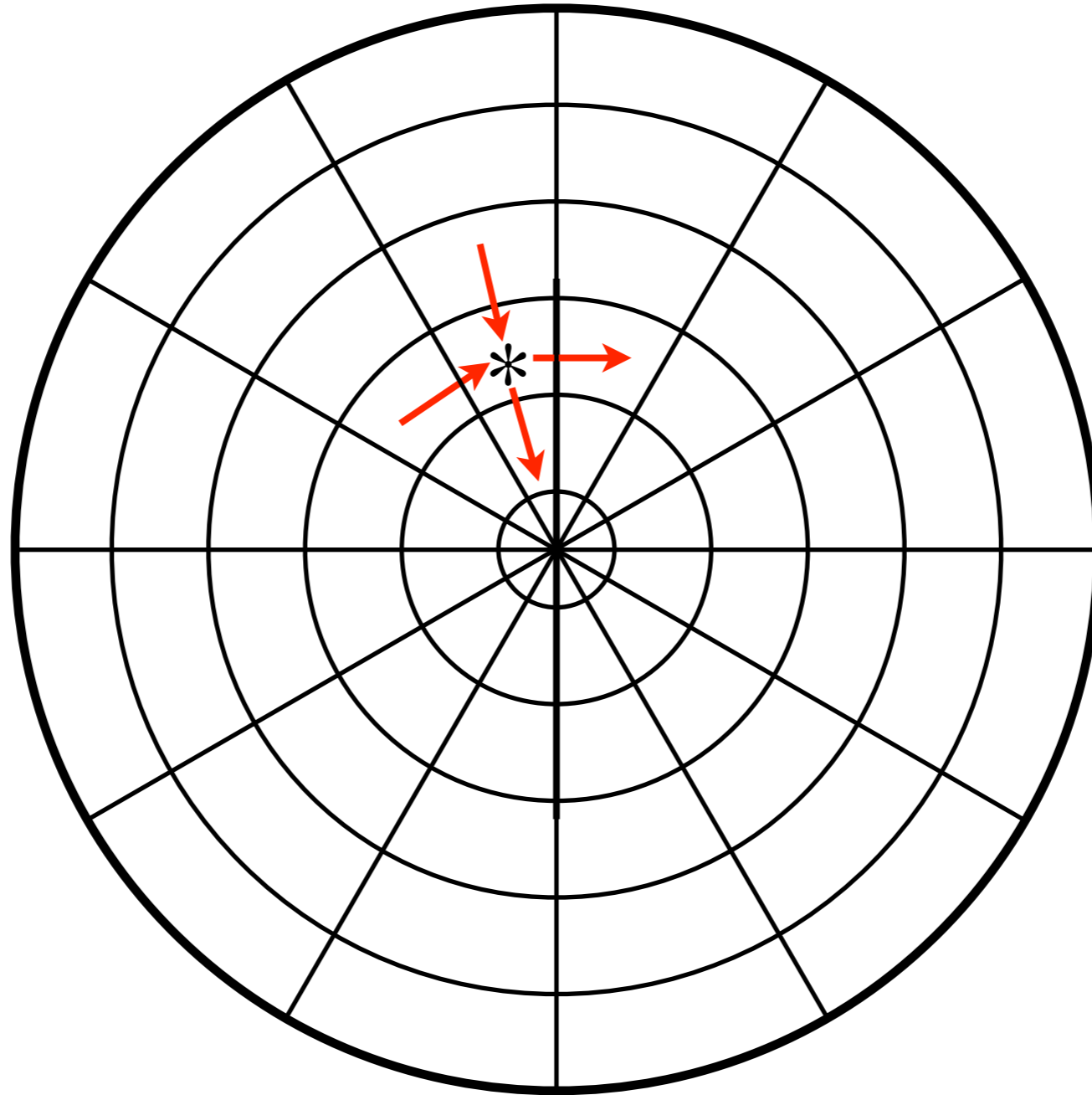
$$\frac{\partial b}{\partial t} = -\nabla \cdot \mathbf{j} + r(\mathbf{x}, t)$$



$$V^* \frac{d\bar{b}^*}{dt} = \sum_{k \in \text{faces}^*} F_k(t) + V^* \bar{r}^*(t)$$

Finite Volume Method

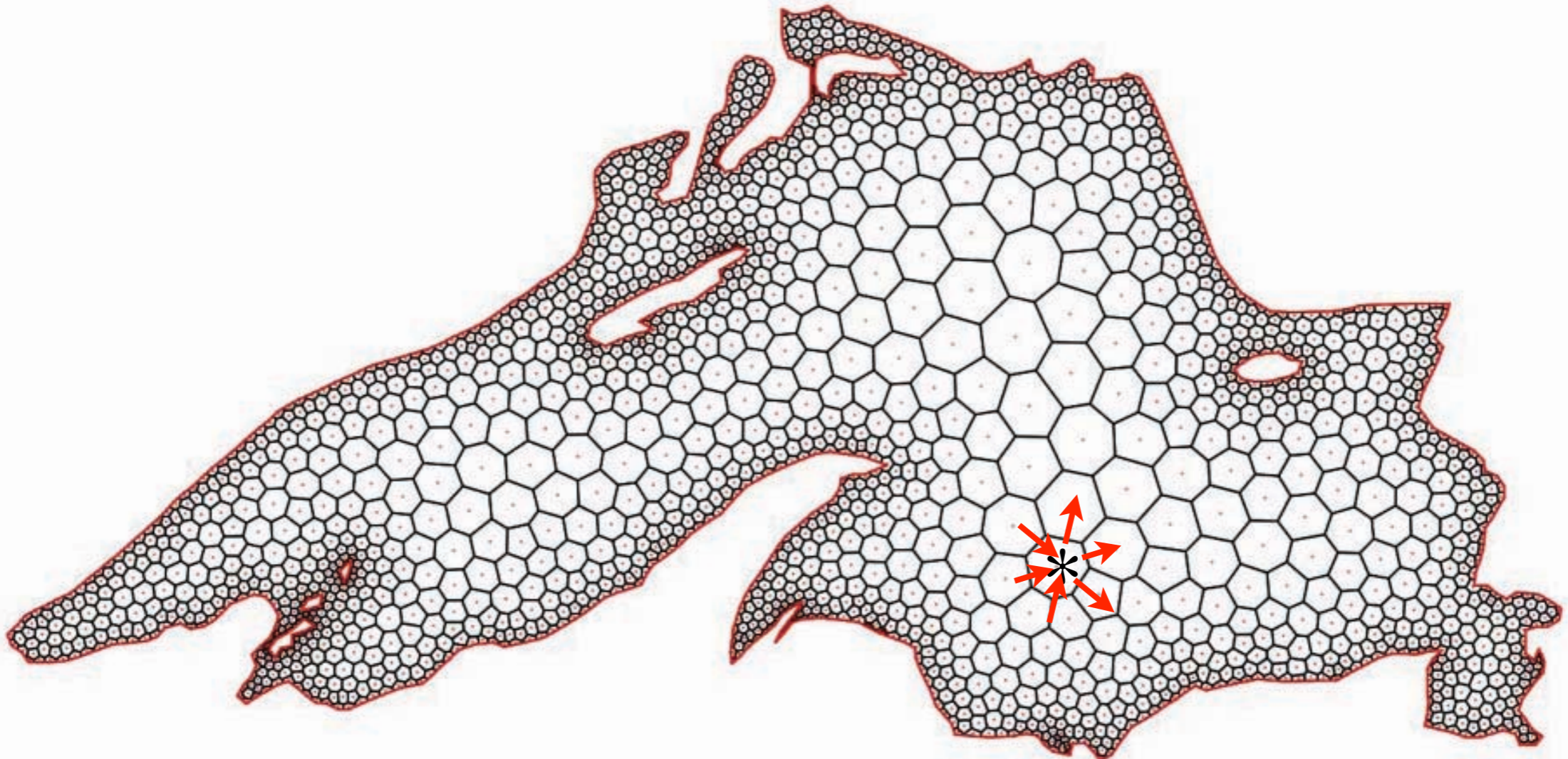
$$\frac{\partial b}{\partial t} = -\nabla \cdot \mathbf{j} + r(\mathbf{x}, t)$$



$$V^* \frac{d\bar{b}^*}{dt} = \sum_{k \in \text{faces}^*} F_k(t) + V^* \bar{r}^*(t)$$

Finite Volume Method

$$\frac{\partial b}{\partial t} = -\nabla \cdot \mathbf{j} + r(\mathbf{x}, t)$$

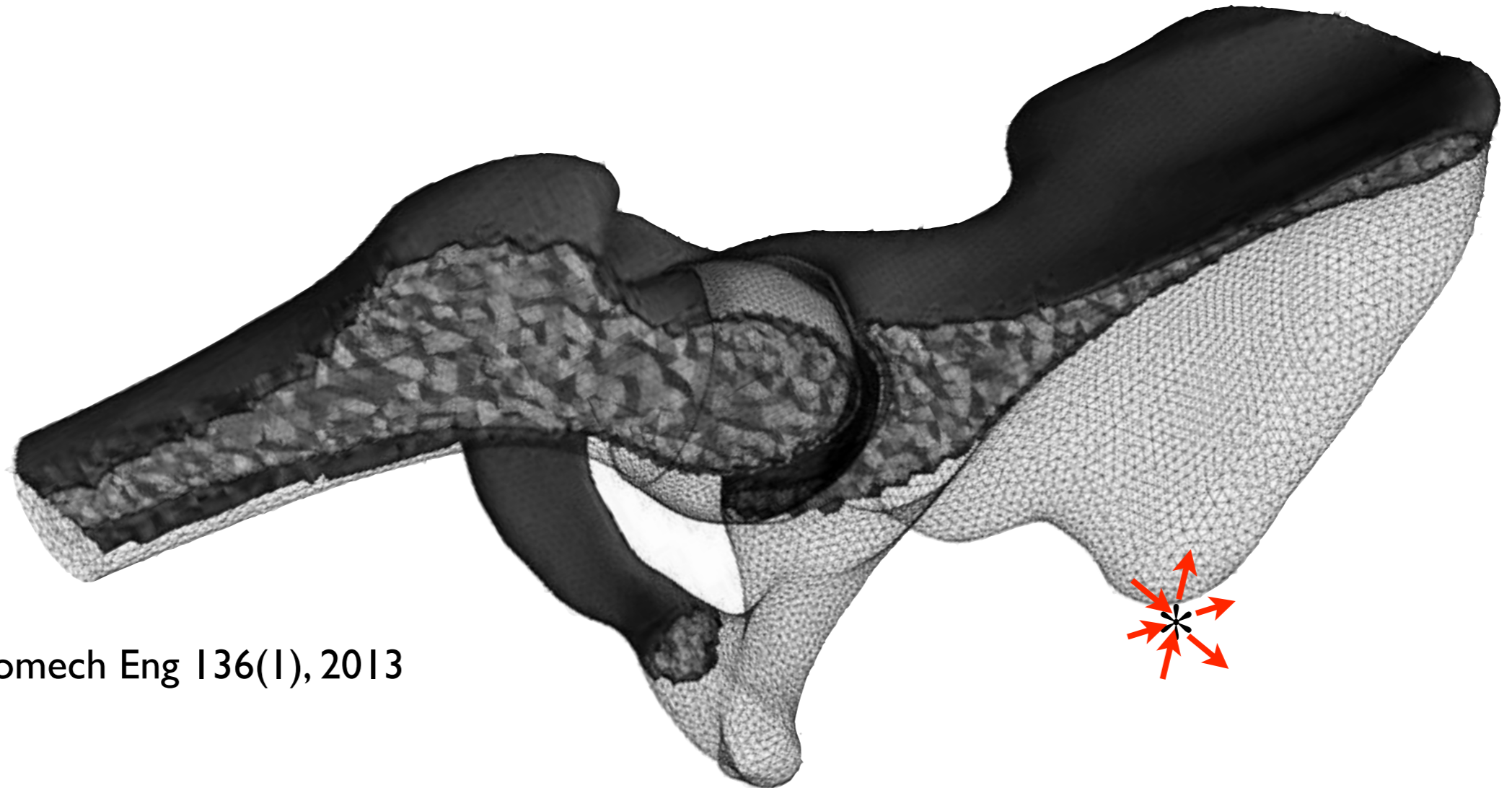


© INRIA. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use/>.

$$V^* \frac{d\bar{b}^*}{dt} = \sum_{k \in \text{faces}^*} F_k(t) + V^* \bar{r}^*(t)$$

Finite Volume Method

$$\frac{\partial b}{\partial t} = -\nabla \cdot \mathbf{j} + r(\mathbf{x}, t)$$



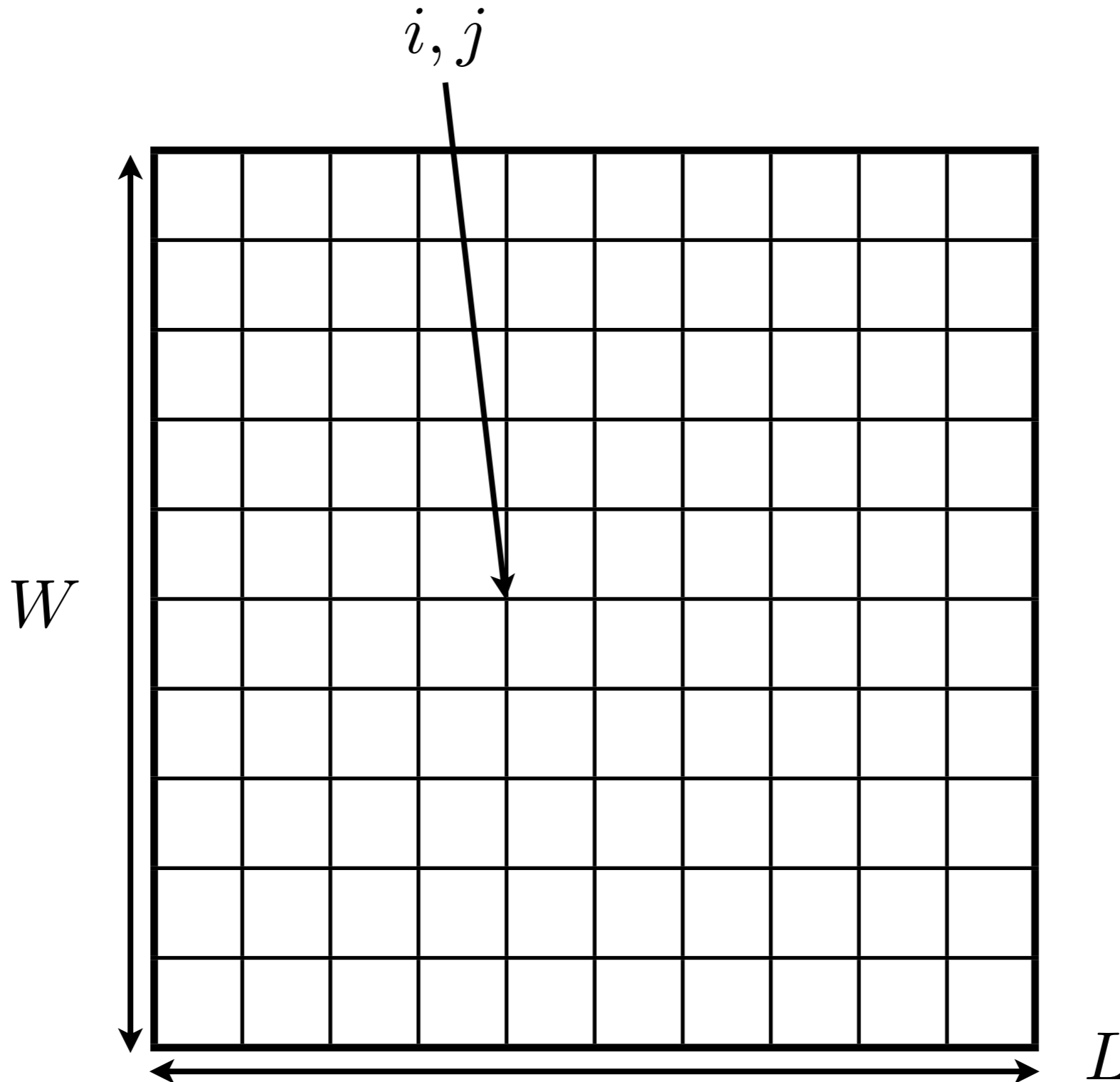
Cardiff et al. J Biomech Eng 136(1), 2013

© Cardiff, Philip et al. License: cc by-nc-nd. Some rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use/>.

$$V^* \frac{d\bar{b}^*}{dt} = \sum_{k \in \text{faces}^*} F_k(t) + V^* \bar{r}^*(t)$$

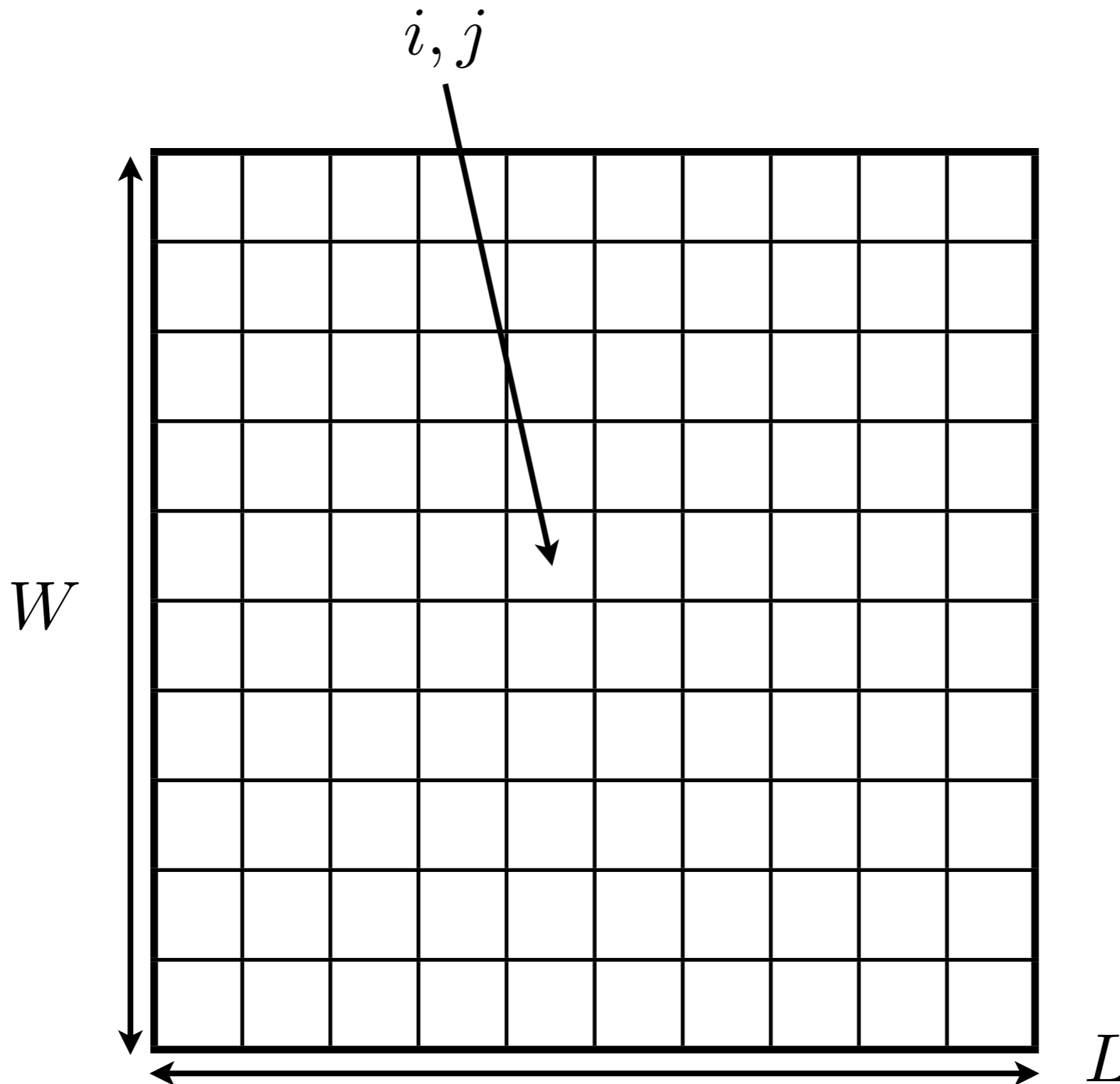
Numerical Solution of PDEs

Step I: domain decomposition
finite difference: nodes



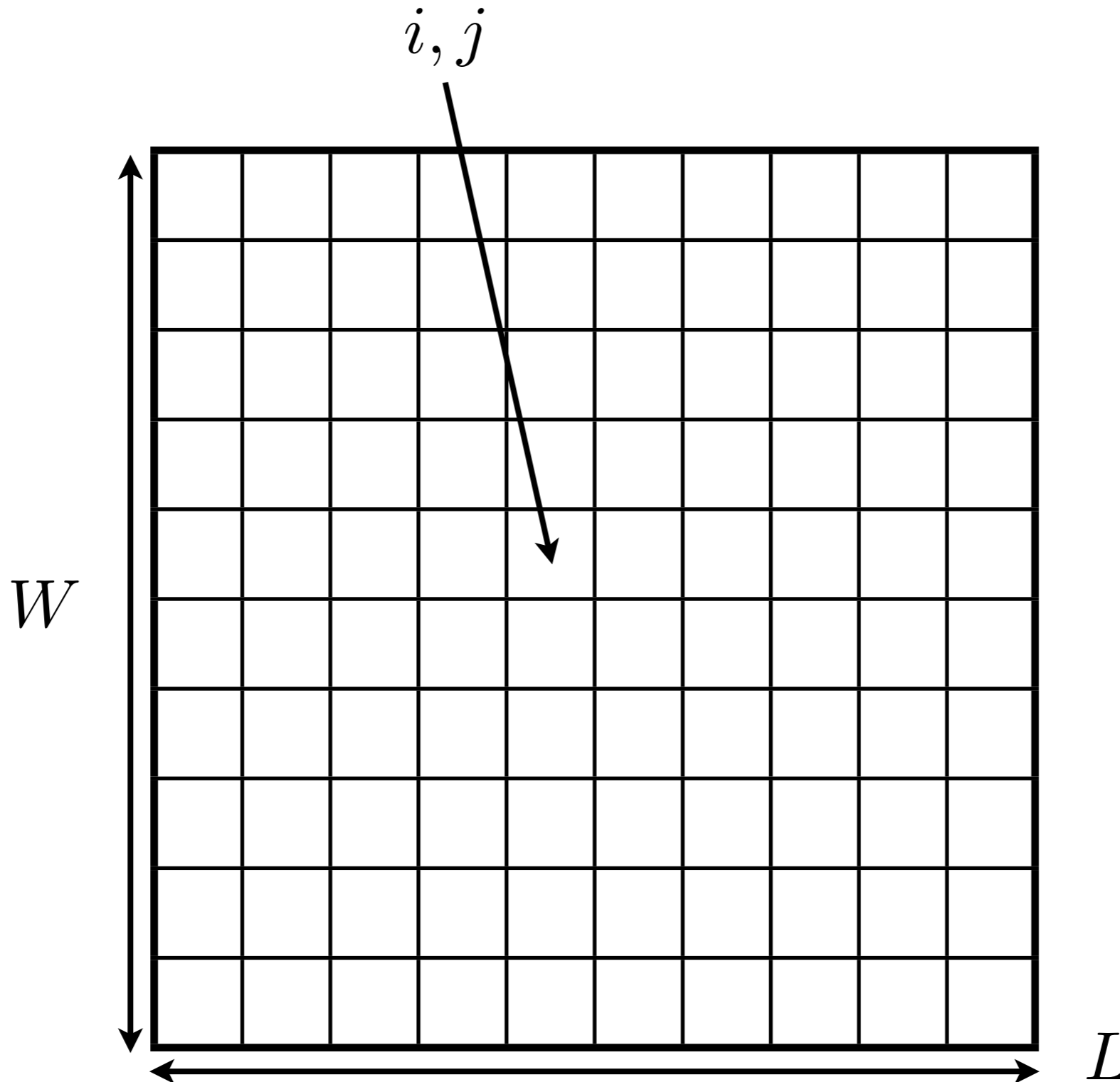
Numerical Solution of PDEs

Step I: domain decomposition
finite volume: cells



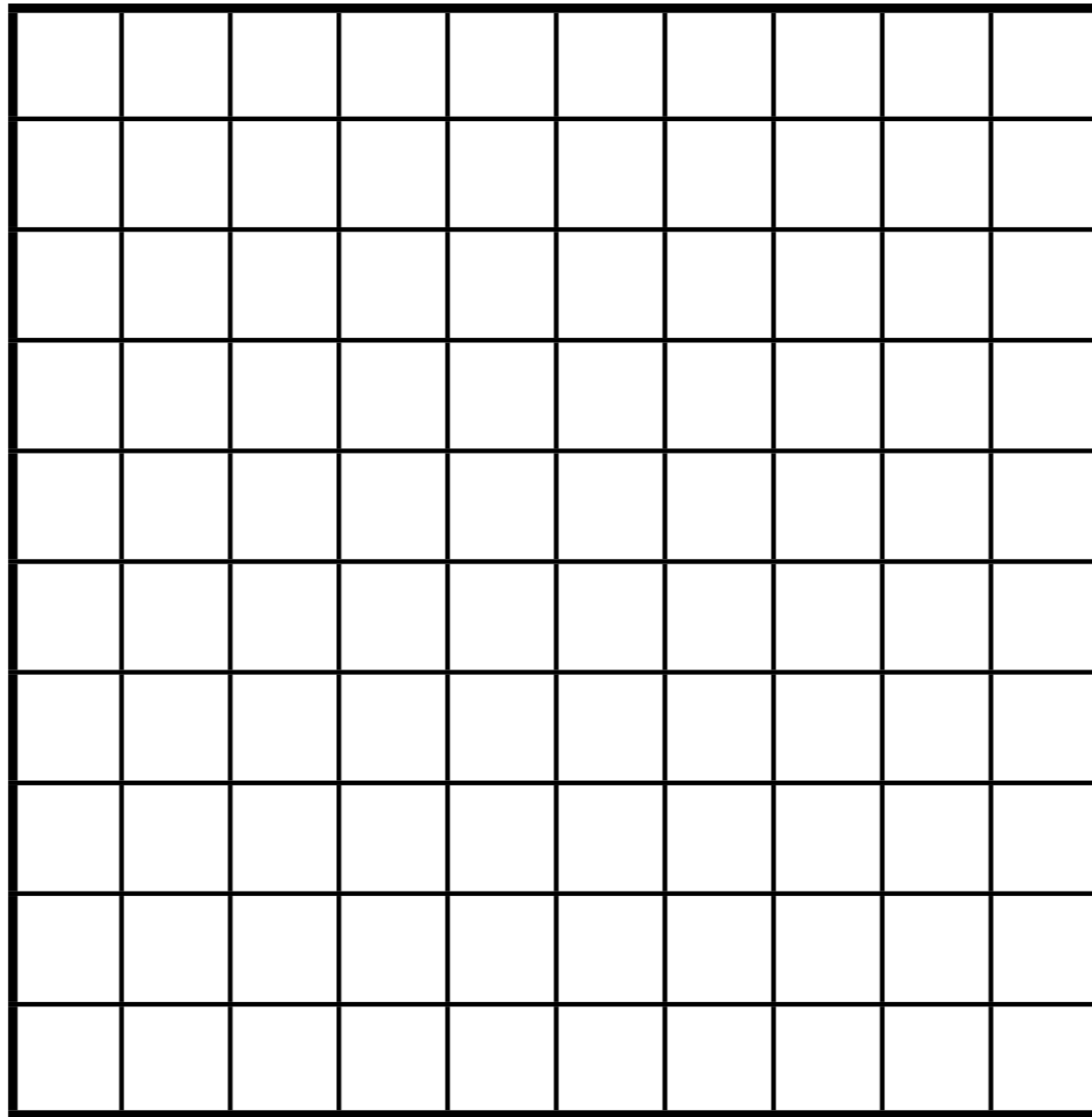
Numerical Solution of PDEs

Step I: domain decomposition
finite element: elements (local basis functions)

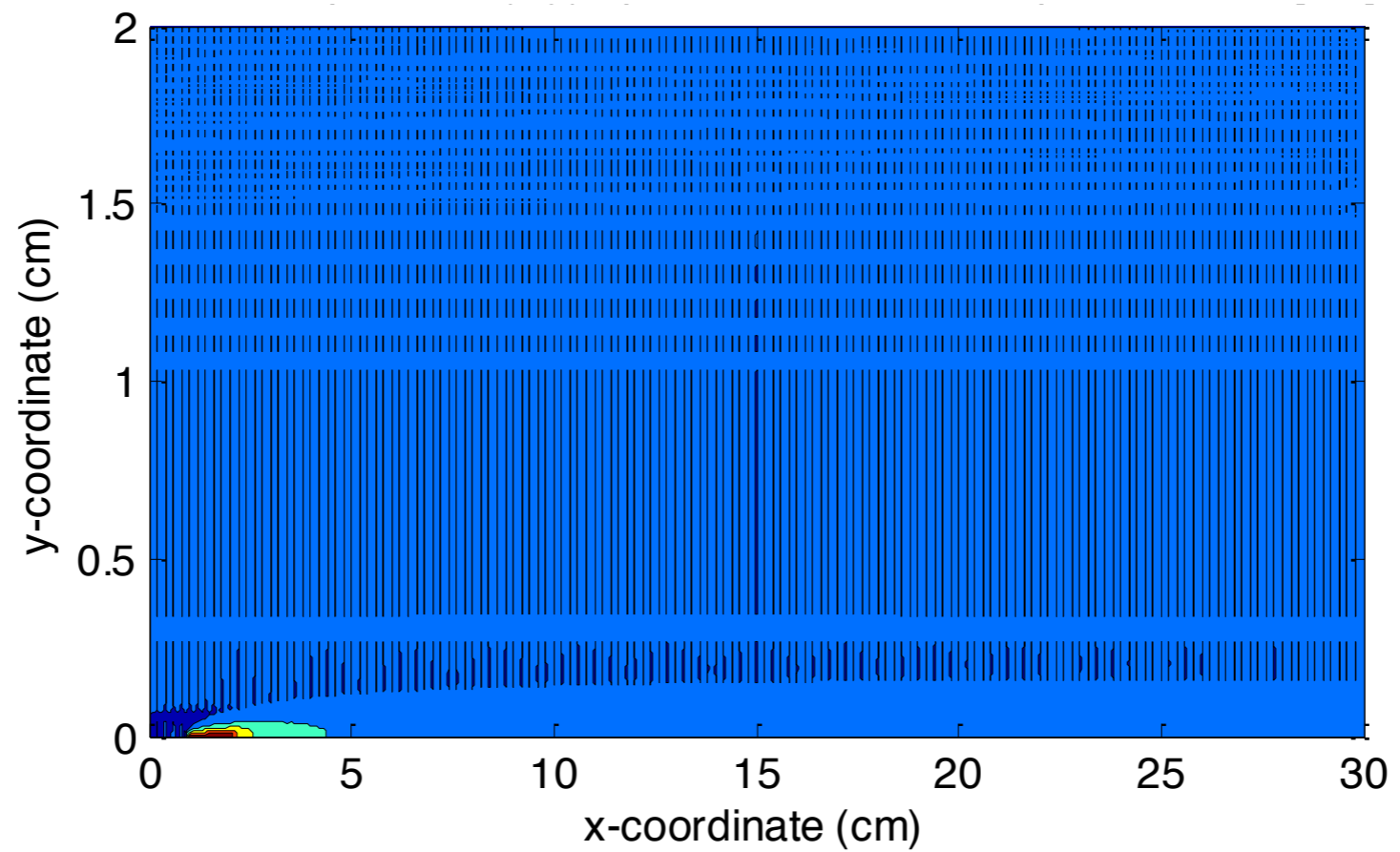
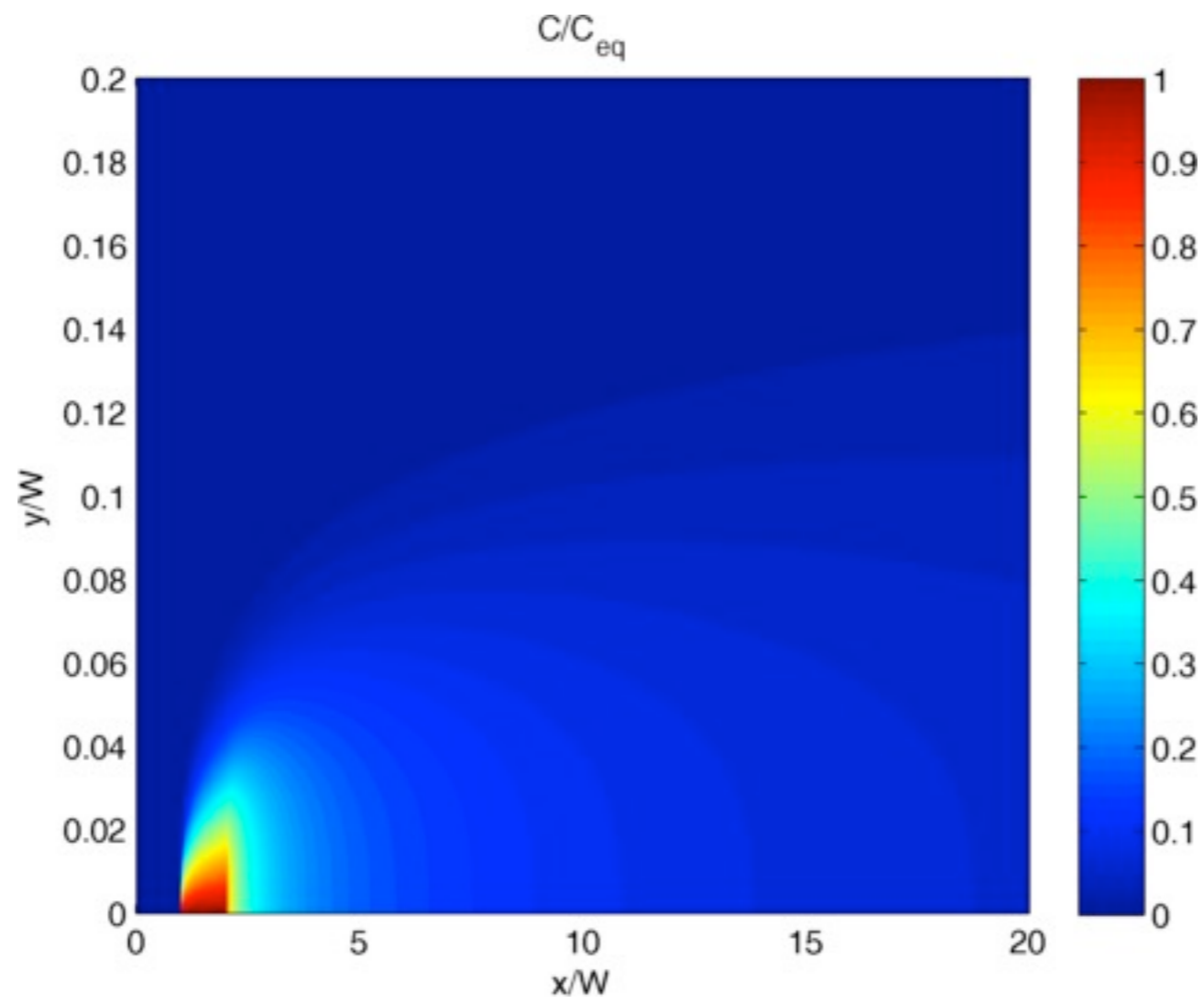


Numerical Solution of PDEs

Step 1: domain decomposition

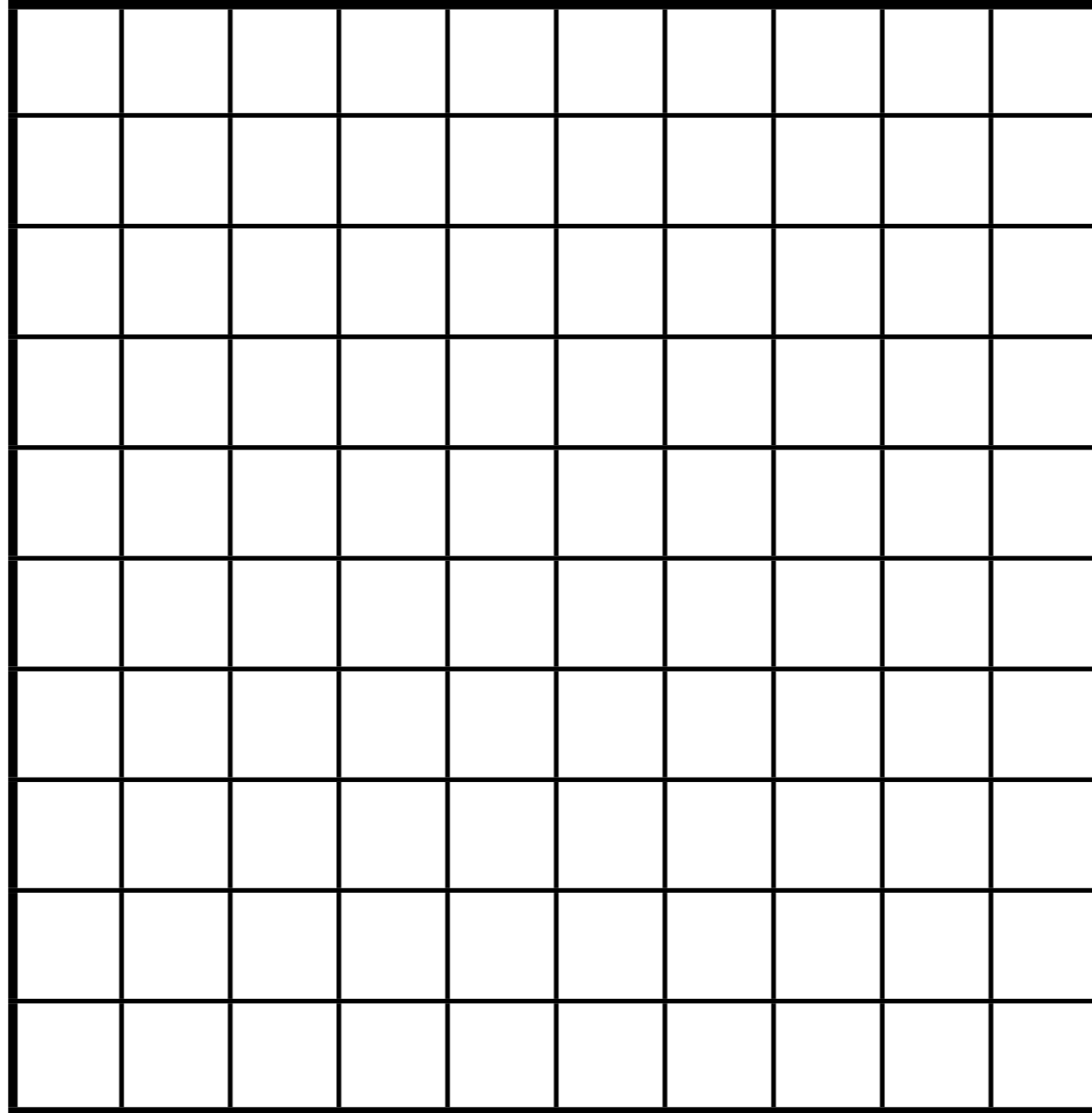


Always choose the spacing between nodes/dimensions of cells to match the physics. Never pick a certain number of nodes or cells *a priori*. That number is irrelevant.



Numerical Solution of PDEs

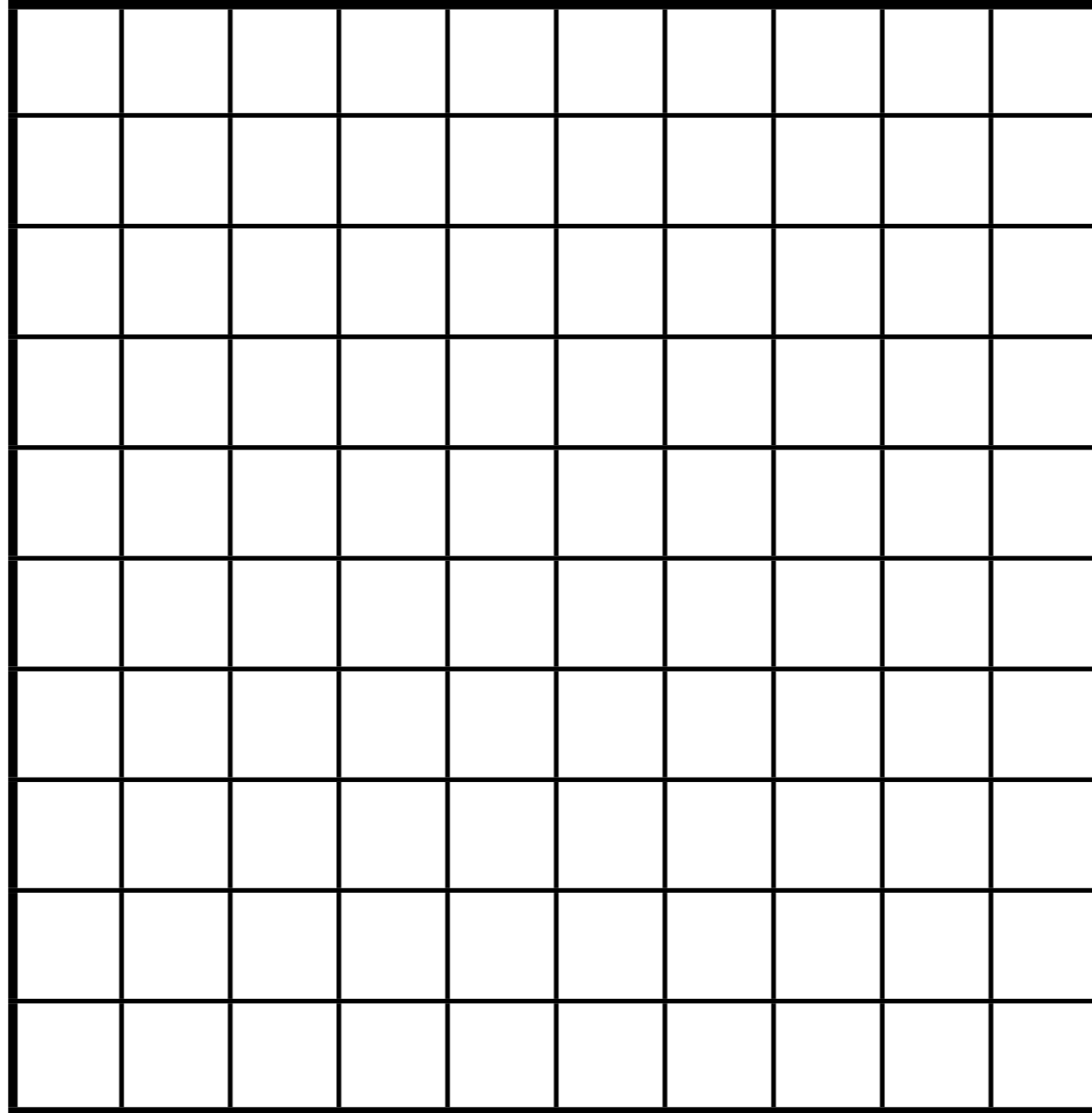
Step 2: formulate an equation to be satisfied at each node/cell



Numerical Solution of PDEs

Step 2: formulate an equation to be satisfied at each node/cell

Example: $\nabla^2 c = 0$ at interior node/cell i,j

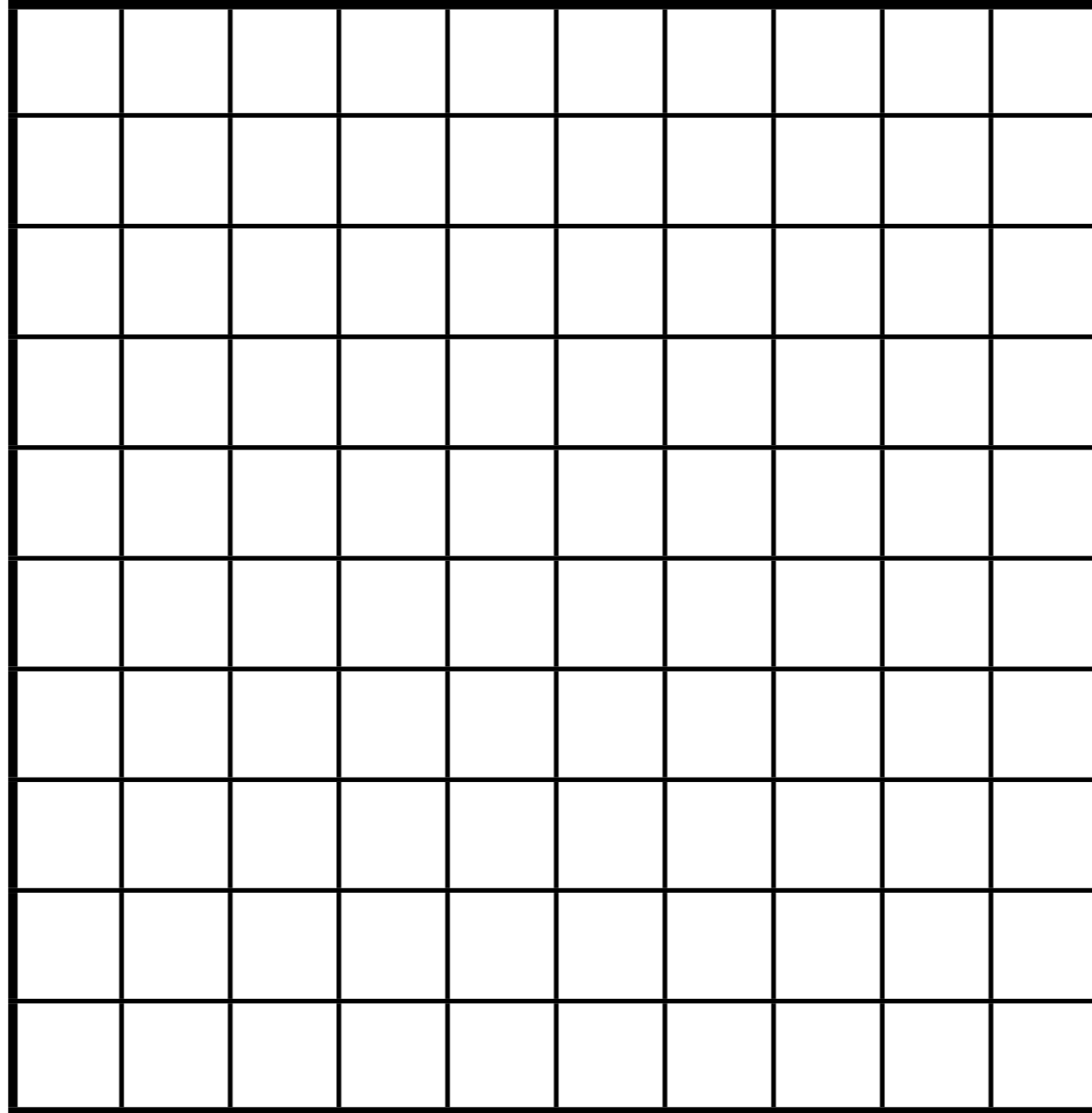


equation i,j : $c_{i+1,j} + c_{i-1,j} + c_{i,j-1} + c_{i,j+1} - 4c_{i,j} = 0$

Numerical Solution of PDEs

Step 2: formulate an equation to be satisfied at each node/cell

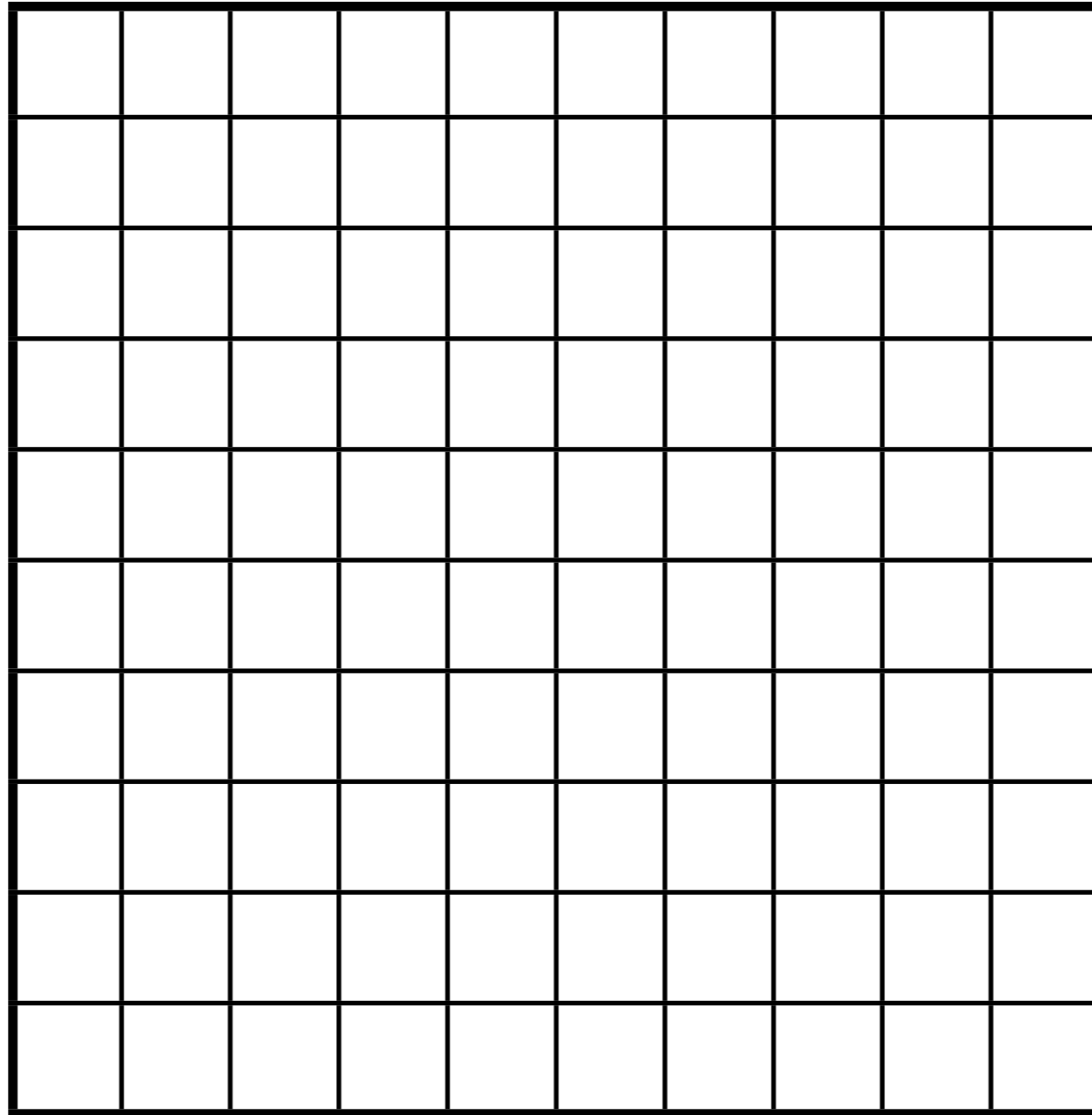
Example: $c = 1$ at boundary node/cell i,j



equation i,j : $c_{i,j} - 1 = 0$

Numerical Solution of PDEs

Step 3: solve the system of equations formulated at each node/cell for the value of unknown function at each node/cell



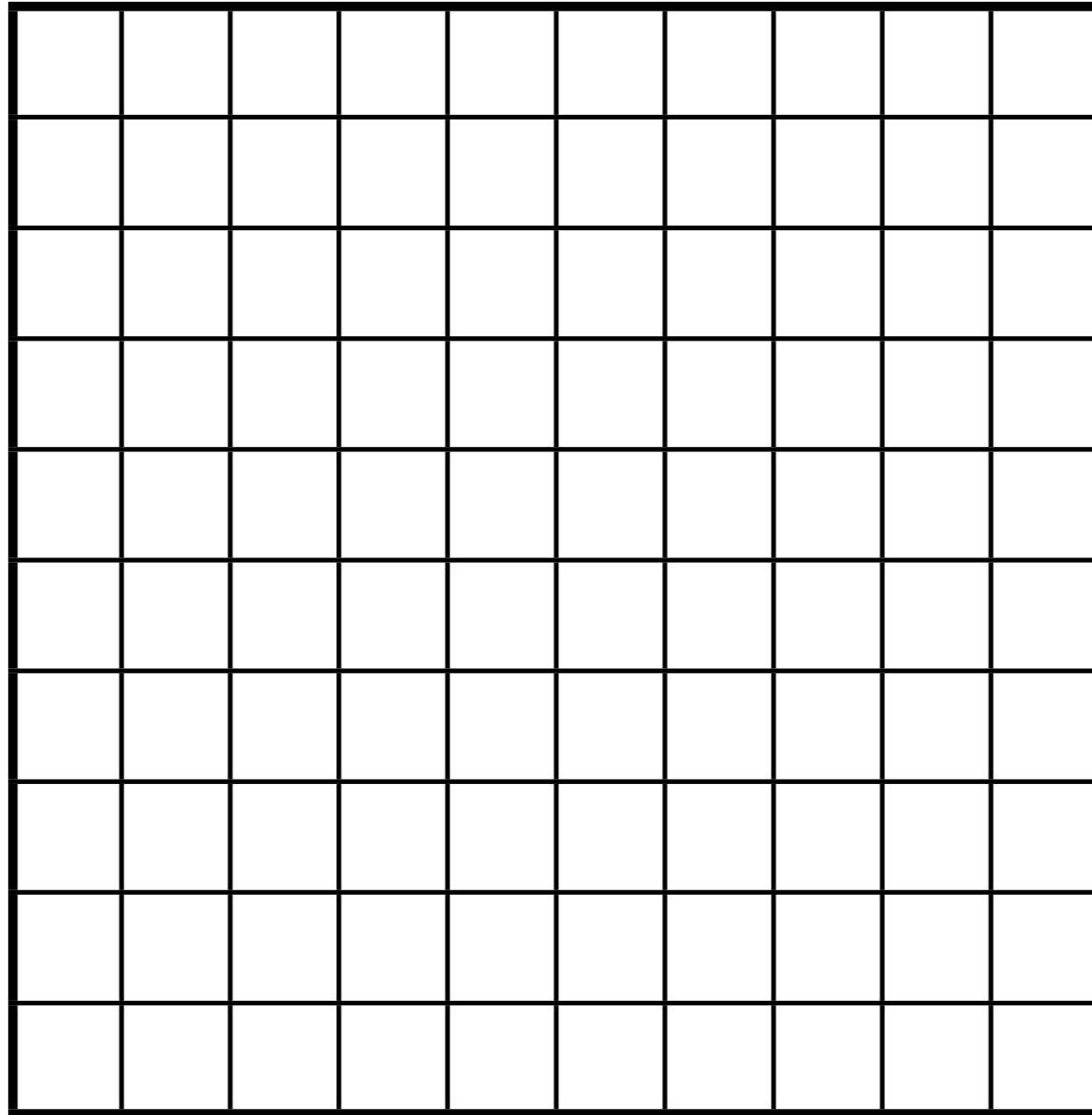
$$\mathbf{f}(\mathbf{c}) = 0$$

If equations are linear, use linear iterative methods

If equations are nonlinear, use nonlinear iterative methods

Numerical Solution of PDEs

Step 3: solve the system of equations formulated at each node/cell for the value of unknown function at each node/cell

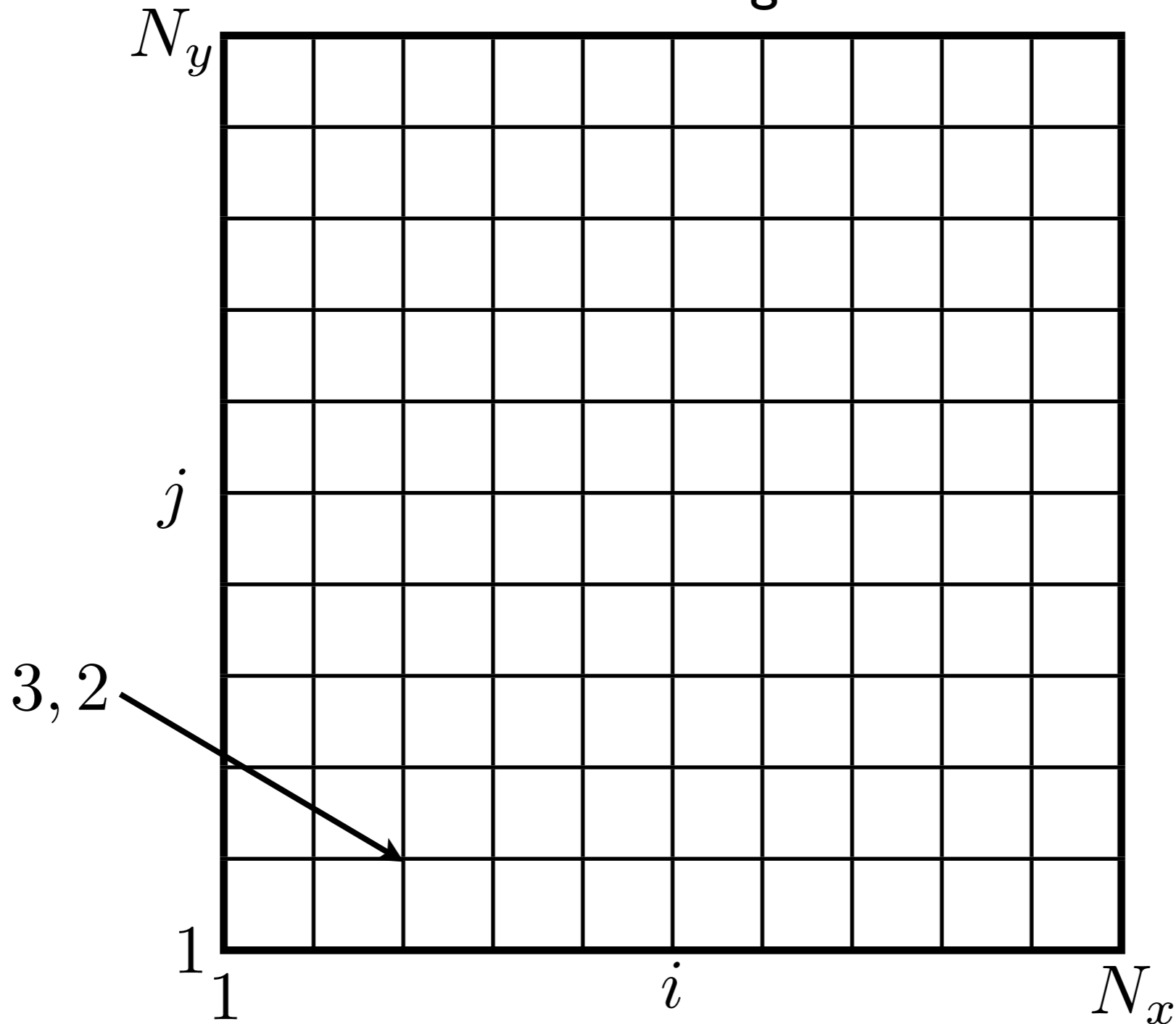


$$\mathbf{f}(\mathbf{c}) = 0$$

\mathbf{c} must be a vector of the unknowns
 \mathbf{f} must be a vector of the equations

Numerical Solution of PDEs

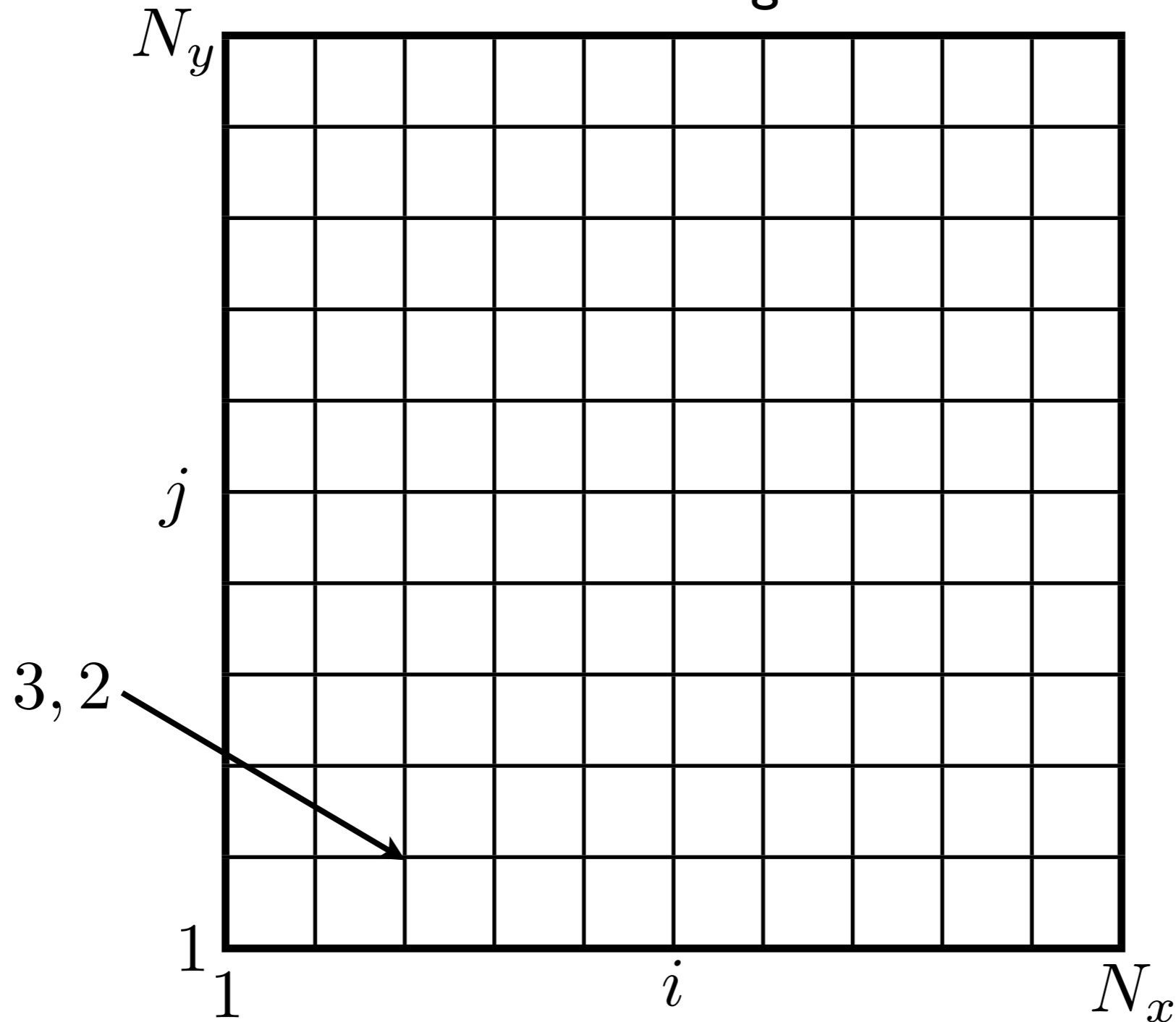
Indexing



$$c_k = c_{i,j} \quad k = i + (j - 1)N_x \quad c_{i,j+1} = c_{k+N_x}$$
$$k = j + (i - 1)N_y \quad c_{i,j+1} = c_{k+1}$$

Numerical Solution of PDEs

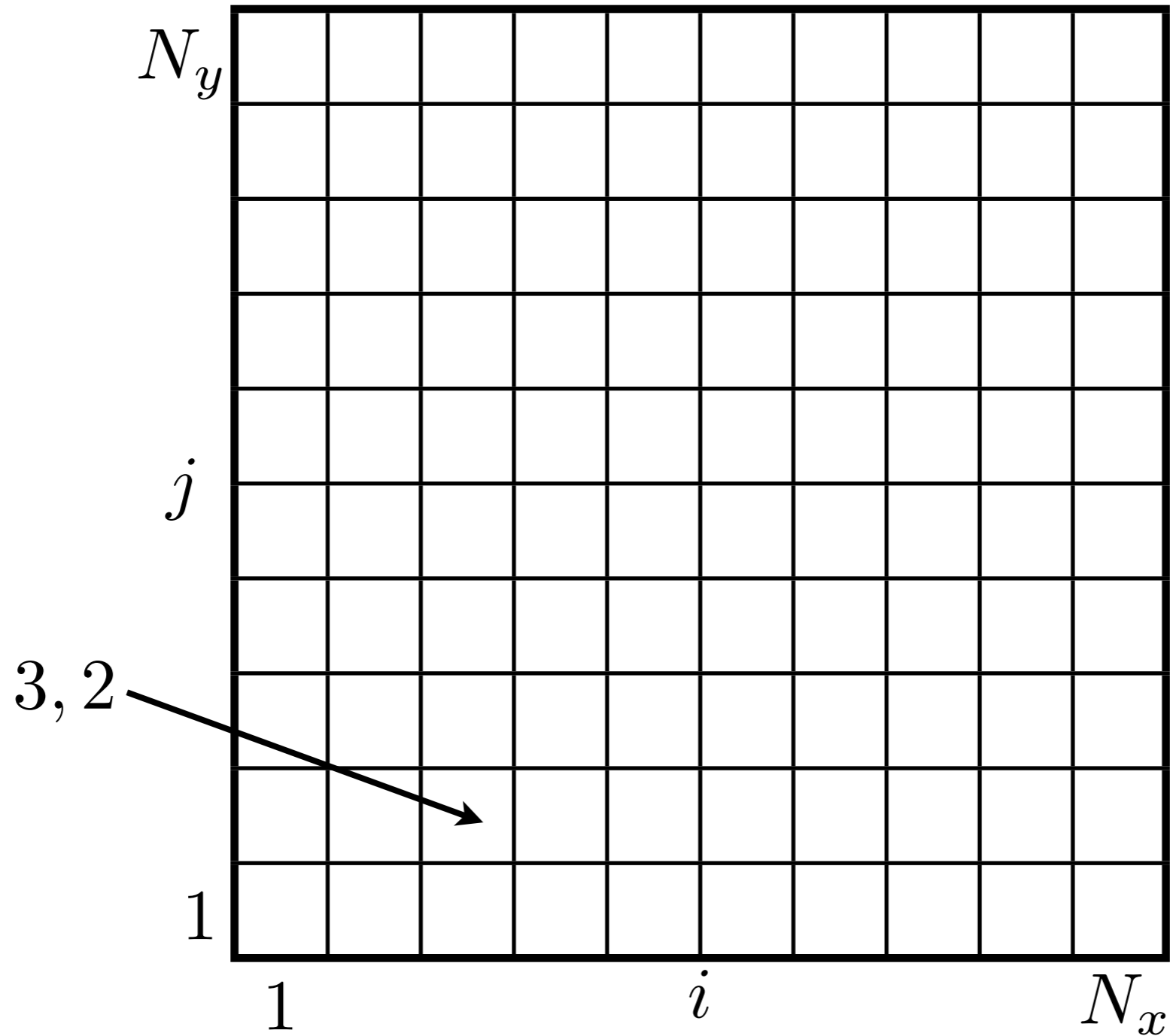
Indexing



$$f_k(\mathbf{c}) = f_{i,j}(\mathbf{c}) \quad \begin{array}{l} k = i + (j - 1)N_x \\ \text{or} \\ k = j + (i - 1)N_y \end{array}$$

Numerical Solution of PDEs

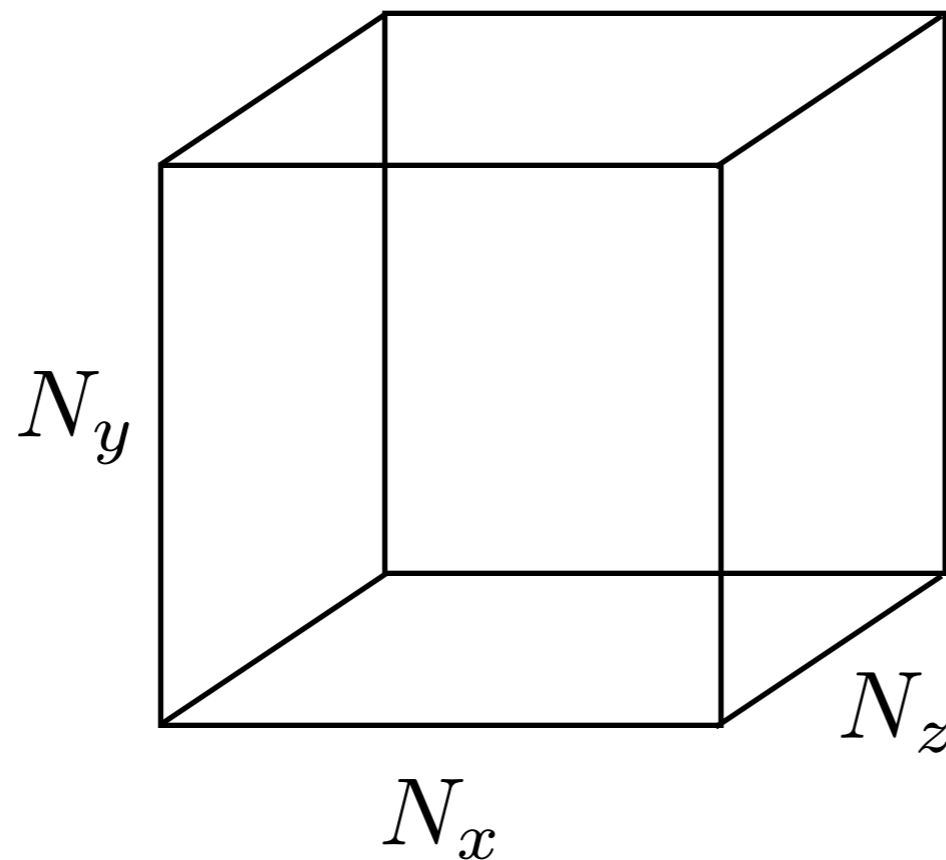
Indexing



$$f_k(\mathbf{c}) = f_{i,j}(\mathbf{c}) \quad \begin{array}{l} k = i + (j - 1)N_x \\ \text{or} \\ k = j + (i - 1)N_y \end{array}$$

Numerical Solution of PDEs

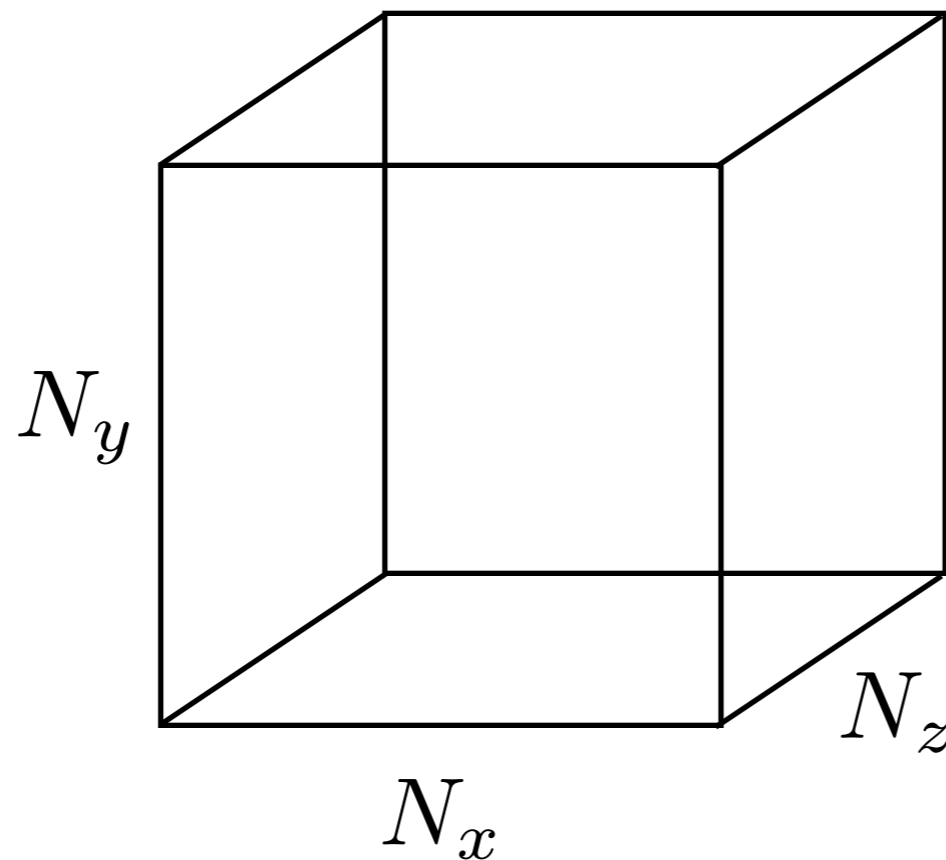
Exercise: write a single index for finite difference nodes in a cubic domain with (N_x, N_y, N_z) nodes in each cartesian direction



$$c_l = c_{i,j,k}, \quad l = ?$$

Numerical Solution of PDEs

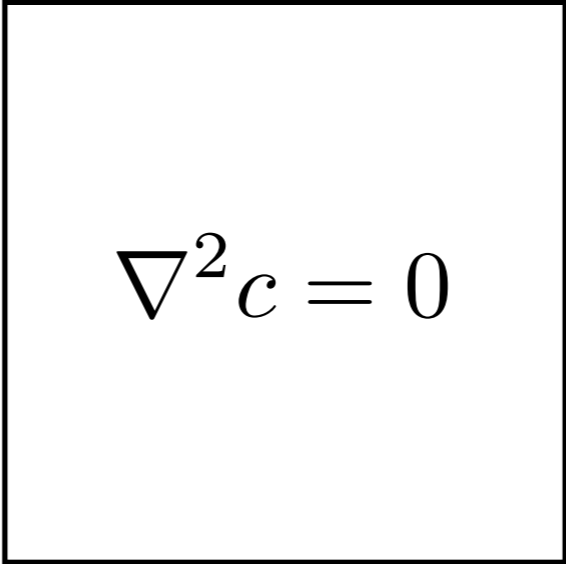
Exercise: write a single index for finite difference nodes in a cubic domain with (N_x, N_y, N_z) nodes in each cartesian direction



$$c_l = c_{i,j,k}, \quad l = i + (j - 1)N_x + (k - 1)N_xN_y$$

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.



A square domain is shown with boundary conditions and a central PDE. The top boundary is labeled $c = 0$, the bottom boundary is labeled $c = 1$, the left boundary is labeled $c = 0$, and the right boundary is labeled $c = 0$. Inside the square, the PDE is given as $\nabla^2 c = 0$.

$$\mathbf{f}(\mathbf{c}) = 0$$

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

```
h = 1 / 10; % Spacing between finite difference nodes
Nx = 1 + 1 / h; % Number of nodes in x-direction
Ny = Nx; % Number of nodes in y-direction

c0 = zeros( Nx * Ny, 1 ); % Initial guess for solution

c = fsolve( @( c ) my_func( c, Nx, Ny ), c0 ); % Find root of FD equations
```

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

```
function f = my_func( c, Nx, Ny )
```

```
    % Loop over all nodes
```

```
    for i = 1:Nx
```

```
        for j = 1:Ny
```

```
            k = i + ( j - 1 ) * Nx; % Compound index
```

```
            % Boundary nodes
```

```
            if ( i == 1 )
```

```
                f( k ) = c( k );
```

```
            elseif ( i == Nx )
```

```
                f( k ) = c( k );
```

```
            elseif( j == 1 )
```

```
                f( k ) = c( k ) - 1;
```

```
            elseif( j == Ny )
```

```
                f( k ) = c( k );
```

```
            % Interior nodes
```

```
            else
```

```
                f( k ) = c( k + 1 ) + c( k - 1 ) + c( k - Nx ) + c( k + Nx ) - 4*c( k );
```

```
            end;
```

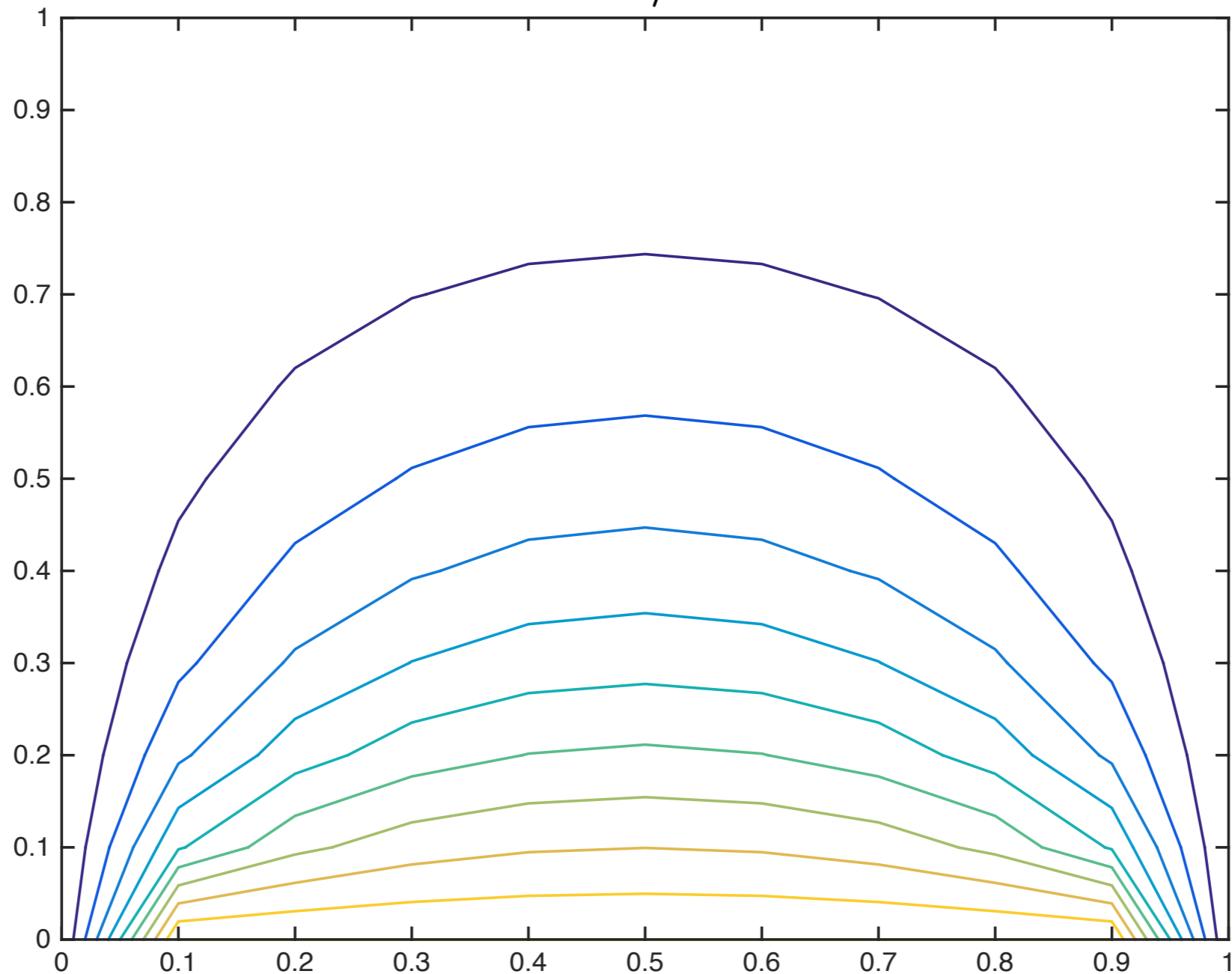
```
        end;
```

```
    end;
```


Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

$$h = 1/10$$

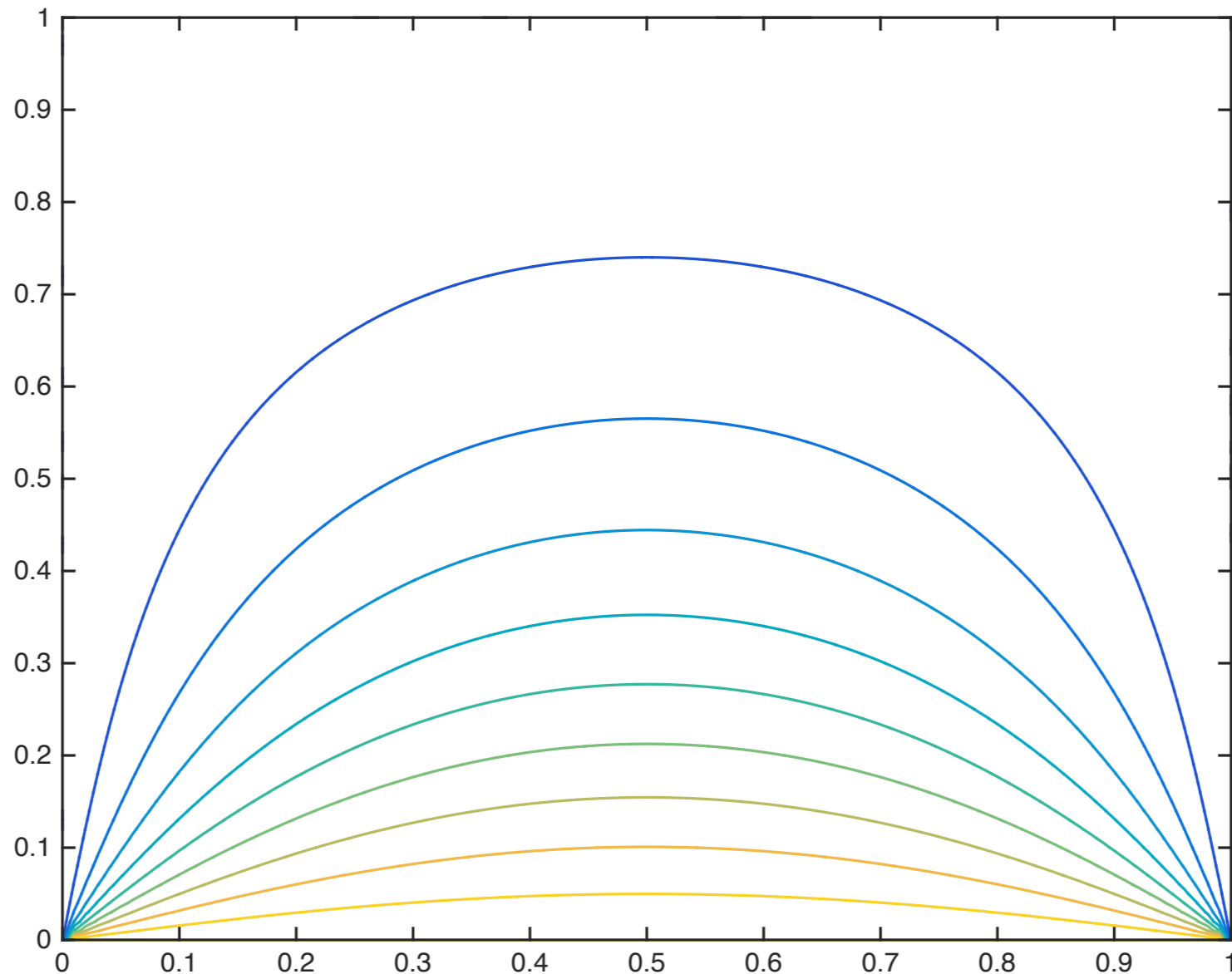


0.08 seconds to solve

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

$$h = 1/100$$



700 seconds to solve!
Why is it almost 10,000x slower?

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

$$\mathbf{f}(\mathbf{c}) = 0 = \mathbf{A}\mathbf{c} - \mathbf{b}$$

```
function [ Ac, b ] = my_func( c, Nx, Ny )
```

```
Ac = sparse( Nx * Ny, 1 );  
b = sparse( Nx * Ny, 1 );
```

```
% Loop over all nodes
```

```
for i = 1:Nx
```

```
    for j = 1:Ny
```

```
        k = i + ( j - 1 ) * Nx; % Compound index
```

```
        % Boundary nodes
```

```
        if ( i == 1 )
```

```
            Ac( k ) = c( k );
```

```
        elseif ( i == Nx )
```

```
            Ac( k ) = c( k );
```

```
        elseif( j == 1 )
```

```
            Ac( k ) = c( k );
```

```
            b( k ) = 1;
```

```
        elseif( j == Ny )
```

```
            Ac( k ) = c( k );
```

```
        % Interior nodes
```

```
        else
```

```
            Ac( k ) = c( k + 1 ) + c( k - 1 ) + c( k - Nx ) + c( k + Nx ) - 4*c( k );
```

```
        end;
```

```
    end;
```

```
end;
```

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

```
h = 1 / 10; % Spacing between finite difference nodes
Nx = 1 + 1 / h; % Number of nodes in x-direction
Ny = Nx; % Number of nodes in y-direction

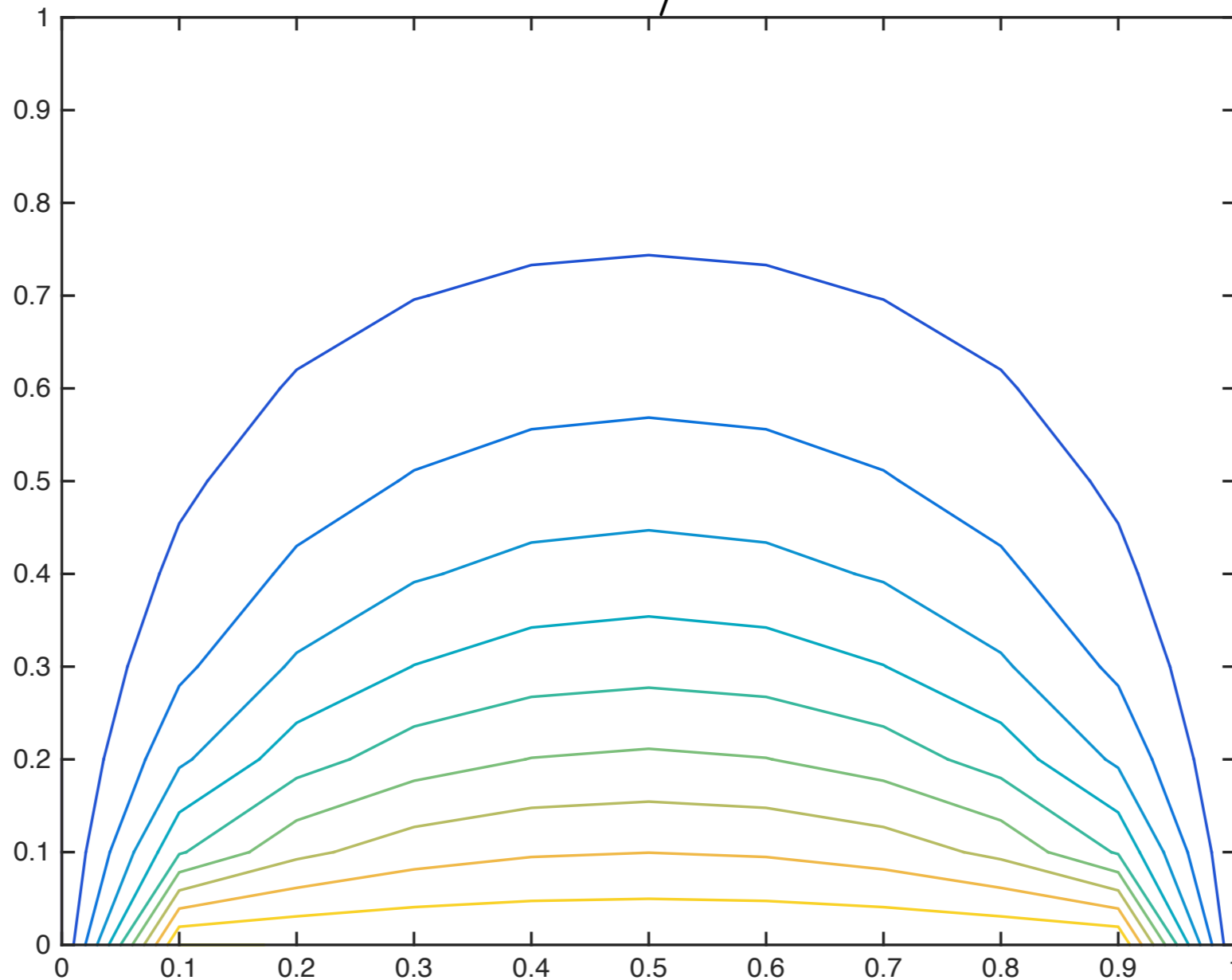
% Calculate RHS of Ac = b
[ Ac, b ] = my_func( zeros( Nx * Ny, 1 ), Nx, Ny );

% Find solution of linear FD equations using the an iterative method
% This is gmres (generalized minimum residual). Other choices include
% bicgstab (conjugate gradient), minres (minimum residual), etc.
% The requires a function that returns A*c given c.
c = gmres( @( c ) my_func( c, Nx, Ny ), b, 100, 1e-6, 100 );
```

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

$$h = 1/10$$

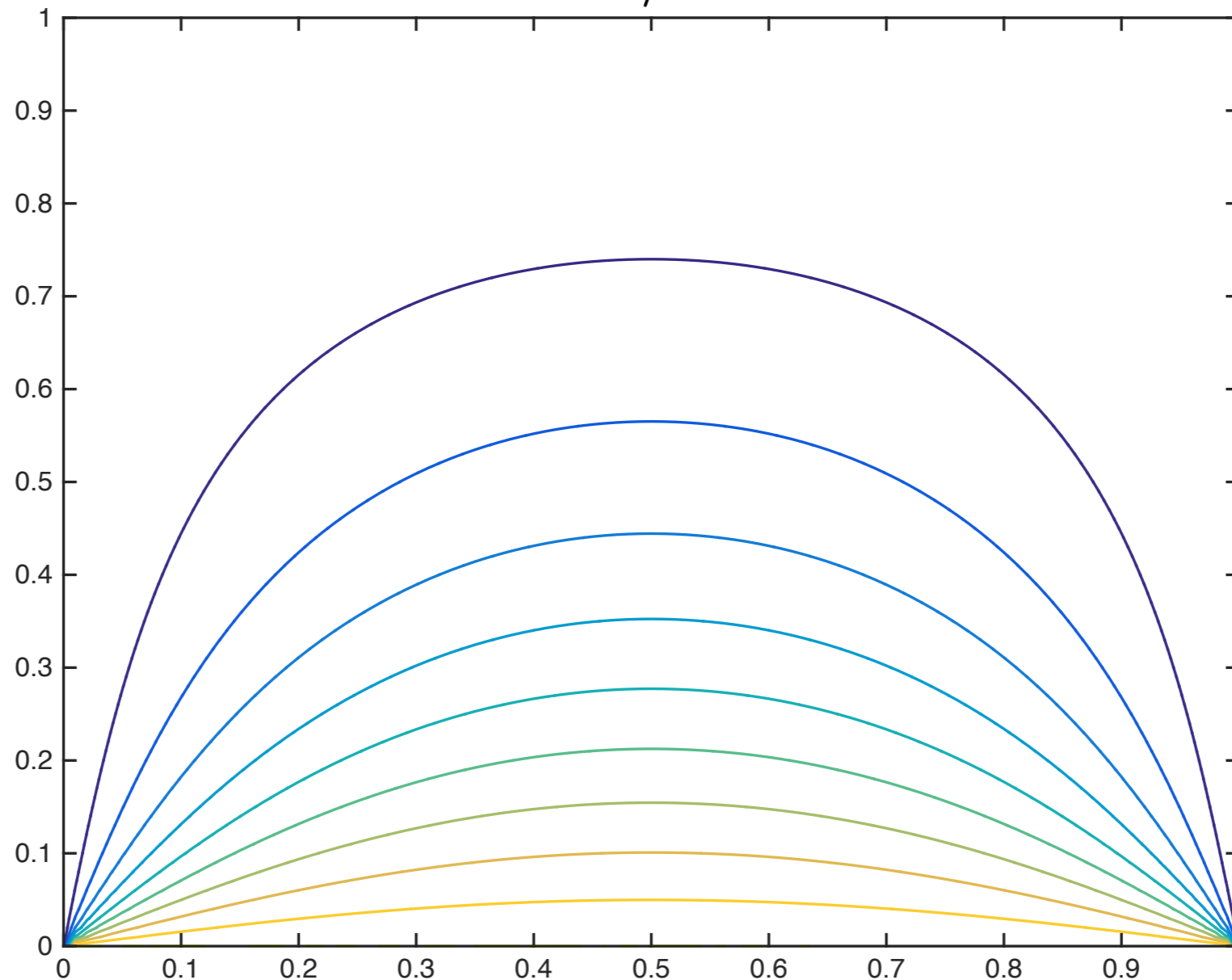


0.015 seconds to solve!

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

$$h = 1/100$$



5 seconds to solve!

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

$$\mathbf{f}(\mathbf{c}) = 0 = \mathbf{A}\mathbf{c} - \mathbf{b}$$

```
function [ Ac, b ] = my_func( c, Nx, Ny )
```

```
Ac = sparse( Nx * Ny, 1 );  
b = sparse( Nx * Ny, 1 );
```

$$k = i + (j - 1)N_x$$

```
% Define indices of boundary points and interior points
```

```
bottom = [ 1:Nx ];  
top = Nx*Ny - [ 1:Nx ];  
left = [ 1:Nx:Nx*Ny ];  
right = [ Nx:Nx:Nx*Ny ];  
interior = setdiff( [ 1:Nx*Ny ], [ left, right, bottom, top ] );
```

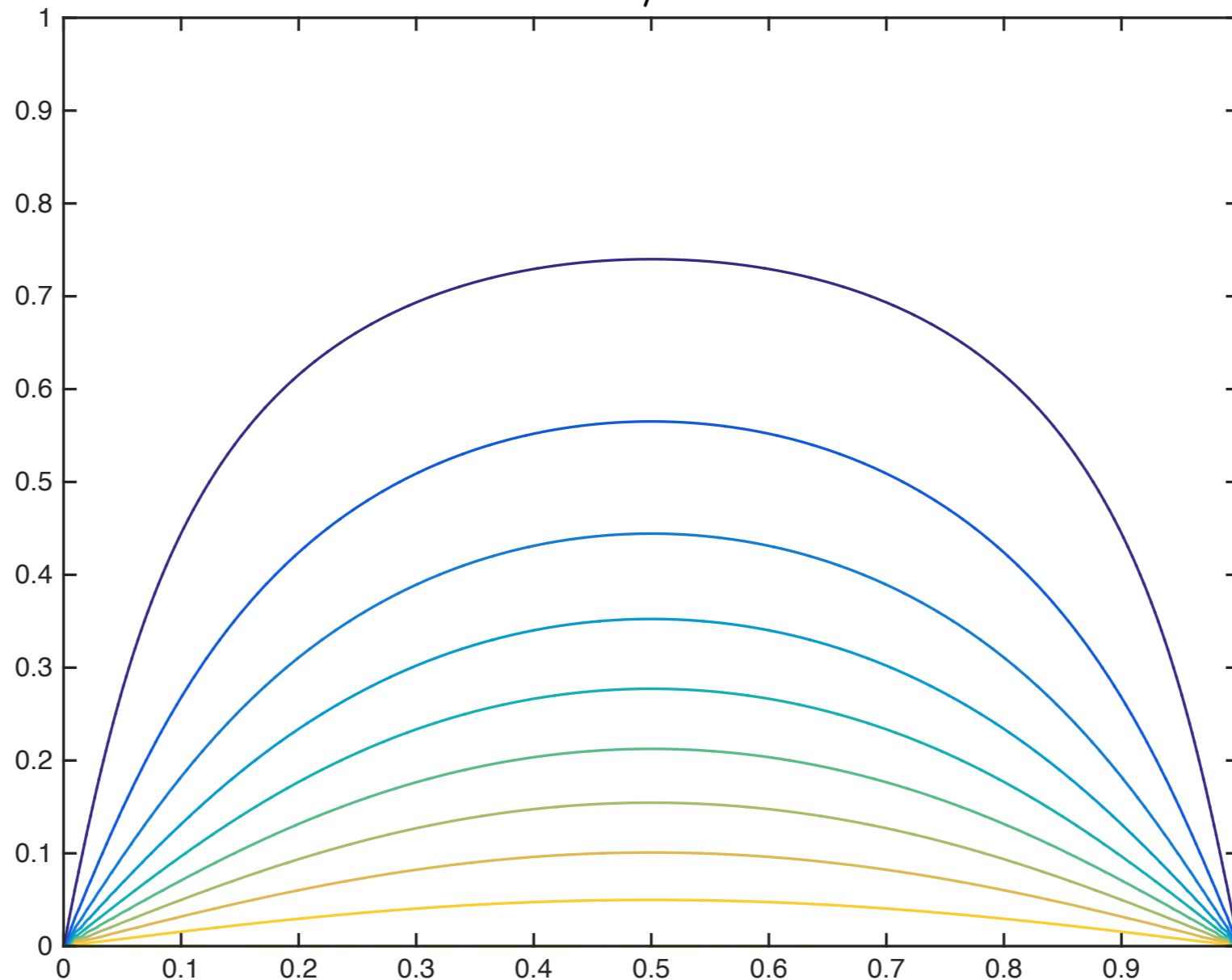
```
Ac( left ) = c( left );  
Ac( right ) = c( right );  
Ac( top ) = c( top );  
Ac( bottom ) = c( bottom );  
b( bottom ) = 1;
```

```
A( interior ) = c( interior - 1 ) + c( interior + 1 ) + c( interior - Nx ) + c( interior + Nx ) ...  
- 4 * c( interior );
```

Numerical Solution of PDEs

Example: solve the diffusion equation in 2-D on a square with side = 1.

$$h = 1/100$$



1.2 seconds to solve!

MIT OpenCourseWare

<https://ocw.mit.edu>

10.34 Numerical Methods Applied to Chemical Engineering

Fall 2015

For information about citing these materials or our Terms of Use, visit: <https://ocw.mit.edu/terms>.