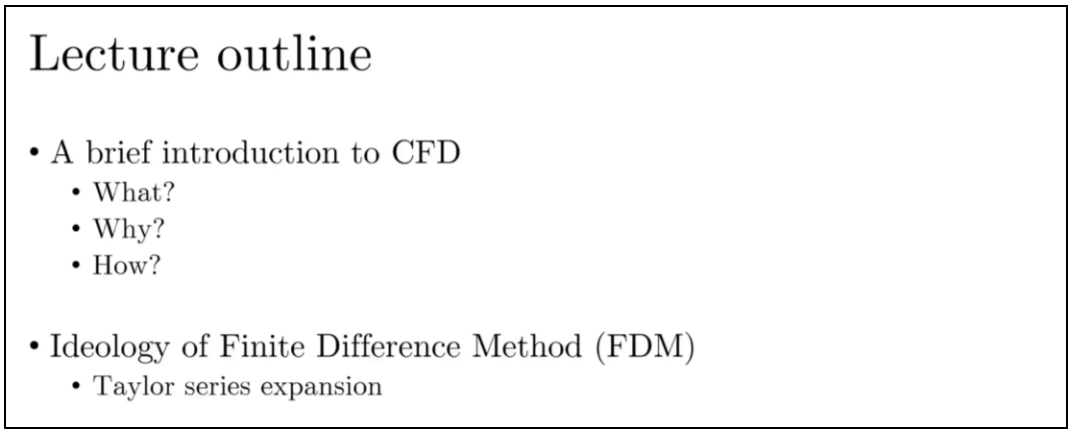
SIMULATING FLUID FLOWS USING PYTHON (SFFP)

Tanmay Agrawal

Carlos Terreros

INDEX

LECTURE 01. INTRODUCTION TO COMPUTATIONAL FLUID DYNAMICS

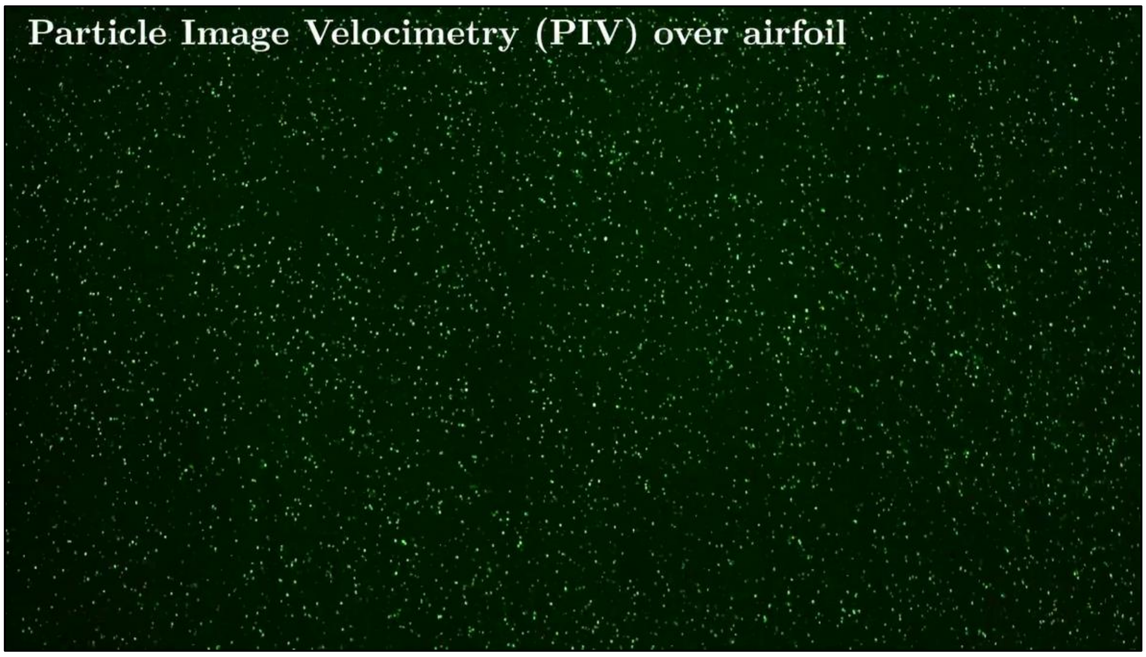


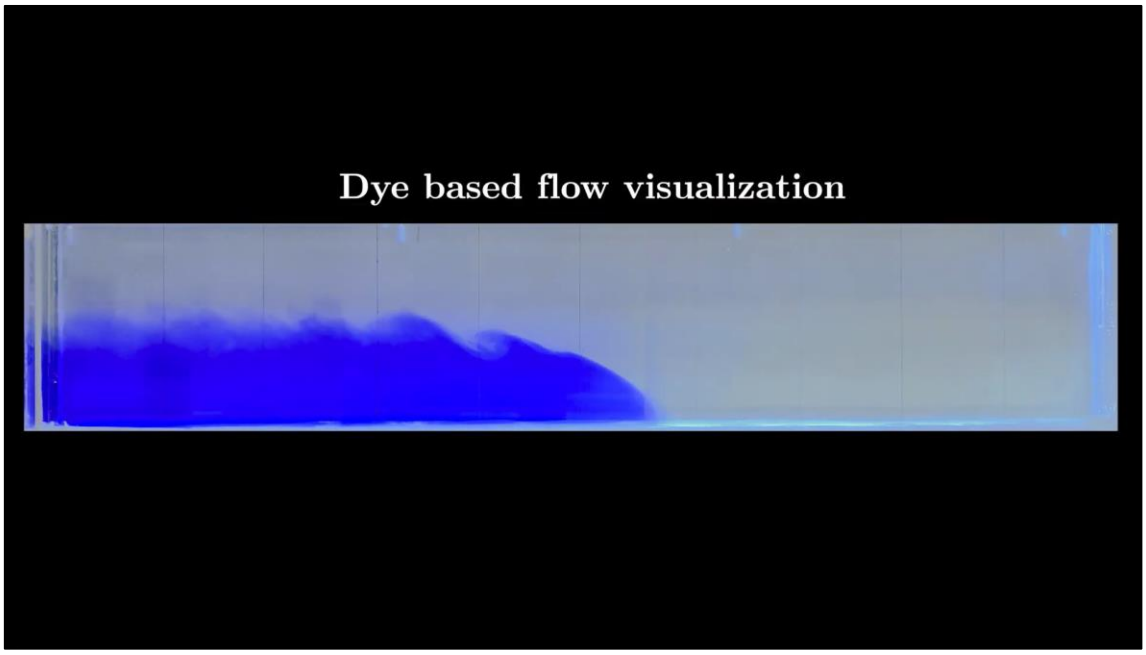
Lecture outline

- A brief introduction to CFD
  - What?
  - Why?
  - How?
- Ideology of Finite Difference Method (FDM)
  - Taylor series expansion

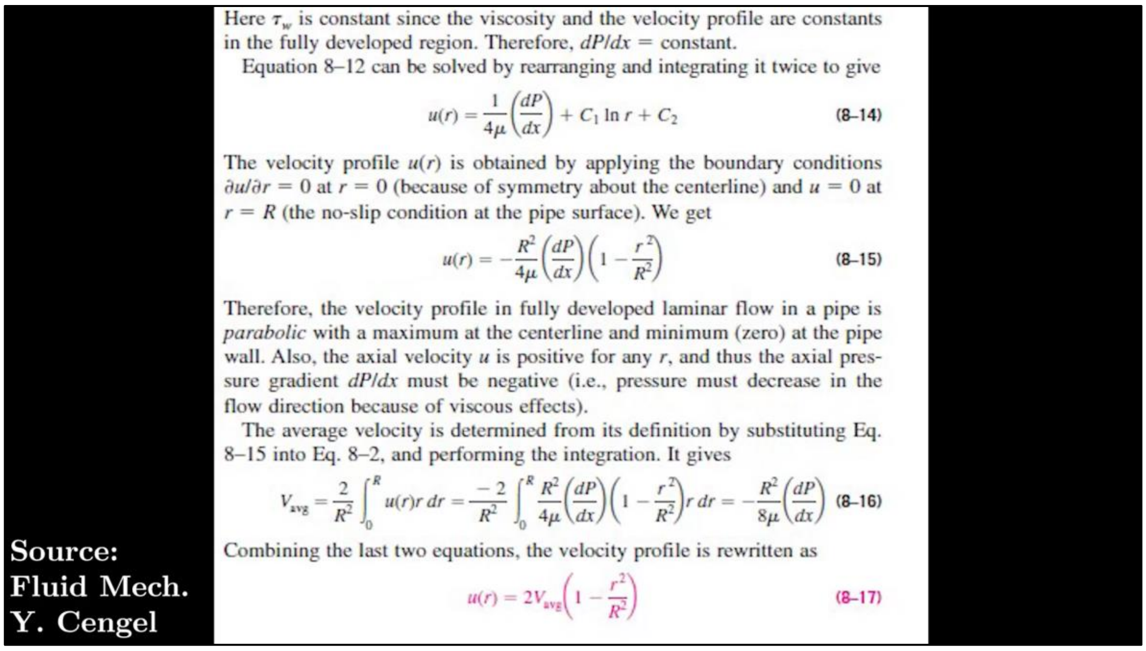
Computational Fluid Dynamics

Experimental Fluid Dynamics:





Theoretical Fluid Dynamics (Analytical Fluid Dynamics):



Here τ_w is constant since the viscosity and the velocity profile are constants in the fully developed region. Therefore, dP/dx = constant.

Equation 8–12 can be solved by rearranging and integrating it twice to give

$$u(r) = \frac{1}{4\mu}\left(\frac{dP}{dx}\right) + C_1 \ln r + C_2 \qquad \textbf{(8–14)}$$

The velocity profile $u(r)$ is obtained by applying the boundary conditions $\partial u/\partial r = 0$ at $r = 0$ (because of symmetry about the centerline) and $u = 0$ at $r = R$ (the no-slip condition at the pipe surface). We get

$$u(r) = -\frac{R^2}{4\mu}\left(\frac{dP}{dx}\right)\left(1 - \frac{r^2}{R^2}\right) \qquad \textbf{(8–15)}$$

Therefore, the velocity profile in fully developed laminar flow in a pipe is *parabolic* with a maximum at the centerline and minimum (zero) at the pipe wall. Also, the axial velocity u is positive for any r, and thus the axial pressure gradient dP/dx must be negative (i.e., pressure must decrease in the flow direction because of viscous effects).

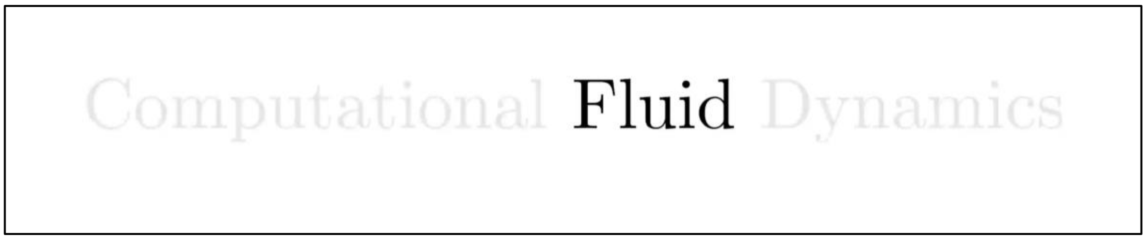
The average velocity is determined from its definition by substituting Eq. 8–15 into Eq. 8–2, and performing the integration. It gives

$$V_{\text{avg}} = \frac{2}{R^2}\int_0^R u(r)r\,dr = \frac{-2}{R^2}\int_0^R \frac{R^2}{4\mu}\left(\frac{dP}{dx}\right)\left(1 - \frac{r^2}{R^2}\right)r\,dr = -\frac{R^2}{8\mu}\left(\frac{dP}{dx}\right) \qquad \textbf{(8–16)}$$

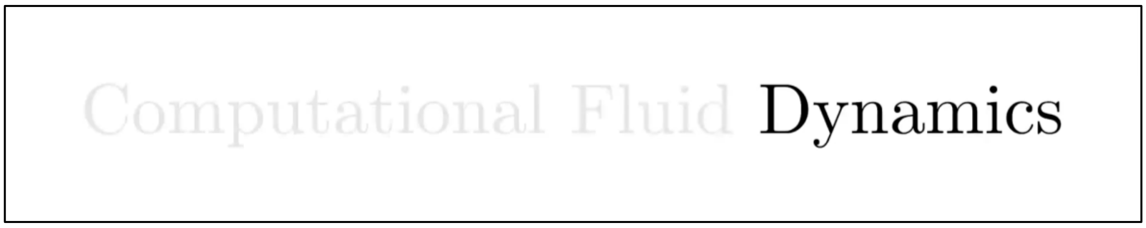
Combining the last two equations, the velocity profile is rewritten as

$$u(r) = 2V_{\text{avg}}\left(1 - \frac{r^2}{R^2}\right) \qquad \textbf{(8–17)}$$

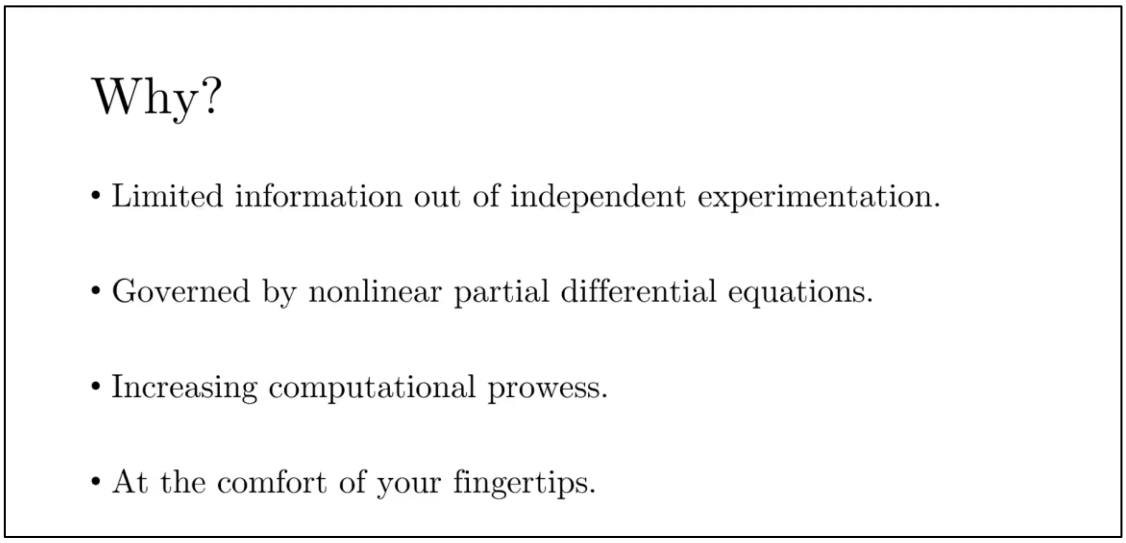
Source: Fluid Mech. Y. Cengel


Computational **Fluid** Dynamics

Any material that can not resist an applied sheer stress: Fluid.


Computational Fluid **Dynamics**

Interest in both the kinematic parameters (velocity, acceleration) and the effect that the variables cause (forces).


Why?

- Limited information out of independent experimentation.
- Governed by nonlinear partial differential equations.
- Increasing computational prowess.
- At the comfort of your fingertips.

Governing equations

- 2D Incompressible flow:

Conservation of mass:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$$

Conservation of momentum:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\vartheta\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+F_x$$

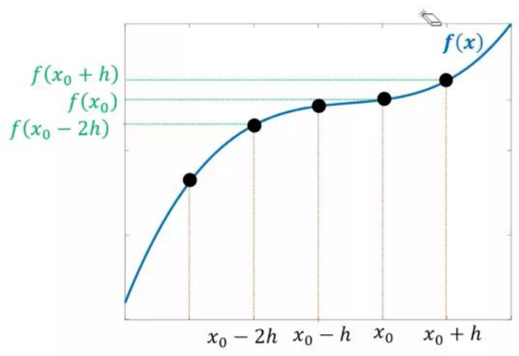
$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\vartheta\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+F_y$$

Suggested books

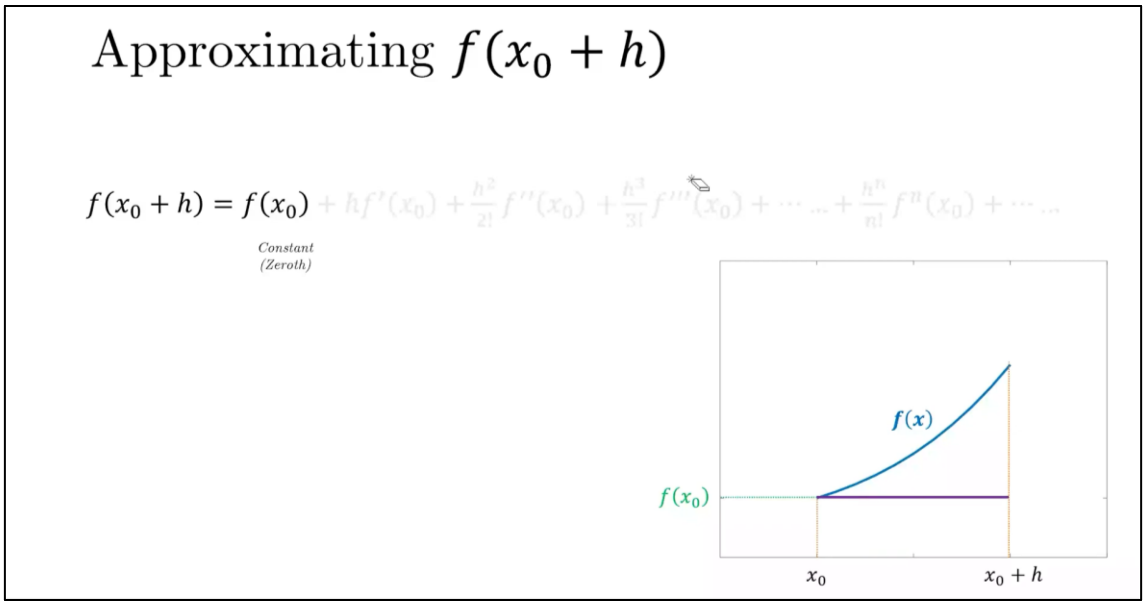
- An Introduction to Computational Fluid Dynamics – The Finite Volume Method
  H. Versteeg and W. Malalasekra
- Numerical Heat Transfer and Fluid Flow
  Suhas V. Patankar
- Numerical Python: Scientific Computing and Data Science
  Robert Johansson

Idea behind Finite Differencing

- **Given:** Value of a function, f, at multiple *grid* points.



Objective: To estimate gradients or derivatives of f.



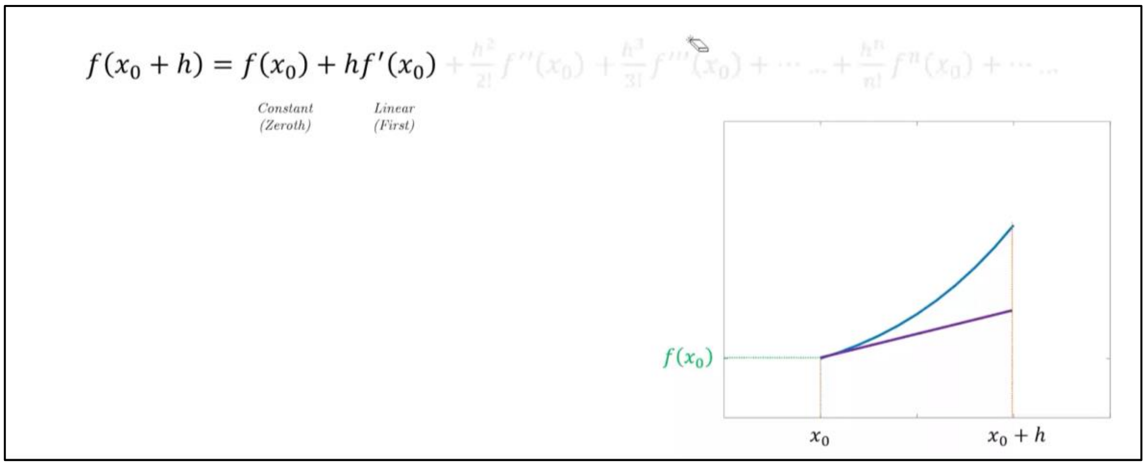
Approximating $f(x_0+h)$

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \cdots\ldots + \frac{h^n}{n!}f^n(x_0) + \cdots\ldots$$

Constant (Zeroth)

Zeroth approximation: represents a flat line.

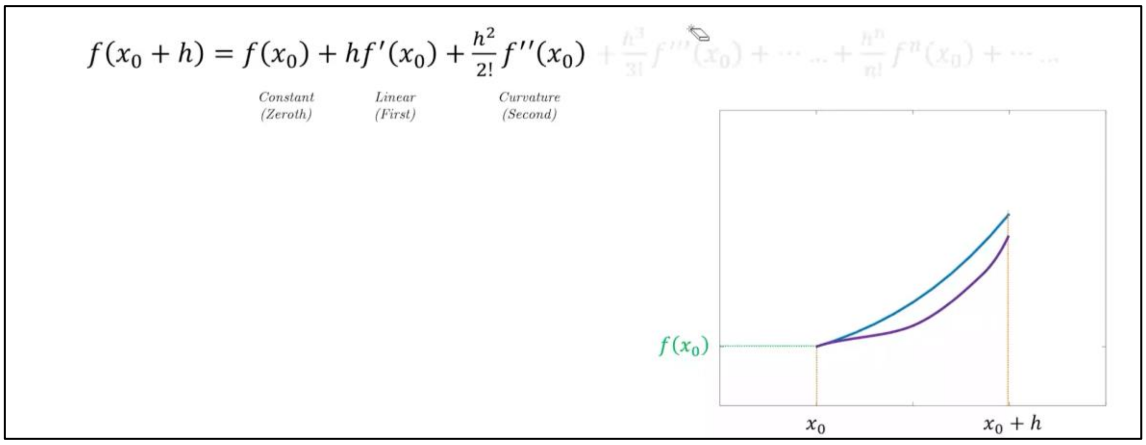
1st derivative represents the slope of the curve at a particular point:



$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \cdots\ldots + \frac{h^n}{n!}f^n(x_0) + \cdots\ldots$$

Constant (Zeroth) *Linear (First)*

2nd derivative gives information about the curvature:



$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \cdots\ldots + \frac{h^n}{n!}f^n(x_0) + \cdots\ldots$$

Constant (Zeroth) *Linear (First)* *Curvature (Second)*

Definition of Taylor Series

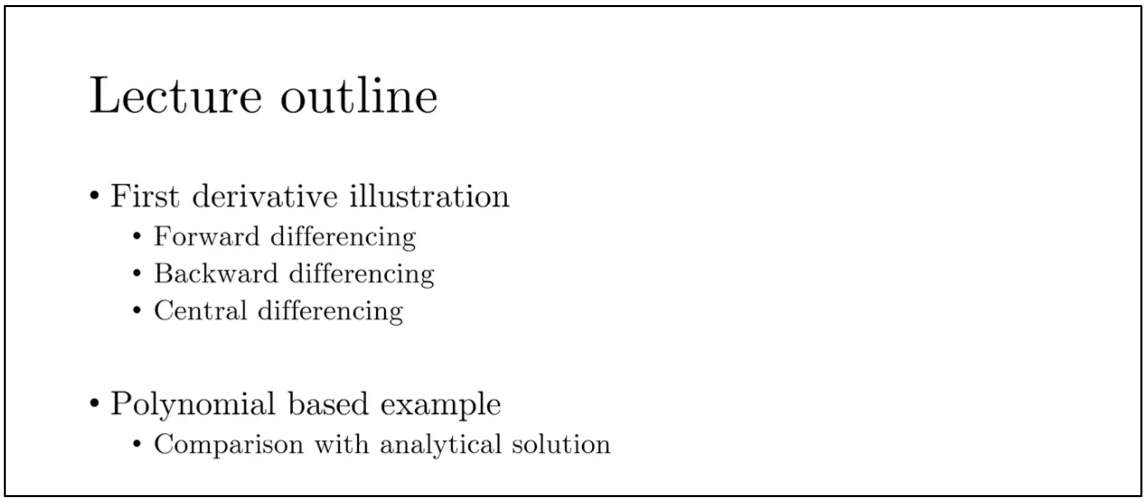
General formulation:

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \cdots \ldots + \frac{h^n}{n!}f^n(x_0) + \cdots \ldots$$

$$f(x_0-h) = f(x_0) - hf'(x_0) + \frac{h^2}{2!}f''(x_0) - \frac{h^3}{3!}f'''(x_0) + \cdots \ldots + (-1)^n\frac{h^n}{n!}f^n(x_0) + \cdots$$

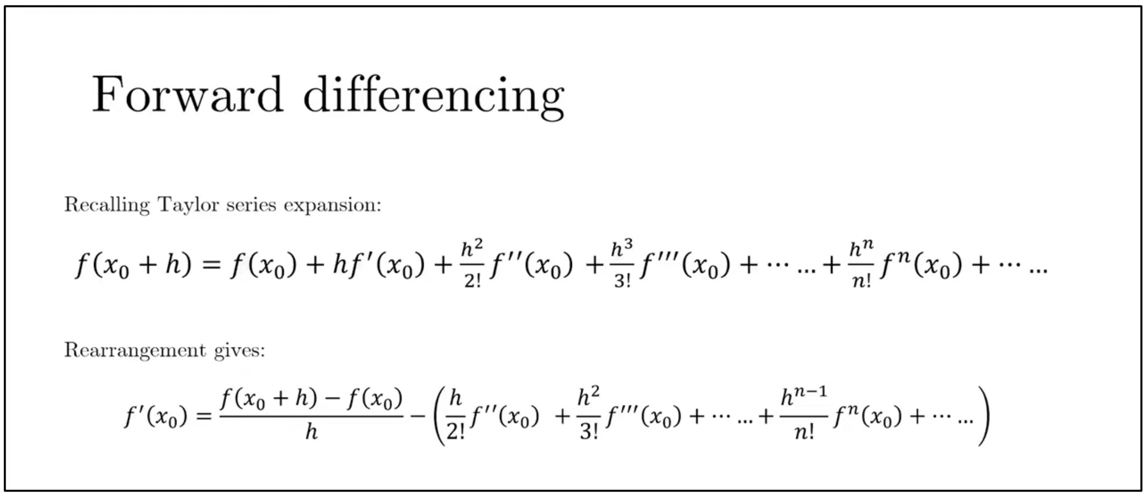
In next class...

- First derivative illustration
  - Forward differencing
  - Backward differencing
  - Central differencing
- Polynomial based example
  - Comparison with analytical solution



Lecture outline

- First derivative illustration
  - Forward differencing
  - Backward differencing
  - Central differencing
- Polynomial based example
  - Comparison with analytical solution



Forward differencing

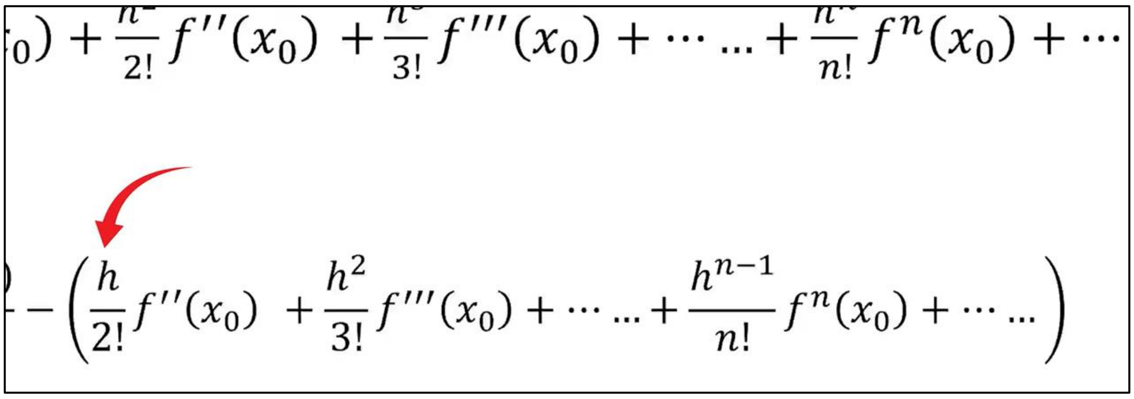
Recalling Taylor series expansion:

$$f(x_0+h)=f(x_0)+hf'(x_0)+\frac{h^2}{2!}f''(x_0)+\frac{h^3}{3!}f'''(x_0)+\cdots\ldots+\frac{h^n}{n!}f^n(x_0)+\cdots\ldots$$

Rearrangement gives:

$$f'(x_0)=\frac{f(x_0+h)-f(x_0)}{h}-\left(\frac{h}{2!}f''(x_0)\ +\frac{h^2}{3!}f'''(x_0)+\cdots\ldots+\frac{h^{n-1}}{n!}f^n(x_0)+\cdots\ldots\right)$$

The first term inside the bracket has a multiple of h:



$$_0)+\frac{h^2}{2!}f''(x_0)\ +\frac{h^3}{3!}f'''(x_0)+\cdots\ldots+\frac{h^n}{n!}f^n(x_0)+\cdots$$

$$-\left(\frac{h}{2!}f''(x_0)\ +\frac{h^2}{3!}f'''(x_0)+\cdots\ldots+\frac{h^{n-1}}{n!}f^n(x_0)+\cdots\ldots\right)$$

The second term has a multiple of h^2:

$$x_0) + \frac{h^2}{2!} f''(x_0) + \frac{h^3}{3!} f'''(x_0) + \cdots \ldots + \frac{h^n}{n!} f^n(x_0) + \cdots$$

$$) - \left(\frac{h}{2!} f''(x_0) + \frac{h^2}{3!} f'''(x_0) + \cdots \ldots + \frac{h^{n-1}}{n!} f^n(x_0) + \cdots \ldots \right)$$

Etc.

In the context of DFC, h is the grid spacing (distance between any two points).

Almost always, $h < 1 \Rightarrow h > h^2 > h^3 > \ldots > h^n$

Regarding the terms of the bracket, it is very likely that the largest term will be the first term itself:

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \left(\frac{h}{2!} f''(x_0) + \frac{h^2}{3!} f'''(x_0) + \cdots \ldots + \frac{h^{n-1}}{n!} f^n(x_0) + \cdots \ldots \right)$$

To write all the bracketed terms in a more compact way, they can be rewritten as:

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - O(h)$$

All the bracketed terms are of the order of h.

This final equation is the First Order Estimate using the Forward Differencing Method:

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - O(h)$$

Approximation Truncation error

Since the bracketed terms have gotten rid of, the series has been truncated. The leading term in the truncation error, or leading term in the truncated series of term is of the order of h, because the first term was multiplied by h.

Backward differencing

Recalling Taylor series expansion:

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2}{2!}f''(x_0) - \frac{h^3}{3!}f'''(x_0) + \cdots \ldots + (-1)^n \frac{h^n}{n!}f^n(x_0) + \cdots$$

Rearrangement gives:

$$f'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h} + \left(\frac{h}{2!}f''(x_0) - \frac{h^2}{3!}f'''(x_0) + \cdots \ldots + (-1)^n \frac{h^{n-1}}{n!}f^n(x_0) + \cdots \ldots\right)$$

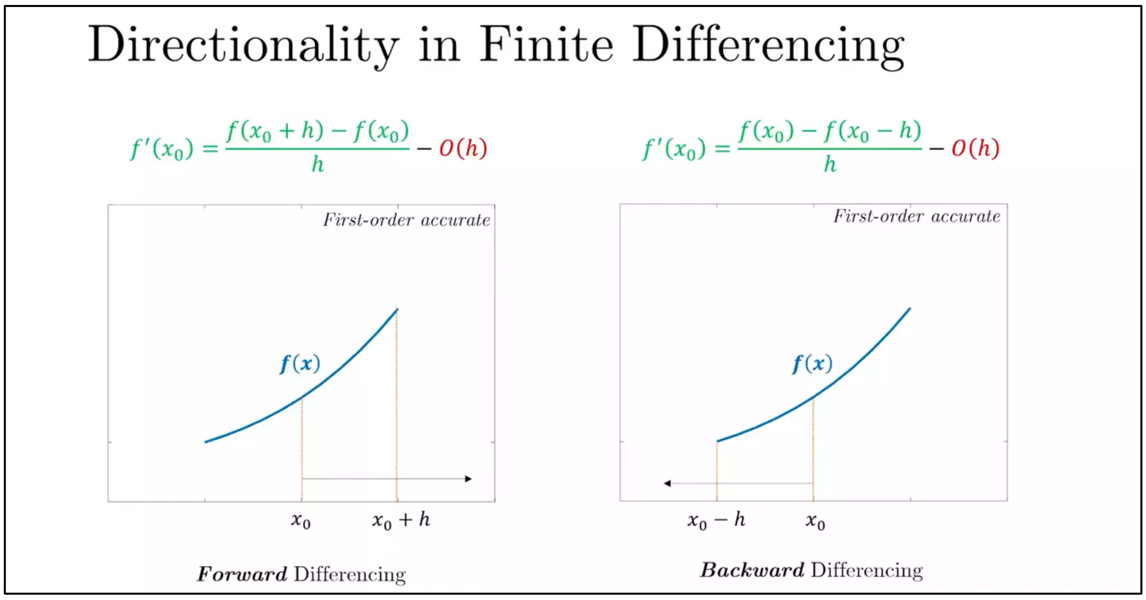
It can be studied in a similar way as the Forward Differencing Method.

First Order Estimate using the Backward Differencing Method:

$$f'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h} - O(h)$$

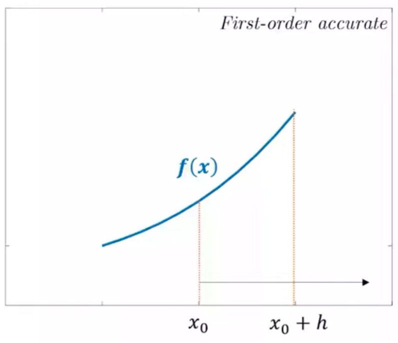
Approximation Truncation error

Where the truncation error is of the order of h.



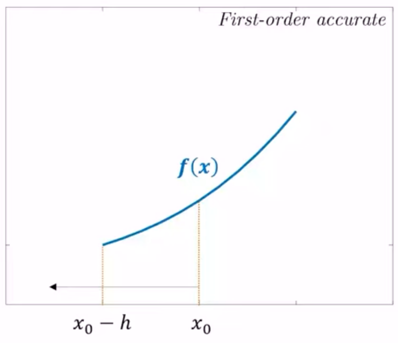
Directionality in Finite Differencing

$$f'(x_0) = \frac{f(x_0+h) - f(x_0)}{h} - O(h)$$



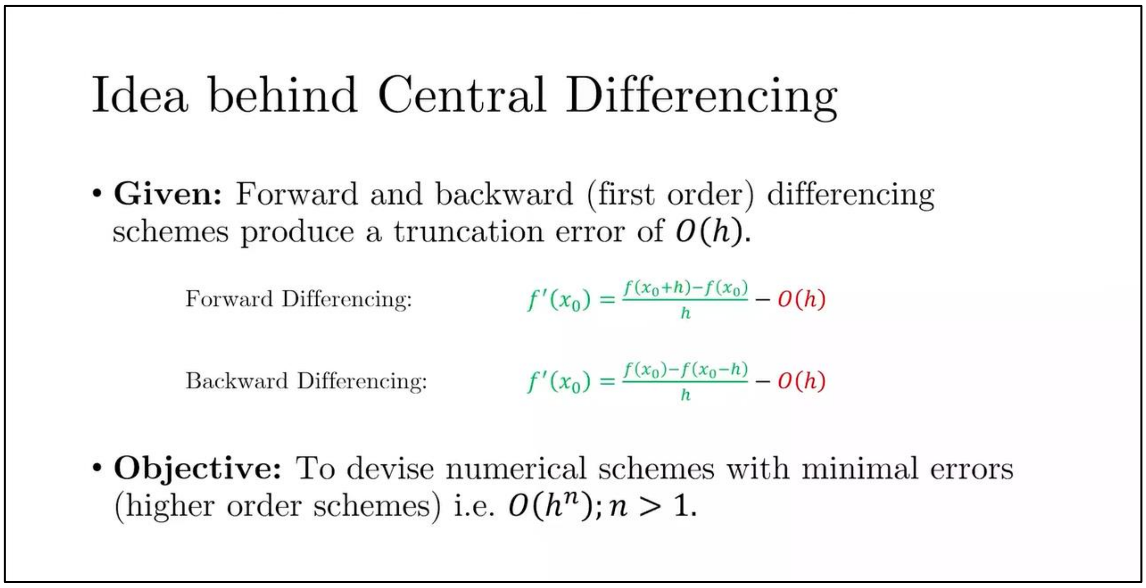
Forward Differencing

$$f'(x_0) = \frac{f(x_0) - f(x_0-h)}{h} - O(h)$$



Backward Differencing

Since the truncation errors are of the order of h (h^1), these schemes are called First Order Accurate.



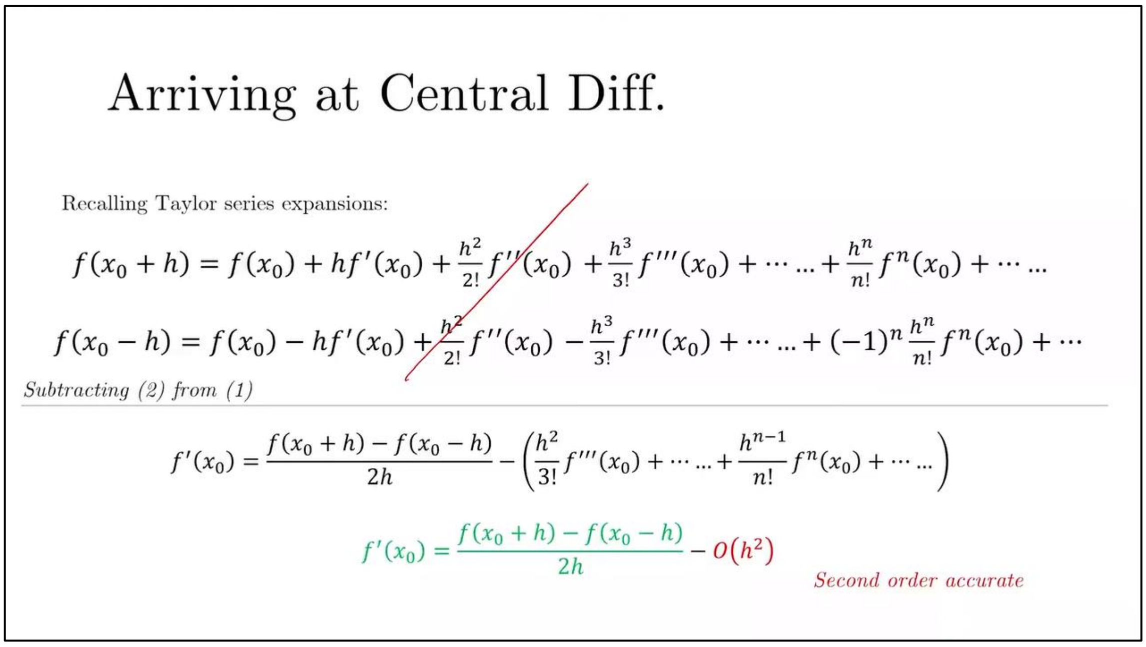
Idea behind Central Differencing

- **Given:** Forward and backward (first order) differencing schemes produce a truncation error of $O(h)$.

Forward Differencing: $f'(x_0) = \frac{f(x_0+h)-f(x_0)}{h} - O(h)$

Backward Differencing: $f'(x_0) = \frac{f(x_0)-f(x_0-h)}{h} - O(h)$

- **Objective:** To devise numerical schemes with minimal errors (higher order schemes) i.e. $O(h^n); n > 1$.



Arriving at Central Diff.

Recalling Taylor series expansions:

$$f(x_0+h) = f(x_0) + hf'(x_0) + \cancel{\frac{h^2}{2!}f''(x_0)} + \frac{h^3}{3!}f'''(x_0) + \cdots \ldots + \frac{h^n}{n!}f^n(x_0) + \cdots \ldots$$

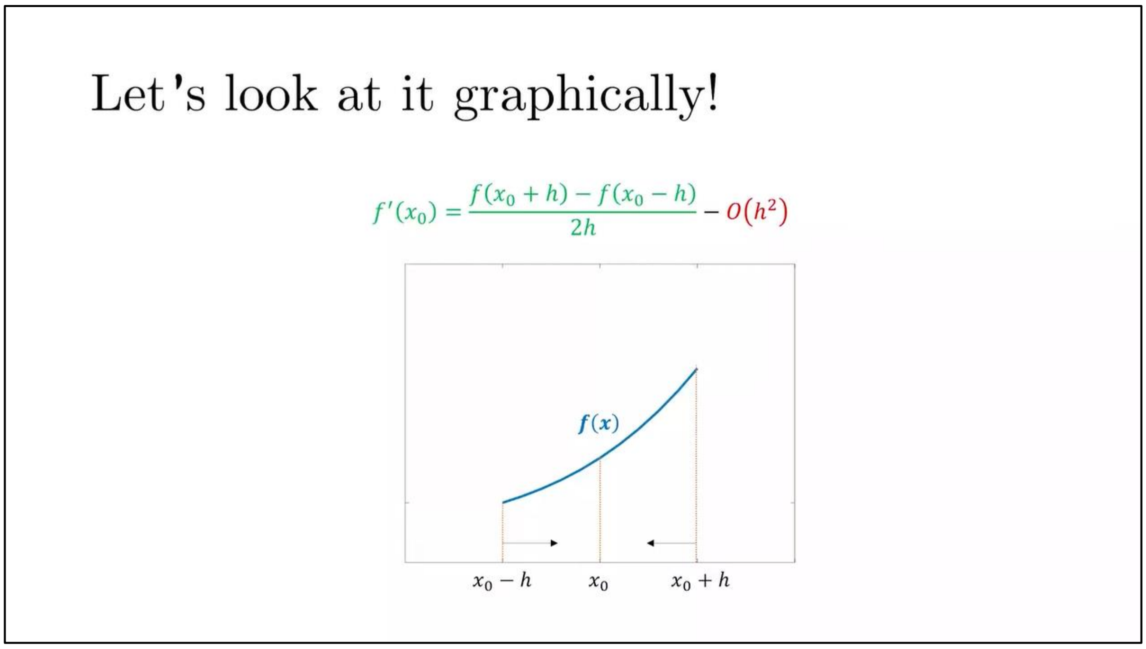
$$f(x_0-h) = f(x_0) - hf'(x_0) + \cancel{\frac{h^2}{2!}f''(x_0)} - \frac{h^3}{3!}f'''(x_0) + \cdots \ldots + (-1)^n\frac{h^n}{n!}f^n(x_0) + \cdots$$

Subtracting (2) from (1)

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h} - \left(\frac{h^2}{3!}f'''(x_0) + \cdots \ldots + \frac{h^{n-1}}{n!}f^n(x_0) + \cdots \ldots\right)$$

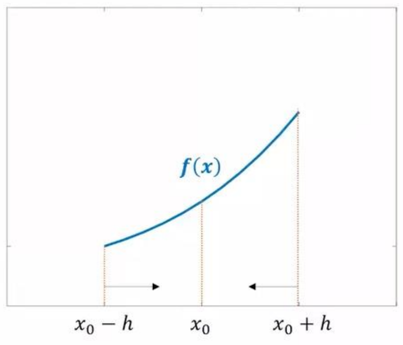
$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h} - O(h^2)$$

Second order accurate



Let's look at it graphically!

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h} - O(h^2)$$



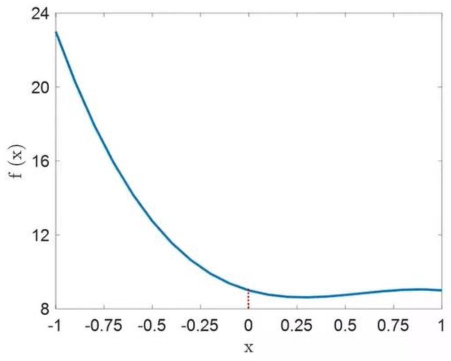
Scheme	Expression	Truncation Error
Forward	$\frac{f(x_0+h)-f(x_0)}{h}$	$O(h)$
Backward	$\frac{f(x_0)-f(x_0-h)}{h}$	$O(h)$
Central	$\frac{f(x_0+h)-f(x_0-h)}{2h}$	$O(h^2)$

Polynomial derivative estimation

$f(x) = -4x^3 + 7x^2 - 3x + 9$

$f'(0) = ?$

Use h = 0.25



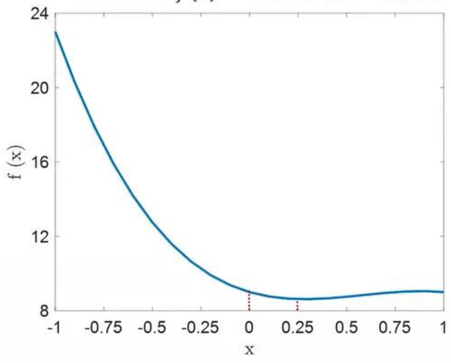
Forward difference estimate

$f(x_0 = 0) = 9$

$f(x_0 + h = 0.25) = 8.625$

$f'(0) = \frac{f(x_0+h) - f(x_0)}{h} = -1.25$

$f(x) = -4x^3 + 7x^2 - 3x + 9$

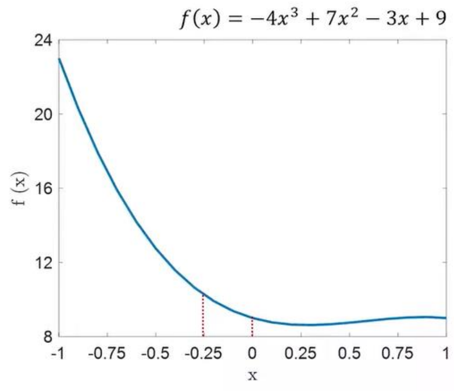


Backward difference estimate

$f(x_0 = 0) = 9$

$f(x_0 - h = -0.25) = 10.25$

$$f'(0) = \frac{f(x_0) - f(x_0 - h)}{h} = -5$$



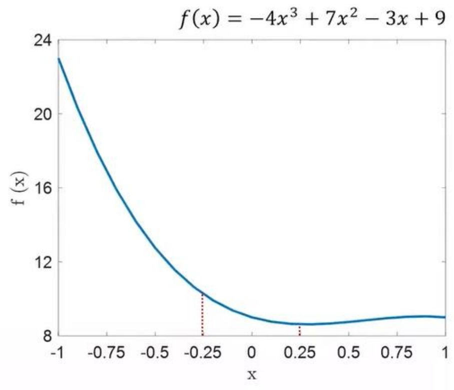
Central difference estimate

$f(x_0 = 0) = 9$

$f(x_0 + h = 0.25) = 8.625$

$f(x_0 - h = -0.25) = 10.25$

$$f'(0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} = -3.25$$



Comparison

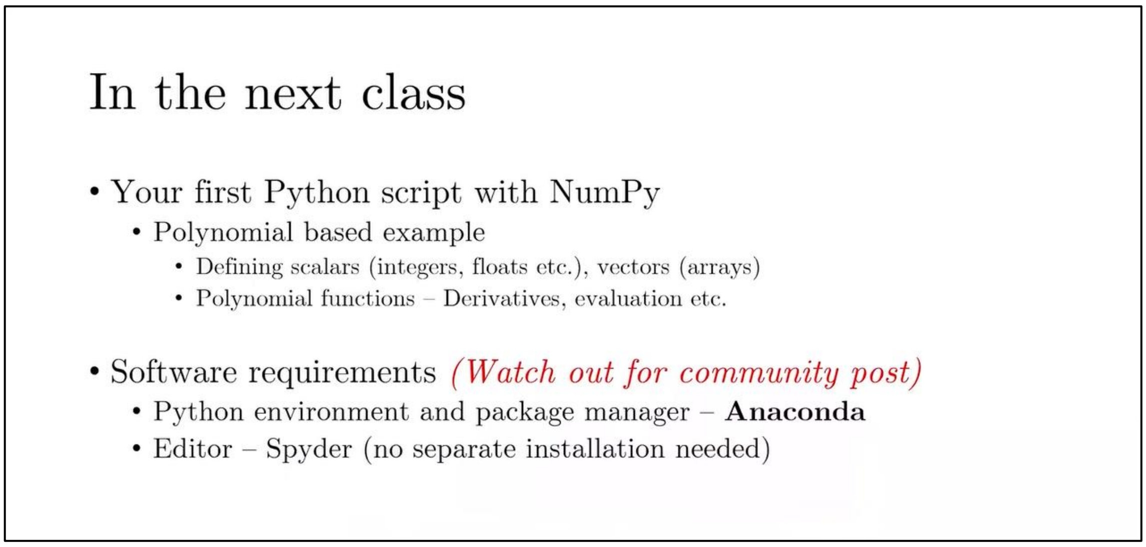
$f(x) = -4x^3 + 7x^2 - 3x + 9$

$f'(x) = -12x^2 + 14x - 3$

$f'(0) = -3$

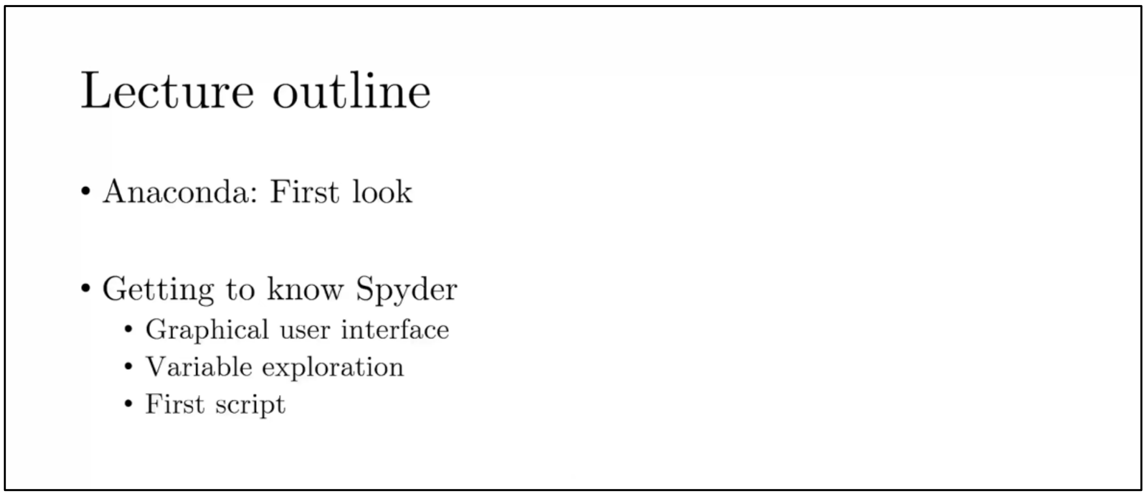
Scheme	Estimation	Absolute error
Forward	-1.25	1.75
Backward	-5	2
Central	-3.25	0.25

The truncation error of the Central Differencing Scheme is much lesser, one order more accurate.



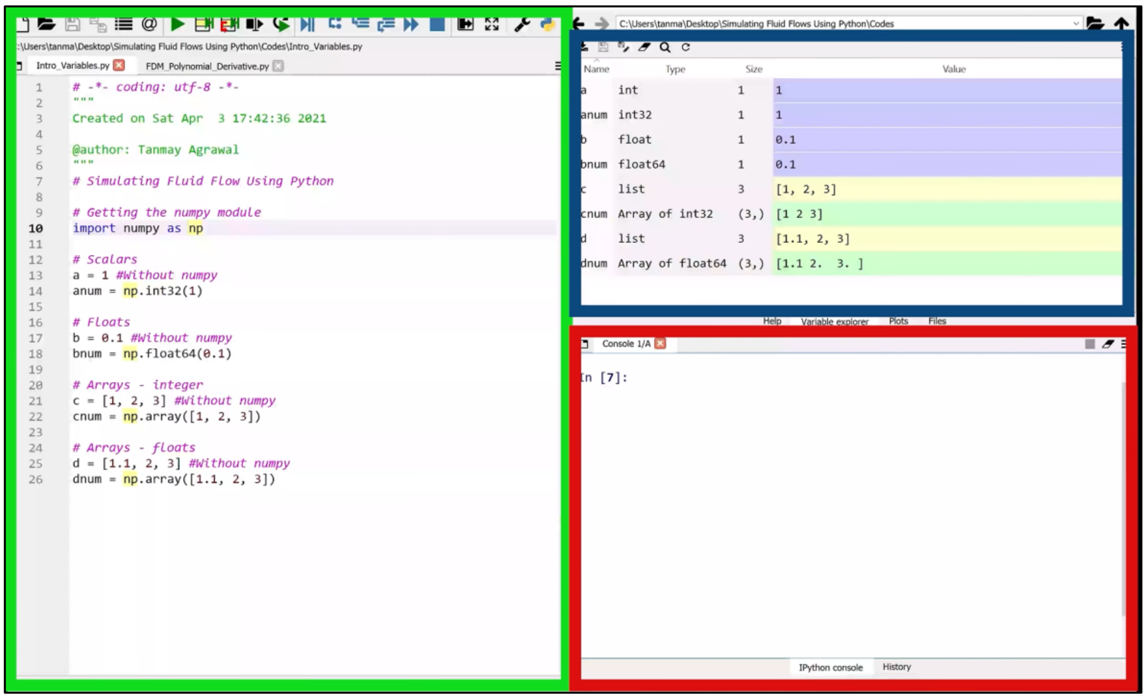
In the next class

- Your first Python script with NumPy
  - Polynomial based example
    - Defining scalars (integers, floats etc.), vectors (arrays)
    - Polynomial functions – Derivatives, evaluation etc.
- Software requirements *(Watch out for community post)*
  - Python environment and package manager – **Anaconda**
  - Editor – Spyder (no separate installation needed)



Lecture outline

- Anaconda: First look
- Getting to know Spyder
  - Graphical user interface
  - Variable exploration
  - First script



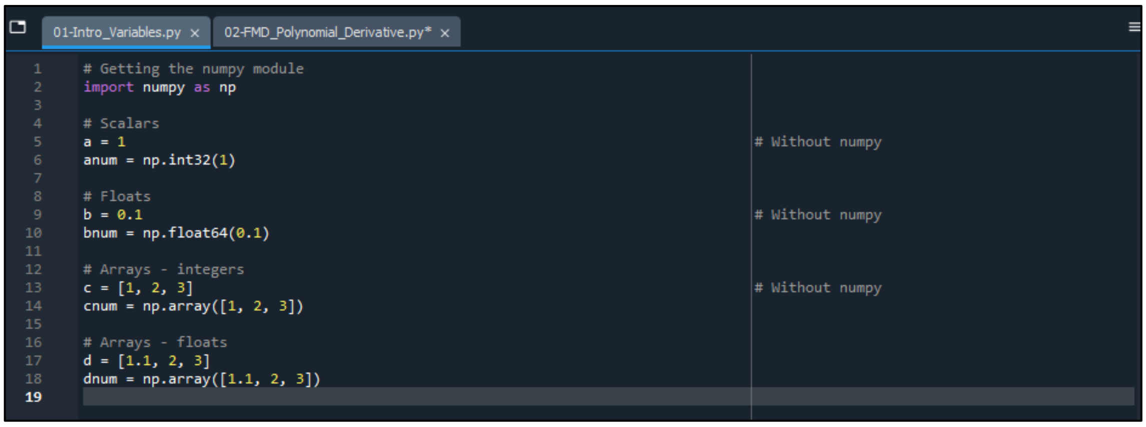
:\Users\tanma\Desktop\Simulating Fluid Flows Using Python\Codes\Intro_Variables.py

```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr  3 17:42:36 2021

@author: Tanmay Agrawal
"""
# Simulating Fluid Flow Using Python

# Getting the numpy module
import numpy as np

# Scalars
a = 1 #Without numpy
anum = np.int32(1)

# Floats
b = 0.1 #Without numpy
bnum = np.float64(0.1)

# Arrays - integer
c = [1, 2, 3] #Without numpy
cnum = np.array([1, 2, 3])

# Arrays - floats
d = [1.1, 2, 3] #Without numpy
dnum = np.array([1.1, 2, 3])
```

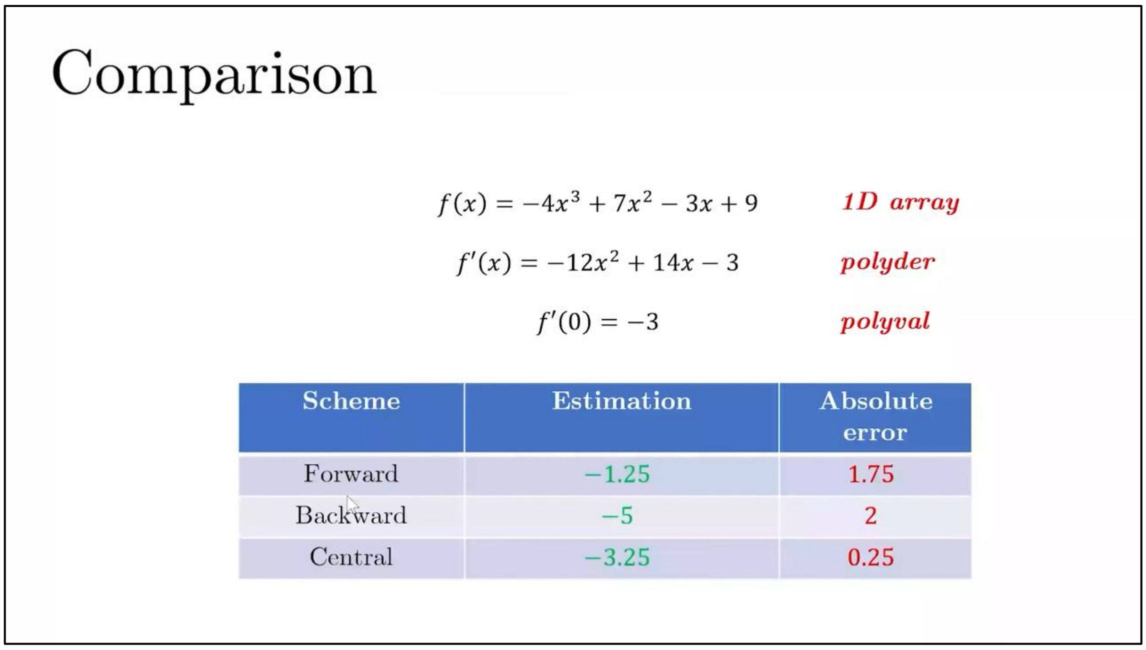
C:\Users\tanma\Desktop\Simulating Fluid Flows Using Python\Codes

Name	Type	Size	Value
a	int	1	1
anum	int32	1	1
b	float	1	0.1
bnum	float64	1	0.1
c	list	3	[1, 2, 3]
cnum	Array of int32	(3,)	[1 2 3]
d	list	3	[1.1, 2, 3]
dnum	Array of float64	(3,)	[1.1 2. 3.]

```
In [7]:
```



```
# Getting the numpy module
import numpy as np

# Scalars
a = 1                                                    # Without numpy
anum = np.int32(1)

# Floats
b = 0.1                                                  # Without numpy
bnum = np.float64(0.1)

# Arrays - integers
c = [1, 2, 3]                                            # Without numpy
cnum = np.array([1, 2, 3])

# Arrays - floats
d = [1.1, 2, 3]
dnum = np.array([1.1, 2, 3])
```



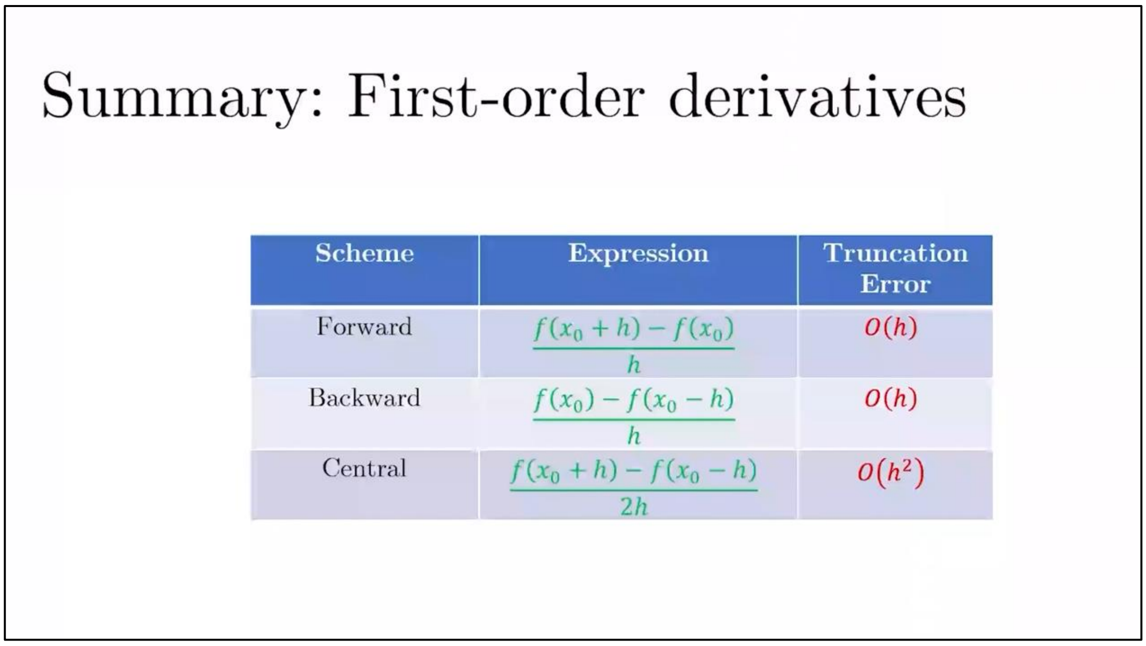
Comparison

$f(x) = -4x^3 + 7x^2 - 3x + 9$ *1D array*

$f'(x) = -12x^2 + 14x - 3$ *polyder*

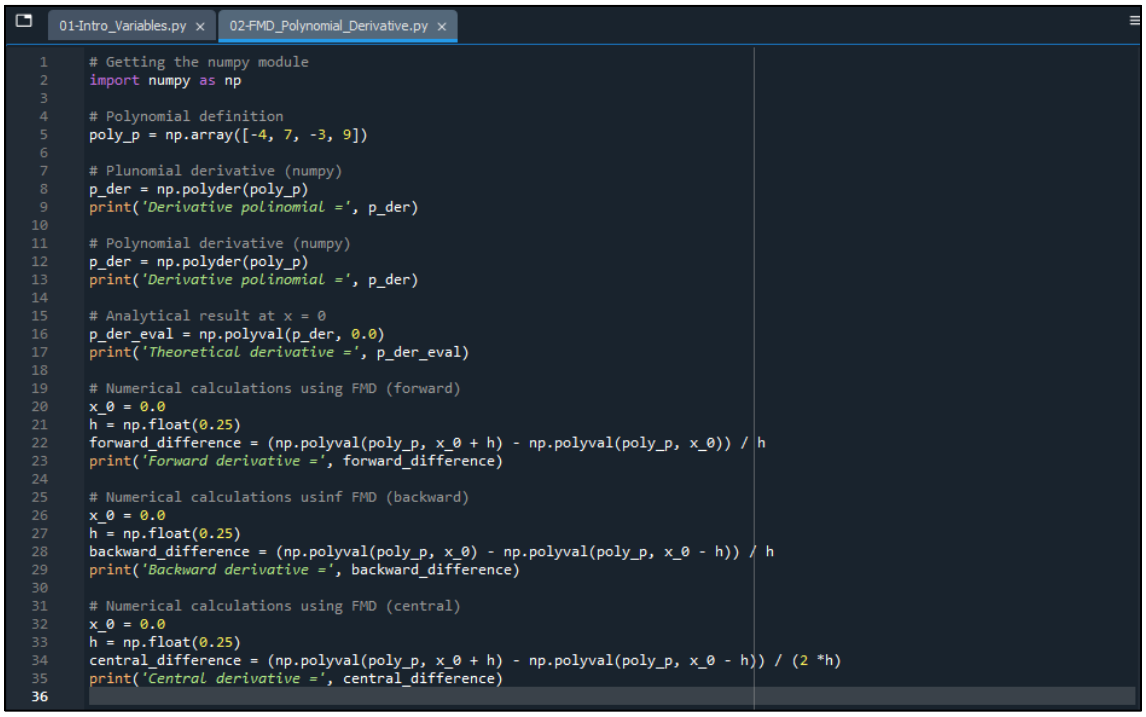
$f'(0) = -3$ *polyval*

Scheme	Estimation	Absolute error
Forward	-1.25	1.75
Backward	-5	2
Central	-3.25	0.25



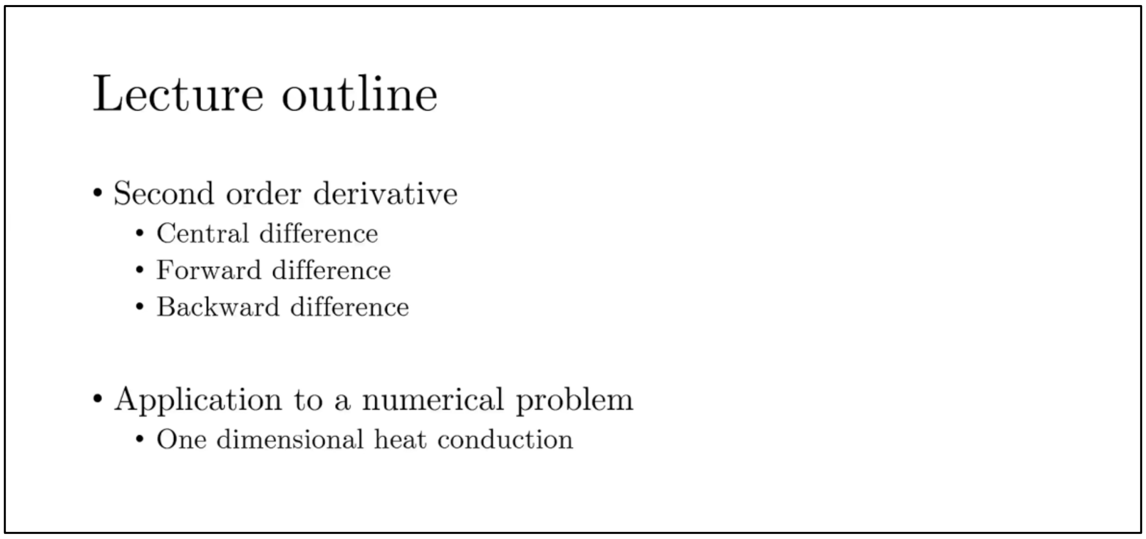
Summary: First-order derivatives

Scheme	Expression	Truncation Error
Forward	$\frac{f(x_0+h)-f(x_0)}{h}$	$O(h)$
Backward	$\frac{f(x_0)-f(x_0-h)}{h}$	$O(h)$
Central	$\frac{f(x_0+h)-f(x_0-h)}{2h}$	$O(h^2)$



```python
# Getting the numpy module
import numpy as np

# Polynomial definition
poly_p = np.array([-4, 7, -3, 9])

# Plunomial derivative (numpy)
p_der = np.polyder(poly_p)
print('Derivative polinomial =', p_der)

# Polynomial derivative (numpy)
p_der = np.polyder(poly_p)
print('Derivative polinomial =', p_der)

# Analytical result at x = 0
p_der_eval = np.polyval(p_der, 0.0)
print('Theoretical derivative =', p_der_eval)

# Numerical calculations using FMD (forward)
x_0 = 0.0
h = np.float(0.25)
forward_difference = (np.polyval(poly_p, x_0 + h) - np.polyval(poly_p, x_0)) / h
print('Forward derivative =', forward_difference)

# Numerical calculations usinf FMD (backward)
x_0 = 0.0
h = np.float(0.25)
backward_difference = (np.polyval(poly_p, x_0) - np.polyval(poly_p, x_0 - h)) / h
print('Backward derivative =', backward_difference)

# Numerical calculations using FMD (central)
x_0 = 0.0
h = np.float(0.25)
central_difference = (np.polyval(poly_p, x_0 + h) - np.polyval(poly_p, x_0 - h)) / (2 *h)
print('Central derivative =', central_difference)
```

LECTURE 04. SECOND DERIVATIVE. 1D HEAT CONDUCTION



Lecture outline

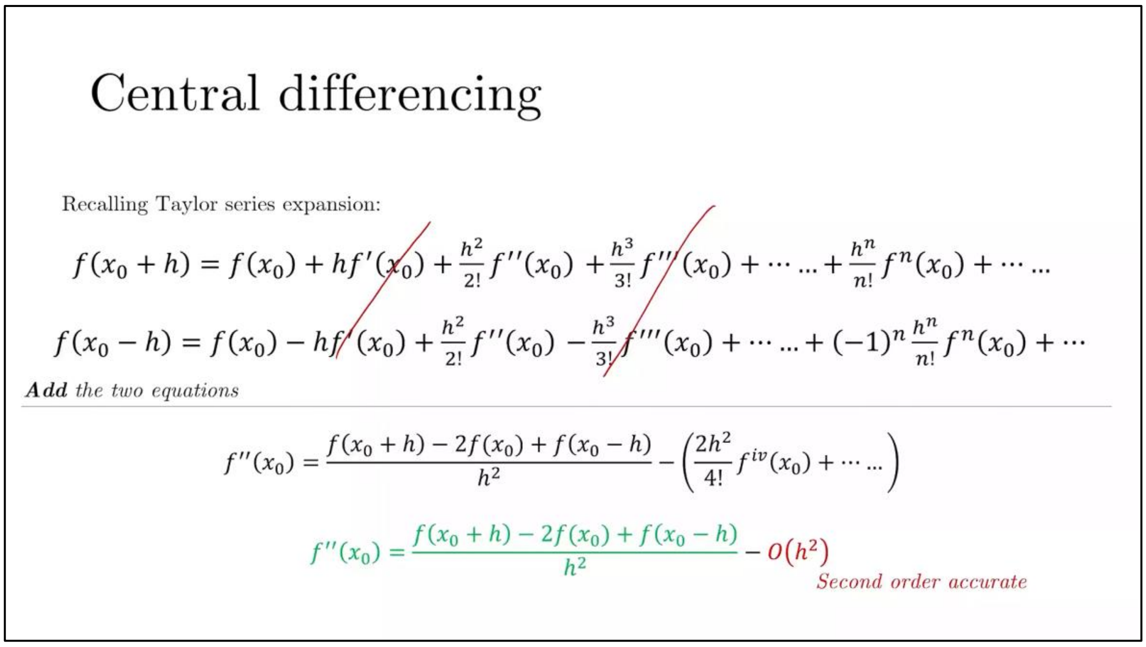
- Second order derivative
  - Central difference
  - Forward difference
  - Backward difference
- Application to a numerical problem
  - One dimensional heat conduction

Momentum equations, viscosity term:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \vartheta\boxed{\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)} + F_x$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \vartheta\boxed{\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)} + F_y$$

Energy equation, diffusion of temperature or any scalar: temperature, concentration, etc.



Central differencing

Recalling Taylor series expansion:

$$f(x_0+h) = f(x_0) + \cancel{hf'(x_0)} + \frac{h^2}{2!}f''(x_0) + \cancel{\frac{h^3}{3!}f'''(x_0)} + \cdots \ldots + \frac{h^n}{n!}f^n(x_0) + \cdots \ldots$$

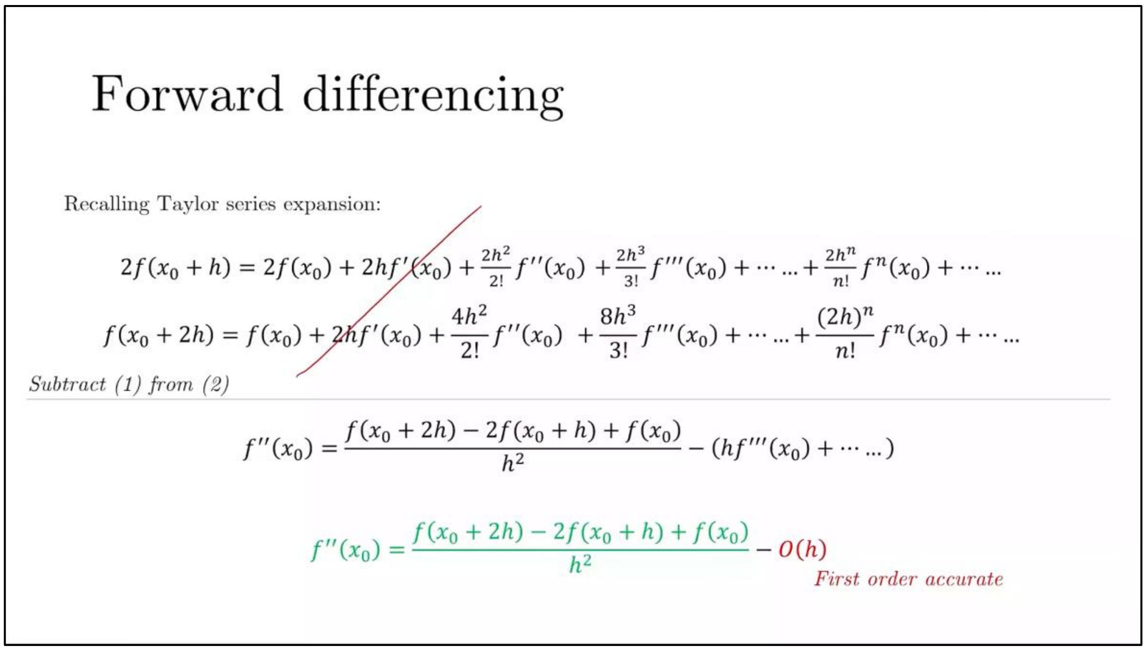
$$f(x_0-h) = f(x_0) - \cancel{hf'(x_0)} + \frac{h^2}{2!}f''(x_0) - \cancel{\frac{h^3}{3!}f'''(x_0)} + \cdots \ldots + (-1)^n\frac{h^n}{n!}f^n(x_0) + \cdots$$

***Add** the two equations*

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} - \left(\frac{2h^2}{4!}f^{iv}(x_0) + \cdots \ldots\right)$$

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} - O(h^2)$$

Second order accurate



Forward differencing

Recalling Taylor series expansion:

$$2f(x_0+h) = 2f(x_0) + \cancel{2hf'(x_0)} + \frac{2h^2}{2!}f''(x_0) + \frac{2h^3}{3!}f'''(x_0) + \cdots \ldots + \frac{2h^n}{n!}f^n(x_0) + \cdots \ldots$$

$$f(x_0+2h) = f(x_0) + \cancel{2hf'(x_0)} + \frac{4h^2}{2!}f''(x_0) + \frac{8h^3}{3!}f'''(x_0) + \cdots \ldots + \frac{(2h)^n}{n!}f^n(x_0) + \cdots \ldots$$

Subtract (1) from (2)

$$f''(x_0) = \frac{f(x_0+2h) - 2f(x_0+h) + f(x_0)}{h^2} - (hf'''(x_0) + \cdots \ldots)$$

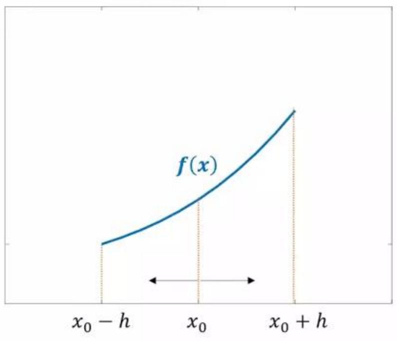
$$f''(x_0) = \frac{f(x_0+2h) - 2f(x_0+h) + f(x_0)}{h^2} - O(h)$$

First order accurate

Let's look at it graphically!

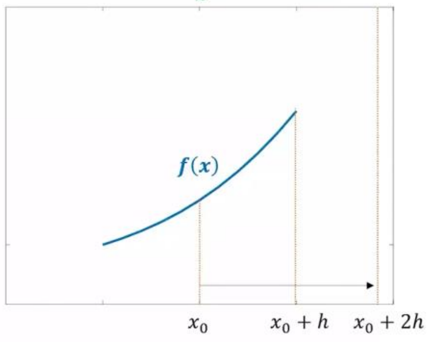
Central difference

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} - O(h^2)$$



Forward difference

$$f''(x_0) = \frac{f(x_0+2h) - 2f(x_0+h) + f(x_0)}{h^2} - O(h)$$

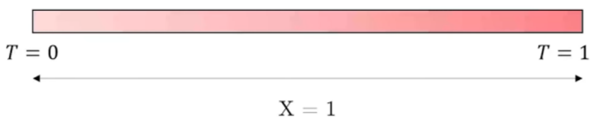


Try the backward difference!

Hint: Use the expansions for $x_0 - h$ and $x_0 - 2h$

$$f''(x_0) = \frac{f(x_0) - 2f(x_0-h) + f(x_0-2h)}{h^2} - O(h)$$

Why bother with all this?

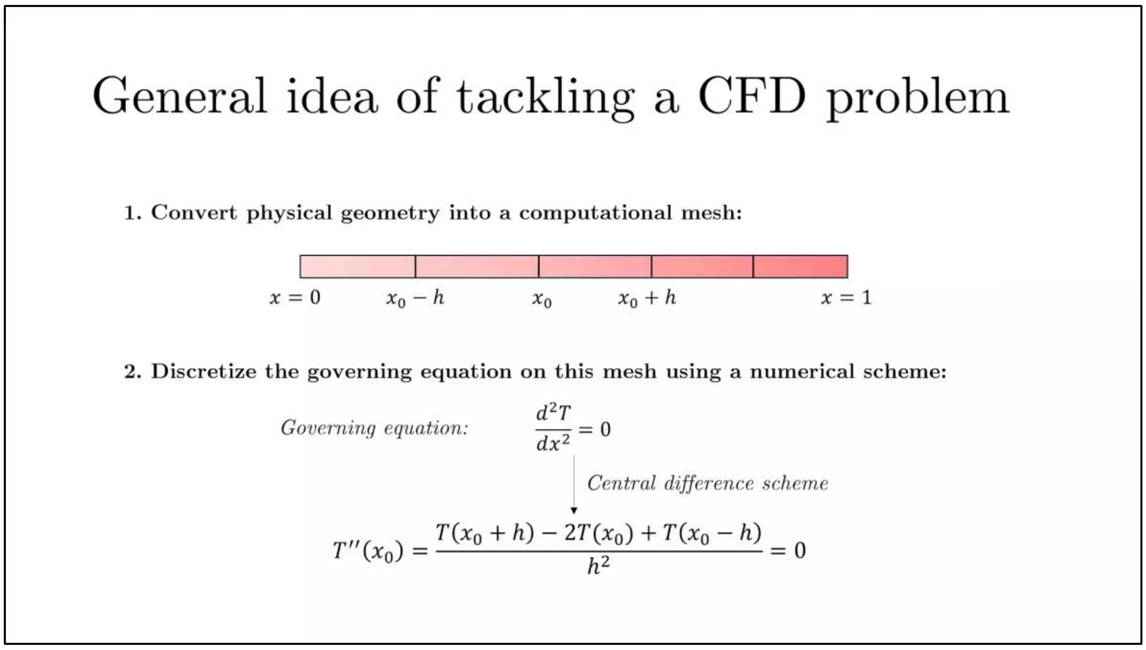


Given: The physics of the situation is governed by the differential equation of pure diffusion:

$$\frac{d^2T}{dx^2} = 0$$

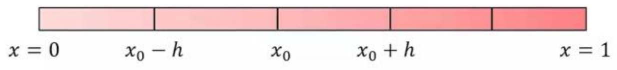
Initially: $T(t = 0, x < 1) = 0$ and $T(t = 0, x = 1) = 1$ (Initial conditions)
At any other instant: $T(t = t, x = 0) = 0$ and $T(t = t, x = 1) = 1$ (Boundary conditions)

Objective: To calculate the temperature distribution, $T(x)$, at equilibrium.



General idea of tackling a CFD problem

1. Convert physical geometry into a computational mesh:

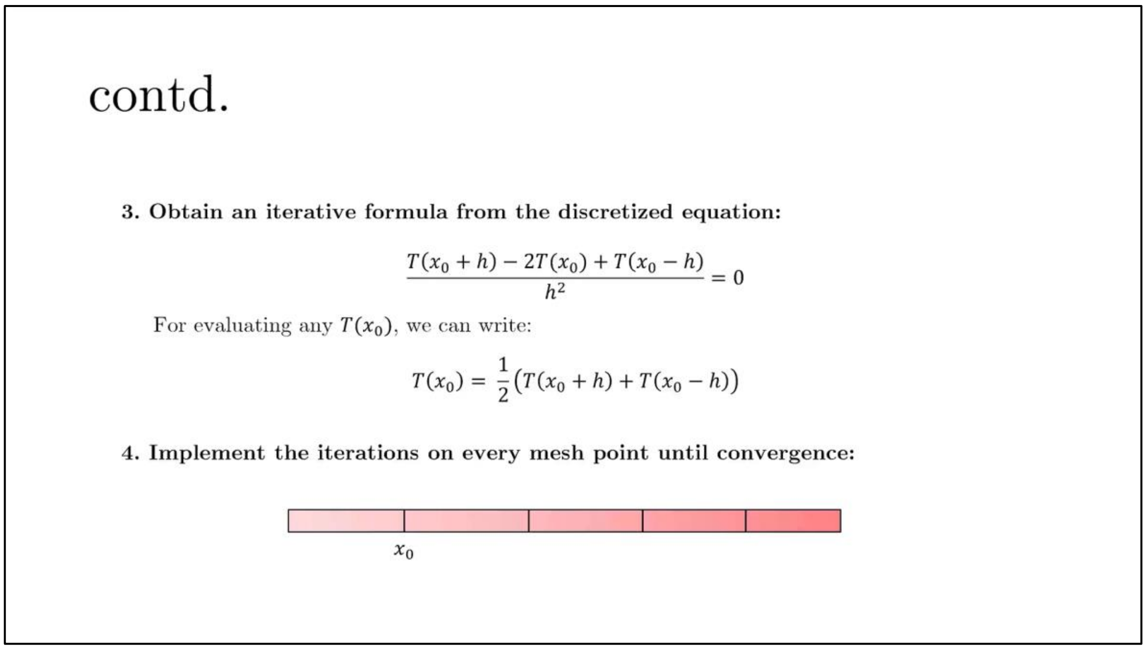


2. Discretize the governing equation on this mesh using a numerical scheme:

Governing equation: $$\frac{d^2T}{dx^2} = 0$$

$$\Big\downarrow \textit{Central difference scheme}$$

$$T''(x_0) = \frac{T(x_0 + h) - 2T(x_0) + T(x_0 - h)}{h^2} = 0$$



contd.

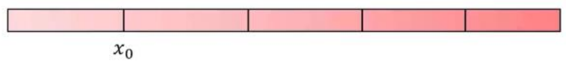
3. Obtain an iterative formula from the discretized equation:

$$\frac{T(x_0 + h) - 2T(x_0) + T(x_0 - h)}{h^2} = 0$$

For evaluating any $T(x_0)$, we can write:

$$T(x_0) = \frac{1}{2}\big(T(x_0 + h) + T(x_0 - h)\big)$$

4. Implement the iterations on every mesh point until convergence:



Definition of convergence criterion?

- There are many ways in which you can call any solution a *converged* solution.

$$\sum_{x=x_0} |T_{new} - T_{old}| < \epsilon$$

$$\sum_{x} |T_{new} - T_{old}| < \epsilon$$

$$\sum_{x} \left|\frac{T_{new} - T_{old}}{T_{old}}\right| < \epsilon$$

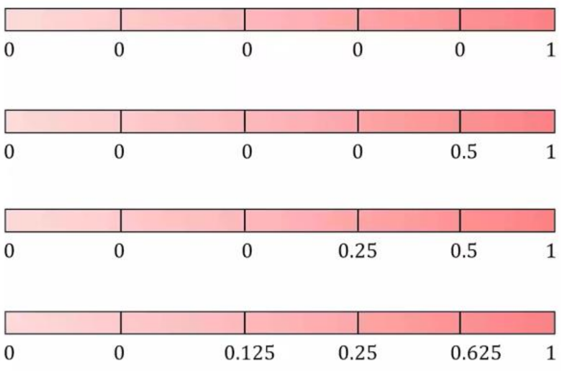
Hand calculations

Initial time:

1st timestep:

2nd timestep:

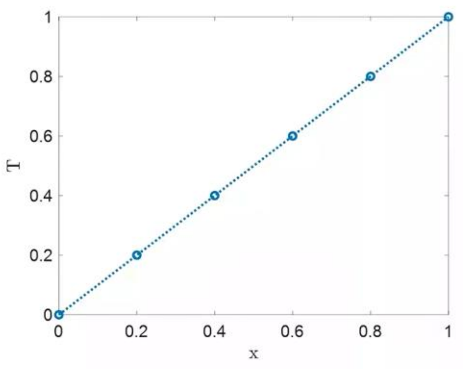
3rd timestep:



$$T(x_0) = \frac{1}{2}\big(T(x_0 + h) + T(x_0 - h)\big)$$

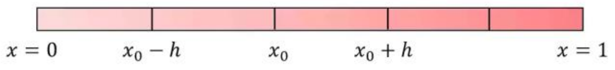
Too tiring...

after around 73 time steps:

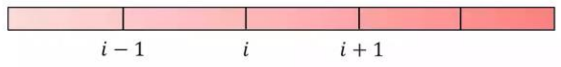


$$\begin{tikzcd}
\frac{d^2T}{dx^2} = 0 \arrow[d, "\text{Integrating twice}"] \\
T = Ax + B \arrow[d, "\begin{matrix} T(0) = 0 \\ T(1) = 1 \end{matrix}"] \\
T = x
\end{tikzcd}$$

Conventional notation



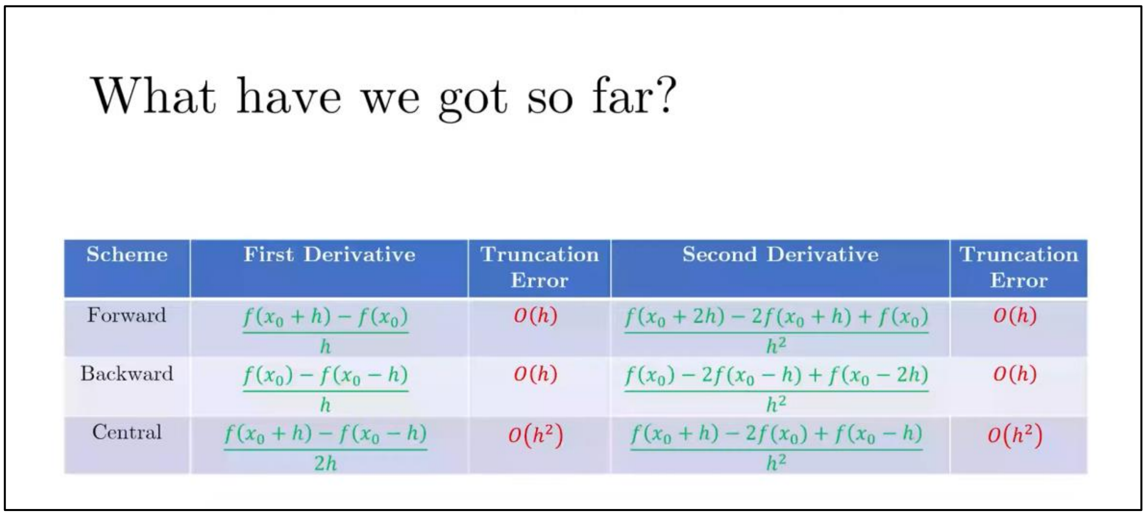
$$T(x_0) = \frac{1}{2}\big(T(x_0 + h) + T(x_0 - h)\big)$$



$$T_i = \frac{1}{2}(T_{i+1} + T_{i-1})$$

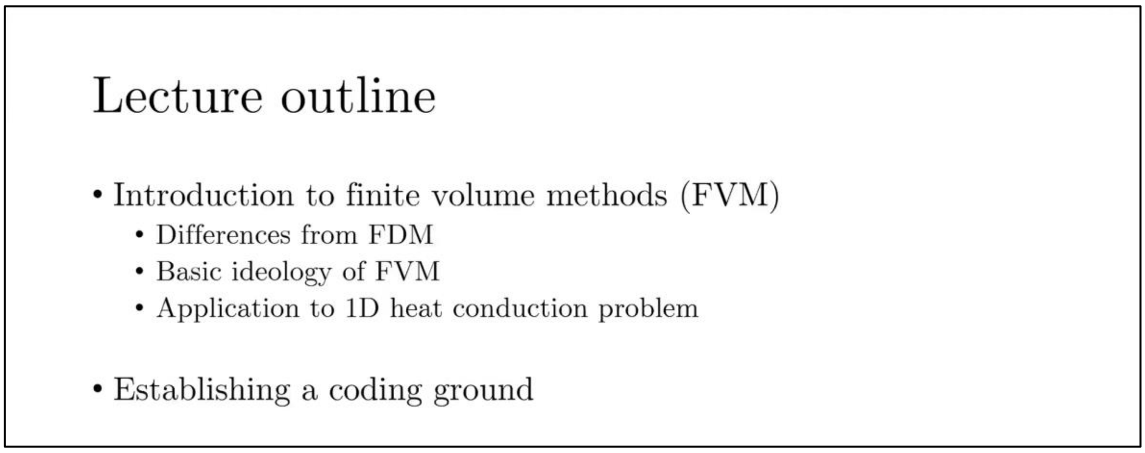
Discretization exercise:

1D convection – diffusion equation $\quad U\frac{dT}{dx} + \frac{d^2T}{dx^2} = 0$



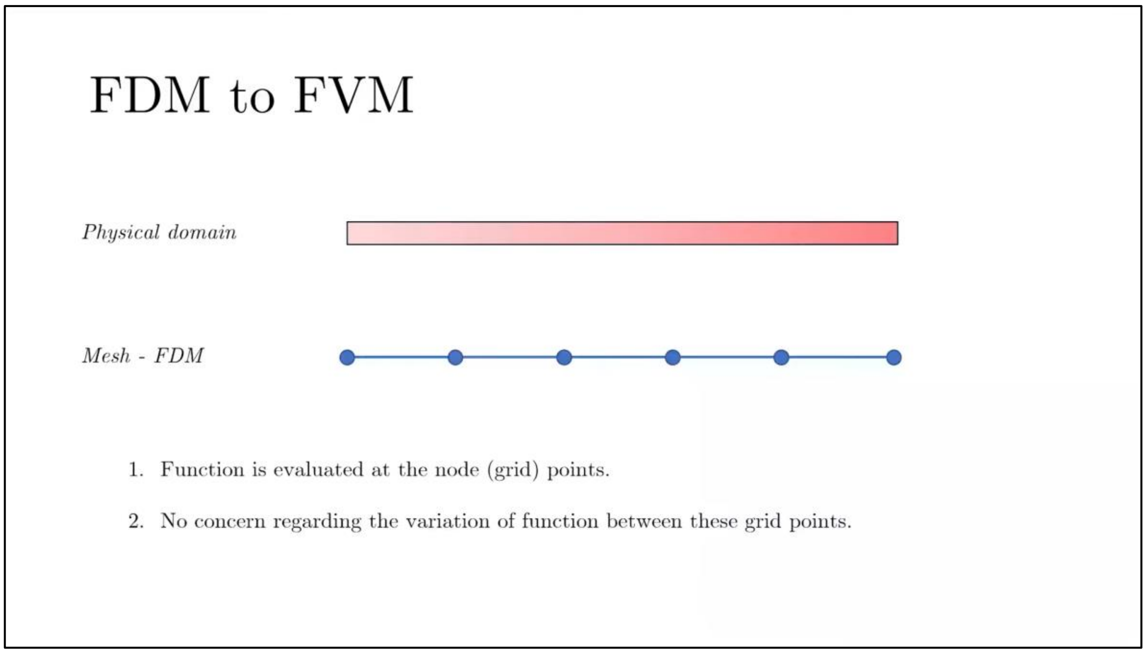
What have we got so far?

Scheme	First Derivative	Truncation Error	Second Derivative	Truncation Error
Forward	$\frac{f(x_0+h)-f(x_0)}{h}$	$O(h)$	$\frac{f(x_0+2h)-2f(x_0+h)+f(x_0)}{h^2}$	$O(h)$
Backward	$\frac{f(x_0)-f(x_0-h)}{h}$	$O(h)$	$\frac{f(x_0)-2f(x_0-h)+f(x_0-2h)}{h^2}$	$O(h)$
Central	$\frac{f(x_0+h)-f(x_0-h)}{2h}$	$O(h^2)$	$\frac{f(x_0+h)-2f(x_0)+f(x_0-h)}{h^2}$	$O(h^2)$



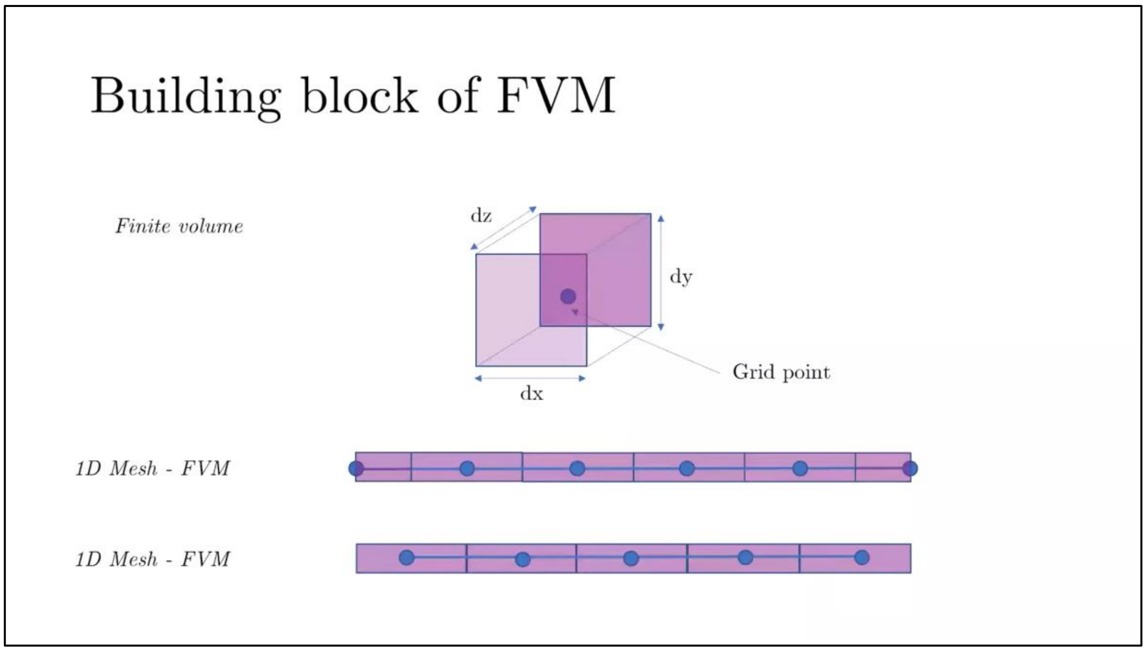
Lecture outline

- Introduction to finite volume methods (FVM)
  - Differences from FDM
  - Basic ideology of FVM
  - Application to 1D heat conduction problem
- Establishing a coding ground



FDM to FVM

1. Function is evaluated at the node (grid) points.
2. No concern regarding the variation of function between these grid points.

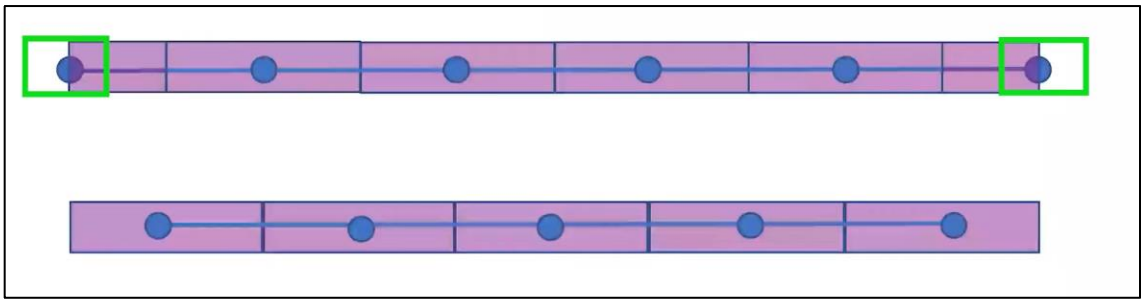


Building block of FVM

Finite volume

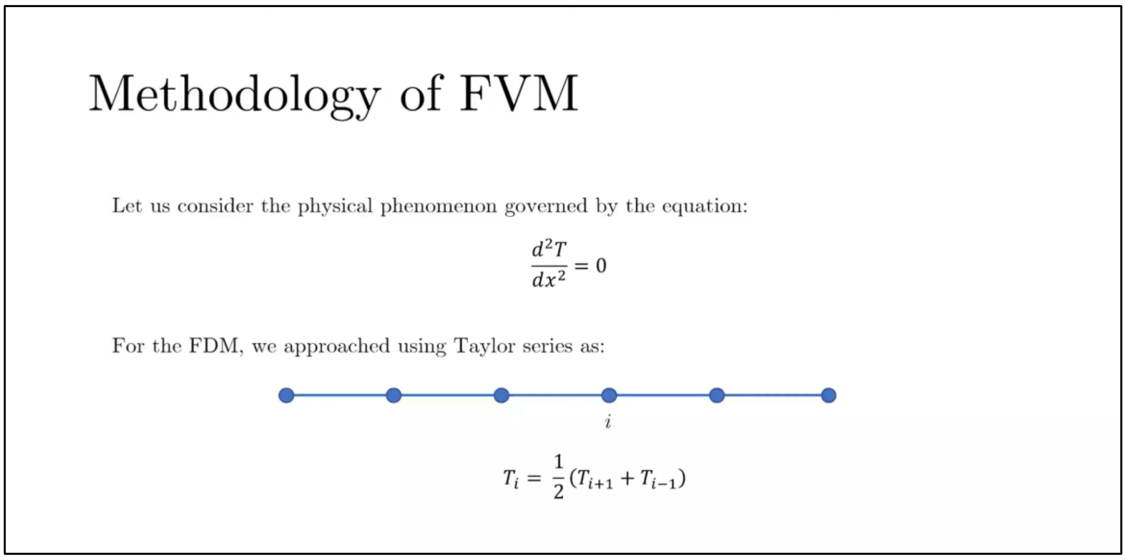
1D Mesh - FVM

1D Mesh - FVM



The top grid has 6 grid points and the volumes defining the boundary conditions are reduced. The bottom grid has 5 grid points and doesn't points in the boundary. The total volume is the same in both meshes.

Using the 1D diffusion as governing equation:



Methodology of FVM

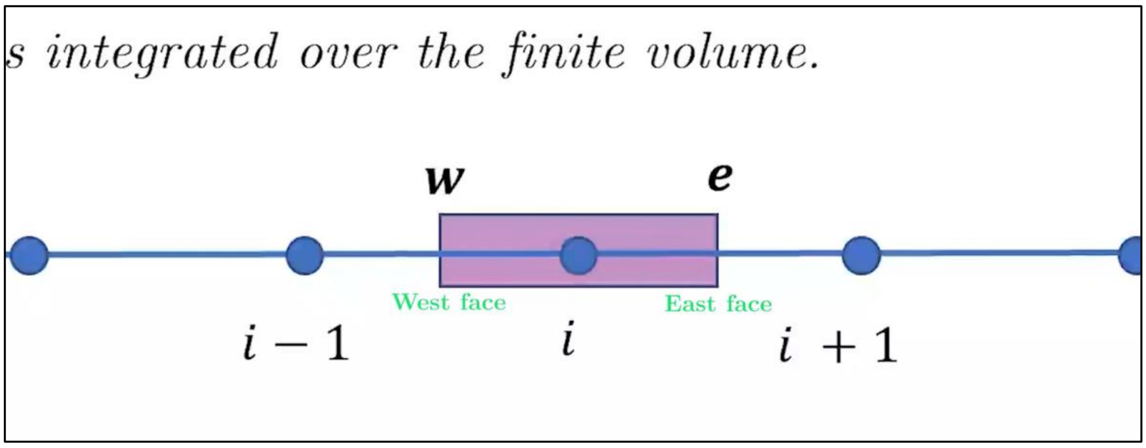
Let us consider the physical phenomenon governed by the equation:

$$\frac{d^2T}{dx^2} = 0$$

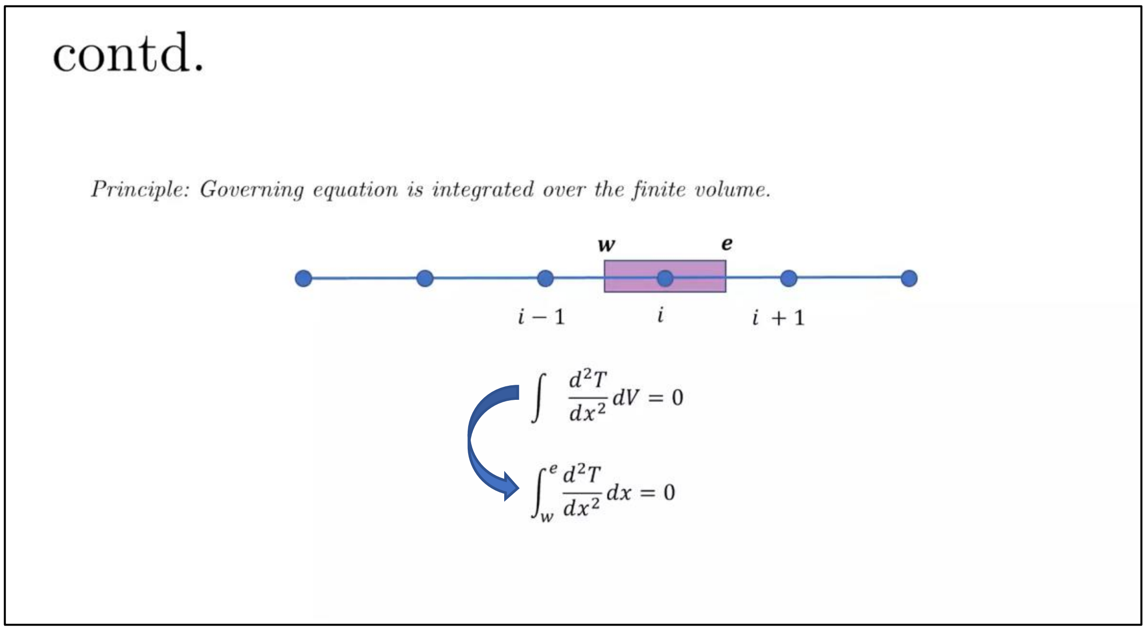
For the FDM, we approached using Taylor series as:

$$T_i = \frac{1}{2}(T_{i+1} + T_{i-1})$$

The faces where the finite volume intercepts the grid: east and west faces.



s integrated over the finite volume.

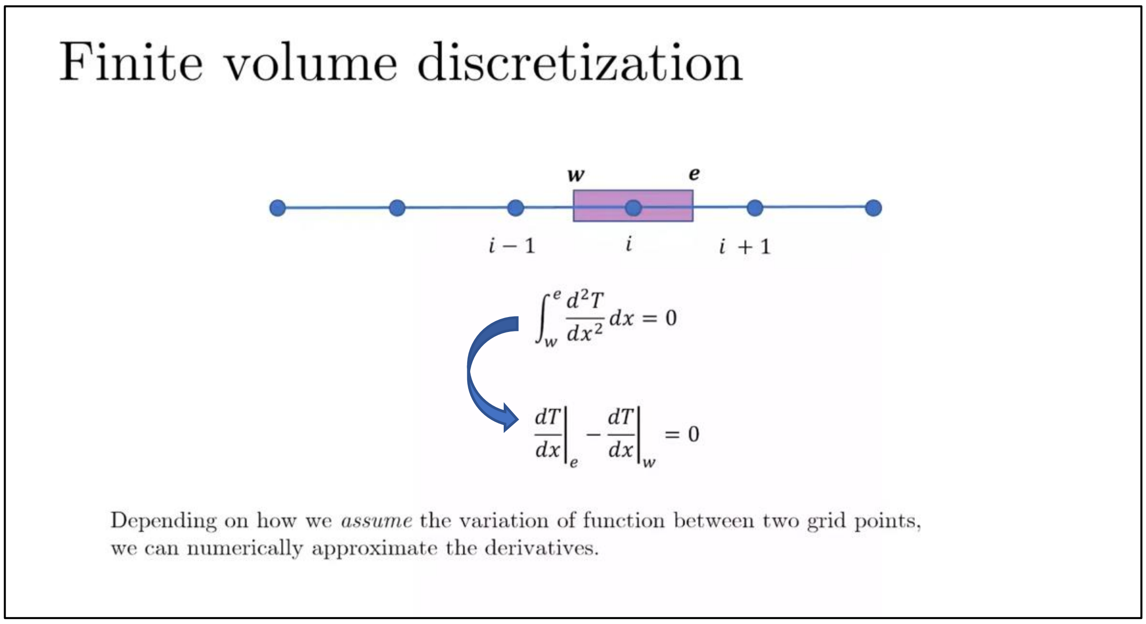


contd.

Principle: Governing equation is integrated over the finite volume.

$$\int \frac{d^2T}{dx^2}\,dV = 0$$

$$\int_w^e \frac{d^2T}{dx^2}\,dx = 0$$



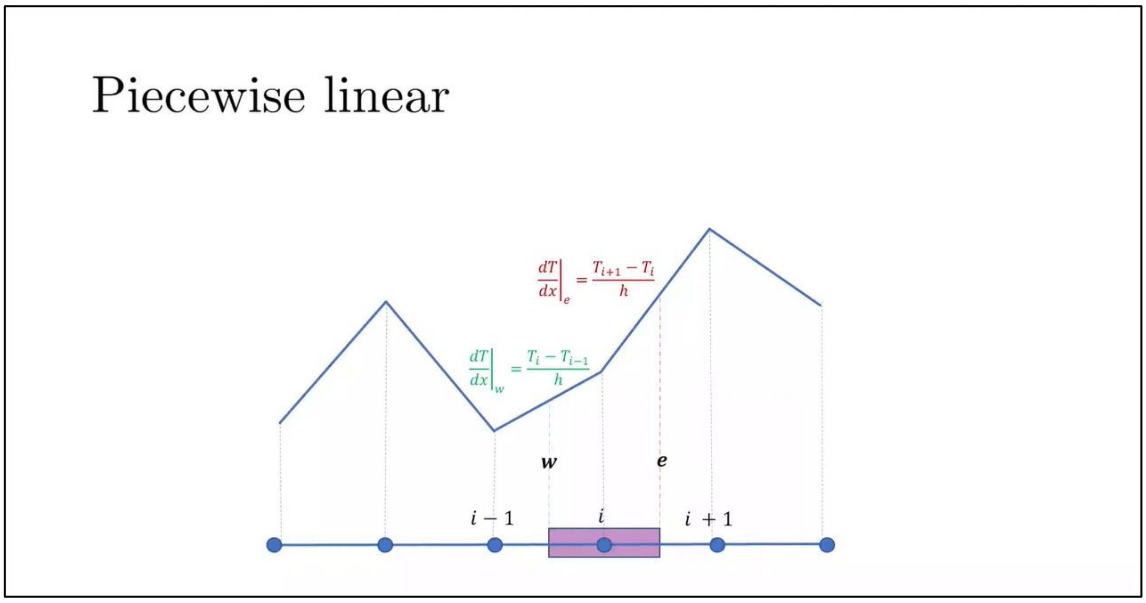
Finite volume discretization

$$\int_w^e \frac{d^2T}{dx^2}\,dx = 0$$

$$\left.\frac{dT}{dx}\right|_e - \left.\frac{dT}{dx}\right|_w = 0$$

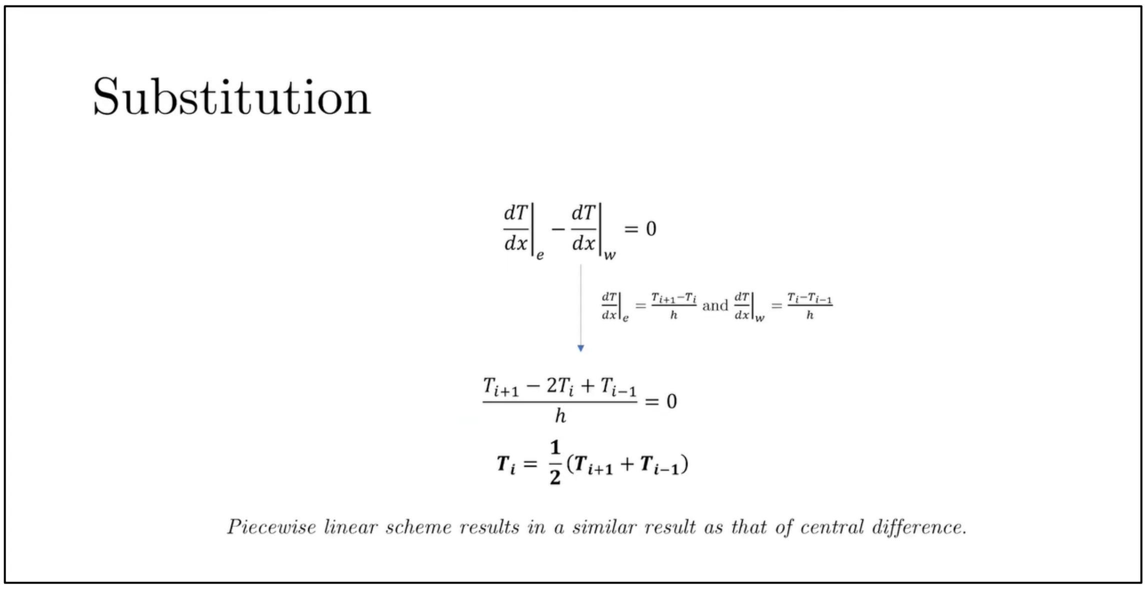
Depending on how we *assume* the variation of function between two grid points, we can numerically approximate the derivatives.

Assuming that the function varies piecewise linearly between 2 grid points:



Piecewise linear

$\left.\frac{dT}{dx}\right|_e = \frac{T_{i+1} - T_i}{h}$
$\left.\frac{dT}{dx}\right|_w = \frac{T_i - T_{i-1}}{h}$
w
e
$i-1$
i
$i+1$



Substitution

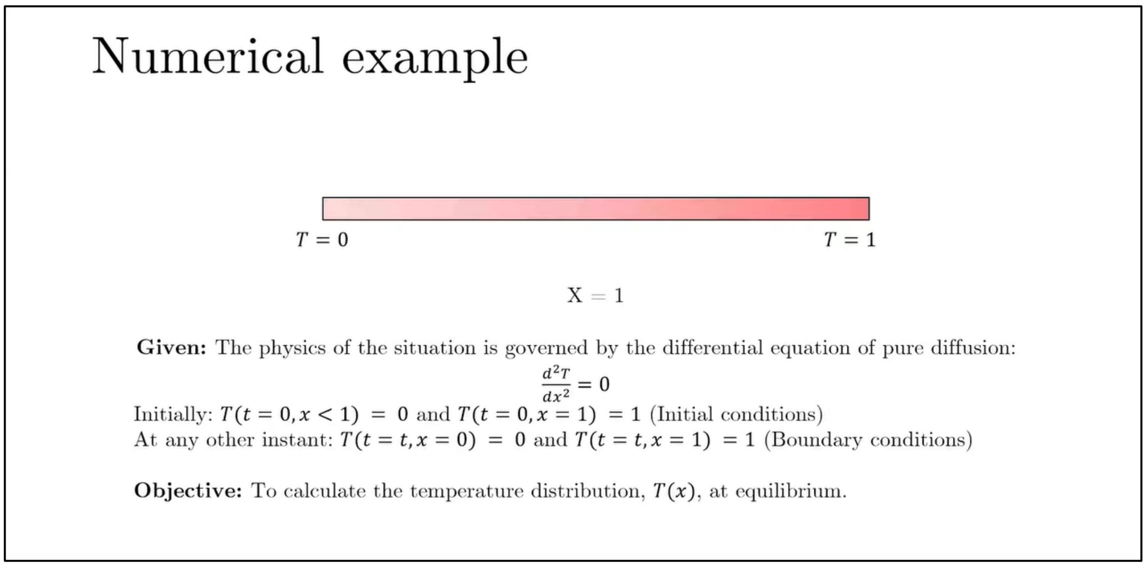
$$\left.\frac{dT}{dx}\right|_e - \left.\frac{dT}{dx}\right|_w = 0$$

$$\left.\frac{dT}{dx}\right|_e = \frac{T_{i+1}-T_i}{h} \text{ and } \left.\frac{dT}{dx}\right|_w = \frac{T_i-T_{i-1}}{h}$$

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{h} = 0$$

$$\boldsymbol{T_i = \frac{1}{2}(T_{i+1} + T_{i-1})}$$

Piecewise linear scheme results in a similar result as that of central difference.



Numerical example

$T = 0$ $T = 1$

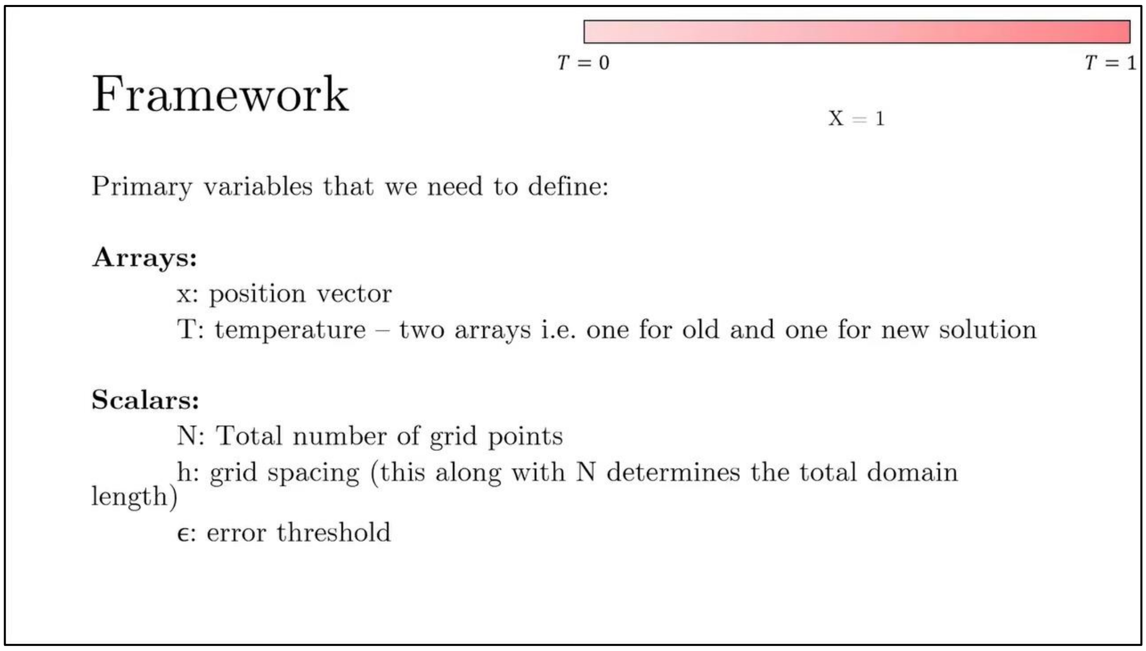
X = 1

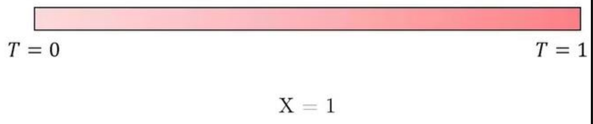
Given: The physics of the situation is governed by the differential equation of pure diffusion:

$$\frac{d^2T}{dx^2} = 0$$

Initially: $T(t = 0, x < 1) = 0$ and $T(t = 0, x = 1) = 1$ (Initial conditions)
At any other instant: $T(t = t, x = 0) = 0$ and $T(t = t, x = 1) = 1$ (Boundary conditions)

Objective: To calculate the temperature distribution, $T(x)$, at equilibrium.





Framework

Primary variables that we need to define:

Arrays:

x: position vector

T: temperature – two arrays i.e. one for old and one for new solution

Scalars:

N: Total number of grid points

h: grid spacing (this along with N determines the total domain length)

ϵ: error threshold