Applied Numerical Methods for Civil Engineering

CGN 3405 - 0002

Review Class for Exam 1

Xinyu Chen

Assistant Professor

University of Central Florida

40% in Exam 1

Components (≈ 10 points for each, 40 points in total):

- Quadratic equations
- Euler's method
- Deflection of cantilever beam
- Taylor series

Format:

- Understand quadratic formula
- Fill in a table of results over different steps (step size $\Delta x = 0.5, 1$) using Euler's methods
- Write down the Taylor series approximation with different orders (one of $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$, $f(x) = \sin(x)$, and $f(x) = e^x$ will be selected for exam)

Questions:

- Easy to calculate by hand

Roots of Equations

Solve quadratic equations with **a leading coefficient of 1**: $x^2 + bx + c = 0$

- Find two numbers who sum equals b and whose product equals c, i.e.,

$$d + e = b \qquad d \cdot e = c$$

by using the zero-product property:

$$(x + d)(x + e) = x^2 + \underbrace{(d + e)}_{=b} x + \underbrace{d \cdot e}_{=c} = 0$$

Example. Roots of a simple quadratic equation.

Let's solve the quadratic equation $x^2 + x - 6 = 0$.

This is a quadratic equation with

$$d + e = b = 1 \qquad d \cdot e = c = -6$$

So we can factor the equation as:

$$(x - 2)(x + 3) = 0$$

As a result, we can find the solutions as $x = 2$ and $x = -3$.

Roots of Equations

Example. Roots of a simple quadratic equation.

Let's solve the quadratic equation $x^2 - 5x + 6 = 0$.

This is standard quadratic equation of the form:

$$ax^2 + bx + c = 0$$

with

$$a = 1, \quad b = -5, \quad c = 6$$

So we can factor the equation as:

$$(x - 2)(x - 3) = 0$$

As a result, we can find the solutions as $x = 2$ and $x = 3$.

Roots of Equations

Quadratic formula. Given $ax^2 + bx + c = 0$ $(a \neq 0)$, we can derive the quadratic formula by completing the square.

1. Move the constant term to the right-hand side:

$$ax^2 + bx = -c$$

2. Divide by a and let the leading coefficient be 1:

$$x^2 + \frac{b}{a}x = -\frac{c}{a}$$

3. Add $\frac{b^2}{4a^2}$ to both sides:

$$x^2 + \frac{b}{a}x + \frac{b^2}{4a^2} = \frac{b^2}{4a^2} - \frac{c}{a}$$

4. Use the square root property:

$$\left(x + \frac{b}{2a}\right)^2 = \frac{b^2 - 4ac}{4a^2} \quad \Rightarrow \quad x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Roots of Equations

Quadratic formula. Given $ax^2 + bx + c = 0$ $(a \neq 0)$, the quadratic formula is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Python programming.

```
1 import numpy as np
2
3 def quad_formula(a, b, c):
4     term = np.sqrt(b**2 - 4*a*c)
5     x1 = (-b + term) / (2*a)
6     x2 = (-b - term) / (2*a)
7     return x1, x2
```

Euler's Method

- **Example**. Given an ODE:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y)$$

with initial condition $y(x_0) = y_0$

- **Euler's formula**:

$$\underbrace{y_{i+1}}_{\text{next value}} = \underbrace{y_i}_{\text{current value}} + \underbrace{\Delta x}_{\text{step size}} \cdot \underbrace{f(x_i, y_i)}_{\text{slope}}$$

$$x_{i+1} = x_i + \underbrace{\Delta x}_{\text{step size}}$$

- **Interpretation**:
  - $f(x_i, y_i)$ = slope at current point
  - Δx = step size (small values!)
  - step size $\times$ slope = predicted change in y
  - Add to current y to get next y

Euler's Method

- **Toy example**: Solve

$$\frac{\mathrm{d}y}{\mathrm{d}x} = x + y$$

with $y(0) = 1$, find $y(1)$ using step size $\Delta x = 0.5$.

- ❶ Initialize $x_0 = 0$ and $y_0 = 1$
- ❷ First step $(0 \to \Delta x)$

$$f(x_0, y_0) = x_0 + y_0 = 1 \quad y_1 = y_0 + \Delta x \cdot f(x_0, y_0) = 1.5 \quad x_1 = x_0 + \Delta x = 0.5$$

- ❸ Second step $(\Delta x \to 2\Delta x)$

$$f(x_1, y_1) = x_1 + y_1 = 2 \quad y_2 = y_1 + \Delta x \cdot f(x_1, y_1) = 2.5 \quad x_2 = x_1 + \Delta x = 1$$

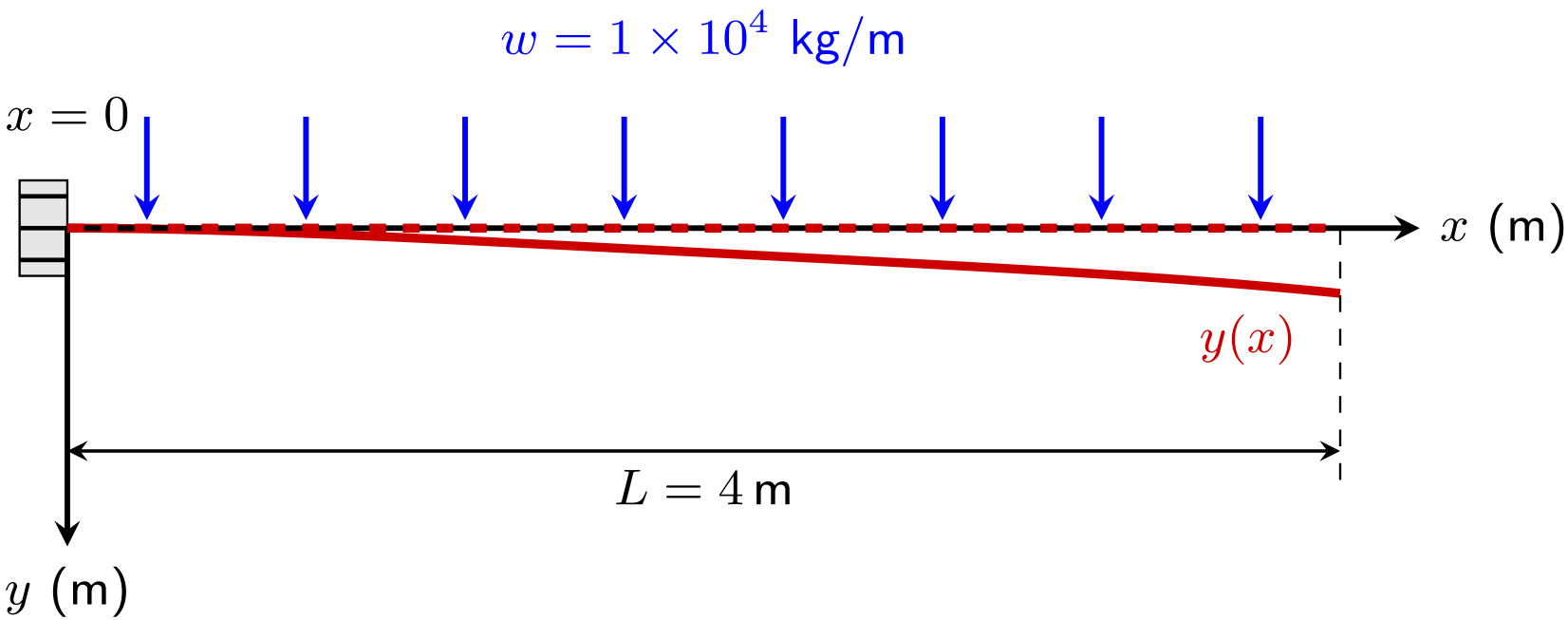
So we have $y(1) \approx y(x_2) = 2.5$.

- Hint (Keep in mind!):

$$\underbrace{y_{i+1}}_{\text{next value}} = \underbrace{y_i}_{\text{current value}} + \underbrace{\Delta x}_{\text{step size}} \cdot \underbrace{f(x_i, y_i)}_{\text{sloop}} \qquad x_{i+1} = x_i + \Delta x$$

Cantilever Beam Deflection

- Use **Euler's method** to find deflection $y(x)$ from $x = 0$ to $x = L$.
- $y(x)$ is downward deflection at point x (x is distance from fixed end).
- **Given parameters**:
  - Uniform load: $w = 1 \times 10^4$ kg/m
  - Beam length: $L = 4$ m
  - Modulus: $E = 2 \times 10^{11}$ Pa (steel)
  - Moment of inertia: $I = 3.25 \times 10^{-4}$ m^4



Cantilever Beam Deflection

- Use **Euler's method** to find deflection $y(x)$ from $x = 0$ to $x = L$.

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \underbrace{\frac{w}{24 \cdot E \cdot I}}_{\text{constant}}(4x^3 - 12Lx^2 + 12L^2x)$$

- x is distance from fixed end.
- $y(x)$ is downward deflection at point x.
- **Given parameters**:
  - Uniform load: $w = 1 \times 10^4$ kg/m
  - Beam length: $L = 4$ m
  - Modulus: $E = 2 \times 10^{11}$ Pa (steel)
  - Moment of inertia: $I = 3.25 \times 10^{-4}$ m^4
- Compute the **constant factor**:

$$c = \frac{w}{24 \cdot E \cdot I} = \frac{10^4}{24 \times (2 \times 10^{11}) \times (3.25 \times 10^{-4})} = 6.41 \times 10^{-6}$$

Cantilever Beam Deflection

- **Idea**: Given the step size $\Delta x = 0.25\,\text{m}$, we start from $y(0) = 0$ and **update the deflection** by

$$\underbrace{y_{i+1}}_{\text{next deflection}} = \underbrace{y_i}_{\text{current deflection}} + \underbrace{\Delta x}_{\text{step size}} \cdot \underbrace{\frac{\mathrm{d}y}{\mathrm{d}x}}_{\text{sloop}}$$

  update the position by

$$\underbrace{x_{i+1}}_{\text{next position}} = \underbrace{x_i}_{\text{current position}} + \underbrace{\Delta x}_{\text{step size}}$$

  where the sloop is given by

$$\frac{\mathrm{d}y}{\mathrm{d}x} = c(4x^3 - 12Lx^2 + 12L^2x)$$

- Number of steps (repeat `for` loop)

$$\frac{L}{\Delta x} = \frac{4}{0.25} = 16\,\text{steps}$$

- Compute the **constant factor**:

$$c = \frac{w}{24 \cdot E \cdot I} = \frac{10^4}{24 \times (2 \times 10^{11}) \times (3.25 \times 10^{-4})} = 6.41 \times 10^{-6}$$

Cantilever Beam Deflection

Deflection table (numerical vs. analytical solution).

$$\text{error} = |y_{\text{analytical}} - y_{\text{numerical}}| \qquad \text{with } |\cdot| \text{ denoting absolute value}$$

Distance x	Analytical solution	Numerical solution	Error
0.25	0.00003688	0	
0.50	0.00014143	0.00007222	0.07 mm
0.75	0.00030491	0.00020763	⋯
1.00	0.00051923	0.00039784	
1.25	0.00077687	0.00063502	
1.50	0.00107091	0.00091196	
1.75	0.00139506	0.00122206	
2.00	0.00174359	0.00155929	
2.25	0.00211140	0.00191827	
2.50	0.00249399	0.00229417	
2.75	0.00288744	0.00268279	
3.00	0.00328846	0.00308053	
3.25	0.00369434	0.00348438	
3.50	0.00410296	0.00389193	0.21 mm
3.75	0.00451285	0.00430138	0.21 mm
4.00	0.00492308	0.00471154	0.21 mm

Note: $1\,\text{meter} = 1,000\,\text{millimeter}$ (mm).

Taylor Series Approximation

Intuition: We predict $f(x_{i+1})$ using information at x_i:

- **Zeroth-order** (constant) approximation:

$$f(x_{i+1}) \approx f(x_i)$$

  Only works if f is **constant** between x_i and x_{i+1}

- **First-order** Taylor approximation:
  Add slope information:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i) \underbrace{(x_{i+1} - x_i)}_{\text{step size } \Delta x}$$

  - Represents a **straight line** (linear approximation)
  - Exact if f is linear

- **Euler's formula**:

$$\underbrace{y_{i+1}}_{\text{next value}} = \underbrace{y_i}_{\text{current value}} + \underbrace{\Delta x}_{\text{step size}} \cdot \underbrace{f(x_i, y_i)}_{\text{slope}} \qquad x_{i+1} = x_i + \underbrace{\Delta x}_{\text{step size}}$$

Taylor Series Approximation

Intuition: We predict $f(x_{i+1})$ using information at x_i:

- **Second-order** Taylor approximation:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2$$

  - Captures quadratic behavior
  - Better accuracy for smooth functions

Taylor Series Approximation

- **General Taylor polynomial**

$$
\begin{aligned}
f(x_{i+1}) \approx f(x_i) & \\
& + f'(x_i)(x_{i+1} - x_i) \\
& + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 \\
& + \cdots \\
& + \frac{f^{(n)}(x_i)}{n!}(x_{i+1} - x_i)^n \\
= & \sum_{k=0}^{n} \frac{f^{(k)}(x_i)}{k!}(x_{i+1} - x_i)^k
\end{aligned}
$$

Higher $n \rightarrow$ better approximation (if function is smooth)

Taylor Series Approximation of a Polynomial

Problem statement:

- Use zeroth- through fourth-order Taylor series expansions to approximate:

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

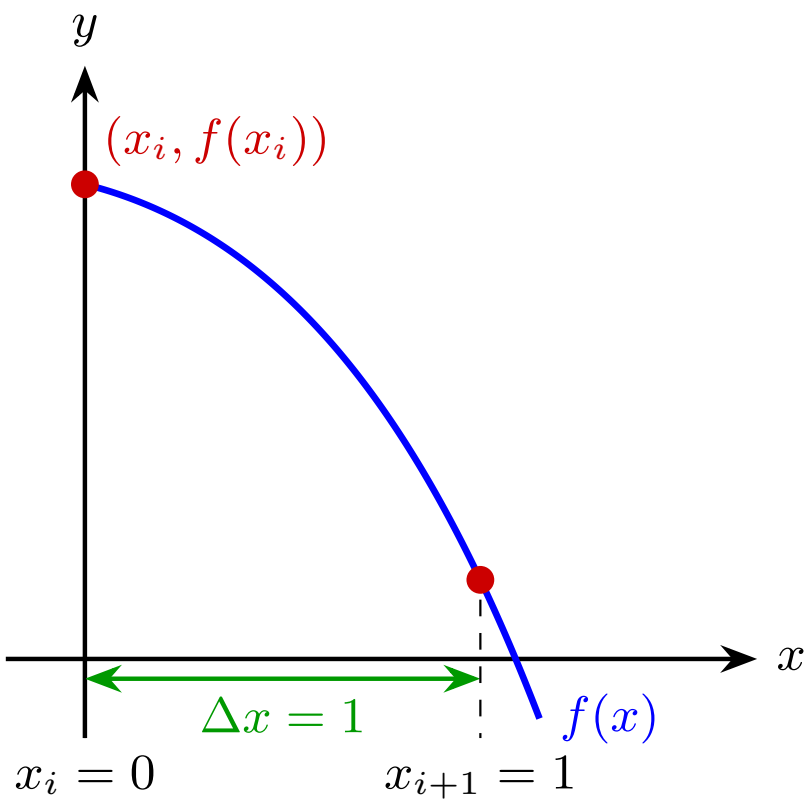
- **Goal**: Predict $f(1)$ using Taylor approximations of increasing order.

  - Function $f(x)$:

$$f(0) = 1.2$$

$$f(1) = -0.1 - 0.15 - 0.5 - 0.25 + 1.2 = 0.2$$

  - True value to predict: $f(1) = 0.2$



Taylor Series Approximation of a Polynomial

First-order approximation for $f(1)$:

- Need first-order derivative at $x_i = 0$:

$$\begin{aligned} & f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2 \\ \Rightarrow \quad & f'(x) = -0.4x^3 - 0.45x^2 - x - 0.25 \\ \Rightarrow \quad & f'(0) = -0.25 \end{aligned}$$

- First-order Taylor series:

$$\begin{aligned} & f(x_{i+1}) \approx f(x_i) + f'(x_i) \cdot (x_{i+1} - x_i) \\ \Rightarrow \quad & f(1) \approx 1.2 + (-0.25) \times 1 = 0.95 \end{aligned}$$

Taylor Series Approximation of a Polynomial

Second-order approximation for $f(1)$:

- Need second-order derivative at $x_i = 0$:

$$\begin{aligned} f(x) &= -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2 \\ \Rightarrow \quad f'(x) &= -0.4x^3 - 0.45x^2 - x - 0.25 \\ \Rightarrow \quad f''(x) &= -1.2x^2 - 0.9x - 1 \\ \Rightarrow \quad f''(0) &= -1 \end{aligned}$$

- Second-order Taylor series:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i) \cdot (x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2$$

$$\Rightarrow \quad f(1) \approx 1.2 - 0.25 \times 1 + \left(\frac{-1}{2}\right) \times 1^2 = 0.45$$

Taylor Series Approximation of a Polynomial

Third-order approximation for $f(1)$:

- Need third-order derivative at $x_i = 0$:

$$
\begin{aligned}
f(x) &= -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2 \\
\Rightarrow \quad f'(x) &= -0.4x^3 - 0.45x^2 - x - 0.25 \\
\Rightarrow \quad f''(x) &= -1.2x^2 - 0.9x - 1 \\
\Rightarrow \quad f'''(x) &= -2.4x - 0.9 \\
\Rightarrow \quad f'''(0) &= -0.9
\end{aligned}
$$

- Third-order Taylor series:

$$f(1) \approx 1.2 - 0.25 \times 1 - 0.5 \times 1^2 + \left(\frac{-0.9}{3!}\right) \times 1^3 = 0.3$$

Taylor Series Approximation of a Polynomial

Fourth-order approximation for $f(1)$:

- Need fourth-order derivative at $x_i = 0$:

$$\begin{aligned} f(x) &= -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2 \\ \Rightarrow \quad f'(x) &= -0.4x^3 - 0.45x^2 - x - 0.25 \\ \Rightarrow \quad f''(x) &= -1.2x^2 - 0.9x - 1 \\ \Rightarrow \quad f'''(x) &= -2.4x - 0.9 \\ \Rightarrow \quad f^{(4)}(x) &= -2.4 \\ \Rightarrow \quad f^{(4)}(0) &= -2.4 \end{aligned}$$

- Fourth-order Taylor series:

$$f(1) \approx 1.2 - 0.25 \times 1 - 0.5 \times 1^2 - 0.15 \times 1^3 - \left(\frac{-2.4}{4!}\right) \times 1^4 = 0.2$$

Taylor Series Approximation of $\sin(x)$

Maclaurin series:

- A Taylor series expansion of a function about 0
- Taylor series approximation:

$$f(x_{i+1}) \approx \sum_{k=0}^{n} \frac{f^{(k)}(x_i)}{k!}(x_{i+1} - x_i)^k$$

- Set $x_i = 0$ and $x_{i+1} = x$, then

$$f(x) \approx \sum_{k=0}^{n} \frac{f^{(k)}(0)}{k!}x^k$$

- Derivatives of $f(x) = \sin(x)$:

$$f'(x) = \underbrace{\cos(x)}_{\cos(0)=1}, \quad f''(x) = -\underbrace{\sin(x)}_{\sin(0)=0}, \quad f'''(x) = -\underbrace{\cos(x)}_{\cos(0)=1}, \quad f^{(4)}(x) = \underbrace{\sin(x)}_{\sin(0)=0}$$

- Formula:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \frac{x^{13}}{13!} - \frac{x^{15}}{15!} + \cdots$$

Taylor Series Approximation of e^x

Maclaurin series:

- A Taylor series expansion of a function about 0

$$f(x) \approx \sum_{k=0}^{n} \frac{f^{(k)}(0)}{k!} x^k$$

- Derivatives of $f(x) = e^x$:

$$f'(x) = f''(x) = \cdots = f^{(n)}(x) = e^x \quad \Rightarrow \quad f^{(n)}(0) = 1 \quad \text{for all } n$$

- Formula:

$$f(x) \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

$$= \sum_{k=0}^{n} \frac{x^k}{k!}$$

Taylor Series Approximation of e^x for $x = 1$

Problem statement:

- Given the exponential function

$$f(x) = e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

- **Goal**: Predict $f(1)$ using Taylor approximations of increasing order.
- True value: $f(1) = e = 2.71828$

```
1 import numpy as np
2
3 print(np.exp(1))
```

Taylor Series Approximation of e^x for $x = 1$

Predict $f(1)$ (true value $f(1) = 2.71828$):

- First-order approximation:

$$\hat{f}(x) = 1 + x \quad \Rightarrow \quad \hat{f}(1) = 1 + 1 = 2$$

- Second-order approximation:

$$\hat{f}(x) = 1 + x + \frac{x^2}{2!} \quad \Rightarrow \quad \hat{f}(1) = 1 + 1 + \frac{1}{2} = 2.5$$

- Third-order approximation:

$$\hat{f}(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} \quad \Rightarrow \quad \hat{f}(1) = 1 + 1 + \frac{1}{2} + \frac{1}{6} = 2.66667$$

- Fourth-order approximation:

$$\hat{f}(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} \quad \Rightarrow \quad \hat{f}(1) = 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} = 2.70833$$

Taylor Series Approximation of e^x for $x = 2$

Predict $f(2)$ (true value $f(2) = 7.38906$, see `print(np.exp(2))`):

- First-order approximation:

$$\hat{f}(x) = 1 + x \quad \Rightarrow \quad \hat{f}(2) = 1 + 2 = 3$$

- Second-order approximation:

$$\hat{f}(x) = 1 + x + \frac{x^2}{2!} \quad \Rightarrow \quad \hat{f}(2) = 1 + 2 + \frac{2^2}{2} = 5$$

- Third-order approximation:

$$\hat{f}(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} \quad \Rightarrow \quad \hat{f}(2) = 1 + 2 + \frac{2^2}{2} + \frac{2^3}{6} = 6.33333$$

- Fourth-order approximation:

$$\hat{f}(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} \quad \Rightarrow \quad \hat{f}(2) = 1 + 2 + \frac{2^2}{2} + \frac{2^3}{6} + \frac{2^4}{24} = 7$$

20% in Exam 1

Format (20 points in total):

- Give you Python codes
- Write down the results

Questions:

- **3 questions** are selected from the sample codes
- Parameters might be a little different
- Make sure you can totally understand these sample codes

Sample 1

```
1 a = 2
2 b = 3
3 print(a + b)   # plus
4 print(a - b)   # minus
5 print(a * b)   # product
6 print(a / b)   # division
7 print(a ** 2) # quadratic function
8 print(a ** 3) # cubic function
```

Corresponding **arithmetic operations**:

Line **3**: $a + b$

Line **4**: $a - b$

Line **5**: $a \cdot b$

Line **6**: $\frac{a}{b}$

Line **7**: a^2

Line **8**: a^3

Note: `a ** n` refers to a to the power of n.

Sample 2

Fibonacci Sequence.

- Given $f(1) = f(2) = 1$, the Fibonacci sequence takes the form of

$$f(n) = f(n-1) + f(n-2),\ n > 2$$

```
1  import numpy as np
2
3  def fib(n):              # Input n>2
4      f = np.zeros(n)
5      f[0] = 1
6      f[1] = 1
7      for i in range(2, n):
8          f[i] = f[i - 1] + f[i - 2]
9      return f[n - 1]
10
11 print(fib(5))
12 print(fib(6))
13 print(fib(7))
```

Sample 3

A system of linear equations.

- Let's solve:

$$\begin{cases} 3x + 2y = 5 \\ x - y = 0 \end{cases} \quad \Rightarrow \quad \begin{bmatrix} 3 & 2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

- Try to solve by hand, and then check with Python.
- Define matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$:

Line **3**: $\boldsymbol{A} = \begin{bmatrix} 3 & 2 \\ 1 & -1 \end{bmatrix}$

Line **4**: $\boldsymbol{b} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$

```
1 import numpy as np
2
3 A = np.array([[3, 2], [1, -1]])
4 b = np.array([5, 0])
5 solution = np.linalg.solve(A, b)
6 print('Solution (x, y):', solution)
```

Sample 4

- Generate sequences of numbers:

```
1  # Count from 0 to 4
2  for i in range(5):
3      print(i)
4
5  # With start and end
6  for i in range(2, 6):
7      print(i)
8
9  # With step
10 for i in range(0, 10, 2):
11     print(i)
```

Line **2-3** Result: 0, 1, 2, 3, 4

Line **6-7** Result: 2, 3, 4, 5

Line **10-11** Result: 0, 2, 4, 6, 8

Sample 5

- Repeat while condition is true:

```
a = [1, 2, 3, 4, 5, 6, 7, 8]
i = 0
while a[i] < 6:
    print(a[i])
    i = i + 1
```

Result: 0, 1, 2, 3, 4, 5

Sample 6

- `np.arange()`: Like Python's `range()`, but returns array

```
1 import numpy as np
2
3 # Bungee jumping velocity
4 delta_t = 0.1
5 t_start = 0
6 t_end = 20
7 time_step = np.arange(t_start, t_end, delta_t)
8 print(time_step)
```

will not count `t_end = 20`.

- Toy examples:

```
1 import numpy as np
2
3 a = np.arange(1, 10, 2) # step size: 2
4 b = np.arange(1, 10, 2.5) # step size: 2.5
```

$$\boldsymbol{a} = (1, 3, 5, 7, 9)^\top \qquad \boldsymbol{b} = (1, 3.5, 6, 8.5)^\top$$

Sample 7

- **Converting matrix into vector**

  Given a matrix $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, there are two strategies:

```
1 import numpy as np
2
3 A = np.array([[1, 2, 3], [4, 5, 6]])
4 a1 = np.reshape(A, (6)) # C-like index ordering
5 print(a1)
6 a2 = np.reshape(A, (6), order = 'F') # Fortran-like
      index ordering
7 print(a2)
```

$$\boldsymbol{a}_1 = (1, 2, 3, 4, 5, 6)^\top \qquad \boldsymbol{a}_2 = (1, 4, 2, 5, 3, 6)^\top$$

- **Converting vector into matrix**

```
1 A1 = np.reshape(a1, (2, 3)) # C-like index ordering
2 print(A1)
3 A2 = np.reshape(a1, (2, 3), order = 'F') # Fortran-like
      index ordering
4 print(A2)
```

Sample 8

- Given a vector

```
import numpy as np
np.random.seed(0)

a = np.random.rand(10)
print(a)
```

Result:

```
[0.5488135  0.71518937 0.60276338 0.54488318 0.4236548
     0.64589411 0.43758721 0.891773   0.96366276
    0.38344152]
```

- Indexing

```
i = 1
j = 7
print(a[i])      # 2nd
print(a[j])      # 8th
print(a[i :])    # 2nd to the last
print(a[: j])    # 1st to 7th
print(a[i : j]) # 2nd to 7th
```

Sample 9

Flip or reverse an array:

```
import numpy as np

a = np.arange(1, 9)
a_flip = np.flip(a)
A = np.arange([[1, 2, 3], [4, 5, 6]])
A0 = np.flip(A, axis = 0)
A1 = np.flip(A, axis = 1)
A_flip = np.flip(A)
print(a_flip)
print(A0)
print(A1)
print(A_flip)
```

Sample 10

Stack two arrays vertically and horizontally:

```
1 import numpy as np
2
3 A = np.array([[1, 2], [3, 4]])
4 B = np.array([[5, 6], [7, 8]])
5 C = np.vstack(A, B)
6 D = np.hspack(A, B)
7 print(C)
8 print(D)
```

Sample 11

- **Factorial of a non-negative integer** n is the product of all positive integers less than or equal to n:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

$$= \begin{cases} 1 & \text{if } n = 1 \\ n \times \underbrace{(n-1)!}_{\text{factorial}} & \text{if } n > 1 \end{cases}$$

```
1 def factorial_r(n):
2     if n == 0:
3         return 1
4     else:
5         return n * factorial_r(n-1)
```

- Toy example: $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

```
1 print(factorial_r(5))
```

- Using NumPy

```
1 import numpy as np
2
3 x = np.prod(np.arange(1, 6))
```

Sample 12

- ℓ_1-norm:

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

- ℓ_2-norm:

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

- ℓ_∞-norm:

$$\|\boldsymbol{x}\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$$

```
1  import numpy as np
2
3  a = np.arange(1, 5)
4  x = np.linalg.norm(a, 1)
5  y = np.linalg.norm(a, 2)
6  z = np.linalg.norm(a, np.inf)
7  s = np.sum(a)
8  print(x)
9  print(y)
10 print(z)
11 print(s)
```

Sample 13

- Inner product:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\top \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$$

- Outer product:

$$\boldsymbol{x}\boldsymbol{y}^\top = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix}$$

```python
import numpy as np

x = np.arange(1, 4)
y = np.arange(4, 7)
print(np.inner(x, y))
print(np.outer(x, y))
```

Sample 14

- Kronecker product:

$$\boldsymbol{X} \otimes \boldsymbol{Y} = \begin{bmatrix} x_{11}\boldsymbol{Y} & x_{12}\boldsymbol{Y} & \cdots & x_{1n}\boldsymbol{Y} \\ x_{21}\boldsymbol{Y} & x_{22}\boldsymbol{Y} & \cdots & x_{2n}\boldsymbol{Y} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1}\boldsymbol{Y} & x_{m2}\boldsymbol{Y} & \cdots & x_{mn}\boldsymbol{Y} \end{bmatrix}$$

- Example:

$$\boldsymbol{X} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \boldsymbol{Y} = \begin{bmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \end{bmatrix}$$

```
1 import numpy as np
2
3 X = np.array([[1, 2], [3, 4]])
4 Y = np.array([[5, 6, 7], [8, 9, 10]])
5 print(np.kron(X, Y))
```

Sample 15

- **Example**: For vector $\boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ and matrix $\boldsymbol{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$, compute the angle between $\boldsymbol{x}$ and $\boldsymbol{Ax}$.
- Matrix-vector multiplication:

$$\boldsymbol{Ax} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$

- Angle:

$$\cos(\theta) = \frac{\boldsymbol{x}^\top (\boldsymbol{Ax})}{\|\boldsymbol{x}\|_2 \cdot \|\boldsymbol{Ax}\|_2} = \frac{1 \times 0 + 2 \times 2 + 1 \times 0}{\sqrt{1^2 + 2^2 + 1^2} \times \sqrt{0^2 + 2^2 + 0^2}} = \frac{\sqrt{6}}{3}$$

```python
import numpy as np

x = np.array([1, 2, 1])
A = np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])
theta = np.arccos(x @ A @ x / (np.linalg.norm(x, 2) * np.
    linalg.norm(A @ x, 2)))
```

Quiz 2

1. What is the formal $f(x) = 0$ for x? **[C]**
   A. Feasibility of equations
   B. Intersections of equations
   C. Roots of equations
   D. Optimum of equations

2. Numerical methods are techniques by which mathematical problems are formulated so that they can be solved with _____? **[D]**
   A. optimization algorithms
   B. matrix computations
   C. black-box tools
   D. arithmetic operations

3. Which of the following best describes a quadratic equation $x^2 - 1 = 0$? **[B]**
   A. A linear equation
   B. A nonlinear equation
   C. A differential equation
   D. A transcendental equation

Quiz 2

4. Is using `import numpy as np` the standard convention within the Python community? **[A]**

   A. True

   B. False

5. Based on the mathematical topics covered in this course, is the equation $f(x) = x^2 - 5x + 6 = 0$ an example of: **[A]**

   A. A root-finding problem for a nonlinear equation

   B. A system of linear equations

   C. An ordinary differential equation

   D. An optimization problem

6. An equation containing a _____ polynomial is called a quadratic equation. **[B]**

   A. first-degree

   B. second-degree

   C. third-degree

   D. fourth-degree

Quiz 2

7. Find all roots of quadratic equation $(x-2)(3x+7)=0$. **[BD]**

   A. -2

   B. 2

   C. $-3/7$

   D. $-7/3$

8. Find all roots of quadratic equation $x^2+8x+15=0$. **[BD]**

   A. 3

   B. -3

   C. 5

   D. -5

9. Find all roots of quadratic equation $x^2-4x-21=0$. **[BC]**

   A. 3

   B. -3

   C. 7

   D. -7

Quiz 2

10. Which line is used to return the roots of quadratic equations? **[D]**

```
1 import numpy as np
2
3 def quad_formula(a, b, c):
4     term = np.sqrt(b**2 - 4*a*c)
5     x1 = (-b + term) / (2*a)
6     x2 = (-b - term) / (2*a)
7     return x1, x2
```

A. Line 4

B. Line 5

C. Line 6

D. Line 7

Quiz 2

11. In Python programming, what is the result of `a ** 3` if we give the input `a = 4`? **[C]**

A. 4

B. 16

C. 64

D. 128

12. In Python programming, what is the result of `a ** 2` if we give the input `a = 3`? **[B]**

A. 3

B. 9

C. 27

D. 81

Quiz 3

1. How to write $\sqrt{b^2 - 4ac}$ in Python? Note that the usage of NumPy is `import numpy as np` **[C]**

   A. `np.sqrt(b ** 2 - 4ac)`

   B. `np.sqrt(b ** 2 - 4 * ac)`

   C. `np.sqrt(b ** 2 - 4 * a * c)`

   D. `np.sqrt(b ** 2 - 4a * c)`

2. A system of linear equations is a set of multiple linear equations with the different variables. **[B]**

   A. True

   B. False

3. The common methods for solving a system of linear equations by hand are Substitution and Elimination. **[A]**

   A. True

   B. False

Quiz 3

4. Optimization is the process of finding the feasible solution to a problem. **[B]**

   A. True

   B. False

5. Optimization can be simplified as follows. It is used to find the value of x that: **[AD]**

   A. minimizes f(x)

   B. convexifies f(x)

   C. constrains f(x)

   D. maximizes f(x)

6. What is the result of `5 ** n` in Python if $n = 4$? **[D]**

   A. 5

   B. 25

   C. 125

   D. 625

Quiz 3

7. What does `np.linalg.solve(A, b)` return? **[C]**

   A. The determinant of matrix A

   B. The inverse of matrix A

   C. The solution for the linear system A x = b

   D. The eigenvalues of matrix A

8. The physical interpretation of numerical integration is the determination of the slope of a curve at a point. **[B]**

   A. True

   B. False

9. In the context of ordinary differential equations (ODEs), what does the term Δt represent in the formula $y_{i+1} = y_i + f(t_i, y_i)\Delta t$? **[B]**

   A. The slope of the function

   B. The time step size

   C. The initial condition

   D. The exact solution

Quiz 3

10. Optimization in engineering can involve minimizing cost or maximizing efficiency. **[A]**

A. True

B. False

11. Which of the following are examples of optimization goals in engineering? (Select all that apply.) **[ABF]**

A. Minimize material usage

B. Maximize traffic flow efficiency

C. Solve linear equations

D. Fit a curve through data points

F. Maximize structural strength

Quiz 3

12. What does the following Python code compute? **[B]**

```
import numpy as np

def quad_formula(a, b, c):
    term = np.sqrt(b**2 - 4*a*c)
    x1 = (-b + term) / (2*a)
    x2 = (-b - term) / (2*a)
    return x1, x2
```

A. The derivative of a quadratic function

B. The roots of a quadratic equation

C. The integral of a quadratic function

D. The minimum of a quadratic function

Quiz 3

13. If `a = 2` and `b = 3` in Python, what will `print(a ** b)` output? **[C]**

    A. 5

    B. 6

    C. 8

    D. 9

14. How to write matrix $\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 1 & -3 \end{bmatrix}$ in Python with NumPy? **[B]**

    A. `A = [[2, 1], [1, -3]]`

    B. `A = np.array([[2, 1], [1, -3]])`

    C. `A = np.array([2, 1], [1, -3])`

Quiz 4

1. The bungee jumper's velocity over time can be modeled using an ordinary differential equation. **[A]**

   A. True

   B. False

2. Which forces are acting on the bungee jumper? (Select all that apply.) **[AD]**

   A. Gravitational force

   B. Tension force from the cord

   C. Centrifugal force

   D. Air resistance

3. In the ODE $\frac{dv}{dt} = g - \frac{c_d}{m}v^2$, what does $\frac{dv}{dt}$ represent? **[B]**

   A. Position

   B. Acceleration

   C. Velocity

   D. Drag coefficient

Quiz 4

4. In the Python function for Euler's method, what does the following line compute? **[B]**

```
a = g - cd / m * (v[i] ** 2)
```

A. Position

B. Acceleration

C. Drag force

D. Terminal velocity

5. The terminal velocity occurs when acceleration is zero. **[A]**

A. True

B. False

6. If mass $m = 50\,\text{kg}$, $g = 9.81\,\text{m/s}^2$, and $c_d = 0.25\,\text{kg/m}$, what is the terminal velocity? **[C]**

A. 19.81 m/s

B. 31.32 m/s

C. 44.29 m/s

D. 50.00 m/s

Quiz 4

7. Which factors affect the accuracy of Euler's method? **[A]**

   A. Time step size t

   B. Initial velocity

   C. Drag coefficient

   D. Mass of the jumper

8. A smaller Δt in Euler's method reduces computational error but increases computation time. **[A]**

   A. True

   B. False

9. In the Fibonacci sequence Python code, what does `f[i] = f[i-1] + f[i-2]` compute? **[B]**

   A. The sum of the first n numbers

   B. The next Fibonacci number

   C. The average of previous two numbers

   D. The product of previous two numbers

Quiz 4

10. Which of the following are advantages of numerical methods like Euler's method? (Select all that apply.) **[AC]**

   A. Can solve problems with no analytical solution

   B. Always gives exact results

   C. Easy to implement in code

11. In the error analysis plot, the error is computed as $|v_{\text{numerical}} - v_{\text{analytical}}|$ **[A]**

   A. True

   B. False

12. What does the following Python function return? **[B]**

```
def analytical_solution(m, g, cd, t):
    v_term = np.sqrt(m * g / cd)
    return v_term * np.tanh(np.sqrt(g * cd / m) * t)
```

   A. Numerical velocity from Euler's method

   B. Analytical velocity as a function of time

   C. Terminal velocity only

   D. Acceleration over time

Quiz 4

13. Which parameters can be adjusted to keep terminal velocity within safe limits? **[B]**

A. Increase mass

B. Increase drag coefficient

C. Decrease gravitational acceleration

D. Use a longer cord

14. In the parameter sensitivity study, if mass increases and drag coefficient stays the same, terminal velocity: **[A]**

A. Increases

B. Decreases

C. Stays the same

D. Becomes zero

15. The Fibonacci sequence example was used to introduce recursive functions in Python. **[A]**

A. True

B. False

Quiz 5

1. Euler's Method is primarily used to solve which type of mathematical problems? **[C]**

   A. Algebraic equations

   B. Partial differential equations

   C. Ordinary differential equations (ODEs)

   D. Linear systems only

2. When is Euler's Method most appropriate to use? **[B]**

   A. When an exact analytical solution is required

   B. When the rate of change is known and an approximate solution is acceptable

   C. When the function has no derivative

   D. When step size is large

Quiz 5

3. In Euler's Method, what does $f(x_i, y_i)$ represent? **[C]**
   A. The next value of y
   B. The step size
   C. The slope at the current point
   D. The numerical error
4. Why should the step size Δx be small when using Euler's Method? **[B]**
   A. To reduce computational cost
   B. To reduce numerical error
   C. To increase instability
   D. To simplify equations
5. What is the effect of increasing the step size in Euler's Method? **[C]**
   A. Higher accuracy
   B. Lower computational speed
   C. Increased numerical error
   D. No effect on results

Quiz 5

6. For the ODE $\frac{dy}{dx} = x + y$ with $y(0) = 1$ and $\Delta x = 0.5$, what is $y(0.5)$ using Euler's method? **[C]**

   A. 1.0

   B. 1.25

   C. 1.5

   D. 2.0

7. Why is calculating cantilever beam deflection important in engineering? **[B]**

   A. To determine beam color

   B. To check safety and meet building codes

   C. To calculate air resistance

   D. To determine beam weight

8. In the cantilever beam problem, what does $y(x)$ represent? **[C]**

   A. Beam stress

   B. Beam length

   C. Downward deflection at distance x

   D. Load intensity

Quiz 5

9. The constant $c = \frac{w}{24 \cdot E \cdot I}$ depends on which parameters? **[B]**

   A. Load, beam length, and position

   B. Load, modulus of elasticity, and moment of inertia

   C. Velocity and acceleration

   D. Step size and position

10. How many Euler steps are required if beam length $L = 4\,\text{m}$ and step size $\Delta x = 0.25\,\text{m}$? **[C]**

   A. 8

   B. 12

   C. 16

   D. 20

11. Why does numerical error increase as distance x increases in the cantilever beam example? **[B]**

   A. Because analytical solutions are unstable

   B. Because Euler's Method accumulates error at each step

   C. Because the load decreases

   D. Because step size changes