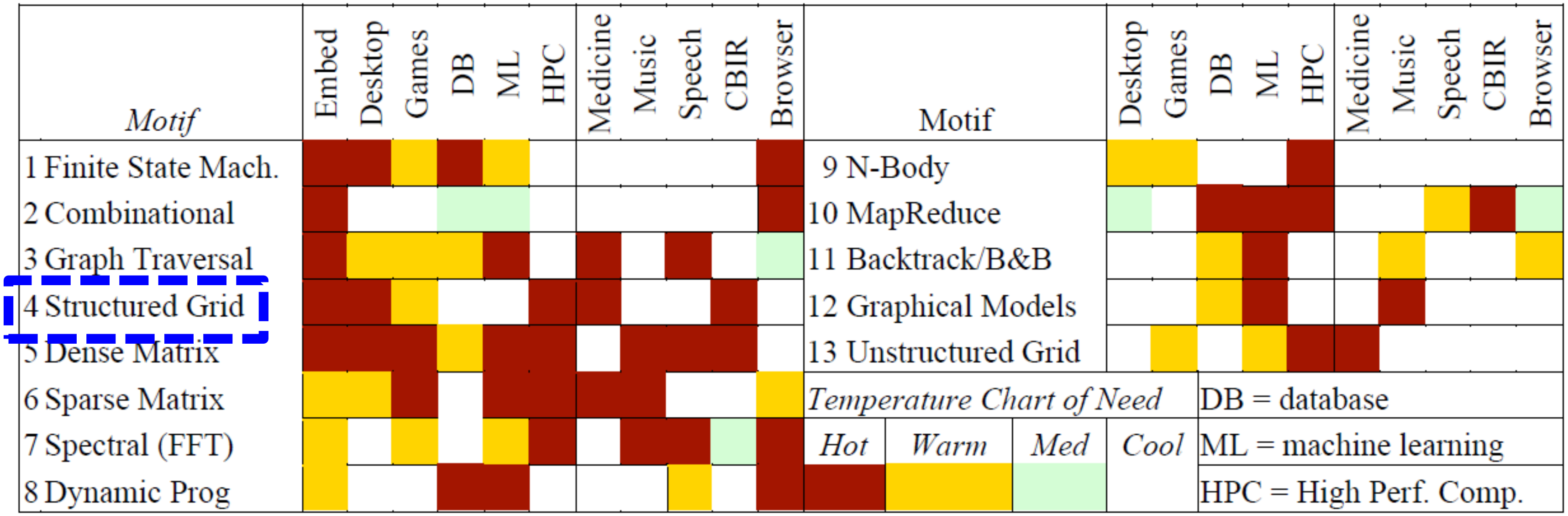
CS601: Software Development for Scientific Computing

Autumn 2023

Week12: Structured Grids

Matrix Algebra and Efficient Computation

- **Pic source: the Parallel Computing Laboratory at U.C. Berkeley: A Research Agenda Based on the Berkeley View (2008)**



Motif	Embed	Desktop	Games	DB	ML	HPC	Medicine	Music	Speech	CBIR	Browser
1 Finite State Mach.	🟥	🟥	🟨	🟥	🟨						🟥
2 Combinational	🟥			🟩	🟩						🟥
3 Graph Traversal	🟥	🟨	🟨	🟨	🟥		🟥		🟥		🟩
4 Structured Grid	🟥	🟥	🟨			🟥	🟥			🟥	
5 Dense Matrix	🟥	🟥	🟥	🟨	🟥	🟥		🟥	🟥	🟥	
6 Sparse Matrix	🟨	🟨	🟥		🟥	🟥	🟥	🟥			🟨
7 Spectral (FFT)	🟨		🟨		🟨	🟥		🟥	🟥	🟩	🟥
8 Dynamic Prog	🟨			🟥	🟥				🟨		🟥

Motif	Desktop	Games	DB	ML	HPC	Medicine	Music	Speech	CBIR	Browser
9 N-Body	🟨	🟨			🟥					
10 MapReduce	🟩		🟥	🟥	🟥			🟨	🟥	🟩
11 Backtrack/B&B			🟨	🟥			🟨			🟨
12 Graphical Models			🟨	🟥			🟥			
13 Unstructured Grid		🟨		🟨	🟥	🟥				

Temperature Chart of Need				DB = database
Hot	*Warm*	*Med*	*Cool*	ML = machine learning
🟥	🟨	🟩		HPC = High Perf. Comp.

Figure 4. Temperature Chart of the 13 Motifs. It shows their importance to each of the original six application areas and then how important each one is to the five compelling applications of Section 3.1. More details on the motifs can be found in (Asanovic, Bodik et al. 2006).

Next..

Discretization

- Cannot store/represent infinitely many continuous values
  - To model turbulent features of flow through a pipe, say, I am interested in `velocity` and `pressure` at all `points` in a region of interest
    1. Represent region of interest as a mesh of small discrete cells - ***discretization spacing***
    2. Solve equations for each cell

Example: `diameter of the pipe = 5cm`
`length=2.5cm`
`discretization spacing = 0.1mm`
`(volume of cylinder =` $\pi r^2 h$`)`

Exercise: *how many variables do you need to declare?*

Discretization

- All problems with ‘continuous’ quantities don’t require discretization
  - Most often they do.
- When discretization is done:
  - How refined is your discretization depends on certain parameters: step-size, cell shape and size. E.g.
    - Size of the largest cell (PDEs in FEM),
    - Step size in ODEs
  - Accuracy of the solution is of prime concern
    - Discretization always gives an approximate solution. Why?
    - Errors may creep in. Must provide an estimate of error.

Accuracy

- Discretization error
  - Is because of the way discretization is done
  - E.g. use more number of rays to minimize discretization error in ray tracing
- Solution error
  - The equation to be solved influences solution error
  - E.g. use more number of iterations in PDEs to minimize solution error
- Accuracy of the solution depends on both solution and discretization errors
- Accuracy also depends on cell shape

Error Estimate

- You will have to deal with errors in the presence of discretization
  - Providing error estimate is necessary
- *Apriori* error estimate
  - Gives insight on whether a discretization strategy is suitable or not
  - Depends on discretization parameter
  - Properties of the (unknown) exact solution
  - Error is bound by: $\mathbf{Ch^p}$ where, C depends on exact solution, h is discretization parameter, and p is a fixed exponent. *Assumption: exact solution is differentiable, typically, p+1 times.*

Error Estimate

- *Aposteriori* error estimate
  - Is estimation of the error in computed (Approximate) solution and does not depend on information about exact solution
  - E.g. Sleipner-A oil rig disaster

Exercise

- *does increasing mesh size always yield better accuracy?*

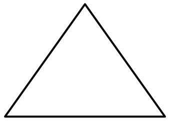
- *does decreasing cell size always yield better accuracy?*

- *How does changing mesh size affect computational cost?*

- *How does changing cell size affect computational cost?*

Cell Shape

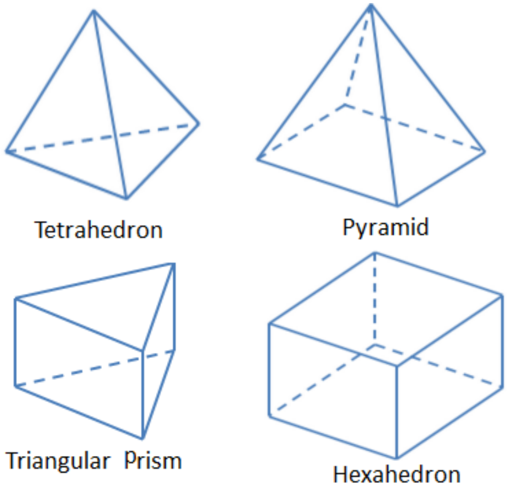
- 2D:



triangle

quadrilateral

- 3D: triangular or quadrilateral faced. E.g.

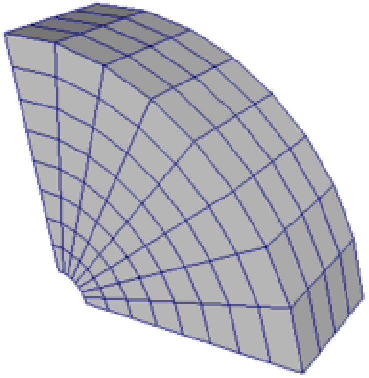


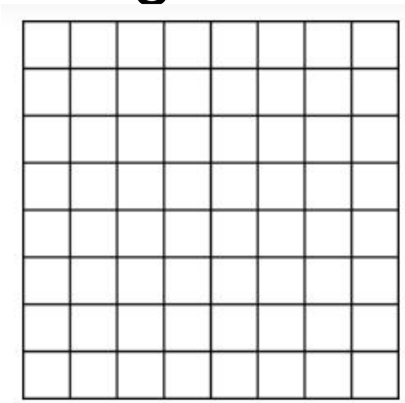
source: wikipedia

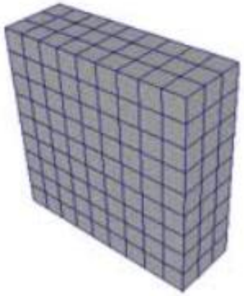
Tetrahedron: 4 vertices, 4 edges, 4△ faces
Pyramid: 5 vertices, 8 edges, 4 △ and 1 ▭ face
Triangular prism: 6 vertices, 9 edges, 2△ and 3▭ faces
Hexahedron: 8 vertices, 12 edges, 6 ▭ faces

Structured Grids

- Have regular connectivity between cells
  - i.e. every cell is connected to a predictable number of neighbor cells
- Quadrilateral (in 2D) and Hexahedra (in 3D) are most common type of cells
- Simplest grid is a rectangular region with uniformly divided rectangular cells (in 2D).







credits: nanohub.org

Structured Grids – Problem Statement

- Given:
  - A geometry
  - A mathematical model (partial differential equation)
  - Certain conditions / constraints / known values etc.
- Goal:
  - Discretize into a grid of cells
  - Approximate the PDE on the grid
  - Solve the PDE on the grid

Structured Grids - Representation

- Because of regular connectivity between cells
  - – Cells can be identified with indices (x,y) or (x,y,z) and neighboring cell info can be obtained.
  - – How about identifying a cell here?

    Given:

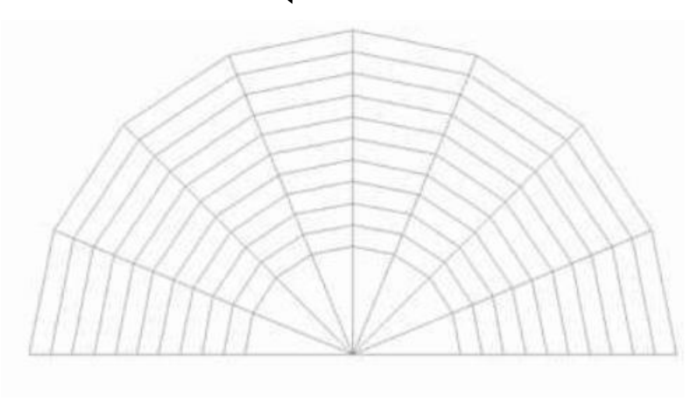
    $\xi =$ ("Xi") radius

    η = ("Eta") angle

    Compute:

$$x = \left(\frac{1}{2} + \xi\right)\cos(\pi\eta)$$

$$y = \left(\frac{1}{2} + \xi\right)\sin(\pi\eta)$$



Structured Grids - Representation

- Assume that we have a grid.
- Task:
  - Approximate Partial Differential Equations (PDEs)
  - Solve/Implement PDEs (turning PDEs into large set of algebraic equations)

Mathematical Model of the Grid

- Partial Differential Equations (PDEs):
  - Navier-Stokes equations to model water, blood flow, weather forecast, aerodynamics etc.
  - Elasticity (Lame-Navier equations)
  - Nutrient transport in blood flow
  - Heat conduction: *how heat conducts/diffuses through a material given the temperature at boundaries?*
  - Mechanics: *how does a mass reach from point p1 to point p2 in shortest time under gravitational forces?*

Notation and Terminology

- $\frac{\partial u}{\partial x} = \ \partial_x u$
- $\frac{\partial^2 u}{\partial x \partial y} = \ \partial_{xy} u$
- $\frac{\partial u}{\partial t} = \ \partial_t u$, t usually denotes time.
- Laplace operator $(\mathbf{L})$: of a two-times continuously differentiable scalar-valued function $u: \mathbb{R}^n \to \mathbb{R}$

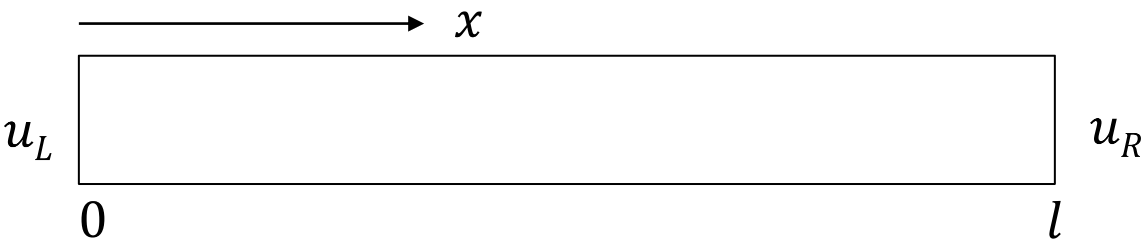
$$\Delta u \quad = \sum_{k=1}^{n} \partial_{kk} u$$

Important PDEs

- Three important types (*not a complete categorization by any means*):
  - Poisson problem: $-\Delta u = f$ (elliptic)
  - Heat equation: $\partial_t u \ - \Delta u = f$ (parabolic. Here, $\partial_t u = \frac{\partial u}{\partial t}$ = partial derivative w.r.t. time)
  - Wave equation: $\partial_t{}^2 u \ - \Delta u = f$ (Hyperbolic. Here, $\partial_t{}^2 u = \frac{\partial^2 u}{\partial t \partial t}$ = second-order partial derivative w.r.t. time)

Application: Heat Equation

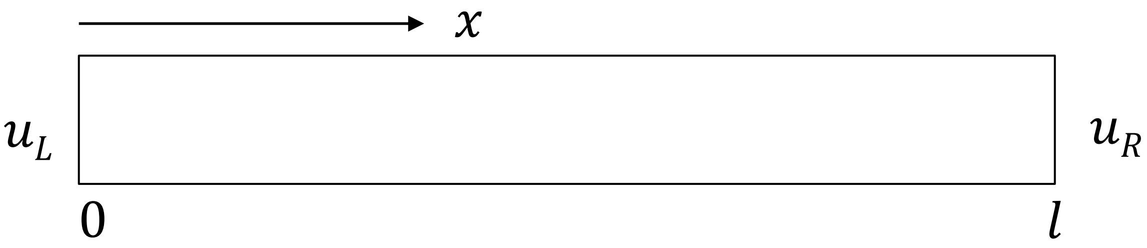
- Example: heat conduction through a rod



- $u = u(x, t)$ is the temperature of the metal bar at distance x from one end and at time t
- Goal: find u

Initial and Boundary Conditions

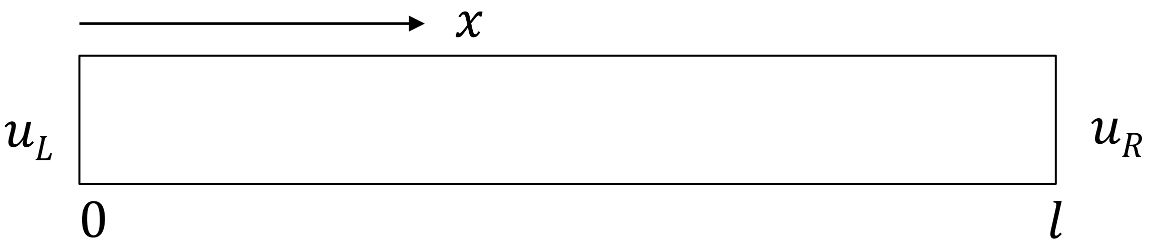
- Example: heat conduction through a rod



- Metal bar has length l and the ends are held at constant temperatures u_L at the left and u_R at the right
- Temperature distribution at the initial time is known $f(x)$, with $f(0) = u_L$ and $f(l) = u_R$

Equations

- Example: heat conduction through a rod



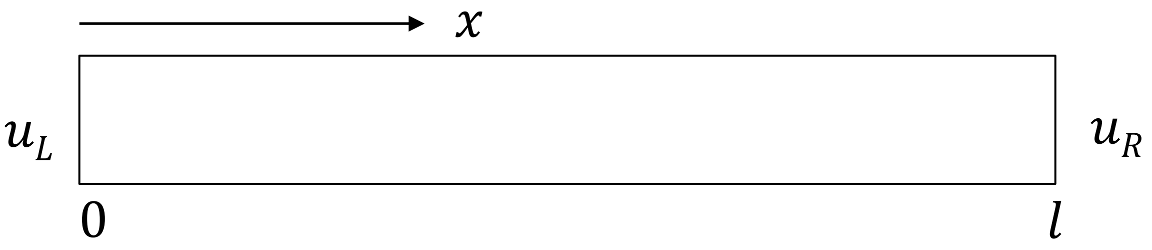
$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \qquad (0 < x < l, t > 0)$$

α is thermal diffusivity

(a constant if the material is homogeneous and isotropic.
copper = 1.14 cm^2 s^{-1}, aluminium = 0.86 cm^2 s^{-1})

Equations

- Example: heat conduction through a rod



$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \qquad (0 < x < l, t > 0)$$

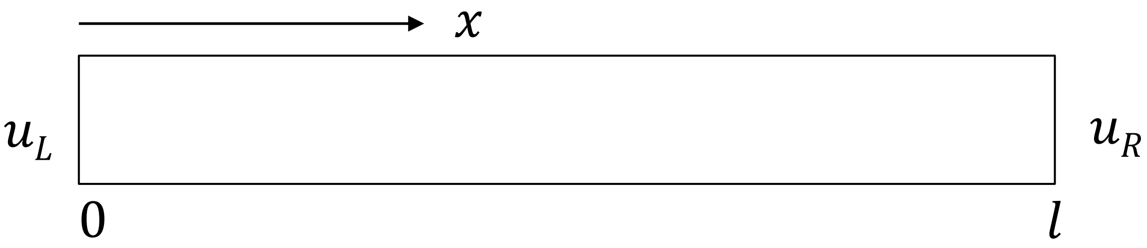
α is thermal diffusivity

(a constant if the material is homogeneous and isotropic. copper = 1.14 cm^2 s^{-1}, aluminium = 0.86 cm^2 s^{-1})

- *Exercise: what kind of a PDE is this? (Poisson/Heat/Wave?)*

Equations

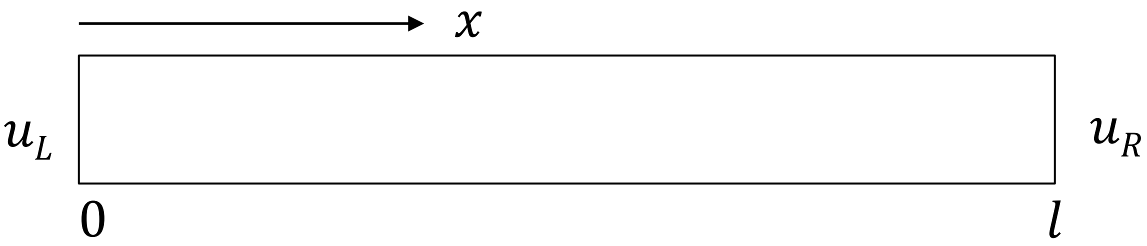
- Example: heat conduction through a rod



$\partial_t u = \alpha \Delta u$ as per the notation mentioned earlier

Equations

- Example: heat conduction through a rod



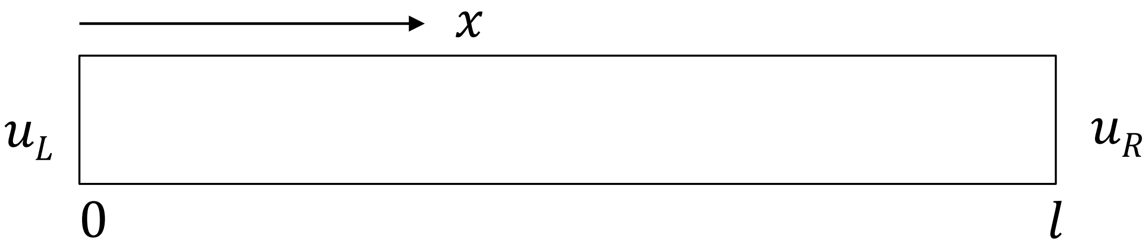
$$\partial_t u = \alpha \Delta u$$

Can also be written as:

$$\partial_t u - \alpha \Delta u = 0$$

Equations

- Example: heat conduction through a rod



$\partial_t u - \alpha \Delta u = 0$,

Based on initial and boundary conditions:

$u(0,t) = u_L$,
$u(l,t) = u_R$,
$u(x,0) \; = \; f(x)$

Equations

- Summarizing:
  1. $\partial_t u - \alpha \Delta u = 0, 0\text{<x<l, t>0}$
  2. $u(0,t) = u_L, t > 0$
  3. $u(l,t) = u_R\ , t > 0$
  4. $u(x,0) =\ f(x), 0 < x < l$

- Solution:

$$u(x,t) \quad = \sum_{m=1}^{\infty} B_m e^{-m^2 \alpha \pi^2 t/l^2} \sin(\frac{m\pi x}{l}) \quad ,$$

where, $B_m = 2/l \int_0^l f(s) \sin\left(\frac{m\pi s}{l}\right) ds$

Equations

- Summarizing:
  1. $\partial_t u - \alpha \Delta u = 0, 0<x<l, t>0$
  2. $u(0,t) = u_L, t > 0$
  3. $u(l,t) = u_R\ , t > 0$
  4. u

But we are interested in a numerical solution

- Solution:

$$u(x,t) \quad = \sum_{m=1}^{\infty} B_m e^{-m^2 \alpha \pi^2 t/l^2} \sin(\frac{m\pi x}{l}) \quad ,$$

where, $B_m = 2/l \int_0^l f(s) \sin\left(\frac{m\pi s}{l}\right) ds$

Approximating Partial Derivatives

- Suppose $y = f(x)$
  - Forward difference approximation to the first-order derivative of f w.r.t. x is:

    $$\frac{df}{dx} \approx \frac{\left(f(x+\delta x)-f(x)\right)}{\delta x}$$

  - Central difference approximation to the first-order derivative of f w.r.t. x is:

    $$\frac{df}{dx} \approx \frac{\left(f(x+\delta x)-f(x-\delta x)\right)}{2\delta x}$$

  - Central difference approximation to the second-order derivative of f w.r.t. x is:

    $$\frac{d^2f}{dx^2} \approx \frac{\left(f(x+\delta x)-2f(x)+f(x-\delta x)\right)}{(\delta x)^2}$$

Approximating Partial Derivatives

- In example heat application $f = u = u(x,t)$ and $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$
  - First, approximating $\frac{\partial u}{\partial t}$:

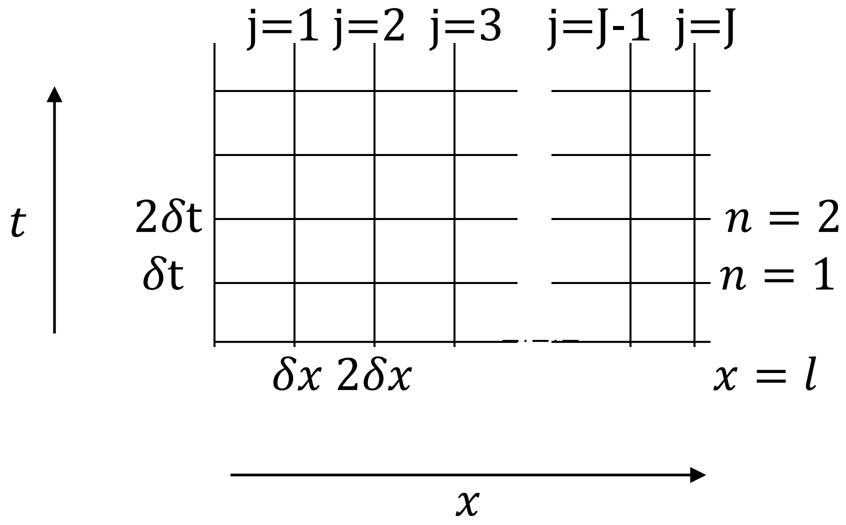
    $\frac{\partial u}{\partial t} \approx \frac{\left(u(x,t+\delta t)-u(x,t)\right)}{\delta t}$, where δt is a small increment in time
  - Next, approximating $\frac{\partial^2 u}{\partial x^2}$:

    $\frac{\partial^2 u}{\partial x^2} \approx \frac{\left(u(x+\delta x,t)-2u(x,t)+u(x-\delta x,t)\right)}{(\delta x)^2}$, where δx is a small

  increment in space (along the length of the rod)

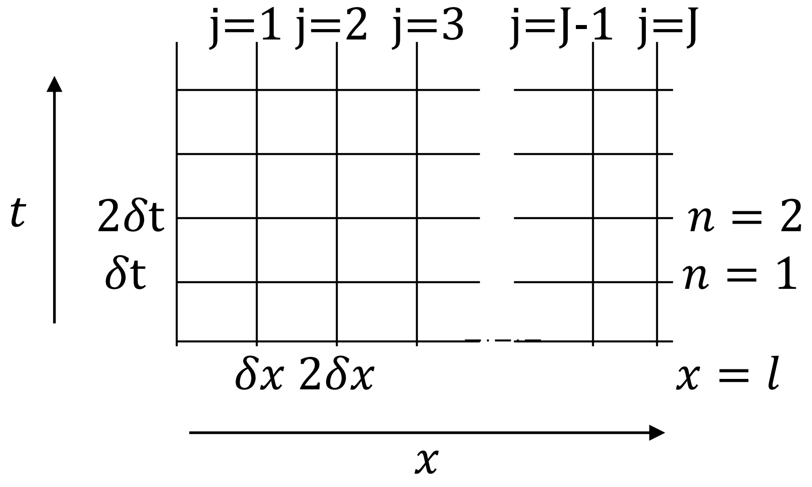
Approximating Partial Derivatives

- Divide length l into J equal divisions: $\delta x = l/J$ (space step)
- Choose an appropriate δt (time step)



Approximating Partial Derivatives

- Find sequence of numbers which approximate u at a sequence of (x, t) points (i.e. at the intersection of horizontal and vertical lines below)



- Approximate the exact solution $u(j \times \delta x, n \times \delta t)$ using the approximation for partial derivatives mentioned earlier

Approximating Partial Derivatives

$$\frac{\partial u}{\partial t} \approx \frac{\big(u(x, t + \delta t) - u(x, t)\big)}{\delta t}$$

$$= \frac{(u_j^{n+1} - u_j^n)}{\delta t}$$

where $\mathrm{u_j}^{\mathrm{n+1}}$ denotes taking j steps along x direction and $n + 1$ steps along t direction

Similarly, $\frac{\partial^2 u}{\partial x^2} \approx \frac{\big(u(x+\delta x,t)-2u(x,t)+u(x-\delta x,t)\big)}{(\delta x)^2}$

$$= \frac{(u_{j+1}^n - 2\, u_j^n + u_{j-1}^n)}{(\delta x)^2}$$

Approximating Partial Derivatives

Plugging into $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$:

$$\frac{(u_j^{n+1} - u_j^n)}{\delta t} = \alpha \frac{(u_{j+1}^n - 2\, u_j^n + u_{j-1}^n)}{(\delta x)^2}$$

This is also called as difference equation because you are computing difference between successive values of a function involving discrete variables.

Approximating Partial Derivatives

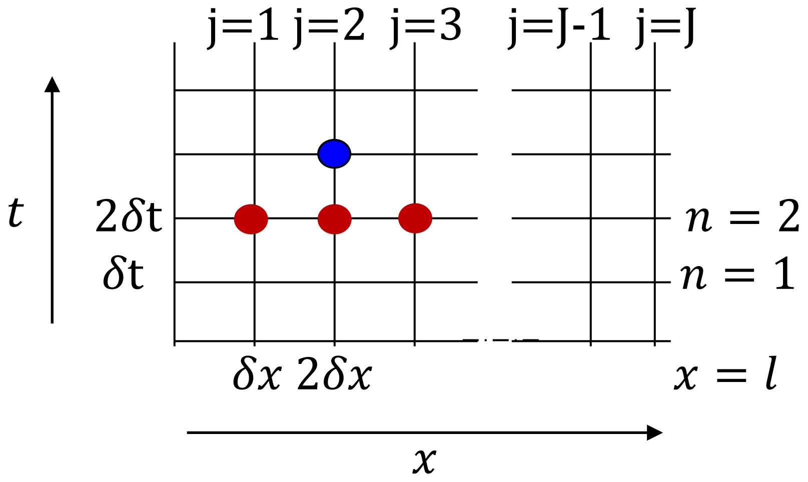
Simplifying:

$$u_j^{n+1} = u_j^n + r(u_{j+1}^n - 2\,u_j^n + u_{j-1}^n)$$
$$= ru_{j-1}^n + (1\ - 2r)u_j^n + ru_{j+1}^n,$$
$$where\ r = \alpha\,\frac{\delta t}{(\delta x)^2}$$

Approximating Partial Derivatives

visualizing,

$$u_j^{n+1} = ru_{j-1}^n + (1 \; - 2r)u_j^n + ru_{j+1}^n$$



To compute the value of function at blue dot, you need 3 values indicated by the red dots – 3-point stencil

Approximating Partial Derivatives

- Initial and boundary conditions tell us that:

  $u(0,t) = u_L$,
  $u(l,t) = u_R$,
  $u(x,0) \; = \; f(x)$

- $u_0^0, u_1^0\, u_2^0, \ldots .\, u_J^0$ are known (at time t=0, the temperature at all points along the distance is known as indicated by $f(x)$ = f_j).
- u_0^1 is $u_{L,}\, u_J^1$ is u_R
- Now compute points on the grid from left-to-right:

Approximating Partial Derivatives

- Now compute points on the grid from left-to-right:

$$u_1^1 = u_1^0 + r(u_0^0 - 2u_1^0 + u_2^0)$$
$$u_2^1 = u_2^0 + r(u_1^0 - 2u_2^0 + u_3^0)$$

.

.

$$u_{J-1}^1 = u_{J-1}^0 + r\left(u_{J-2}^0 - 2u_{J-1}^0 + u_J^0\right)$$

- This constitutes the computation done in the first time step.
- Now do the second time step computation…and so on..

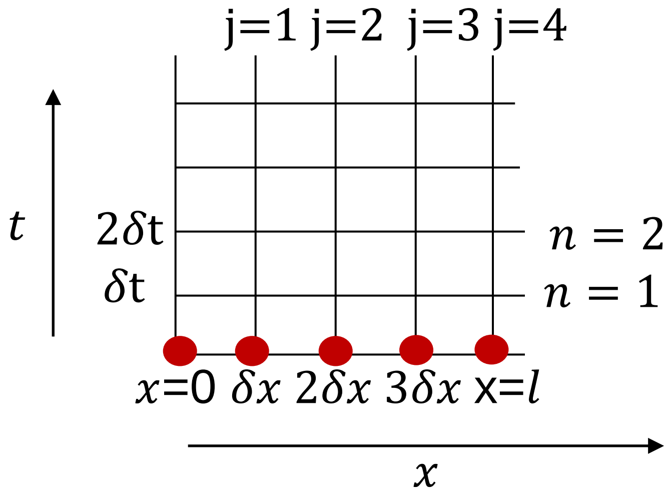
Explicit Difference Method: Stability

- Given: $l = 1,$
  $u(0, t) = u_L = 0,$
  $u(l, t) = u_R = 0,$
  $u(x, 0) = f(x) = x(l - x)$
  $\alpha = 1,$
- Choose: $\delta x = 0.25, \delta t = 0.075$
- Solve.

Explicit Difference Method: Stability

- Initialize u_j^0 values from initial and boundary conditions i.e. *get time-step 0 values*

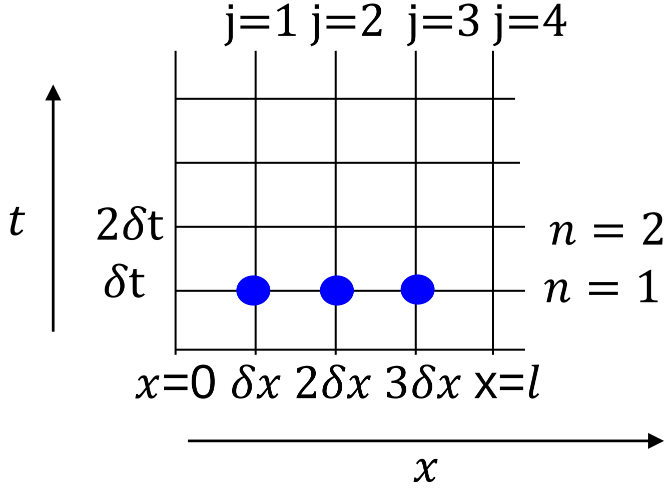
$u_0^0 = 0$
$u_1^0 = f(\delta x) = \delta x(l - \delta x) = .1875$
$u_2^0 = f(2\delta x) = 2\delta x(l - 2\delta x) = .25$
$u_3^0 = f(3\delta x) = 3\delta x(l - 3\delta x) = .1875$
$u_4^0 = 0$



Explicit Difference Method: Stability

- Compute time-step 1 values

$$u_j^{n+1} = ru_{j-1}^n + (1 - 2r)u_j^n + ru_{j+1}^n$$

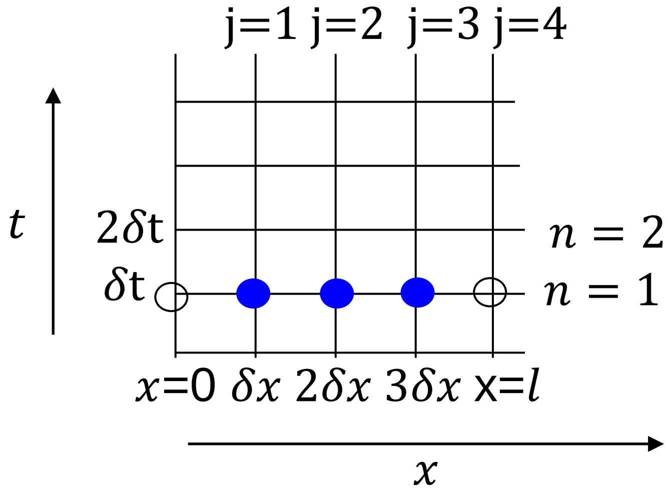


Explicit Difference Method: Stability

- Compute time-step 1 values

$$u_j^{n+1} = ru_{j-1}^n + (1 - 2r)u_j^n + ru_{j+1}^n$$

What about values of $u(x, t)$ at ○ ?



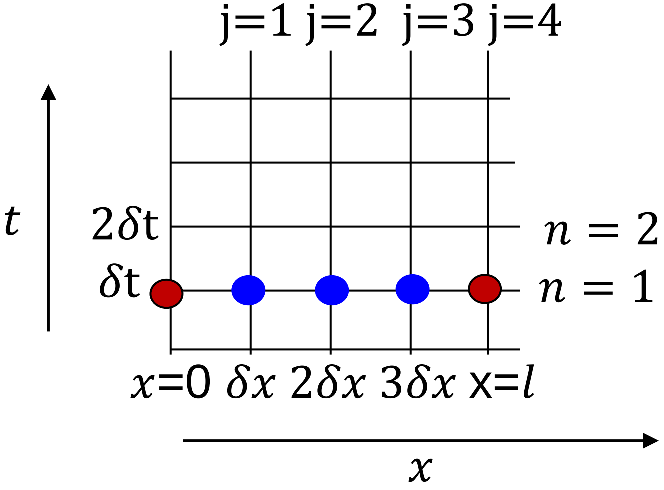
Explicit Difference Method: Stability

- Compute time-step 1 values

$$u_j^{n+1} = ru_{j-1}^n + (1 - 2r)u_j^n + ru_{j+1}^n$$

What about values of $u(x, t)$ at ○ ?

Get it from boundary conditions



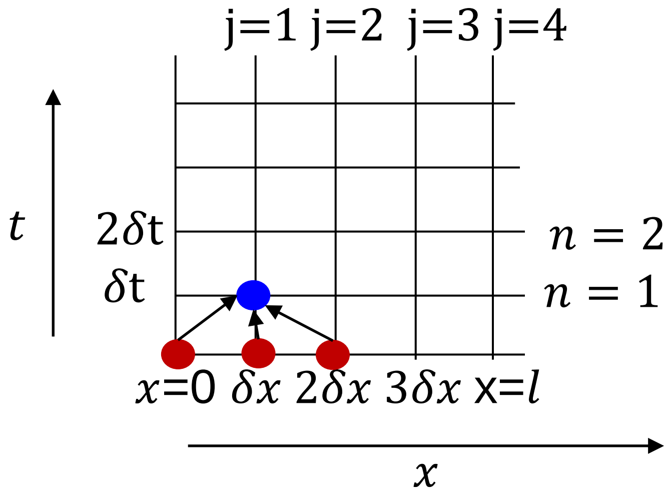
Explicit Difference Method: Stability

- Compute time-step 1 values

$$u_j^{n+1} = ru_{j-1}^n + (1 - 2r)u_j^n + ru_{j+1}^n$$

$r = \alpha\delta t/(\delta x)^2$ = 1.2

$u_1^1 = u_1^0 + r(u_0^0 - 2u_1^0 + u_2^0)$ = 0.03678



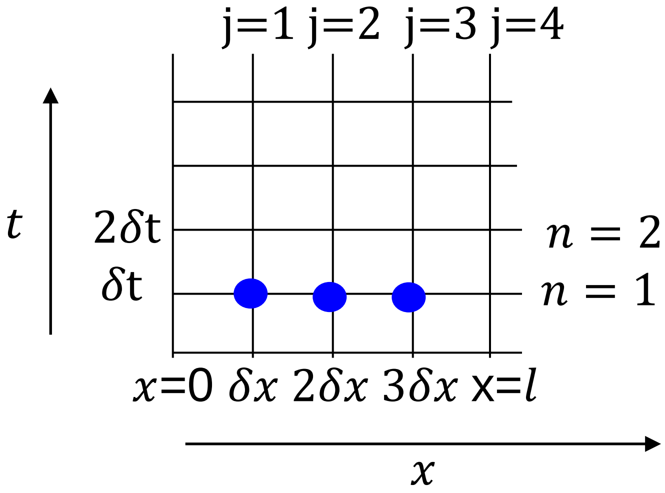
Explicit Difference Method: Stability

- Compute time-step 1 values

$$u_j^{n+1} = ru_{j-1}^n + (1 - 2r)u_j^n + ru_{j+1}^n$$

$r = \alpha\delta t/(\delta x)^2$ = 1.2

$u_1^1 = u_1^0 + r(u_0^0 - 2u_1^0 + u_2^0)$=0.03678
$u_2^1 = u_2^0 + r(u_1^0 - 2u_2^0 + u_3^0)$=0.1
$u_3^1 = u_3^0 + r(u_2^0 - 2u_3^0 + u_4^0)$=0.03678



Explicit Difference Method: Stability

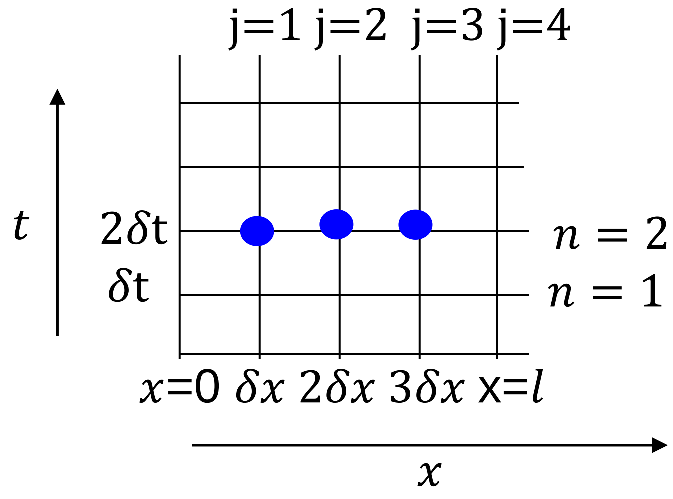
- Compute time-step 2 values

$$u_j^{n+1} = ru_{j-1}^n + (1 - 2r)u_j^n + ru_{j+1}^n$$

$u_1^2 = u_1^1 + r(u_0^1 - 2u_1^1 + u_2^1)$=0.06851

$u_2^2 = u_2^1 + r(u_1^1 - 2u_2^1 + u_3^1)$=**-0.05173**

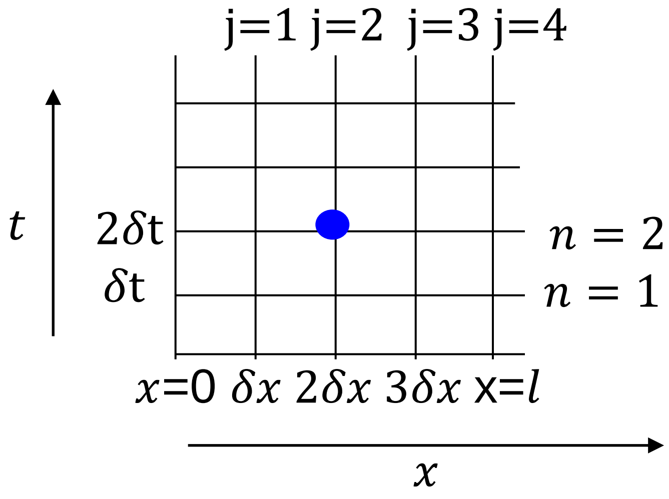
$u_3^2 = u_3^1 + r(u_2^1 - 2u_3^1 + u_4^1)$=0.06851



Explicit Difference Method: Stability

- Temperature at $2\delta x$ after $2\delta t$ time units went into negative! (when the boundaries were held constant at 0)
  - Example of *instability*

$$u_2^2 = u_2^1 + r(u_1^1 - 2u_2^1 + u_3^1) = \mathbf{-0.05173}$$



The solution is stable (for heat diffusion problem) only if the approximations for $u(x,t)$ do not get bigger in magnitude with time

Explicit Difference Method: Stability

- The solution for heat diffusion problem is stable only if:

$$r \leq \frac{1}{2}$$

Therefore, choose your time step in such a way that:

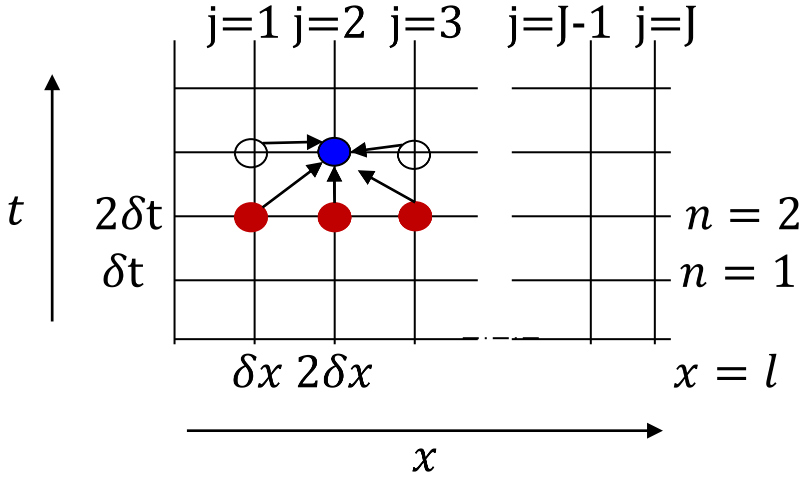
$$\delta t \leq \frac{\delta x^2}{2\alpha}$$

But this is a severe limitation!

Implicit Method: Stability

- Overcoming instability:

$$u_j^{n+1} = u_j^n + 1/2\, \mathrm{r}(\, u_{j-1}^n - 2u_j^n + u_{j+1}^n + u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}\ )$$



To compute the value of function at blue dot, you need 6 values indicated by the red dots (known) and 3 additional ones (unknown) above

Suggested Reading

- *J.W. Thomas. Numerical Partial Differential Equations: Finite Difference Methods*

- *Parabolic PDEs: https://learn.lboro.ac.uk/archive/olmp/olmp_resources/pages/workbooks_1_50_jan2008/Workbook32/32_4_prblc_pde.pdf*