CS601: Software Development for Scientific Computing

Autumn 2021

Week3: Structured Grids (Contd..), Version Control System (Git and GitHub), Intermediate C++

Last Week..

- Program Development Environment – Demo
- ‘C’ subset of C++ and reference variables in C++
- Discretization and issues
  - scalability, approximation, and errors (discretization error and solution error), error estimates
  - mesh of cells/elements, cell shapes and sizes
- Structured Grids
  - ‘Regularity’ of cell connectivity (e.g. neighbors are similar kind of cells)
  - Case study – problem statement, representation (e.g. 2D arrays)

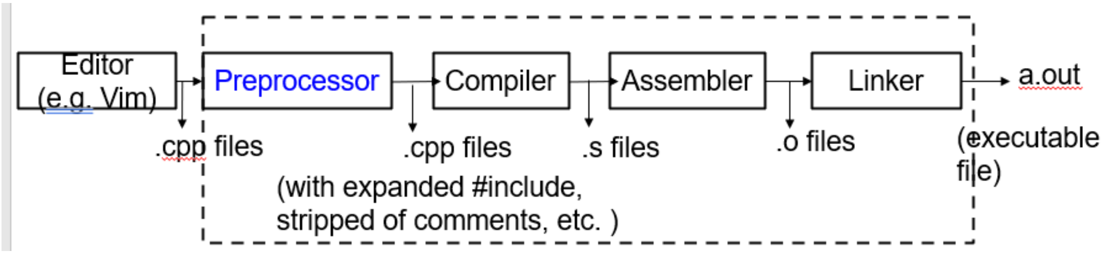
Review of Solution to Exercise: Product of Vectors

- Input sanity check using `istringstream`
- Good programming style: separation of the interface from implementation
  - Streams
  - Passing arrays to functions
  - Pragmas and preprocessor directives
  - Namespaces
- In the sample code, we have so many versions!

Demo

Detour - Conditional Compilation

- Set of 6 preprocessor directives and an operator.
  - #if
  - #ifdef
  - #ifndef
  - #elif
  - #else
  - #endif
- Operator ‘defined’



#if

```
#if <constant-expression>
cout<<“CS601”;
#endif
```

//This line is compiled only if `<constant-expression>` evaluates to a value > 0 while preprocessing

```
#define COMP 0
#if COMP
cout<<“CS601”
#endif
```

No compiler error

```
#define COMP 2
#if COMP
cout<<“CS601”
#endif
```

Compiler throws error about missing semicolon

#ifdef

```
#ifdef identifier
cout<<"CS601";
#endif
```

//This line is compiled only if identifier is defined before the previous line is seen while preprocessing.

`identifier` does not require a value to be set. Even if set, does not care about 0 or > 0.

```
#define COMP
#ifdef COMP
cout<<"CS601"
#endif
```

```
#define COMP 0
#ifdef COMP
cout<<"CS601"
#endif
```

```
#define COMP 2
#ifdef COMP
cout<<"CS601"
#endif
```

All three snippets throw compiler error about missing semicolon

#else and #elif

```
1. #ifdef identifier1
2. cout<<“Summer”
3. #elif identifier2
4. cout<<“Fall”;
5. #else
6. cout<<“Spring”;
7. #endif
```

//preprocessor checks if identifier1 is defined. if so, line 2 is compiled. If not, checks if identifier2 is defined. If identifier2 is defined, line 4 is compiled. Otherwise, line 6 is compiled.

`defined` operator

Example:

```
#if defined(COMP)
cout<<“Spring”;
#endif
```

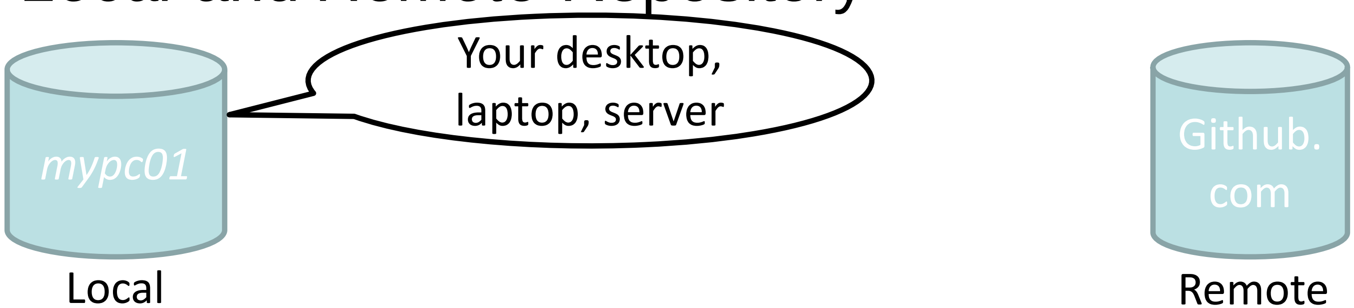
//same as if #ifdef COMP

```
#if defined(COMP1) || defined(COMP2)
cout<<“Spring”;
#endif
```

//if either `COMP1` or `COMP2` is defined, the `printf` statement is compiled. As with `#ifdef`, `COMP1` or `COMP2` values are irrelevant.

Git

- Example of a Version Control System
  - Manage versions of your code – access to different versions when needed
  - Lets you collaborate
- ‘Repository’ – term used to represent storage
  - *Local* and *Remote* Repository



Git – Creating Repositories

- Two methods:
  1. ‘Clone’ / Download an existing repository from GitHub
  2. Create local repository first and then make it available on GitHub

Method 1: `git clone` for creating local working copy

– 'Clone' / Download an existing repository from GitHub – get your own copy of source code

- `git clone` (when a remote repository on GitHub.com exists)

```
nikhilh@ndhpc01:~$ git clone git@github.com:IITDhCSE/dem0.git
Cloning into 'dem0'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
nikhilh@ndhpc01:~$
```

Method 2: `git init` for initializing local repository

– Create local repository first and then make it available on GitHub

1. `git init`

converts a directory to Git local repo

```
nikhilh@ndhpc01:~$ mkdir dem0
nikhilh@ndhpc01:~$ cd dem0/
nikhilh@ndhpc01:~/dem0$ git init
Initialized empty Git repository in /home/nikhilh/dem0/.git/
nikhilh@ndhpc01:~/dem0$ ls -a
.  ..  .git
```

`git` add for staging files

`2. git add`

‘stage’ a file i.e. prepare for saving the file on local repository

```
nikhilh@ndhpc01:~$ ls -a dem0/
.  ..  README
nikhilh@ndhpc01:~$ cd dem0/
nikhilh@ndhpc01:~/dem0$ git init
Initialized empty Git repository in /home/nikhilh/dem0/.git/
nikhilh@ndhpc01:~/dem0$ git add README
```

Note that creating a file, say, README2 in dem0 directory does not *automatically* make it part of the local repository

`git commit` for saving changes in local repository

3. `git commit`

'commit' changes i.e. save all the changes (adding a new file in this example) in the local repository

```
nikhilh@ndhpc01:~/dem0$ git commit -m "Saving the README file in local repo."
[master (root-commit) 99d0a63] Saving the README file in local repo.
 1 file changed, 1 insertion(+)
 create mode 100644 README
```

How to save changes done when you must overwrite an existing file?

Method 2 only: `git branch` for branch management

4. `git branch –M master`

rename the current as ‘`master`’ (-M for force rename even if a branch by that name already exists)

Method 2 only: `git remote add`

5. `git remote add origin git@github.com:IITDhCSE/dem0.git` – prepare the local repository to be managed as a tracked

```
nikhilh@ndhpc01:~/dem0$ git remote add origin git@github.com:IITDhCSE/dem0.git
```

command to manage remote repo.

associates a name ‘origin’ with the remote repo’s URL

The URL of the repository on GitHub.com.

- This URL can be that of any other user’s or server’s address.
- uses SSH protocol
  - HTTP protocol is an alternative. Looks like: https://github.com/IITDhCSE/dem0.git

Method 2 only: GitHub Repository Creation

5.a) Create an empty repository on GitHub.com

(name must be same as the one mentioned previously – `dem0`)

`git push` for saving changes in remote repo

6. `git push -u origin master` – ‘push’ or save all the changes done to the ‘master’ branch in local repo to remote repo. *(necessary for guarding against deletes to local repository)*

```
nikhilh@ndhpc01:~/dem0$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 284 bytes | 47.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:IITDhCSE/dem0.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

syntax: git push <remotename> <branchname>

what does the –u option do?

Git – Releasing Code

– Tagging

- Check for unsaved changes in local repository.

```
nikhilh@ndhpc01:~/dem0$ git status .
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

- Create a tag and associate a comment with that tag

```
nikhilh@ndhpc01:~/dem0$ git tag -a VERSION1 -m "Release version 1 implements feature XYZ"
```

- Save tags in remote repository

```
nikhilh@ndhpc01:~/dem0$ git push --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 191 bytes | 95.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To github.com:IITDhCSE/dem0.git
 * [new tag]         VERSION1 -> VERSION1
```

Git – Recap..

```
1. git clone (creating a local working copy)
2. git add (staging the modified local copy)
3. git commit (saving local working copy)
4. git push (saving to remote repository)
5. git tag (Naming the release with a label)
6. git push --tags (saving the label to remote)
```

- Note that commands 2, 3, and 4 are common to Method 1 and Method 2.
- Please read https://git-scm.com/book/en/v2 for details

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 (Laplace / Poisson equation): *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 $u: \mathbb{R}^n \to \mathbb{R}$

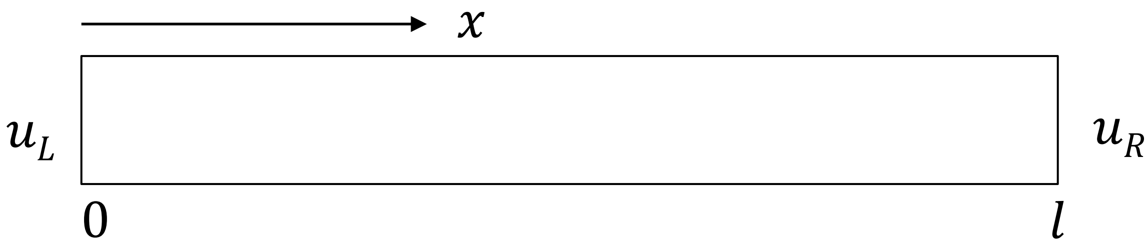
$$\Delta u \quad = \sum_{k=1}^{n} \partial_{kk} u$$

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

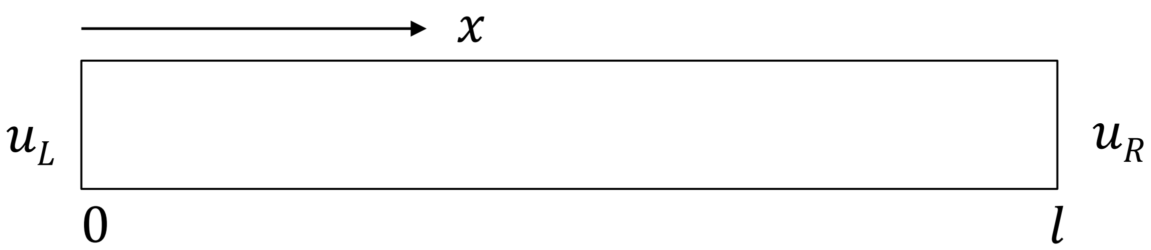
- Example: heat conduction through a rod



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

Initial and Boundary Conditions

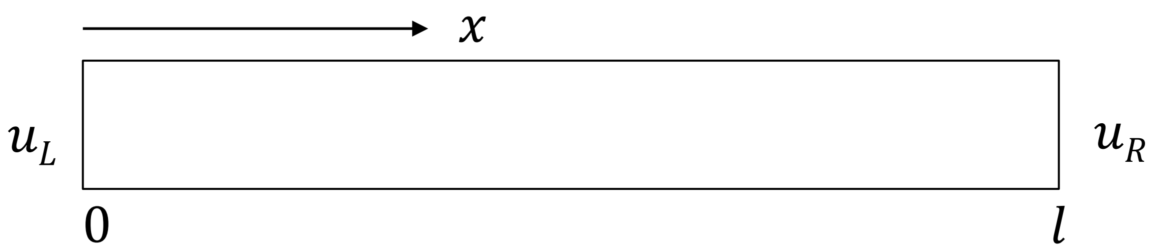
- Example: heat conduction through a rod



- 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



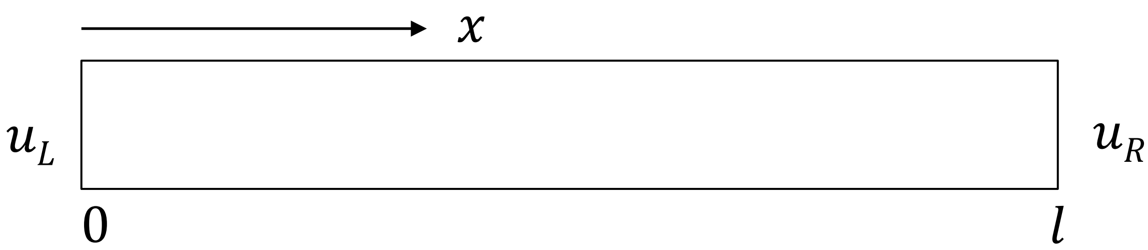
$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \qquad (0 < x < l, t > 0)$$

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



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

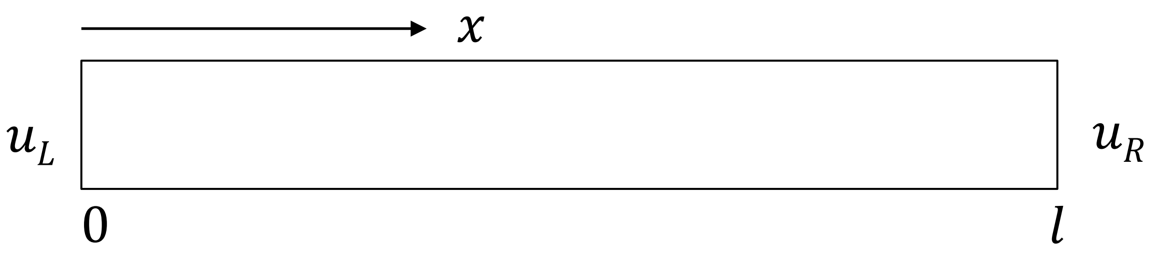
α is thermal diffusivity

(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

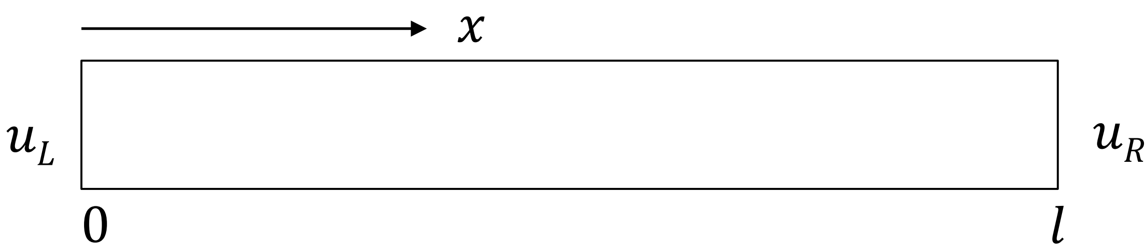
- Example: heat conduction through a rod



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

Equations

- Example: heat conduction through a rod



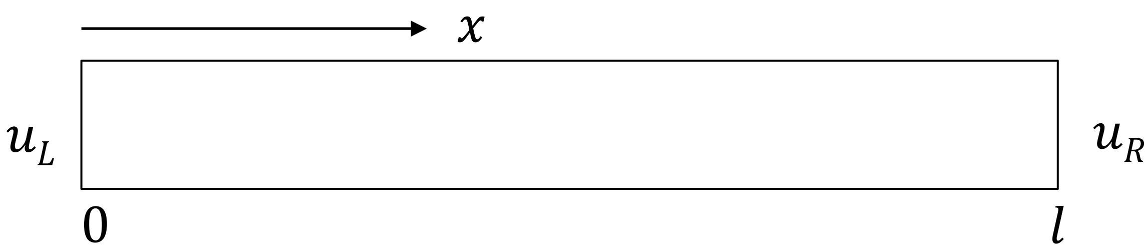
$$\partial_t u = \alpha \Delta u$$

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. $\partial_t u - \alpha \Delta u = 0, 0<\mathrm{x}<\mathrm{l}, \mathrm{t}>0$
  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. *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:
    $$\frac{d^2 f}{dx^2} \approx \frac{\left(f(x+\delta x) - 2f(x) + f(x-\delta x)\right)}{(\delta x)^2}$$

Approximating Partial Derivatives

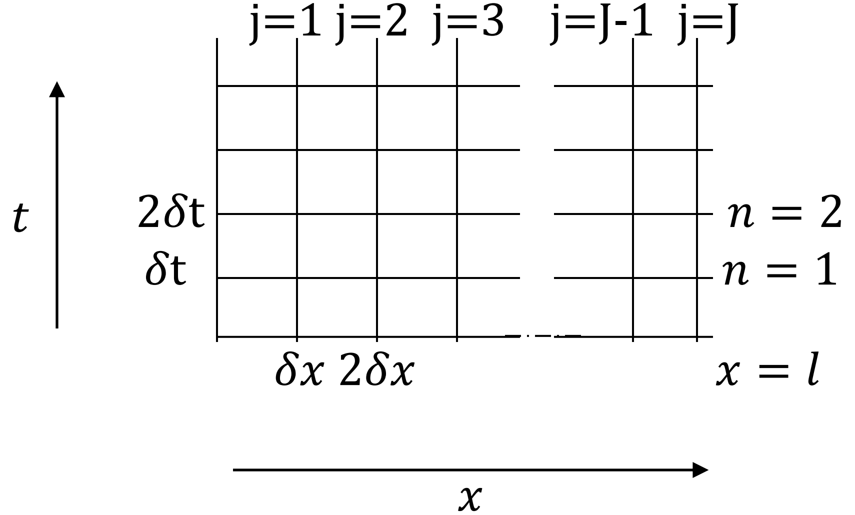
- In example heat application $f = u = u(x,t)$ and $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$
  - First, approximating $\frac{\partial u}{\partial t}$:

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

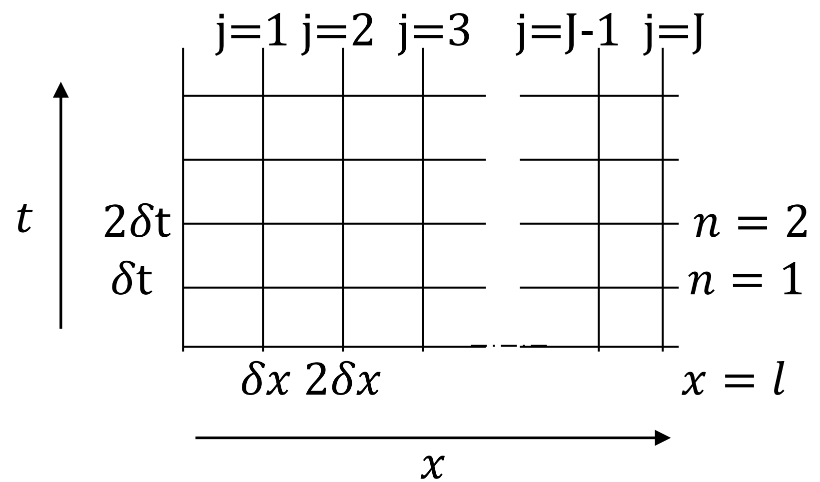
Approximating Partial Derivatives

- 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

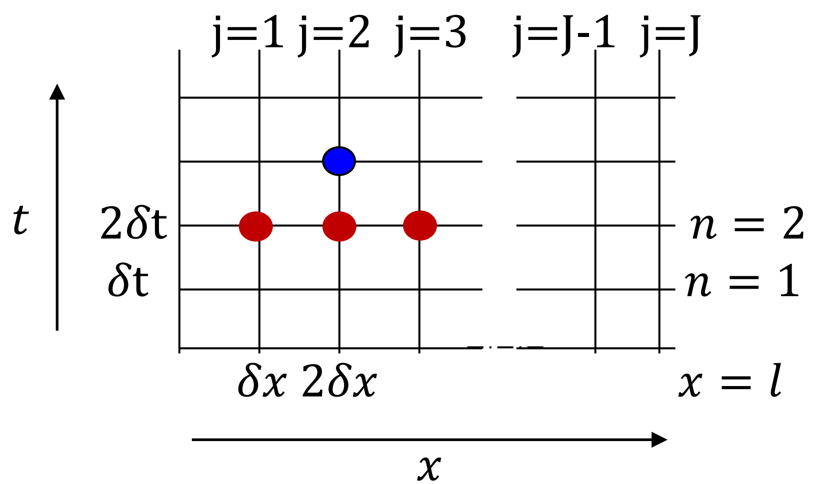
Simplifying:

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

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..

Numerical Methods for Solving PDEs

- Finite Difference Methods
- Finite Volume Methods
- Finite Element Methods
- Boundary Elements Methods
- Isogeometric Analysis
- Spectral Methods

Programming Assignment 1: heads-up

- *Steady-state* heat equation for a metal plate with boundaries at constant temperature

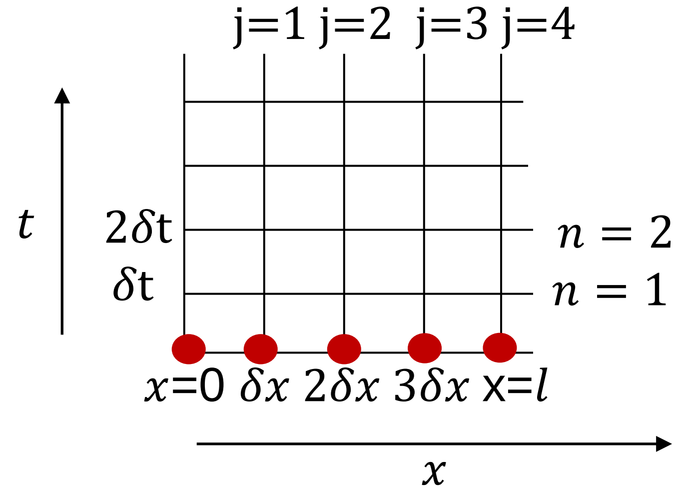
Explicit Difference Method: Stability

- 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*

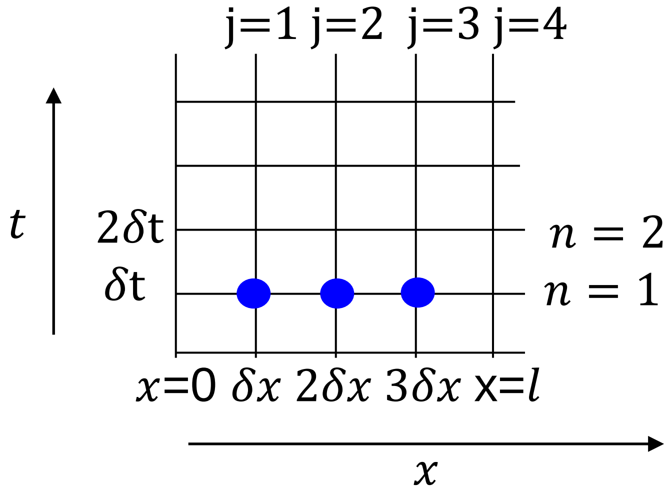
$u_0^0 = 0$
$u_1^0 = f(\delta x) = \delta x(l - \delta x) = .1875$
$u_2^0 = f(2\delta x) = 2\delta x(l - 2\delta x) = .25$
$u_3^0 = f(3\delta x) = 3\delta x(l - 3\delta x) = .1875$
$u_4^0 = 0$



Explicit Difference Method: Stability

- Compute time-step 1 values

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

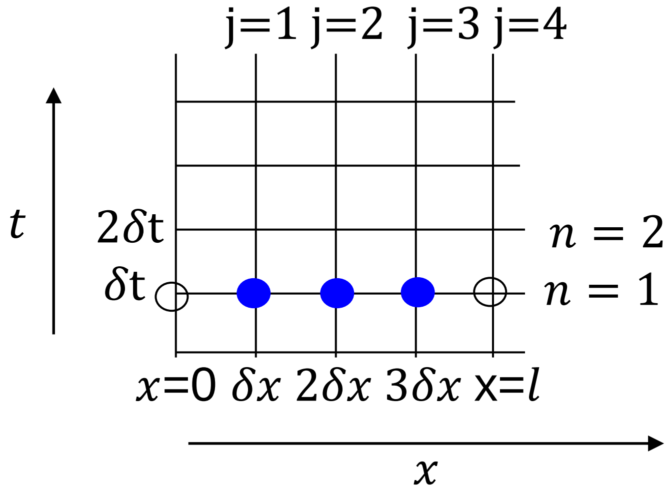


Explicit Difference Method: Stability

- 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 ○ ?



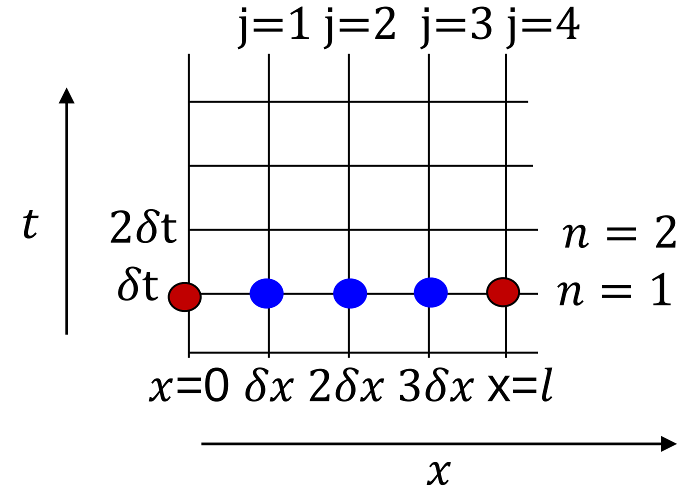
Explicit Difference Method: Stability

- 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



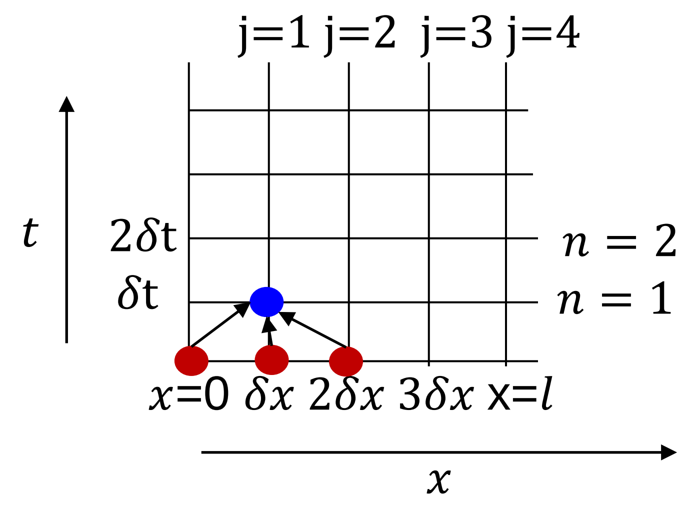
Explicit Difference Method: Stability

- 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



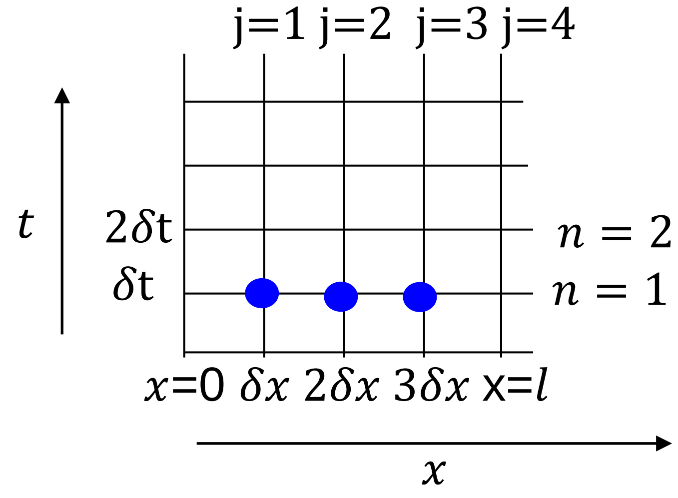
Explicit Difference Method: Stability

- 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

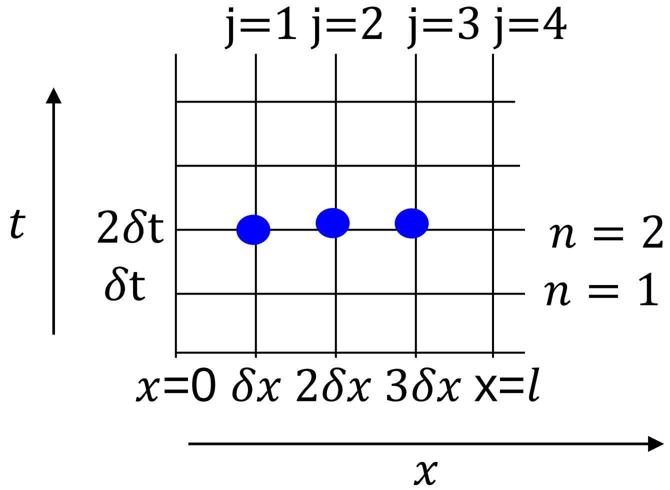


Explicit Difference Method: Stability

- Compute time-step 2 values

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

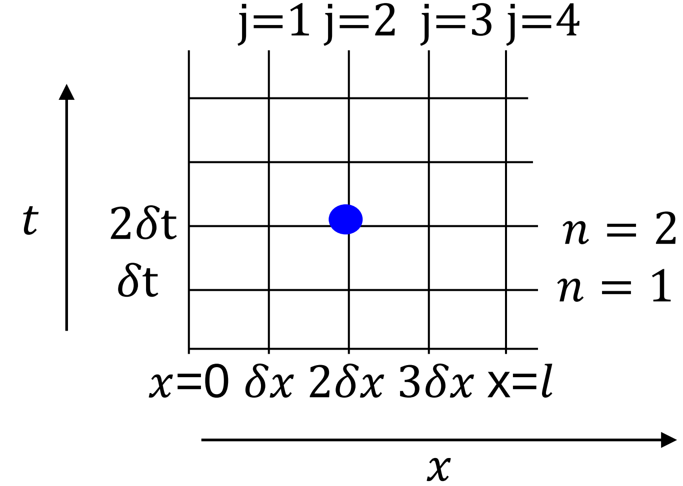
$u_1^2 = u_1^1 + r(u_0^1 - 2u_1^1 + u_2^1)$=0.06851
$u_2^2 = u_2^1 + r(u_1^1 - 2u_2^1 + u_3^1)$=**-0.05173**
$u_3^2 = u_3^1 + r(u_2^1 - 2u_3^1 + u_4^1)$=0.06851



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:

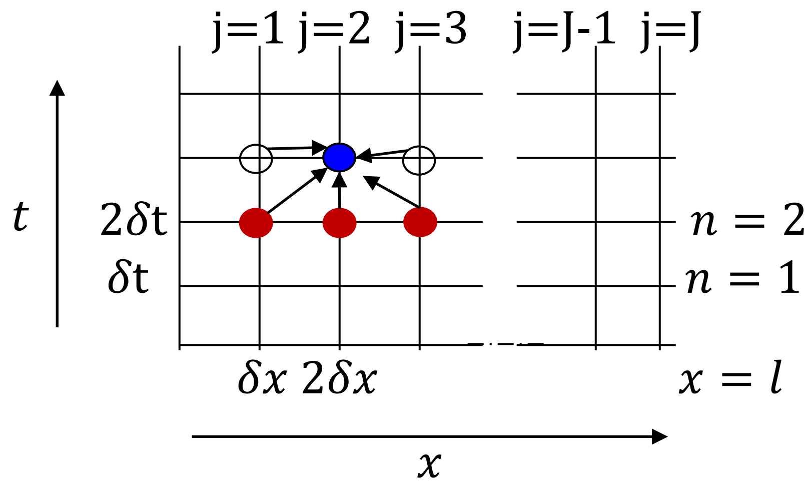
$$\delta t \leq \frac{\delta x^2}{2\alpha}$$

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

Implicit Method: Stability

- Overcoming instability:

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

- Extra work involved to determine the values of unknowns in a time step
  - Solve a system of simultaneous equations. Is it worth it?

Definitions

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

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \qquad (1)$$

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

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*