Introduction to Programming for Control & Application

IK6016 Control and Optimization Technology

Table of contents

Getting Started

For Python

- **Install Python**
  - Download and install Python from https://www.python.org/downloads/
- **Install Jupyter using Anaconda**
  - Download Anaconda from https://www.anaconda.com/download/
  - To run the notebook, type `jupyter notebook` in Anaconda Prompt

For MATLAB

- Download MATLAB from https://www.mathworks.com/downloads/
- Follow the installment guide in https://www.mathworks.com/help/install/ug/install-products-with-internet-connection.html

01

Basic Programming

Variables and arithmetic operations, matrix operations, loop and conditionals, plotting graphs, and defining functions

Variables in Programming

What is a variable?

- A storage location in the program where we can store data (numbers, text, etc.)

A variable has a name and a value, where the value can change during the execution of the program.

Example in Python:

```python
# Define a variable and assign a value
x = 10
y = 5

# Use the variables in operations
z = x + y
print("Result of x + y:", z)
```

Example in MATLAB:

```matlab
% Define a variable and assign a value
x = 10;
y = 5;

% Use the variables in operations
z = x + y;
disp(['Result of x + y: ', num2str(z)]);
```

Arithmetic Operations

Arithmetic operations allow us to perform mathematical calculations.

- Addition (+)
- Substraction (-)
- Multiplication (*)
- Division (/)
- Exponentiation (Python: **, MATLAB: ^)

Example in Python:

```
x = 10
y = 5

print("Addition:", x + y)
print("Subtraction:", x - y)
print("Multiplication:", x * y)
print("Division:", x / y)
print("Exponentiation:", x ** 2)
```

Example in MATLAB:

```
x = 10;
y = 5;

disp(['Addition: ', num2str(x + y)]);
disp(['Subtraction: ', num2str(x - y)]);
disp(['Multiplication: ', num2str(x * y)]);
disp(['Division: ', num2str(x / y)]);
disp(['Exponentiation: ', num2str(x ^ 2)]);
```

Matrix in Programming

What is a matrix?

- A two-dimensional array of numbers, arranged in rows and columns

Notation example:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Example in Python:

```
import numpy as np

A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

print(A)
```

Example in MATLAB:

```
A = [1 2 3; 4 5 6; 7 8 9]
```

Matrix Operations

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \qquad B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

Addition:

$$C = A + B = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix}$$

Substraction:

$$D = A - B = \begin{bmatrix} -8 & -6 & -4 \\ -2 & 0 & 2 \\ 4 & 6 & 8 \end{bmatrix}$$

Example in Python:

```
Import numpy as np

A = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
B = np.array([[9, 8, 7],[6, 5, 4],[3, 2, 1]])
C = A + B  # Matrix addition
D = A - B  # Matrix subtraction
print(C)
print(D)
```

Example in MATLAB:

```
A = [1 2 3; 4 5 6; 7 8 9]
B = [9 8 7; 6 5 4; 3 2 1];
C = A + B  % Matrix addition
D = A - B  % Matrix subtraction
disp(C)
disp(D)
```

Matrix Operations

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \qquad B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

Matrix multiplication:

$$E = A \cdot B$$

$$= \begin{bmatrix} (1 \times 9 + 2 \times 6 + 3 \times 3) & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & 90 \end{bmatrix}$$

Element-wise multiplication:

$$D = A \odot B = \begin{bmatrix} 9 & 16 & 21 \\ 24 & 25 & 24 \\ 21 & 16 & 9 \end{bmatrix}$$

Example in Python:

```
Import numpy as np

A = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
B = np.array([[9, 8, 7],[6, 5, 4],[3, 2, 1]])
E = np.dot(A, B)  # Matrix multiplication
F = A * B  # Element-wise multiplication
print(E)
print(F)
```

Example in MATLAB:

```
A = [1 2 3; 4 5 6; 7 8 9]
B = [9 8 7; 6 5 4; 3 2 1];
E = A * B  % Matrix multiplication
F = A .* B  % Element-wise multiplication
disp(E)
disp(F)
```

Matrix Operations

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \qquad B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

Matrix multiplication:

$$E = A \cdot B = \begin{bmatrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & 90 \end{bmatrix}$$

Element-wise multiplication:

$$F = A \odot B = \begin{bmatrix} (9 \times 1) & 16 & 21 \\ 24 & 25 & 24 \\ 21 & 16 & 9 \end{bmatrix}$$

Example in Python:

```
Import numpy as np

A = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
B = np.array([[9, 8, 7],[6, 5, 4],[3, 2, 1]])
E = np.dot(A, B)  # Matrix multiplication
F = A * B  # Element-wise multiplication
print(E)
print(F)
```

Example in MATLAB:

```
A = [1 2 3; 4 5 6; 7 8 9]
B = [9 8 7; 6 5 4; 3 2 1];
E = A * B  % Matrix multiplication
F = A .* B  % Element-wise multiplication
disp(E)
disp(F)
```

Matrix Operations

$$A = \begin{bmatrix} 2 & 1 & 3 \\ 0 & 2 & 4 \\ 1 & 1 & 2 \end{bmatrix}$$

Transpose:

$$A^T = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 2 & 1 \\ 3 & 4 & 2 \end{bmatrix}$$

Inversion:

$$A^{-1} = \begin{bmatrix} 0 & 1/8 & 1/4 \\ 1/2 & -1/8 & -1 \\ 1/4 & -1/8 & -1/2 \end{bmatrix}$$

Note: A matrix must be square and non-singular to have an inverse.

Example in Python:

```
Import numpy as np

A = np.array([[2, 1, 3],[0, 2, 4],[1, 1, 2]])
A_T = np.transpose(A)  # Matrix transpose
A_inv = np.linalg.inv(A)  # Matrix inversion

print(A_T)
print(A_inv)
```

Example in MATLAB:

```
A = [2 1 3; 0 2 4; 1 1 2]
A_T = A’  % Matrix transpose
A_inv = inv(A)  % Matrix inversion

disp(A_T)
disp(A_inv)
```

Matrix Operations

Identity Matrix:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Linear Equation:

$$A = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 2 & 1 \\ 3 & 4 & 2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Linear equation:

$$A\mathbf{x} = \mathbf{b}$$

The solution:

$$\mathbf{x} = A^{-1}\mathbf{b}$$

Example in Python:

```
Import numpy as np

I = np.eye(3)  # 3x3 identity matrix

A = np.array([[2, 1, 3],[0, 2, 4],[1, 1, 2]])
b = np.array([1, 2, 3])
x = np.linalg.solve(A, b) # Solving linear eq.

print(I)
print(x)
```

Example in MATLAB:

```
I = eye(3)  % 3x3 identity matrix

A = [2 1 3; 0 2 4; 1 1 2]
b = [1; 2; 3];
x = A\b # Solving linear eq.

disp(I)
disp(x)
```

Loops

Loop allow us to execute a block of code repeatedly based on a condition.

2 types of loops:

- **For loop** → Iterates over a sequence
- **While loop** → Repeats as long as a condition is true

Example in Python:

```
# For loop
for i in range(5):  # Iterates from 0 to 4
    print(i)

# While loop
i = 0
while i < 5:
    print(i)
    i += 1
```

Example in MATLAB:

```
% For loop
for i = 0:5
    disp(i)
end

% While loop
i = 0;
while i < 5
    disp(i)
    i = i + 1;
end
```

Loops

Use case: Calculate the sum of numbers

Example in Python:

```
# Using for loop
total = 0
for i in range(1, 11):
    total += i
print("Sum (for loop):", total)

# Using while loop
i = 0
total = 0
While i < 11:
    total += i
    i += 1
print("Sum (while loop):", total)
```

Example in MATLAB

```
total = 0;
for i = 1:10
    total = total + i;
end
disp(['Sum (for loop): ', num2str(total)])

# Using while loop
i = 0
total = 0
while i < 11
    total = total + i;
    i = i + 1;
end
Disp(['Sum (while loop): ', num2str(total)])
```

Conditionals

Conditionals allow us to execute different blocks of code based on whether a condition is true or false.

Types of conditionals:

- **If statement** → Executes a block of code if a condition is true
- **If-else statement** → Executes one block of code if the condition is true, and another block if it is false
- **If-elif-else statement** → Checks multiple conditions

Example in Python:

```
x = 10
if x > 10:
    print("x is greater than 10")
elif x == 10:
    print("x is equal to 10")
else:
    print("x is less than 10")
```

Example in MATLAB:

```
x = 10;
if x > 10
    disp('x is greater than 10')
elseif x == 10
    disp('x is equal to 10')
else
    disp('x is less than 10')
end
```

Conditionals

Use case: Determine if a number is even or odd

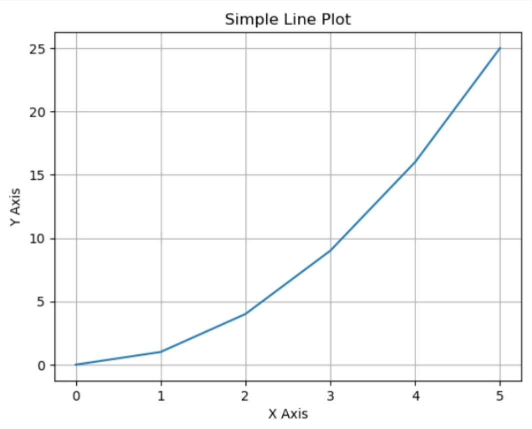
Example in Python:

```python
number = 7
if number % 2 == 0:
    print("Number is even")
else:
    print("Number is odd")
```

Example in MATLAB

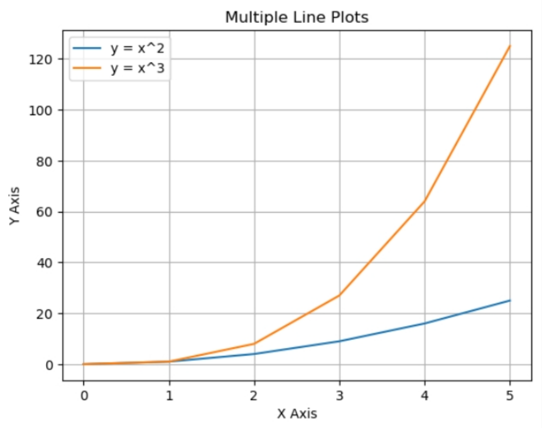
```matlab
number = 7;
if mod(number, 2) == 0
    disp('Number is even')
else
    disp('Number is odd')
end
```

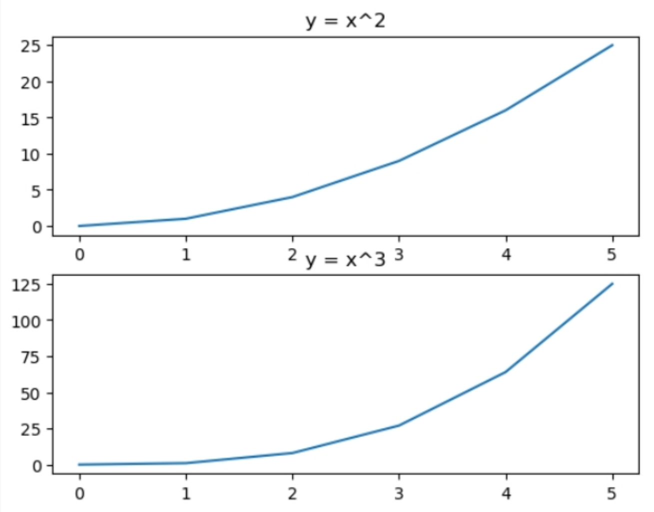
Plotting Graphs



Used for visualizing data, results from simulations, and analyzing system responses in control systems.

Several kind of graphs:





Example: Simple Line Plot

Example in Python:

```python
import matplotlib.pyplot as plt

# Sample data
x = [0, 1, 2, 3, 4, 5]
y = [0, 1, 4, 9, 16, 25]

# Create a line plot
plt.plot(x, y)

# Add labels and title
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Simple Line Plot')

# Show the plot
plt.show()
```

Example in MATLAB:

```matlab
x = 0:5;
y = x.^2;

% Create a line plot
plot(x, y)

% Add labels and title
xlabel('X Axis')
ylabel('Y Axis')
title('Simple Line Plot')

% Display the grid
grid on
```

Example: Multiple Line Plot

Example in Python:

```python
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]
y1 = [0, 1, 4, 9, 16, 25]
y2 = [0, 1, 8, 27, 64, 125]

plt.plot(x, y1, label="y = x^2")
plt.plot(x, y2, label="y = x^3")

plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Multiple Line Plots')
plt.legend()  # Add a legend
plt.grid(True)
plt.show()
```

Example in MATLAB:

```matlab
x = 0:5;
y1 = x.^2;
y2 = x.^3;

plot(x, y1, 'b-', x, y2, 'r--')
xlabel('X Axis')
ylabel('Y Axis')
title('Multiple Line Plots')
legend('y = x^2', 'y = x^3')
grid on
```

Functions

What is functions for?

- To encapsulate code into reusable blocks
- To improve readability, simplify debugging, and avoid code duplication

Functions take inputs, perform operations, and return outputs.

General form in Python:

```
def function_name(parameters):
    # Function body
    return result
```

General form in MATLAB:

```
function output = function_name(inputs)
    % Function body
    output = some_operation(inputs);
end
```

Functions

Example in Python:

```
def square(x):
    return x ** 2

# Using the function
result = square(4)
print(result)  # Output: 16
```

Example in MATLAB:

```
function result = square(x)
    result = x^2;
end

% Using the function
result = square(4);
disp(result)  % Output: 16
```

Note:

- Functions can take more than one parameter
- Function can return multiple values

Lambda Functions

Sometimes, simple functions can be defined without a name.

In this case, we can use:

- **Lambda functions** (Python)
- **Anonymous functions** (MATLAB)

These functions are often used in situations where a full function definition would be unnecessary.

Example in Python:

```python
square = lambda x: x ** 2
print(square(4))  # Output: 16
```

Example in MATLAB:

```matlab
square = @(x) x^2;
disp(square(4))  % Output: 16
```

Assignment 1

1. **Variables and Arithmetic Operations:**
   Write a Python/MATLAB program that calculates the area of a circle given the radius. Use the formula $A = \pi r^2$ with $r = 5$.

2. **Conditional Statements:**
   Write a Python/MATLAB program that asks the user for a number and prints whether it is a positive, negative, or zero. *Hint: use input() function to ask for input from user.*

3. **Loops – For Loop:**
   Write a Python/MATLAB program that calculates the sum of all even numbers between 1 and a number n (you can use any number for n).

Assignment 1

4. **Loops – While Loop:**
   Write a Python/MATLAB program that continuously asks the user for numbers and prints their cumulative sum. The loop should stop when the user enters a negative number.

5. **Matrix Operations:**
   Create two 3×3 matrices and:
   a. Add the matrices
   b. Multiply the matrices
   c. Find the determinant of the second matrix

Assignment 1

6. Functions:

Write a Python/MATLAB function called `time_constants` that takes the damping coefficient ζ and natural frequency ω_n of a second-order system as inputs, and returns the system's settling time t_s and peak time t_p. The formulas are:

$$t_s = \frac{4}{\zeta\omega_n}, \qquad t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}$$

Test the function with $\zeta = 0.5$ and $\omega_n = 2$.

7. Plotting Graphs:

Plot the function $y = \cos(x)$ for x ranging from 0 to 2π using Python or MATLAB. Add gridlines and labels to the plot.

Resources to Study

Course materials: irinamrdhtllh/ik6016-lecture-notes (github.com)

Python

- Python: 3.12.6 Documentation (python.org)
- Numpy: NumPy documentation — NumPy v2.1 Manual
- Matplotlib: Matplotlib documentation — Matplotlib 3.9.2 documentation
- Scipy: SciPy documentation — SciPy v1.14.1 Manual
- Control: Python Control Systems Library — Python Control Systems Library 0.10.1 documentation (python-control.readthedocs.io)

There's only one way to master programming:
Read the documentation, implement, and keep experimenting.