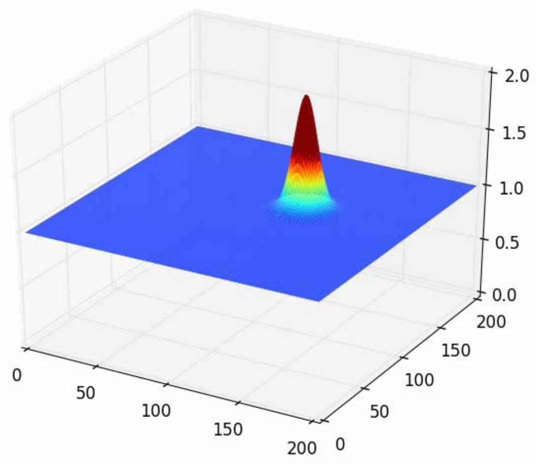


偏微分方程式と物理現象

Partial Differential Equations for Science and Engineering

Outline of Class

Instructor

- VARQUEZ Alvin Christopher Galang

  Field：Urban meteorology and hydrology

  Department of Transdisciplinary Science and Engineering

  School of Environment and Society

  Ishikawadai Rm. 403 I4-9

  e-mail: varquez.a.aa@m.titech.ac.jp

  office: 03-5734-2768

Recommended Textbook

- Advanced Engineering Mathematics by Erwin Kreyszig
  (Available in Tokyo Tech library)

Outline of Class

- Fundamental knowledge of PDE, Analytical and Numerical Solutions.
- Lectures and exercises on problems in Science and Engineering, deriving the PDE for physical phenomena, properties of PDE, methods for obtaining analytical and numerical solutions.
- Acquire the ability to perform numerical analyses by combining lectures with computational exercises.

Outline of Class

Schedule:

- Lecture 1 Definition and types of PDE
- Lecture 2 Modeling of flow phenomena with the hyperbolic PDE
- Lecture 3 Analytical solution of the hyperbolic PDE (D'Alembert Solution)
- Lecture 4 Analytical solution of the hyperbolic PDE (Fourier series)
- Lecture 5 Modeling of diffusion phenomena with the parabolic PDE
- Lecture 6 Analytical solution of the parabolic PDE
- Lecture 7 Derivation and solution of elliptical equation Poisson
- Lecture 8 Test level of understanding (Exercise Problems for Lecture 1-7)
- Lecture 9 Introduction to Numerical analysis
- Lesson 10・11 Numerical solution of PDE – Parabolic equation
- Lesson 12・13 Numerical solution of PDE – Combined Parabolic/Hyperbolic
- Lesson 14・15 Numerical solution of PDE – Hyperbolic equation
- Term-end Report

Grading System

mid-term examination (45%), term-end report (45%), short exercises (10%)

Today’s Objectives

- Schedule and Strategy
- Finite Difference Method
- Basic algorithm structure, outputting, and animation using python and ffmpeg.

SCHEDULE & STRATEGY

Style

- Sit and work together with assigned group to building the models in groups. May freely consult with other groups (free discussion) and myself.
- Tasks are given on the day.
- Concepts on numerical methods are taught on Mondays, computer programming on Thursdays.
- Compile a full report of tasks by November 27, 2017. (English language)
- Submission style: Print-outs and Soft-copies (zipped pdf and animations).
- Submission deadline: November 27, 2017, 17:15

Grouping

Group			
1	Kiyo Kei	Ratchatawijin Maythawee	
2	Thumwanit Napat	Chin Riyu	
3	Zhou Lijun	Lerdbussarakam Tanad	Shagdar Zolbayar
4	Tsamara Tsani	Ka Kinzei	
5	Pongpattanayok Supatat	Jiang Lechuan	
6	Tumurbaatar Uyanga	Kiyo Yu	
7	Kadohiro Yasuki	Dolgormaa Banzragch	
8	Ka Jiyuntaku	Rudjito Rezkita Ramadhani	
9	Nasution Ghiffari Aby Malik	Kobayashi Yuki	
10	Chaijirawiwat Chawit	Chin Keitoku	
11	Sato Yaoki	Purevsuren Norovsambuu	
12	Do Khanh N	Kazama Tomohiro	

Target PDEs

- 2-D diffusion phenomena
  - November 2, 2017 (Thursday)
- 2-D Burger's equation
  - November 9, 2017 (Thursday)
- 1-D Advection (Stability tests)
  - November 16, 2017 (Thursday)

FINITE DIFFERENCE METHOD

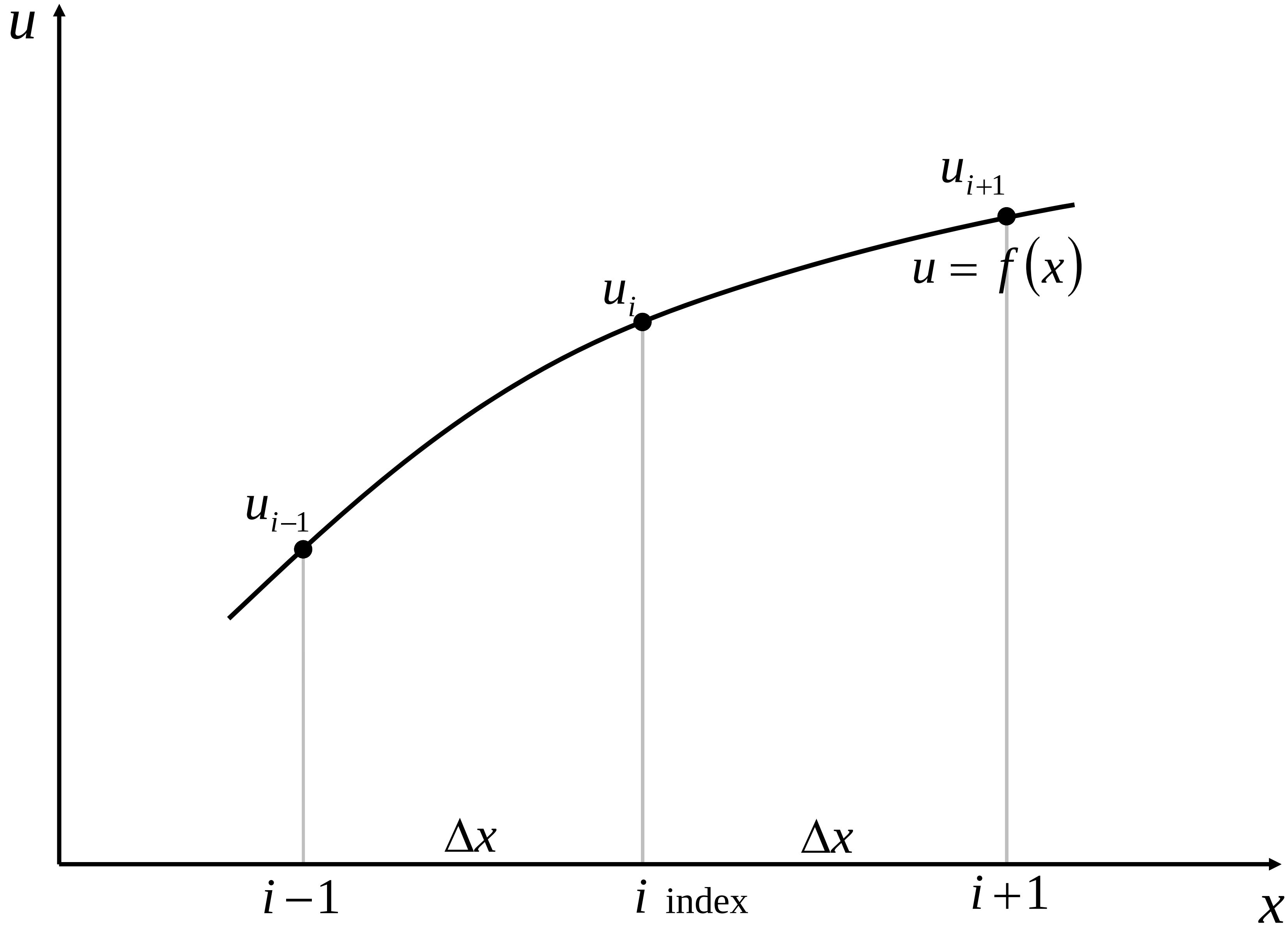
Discretization of Differential Equations

- Finite Difference Method (FDM) (1910~present)
  - Formulation is easy.
  - Widely popular.
  - Mesh must be structured.
  - Neumann boundary conditions are approximated.
- Finite Element Method (FEM) (1956~present)
  - More time consuming for solving non-linear PDEs.
  - On-going development and gaining popularity.
  - Good for complex geometries and unstructured meshes.
  - Neumann boundary conditions are enforced.
- Finite Volume Method (recent)
  - Derived from either FDM or FEM

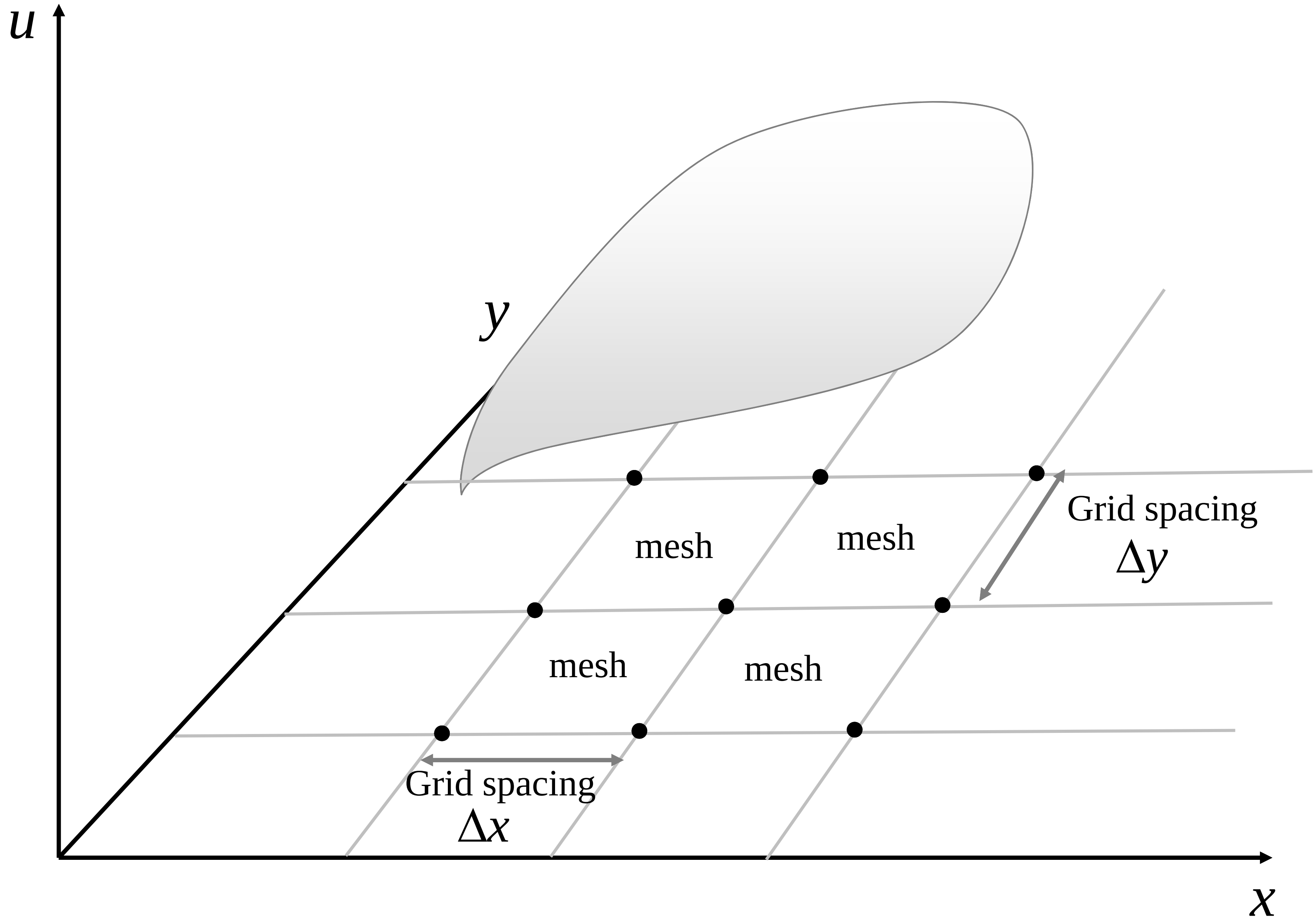
Basic idea of FDM

Derivatives in differential equations are written in terms of *discrete* quantities of dependent and independent variables, resulting in *simultaneous algebraic equations* with all unknowns prescribed at *discrete mesh points* for the entire *domain*.

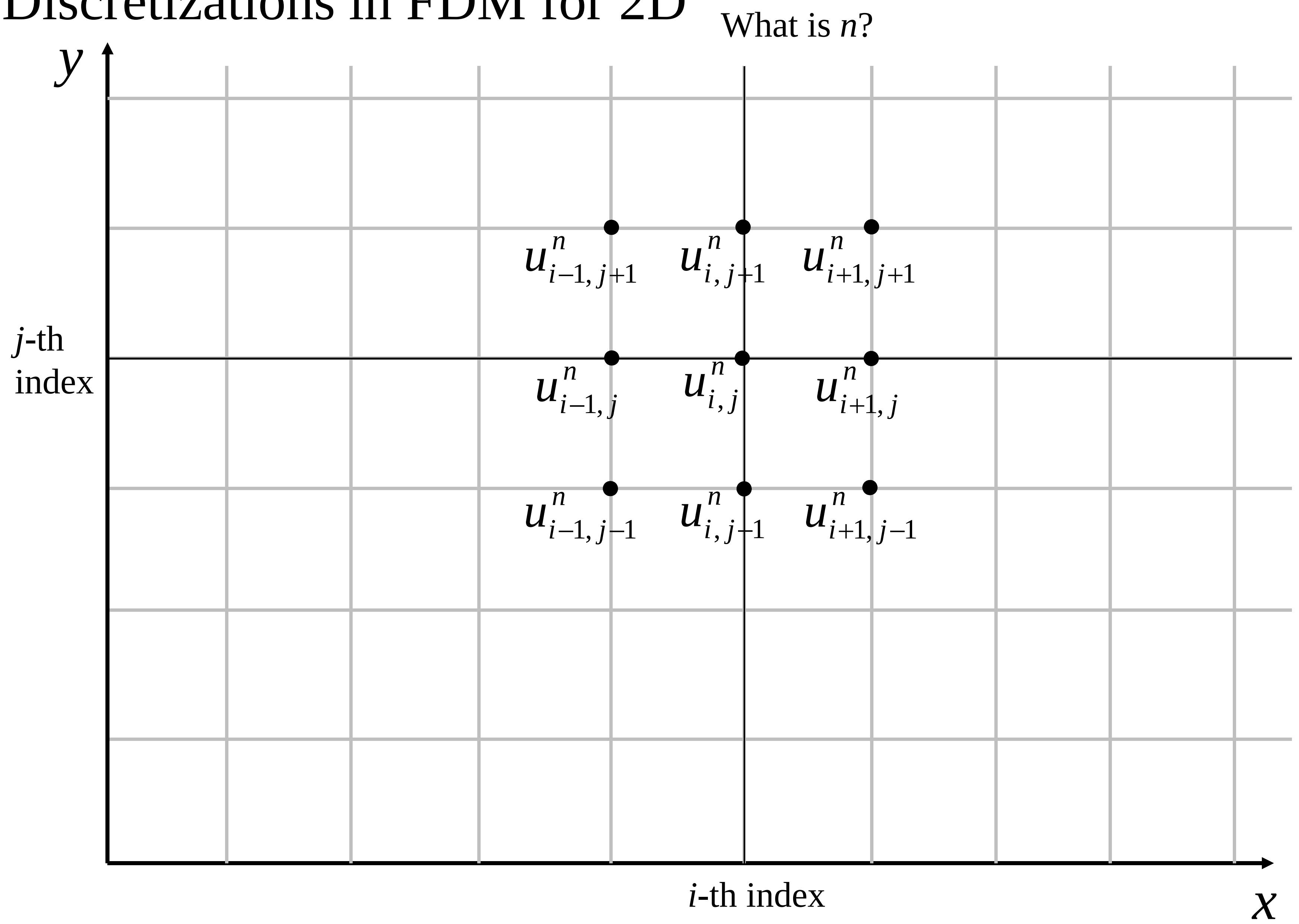
Discretizations in FDM



Discretizations in FDM for 2D



Discretizations in FDM for 2D



Basic FDM Methods

Consider a function $u(x)$ and its derivative at point x,

$$\frac{\partial u(x)}{\partial x} = \lim_{\Delta x \to 0} \frac{u(x+\Delta x) - u(x)}{\Delta x}$$

Expanding $u(x+\Delta x)$ using Taylor's series about $u(x)$,

$$u(x+\Delta x) = u(x) + \Delta x \frac{\partial u(x)}{\partial x} + \frac{(\Delta x)^2}{2} \frac{\partial^2 u(x)}{\partial x^2} + \frac{(\Delta x)^3}{3!} \frac{\partial^3 u(x)}{\partial x^3} + \ldots$$

Substituting $u(x+\Delta x)$ to the derivative function,

$$\frac{\partial u(x)}{\partial x} = \lim_{\Delta x \to 0} \left(\frac{\partial u(x)}{\partial x} + \frac{(\Delta x)}{2} \frac{\partial^2 u(x)}{\partial x^2} + \ldots \right)$$

Basic FDM Methods

Simplifying,

$$\frac{u(x+\Delta x)-u(x)}{\Delta x}=\frac{\partial u(x)}{\partial x}+\frac{(\Delta x)}{2}\frac{\partial^2 u(x)}{\partial x^2}+\ldots=\frac{\partial u(x)}{\partial x}+\mathrm{O}(\Delta x)$$

Applying our pre-defined indexing,

$$\frac{u_{i+1}-u_i}{\Delta x}=\frac{\partial u(x)}{\partial x}+\frac{(\Delta x)}{2}\frac{\partial^2 u(x)}{\partial x^2}+\ldots=\frac{\partial u(x)}{\partial x}+\mathrm{O}(\Delta x)$$

Re-arranging we get an approximation (forward difference),

$$\frac{\partial u(x)}{\partial x}=\frac{u_{i+1}-u_i}{\Delta x}+\mathrm{O}(\Delta x)$$

Basic FDM Methods

Expanding $u(x+\Delta x)$ using Taylor's series about $u(x)$,

$$u(x+\Delta x)=u(x)+\Delta x\frac{\partial u(x)}{\partial x}+\frac{(\Delta x)^2}{2}\frac{\partial^2 u(x)}{\partial x^2}+\frac{(\Delta x)^3}{3!}\frac{\partial^3 u(x)}{\partial x^3}+\ldots$$

$$\boxed{\frac{u_{i+1}-u_i}{\Delta x}=\frac{\partial u(x)}{\partial x}}+\frac{(\Delta x)}{2}\frac{\partial^2 u(x)}{\partial x^2}+\ldots=\frac{\partial u(x)}{\partial x}+\mathrm{O}(\Delta x)$$

Expanding $u(x-\Delta x)$ using Taylor's series about $u(x)$,

$$u(x-\Delta x)=u(x)-\Delta x\frac{\partial u(x)}{\partial x}+\frac{(\Delta x)^2}{2}\frac{\partial^2 u(x)}{\partial x^2}-\frac{(\Delta x)^3}{3!}\frac{\partial^3 u(x)}{\partial x^3}+\ldots$$

$$\boxed{\frac{u_i-u_{i-1}}{\Delta x}=\frac{\partial u(x)}{\partial x}}-\frac{(\Delta x)}{2}\frac{\partial^2 u(x)}{\partial x^2}+\ldots=\frac{\partial u(x)}{\partial x}+\mathrm{O}(\Delta x)$$

Basic FDM Methods

$$\frac{u_{i+1}-u_i}{\Delta x}=\frac{\partial u(x)}{\partial x}+\frac{(\Delta x)}{2}\frac{\partial^2 u(x)}{\partial x^2}+\ldots=\frac{\partial u(x)}{\partial x}+\mathrm{O}(\Delta x)$$

$$\frac{u_i-u_{i-1}}{\Delta x}=\frac{\partial u(x)}{\partial x}-\frac{(\Delta x)}{2}\frac{\partial^2 u(x)}{\partial x^2}+\ldots=\frac{\partial u(x)}{\partial x}+\mathrm{O}(\Delta x)$$

Adding both equations,

$$\boxed{\frac{u_{i+1}-u_{i-1}}{2\Delta x}=\frac{\partial u(x)}{\partial x}}+\frac{(\Delta x)^2}{3!}\frac{\partial^3 u(x)}{\partial x^3}+\ldots=\frac{\partial u(x)}{\partial x}+\mathrm{O}(\Delta x^2)$$

Basic FDM Methods

Approximation for the 1st order,

$$\frac{\partial u(x)}{\partial x} = \frac{u_{i+1} - u_i}{\Delta x} + \mathrm{O}(\Delta x)$$ Forward Difference

$$\frac{\partial u(x)}{\partial x} = \frac{u_i - u_{i-1}}{\Delta x} + \mathrm{O}(\Delta x)$$ Backward Difference

$$\frac{\partial u(x)}{\partial x} = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + \mathrm{O}(\Delta x^2)$$ Central Difference

Approximation for the 2nd order,

$$\frac{\partial^2 u(x)}{\partial x^2} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + \mathrm{O}(\Delta x^2)$$

Considering the diffusion equation (w/ constant coefficient),

$$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0$$

Utilizing the FTCS Method,

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \mathrm{O}\left(\Delta t, \Delta x^2\right)$$

$$\boxed{u_i^{n+1} = u_i^n + \left(\frac{\alpha \Delta t}{\Delta x^2}\right)\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right)}$$

Treatment of Boundary Conditions

Dirichlet conditions,
The *u* values at the boundaries are fixed for all *n*.

Neumann conditions,
The approximation for the derivate at the boundary is

$$\frac{u_{i+1}^{n} - u_{0}^{n}}{2\Delta x} = G$$

$$u_0^n = u_{i+1}^n - 2\Delta x\left(G\right)$$ (Ghost point)

Mission for the week

$$\frac{\partial T}{\partial t} - \alpha\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) = 0$$

Construct a diffusion model for a 2-D heat plate with dimensions 100 m. by 100 m.

1. Desired initial condition (distributed).
2. Investigate w/ or without Dirichlet or Neumann Boundary Conditions.
3. Investigate various influences of $d = \dfrac{\alpha \Delta t}{\Delta x^2}$. What is happening when its values range from 0. to 1.0.
4. Visualize the results by animation.

Python Programming (python.org)

- High-level programming language for general purpose programming.
- Fewer lines of code compared to C++ and Java.
- Most linux operating systems have built-in Python.
- Compatible with various OS.
- Plotting can be done simultaneously within the program.
- Large resources online.
- Not named after a “snake”.

Summary of basic python structure in this class

1. Import Statement (numpy, matplotlib)
2. Function Assignment (def)
3. The body - Main function (def main())

ffmpeg (ffmpeg.org)

1. Free software project that produces libraries and programs for handling multimedia data.
2. Fast video/audio conversion and editing tool.
3. Capable of creating high quality time-lapses.

DEMONSTRATING PYTHON PROGRAM WITH ANIMATION

Given $y(t,x)$ and the differential equation,

$$\frac{\partial y}{\partial t} = 1$$

Dirichlet boundary condition

$$y(0, x) = 0$$

Visualize the change of y from t = 0 to 10 at 0.1 t-intervals from the region x = 0 to 10 with 100 grid points.

```python
import numpy as np
import matplotlib.pyplot as plt
import os
```

Change work directory to location of script.

```python
directory = os.path.dirname(os.path.realpath(__file__))
os.chdir(directory)
```

```python
def get_value(it):
    y = t
    return t
```

Function declaration.

Main body of the program.

```python
def main():
    t = np.arange(0.,10.,0.1)
    x = np.linspace(0.,10.,num=100)
    y = np.copy(x)
    y[:] = 0.
    icount = 1
```

Setting boundaries, descritization

```python
    for it in t:
        for i in range(0,y.shape[0]):
            y[i] = get_value(it)
```

Looping in time and space.

```python
        plt.clf()
        plt.plot(x,y)
        plt.xlim(0,10)
        plt.ylim(0,10)
```

Plot settings.

```python
        plt.savefig("test_%03di.jpg"%(icount))
        icount+=1
```

Save to file.

```python
if __name__ == "__main__":
    main()
```

Calls the main function. Complete program syntax.

ffmpeg command

```
ffmpeg -r 1 -start_number 1 -i test_%03d.jpg -vcodec libx264 trial.mp4
```

-r [frames/images per second]

-start_number [initial file number]

-I test_%03d.jpg (test_001.jpg,test_002.jpg)

-vcodec [codec for video]

trial.mp4 (output video in mp4 format).

Textbook reference

- Computational Fluid Dynamics by T.J. Chung
  - Downloadable via Tokyo Tech OPAC