CS601: Software Development for Scientific Computing

Autumn 2024

Week11: Intermediate C++, Structured Grids

this

- Implicit variable defined by the compiler for every class
  - E.g. `MyVec *this;`
- All member functions have `this` as an implicit first argument
  - E.g.

```
int MyVec::GetVecLen() const;
```

would actually be:

```
int MyVec::GetVecLen(MyVec* this) const;
```

Overloading +=

- `MyVec v1;`
  `v1+=3;`
- `MyVec& MyVec::operator+=(double)`

Overloading +=

- `MyVec v1;`

  `v1+=3;`
  - `MyVec& MyVec::operator+=(double)`
- `MyVec v2;`

  `v2+=v1;`
  - `MyVec& MyVec::operator+=(const MyVec& rhs)`
  - What if you make the return value above `const`?

    Disallow: `(v2+=v1)+=3;`

Overloading +

- `v1=v1+3;`

  Single-argument constructors: *allow implicit conversion from a particular type to initialize an object.*

  – `const MyVec MyVec::operator+(double val)`
- `v3=v1+v2;`

  1. `const MyVec MyVec::operator+(const MyVec& vec2) const;`

  OR

  2. `friend const MyVec operator+(const MyVec& lhs, const MyVec& rhs);`

v1=3+v1 is compiler error! Why?

Operator Overloading - Guidelines

- If a binary operator accepts operands of different types and is commutative, both orders should be overloaded
- Consistency:
  - If a class has ==, it should also have !=
  - += and + should result in identical values
  - define your copy assignment operator if you have defined a copy constructor

Refer to demo example

Class Templates

```
//CS601: example code used to show class declaration.
#ifndef MYVEC_H
#define MYVEC_H
class MyVec{
        //private attributes
        double* data;
        int vecLen;
public:
        MyVec(){data=0;vecLen=0;}
```

What if user wants to have a MyVec class with integer `data`?

Class Templates

- Like function templates but for templating classes

Refer to demo example

Standard Template Library (STL)

- Large set of frequently used data structures and algorithms
  - Defined as *parametrized* data types and functions
  - Types to represent complex numbers and strings, algorithms to sort, get random numbers etc.
- Convenient and bug free to use these libraries
- E.g. `vector, map, queue, pair, sort` etc.
- Use your own type only for efficiency considerations - *only if you are sure!*

STL - Motivation

Real-world view
source:wikipedia

Coconut meat, raw

Nutritional value per 100 g (3.5 oz)		
Energy	354 kcal (1,480 kJ)	
Carbohydrates	15.23 g	
Sugars	6.23 g	
Dietary fiber	9.0 g	
Fat	33.49 g	
Saturated	29.698 g	
Monounsaturated	1.425 g	
Polyunsaturated	0.366 g	
Protein	3.33 g	
Tryptophan	0.039 g	
Threonine	0.121 g	
Isoleucine	0.131 g	
Leucine	0.247 g	
Lysine	0.147 g	
Methionine	0.062 g	
Cystine	0.066 g	
Phenylalanine	0.169 g	
Tyrosine	0.103 g	
Valine	0.202 g	
Arginine	0.546 g	
Histidine	0.077 g	
Alanine	0.170 g	
Aspartic acid	0.325 g	
Glutamic acid	0.761 g	
Glycine	0.158 g	
Proline	0.138 g	
Serine	0.172 g	
Vitamins	**Quantity**	**%DV†**

Consider the nutrients (constituents) present in edible part of coconut.
How would you capture the Real-world view in a Program?

```
vector<pair<string, float> > constituents;
```

Container

- Holder of a collection of objects
- Is an object itself
- Different types:
  - sequence container
  - associative container (ordered/unordered)
  - container adapter

Sequence Container

- Provide fast sequential access to elements
- Factors to consider:
  - Cost to add/delete an element
  - Cost to perform non-sequential access to elements

container name	**comments**
`vector`	Flexible array, fast random access
`string`	Like `vector`. Meant for sequence of characters
`list/slist`	doubly/singly linked list. Sequential access to elements (bidirectional/unidirectional).
`deque`	Double-ended queue. Fast random access, Fast append
`array`	Intended as replacement for ‘C’-style arrays. Fixed-sized.

Container Adapter

- Provide an interface to sequence containers
  - `stack, queue, priority_queue`

Associative Container

- Implement sorted data structures for efficient searching (O(log n)) complexity.
  - Set, map, multiset, multimap

container name	comments
`set`	Collection of unique sorted keys. Implemented as class template
`map`	Collection of key-value pairs sorted by unique keys. Implemented as class template

Unordered Associative Container

- Implement hashed data structures for efficient searching (O(1) best-case, O(n) worst-case complexity).
  - `unordered_set, unordered_map, unordered_multiset, unordered_multimap`

Matrix Algebra and Efficient Computation

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

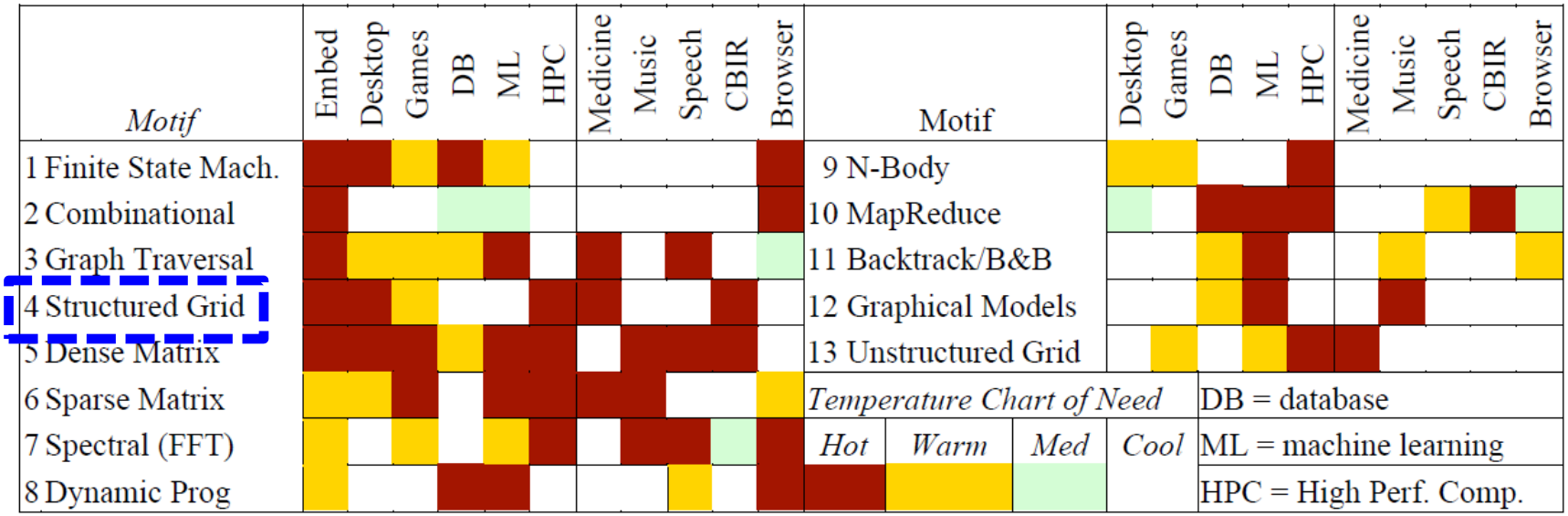


Figure 4. Temperature Chart of the 13 Motifs. It shows their importance to each of the original six application areas and then how important each one is to the five compelling applications of Section 3.1. More details on the motifs can be found in (Asanovic, Bodik et al. 2006).

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 = πr²h)
```

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

Discretization

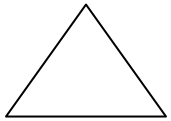
- All problems with ‘continuous’ quantities don’t require discretization
  - Most often they do.
- When discretization is done:
  - How refined is your discretization depends on certain parameters: step-size, cell shape and size. E.g.
    - Size of the largest cell (PDEs in FEM),
    - Step size in ODEs
  - 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

Cell Shape

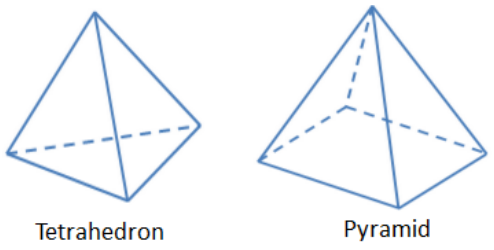
- 2D:



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

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: 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

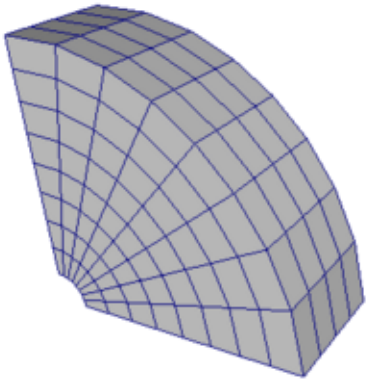
- *Aposteriori* error estimate
  - Is estimation of the error in computed (Approximate) solution and does not depend on information about exact solution
  - E.g. Sleipner-A oil rig disaster

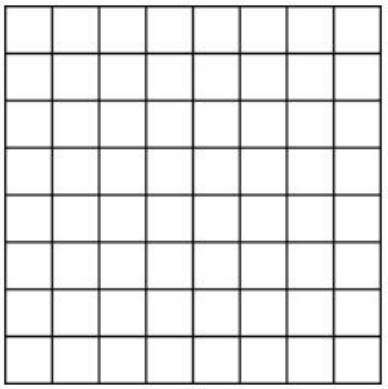
Exercise

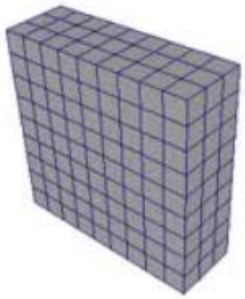
- *does increasing mesh size always yield same or better accuracy?*
- *does decreasing cell size always yield same or better accuracy?*
- *How does changing mesh size affect computational cost?*
- *How does changing cell size affect computational cost?*

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 partial differential equation
  - Initial and boundary conditions
- Goal:
  - Discretize into a grid of cells
  - Approximate the PDE on the grid
  - Solve the PDE on the grid

PDEs

- consider a function $u = u(x,t)$ satisfying the second-order PDE:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial t} + C\frac{\partial^2 u}{\partial t^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial t} + Fu = G\ ,$$

Where A-G are given functions. This is a PDE of type:

- Parabolic: if $B^2 - 4AC = 0$
- Elliptic: if $B^2 - 4AC < 0$
- Hyperbolic: if $B^2 - 4AC > 0$

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 $u: \mathbb{R}^n \to \mathbb{R}$

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

PDEs

- consider a function $u = u(x,t)$ satisfying the second-order PDE:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial t} + C\frac{\partial^2 u}{\partial t^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial t} + Fu = G\ ,$$

Where A-G are given functions. This is a PDE of type:

- Parabolic: if $B^2 - 4AC = 0$ Heat equation: $\partial_t u - \Delta u = f$
- Elliptic: if $B^2 - 4AC < 0$ Poisson problem: $-\Delta u = f$
- Hyperbolic: if $B^2 - 4AC > 0$

Wave equation: $\partial_t{}^2 u - \Delta u = f$

Boundary and Initial Value Problems

- Boundary Value Problems
  - PDE contains independent variables that are only spatial in nature (do not contain time).
  - E.g. $\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$

- Initial Value Problems
  - PDE contains independent variables that are spatial and temporal in nature.
  - E.g. $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$

Definitions (Laplace Equation and Poisson Equation)

- Consider a region of interest R in, say, xy plane. The following is a *boundary-value problem*:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \qquad \text{,where}$$

f is a given function in R and

$u = g$,where

the function g tells the value of function u at boundary of R

- if $f = 0$ everywhere, then Eqn. (1) is Laplace's Equation
- if $f \neq 0$ somewhere in R, then Eqn. (1) is Poisson's Equation

Boundary Conditions and Classification

- Essential / Dirichlet
  - Value of the dependent variable is specified
  - E.g. temperature at the edges of the rod are constant 0°
- Neumann / Natural
  - Value of the dependent variable is specified as gradient of the dependent variable T e.g. dT/dx.
- Mixed / Robin
  - value of the dependent variable is specified as a function of the gradient. E.g. $-K(dTdx)x=L=hA(T-T\infty)$

Approximating PDEs

Finite Difference Method

- 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:

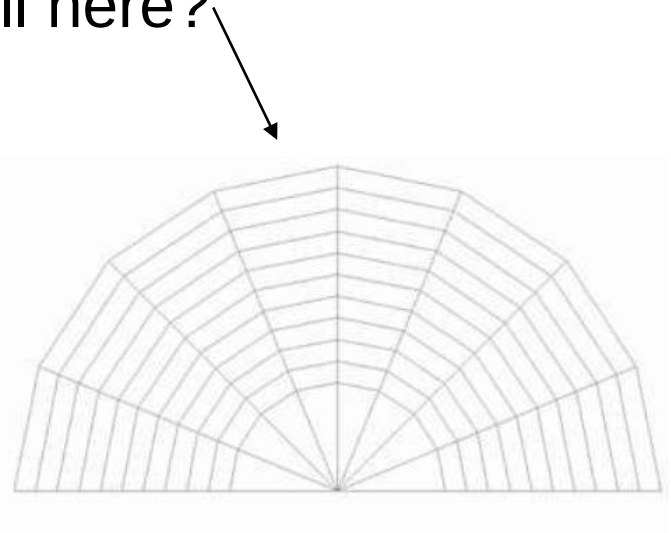
$$\frac{df}{dx} \approx \frac{\left(f(x+\delta x) - f(x-\delta x)\right)}{2\delta x}$$

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

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

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:

ξ = ("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

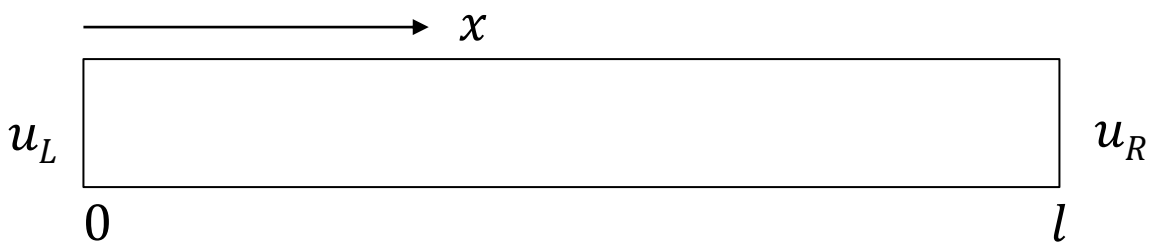
- Partial Differential Equations (PDEs):
  - Navier-Stokes equations to model water, blood flow, weather forecast, aerodynamics etc.
  - Elasticity (Lame-Navier equations)
  - Nutrient transport in blood flow
  - Heat conduction: *how heat conducts/diffuses through a material given the temperature at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

Recall: 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

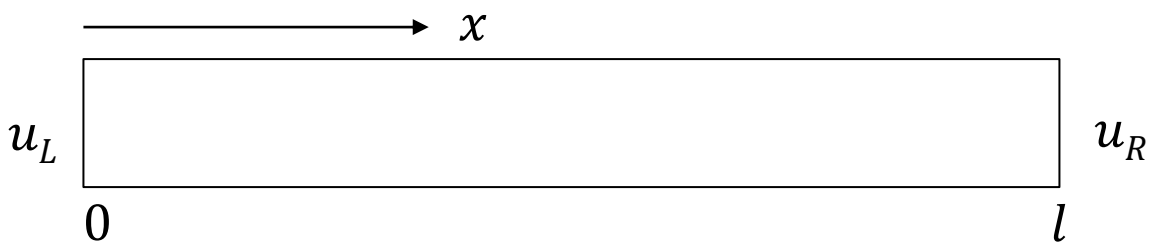
- Example: heat conduction through a rod



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

Initial and Boundary Conditions

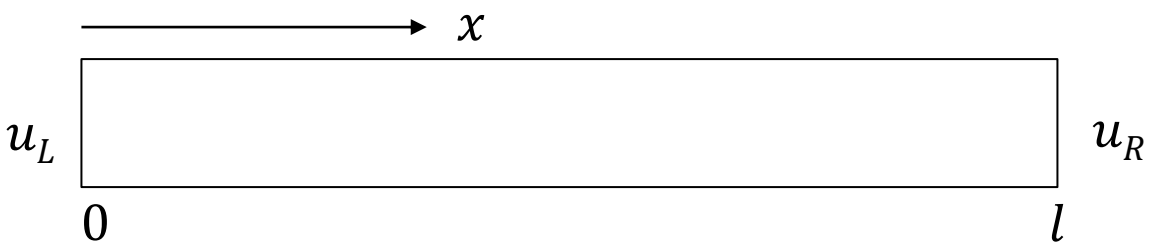
- Example: heat conduction through a rod



- 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



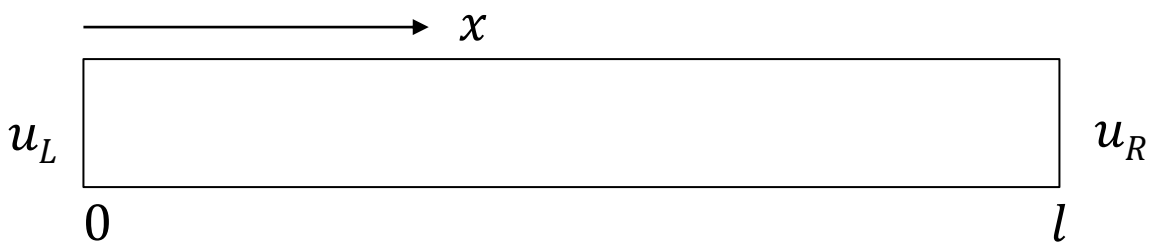
$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \qquad (0 < x < l, t > 0)$$

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



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

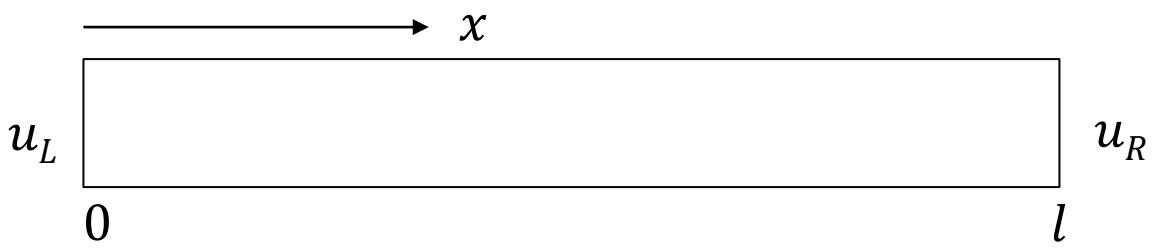
α is thermal diffusivity

(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

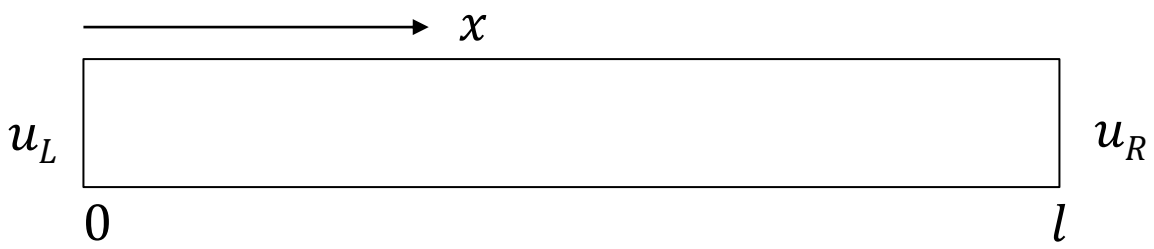
- Example: heat conduction through a rod



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

Equations

- Example: heat conduction through a rod



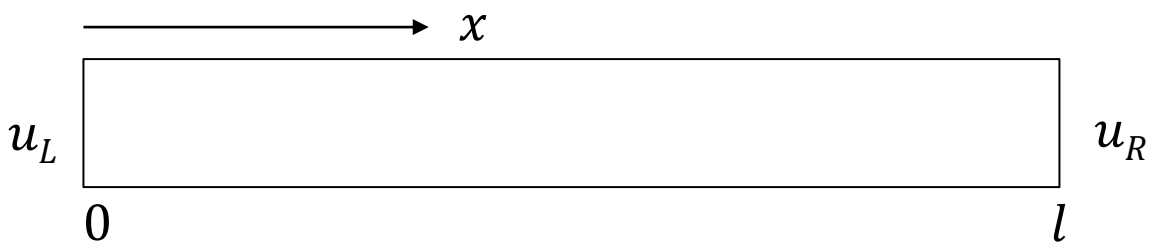
$$\partial_t u = \alpha \Delta u$$

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. $\partial_t u - \alpha \Delta u = 0, 0<x<l, t>0$
  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:

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

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:

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

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

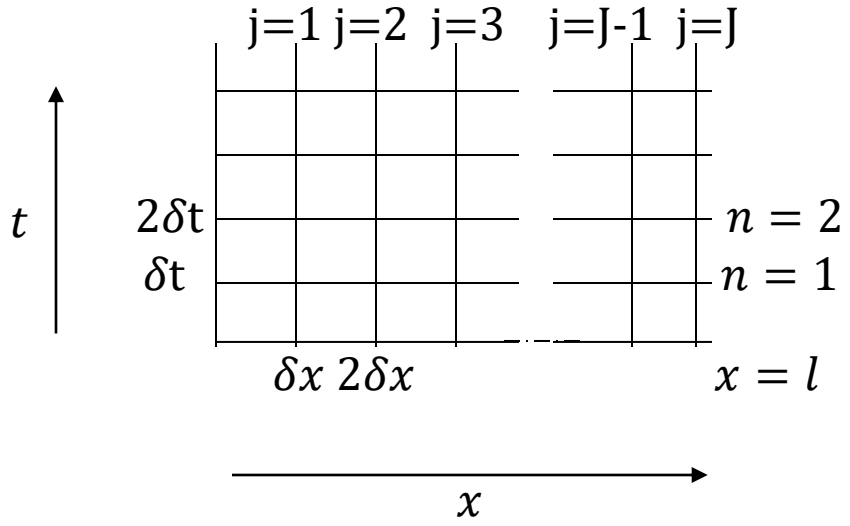
- In example heat application $f = u = u(x,t)$ and $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$
  - First, approximating $\frac{\partial u}{\partial t}$:

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

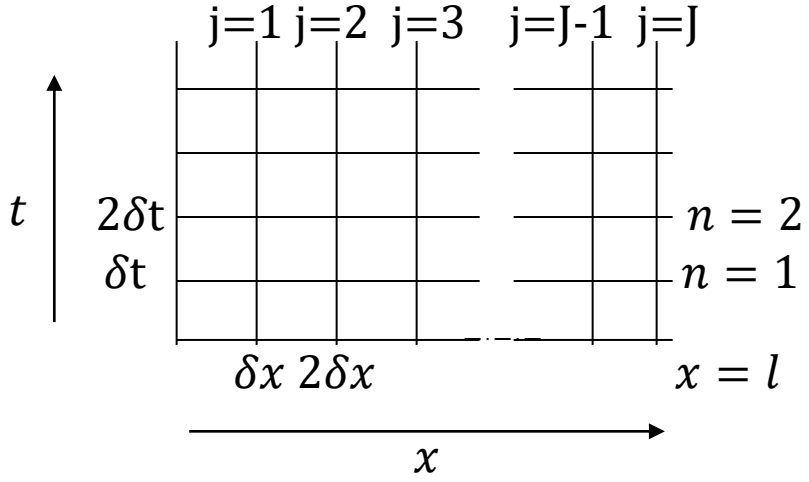
Approximating Partial Derivatives

- 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

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

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

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

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

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

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

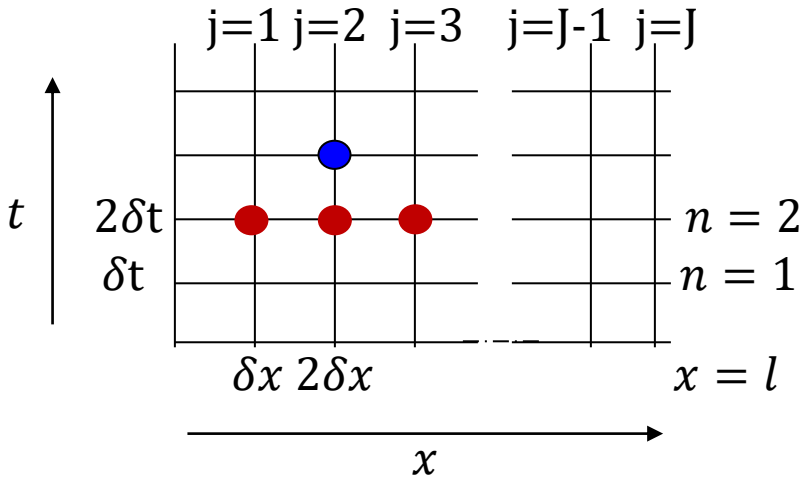
Simplifying:

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

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