

Chương mở đầu:



HƯỚNG DẪN PYTHON CƠ BẢN

Nội dung của chương.

1. Giới thiệu Python, 5-6
2. Hướng dẫn cài đặt, 7-14
 - 2.1. Địa chỉ tải và cài đặt ứng dụng Python gốc, 7-8
 - 2.2. Cài đặt các gói bổ sung cho Python, 9-10
 - 2.3. Địa chỉ tải và cài đặt ứng dụng VSCode gốc, 11-13
 - 2.4. Cài đặt các phần mở rộng cho VSCode, 14
3. Giao diện và sử dụng cơ bản, 15-24
 - 3.1. Giao diện Python gốc, 15-16
 - 3.2. Giao diện Python trên VSCode, 17
 - 3.3. Giao diện Jupyter trên VSCode, 18
 - 3.4. Cài đặt môi trường làm việc, 19-22
 - 3.4.1. Trả lại mọi cài đặt trên VSCode về mặc định, 19
 - 3.4.2. Gỡ sạch ứng dụng Python, VSCode để cài mới, 20-21
 - 3.4.3. Cài đặt kích thước của chữ trong VSCode, 22
 - 3.5. Các ký tự đặc biệt trong Python và phím tắt trong Jupyter, 23-24
 - 3.5.1. Các ký tự đặc biệt trong Python, 23
 - 3.5.2. Các tổ hợp phím tắt trong Jupyter, 24

Nội dung của chương.

4. Function (hàm) trong Python, 25-42

 4.1. Hàm cơ bản có sẵn trong Python, 25-28

 4.2. Hàm trong các gói cài đặt bổ sung, 29-31

 4.3. Hàm do người dùng tự định nghĩa, tạo ra, 32-33

 4.3.1. Hàm được tạo bằng từ khóa **lambda**, 32

 4.3.2. Hàm được tạo bằng từ khóa **def**, 33

 4.4. Vị trí tạo, và nơi lưu trữ của hàm, 34-42

 4.4.1. Hàm tạo trong chương trình chính, 34

 4.4.2. Hàm tạo trên một file (Module) khác, 35-36

 4.4.3. Hàm tạo trên nhiều file (Module) trong một folder (Package) khác, 37-42

5. Các toán tử trong Python, 43-47

 5.1. Toán tử số học, 43

 5.2. Toán tử gán, 44

 5.3. Toán tử quan hệ, 45

 5.4. Toán tử logic, 46

 5.5. Toán tử tìm kiếm, xác thực, 47

Nội dung của chương.

6. Các mệnh đề, vòng lặp điều khiển, 48-54

 6.1. Điều kiện if, 48-52

 6.1.1. Điều kiện if một nhánh rẽ, 47

 6.1.2. Điều kiện if hai nhánh rẽ, 49

 6.1.3. Điều kiện if nhiều hơn hai nhánh rẽ, 50

 6.1.4. Điều kiện if nhiều nhánh rẽ lồng nhau, 52

 6.2. Vòng lặp for, 53

 6.3. vòng lặp while, 54

7. Sử dụng gói Numpy cơ bản, 55-71

8. Sử dụng gói Matplotlib cơ bản, 72-74

9. Sử dụng gói Sympy cơ bản, 75-76

10. Hàm xuất định dạng bảng trong gói Tabulate, 77

1. Giới thiệu Python.

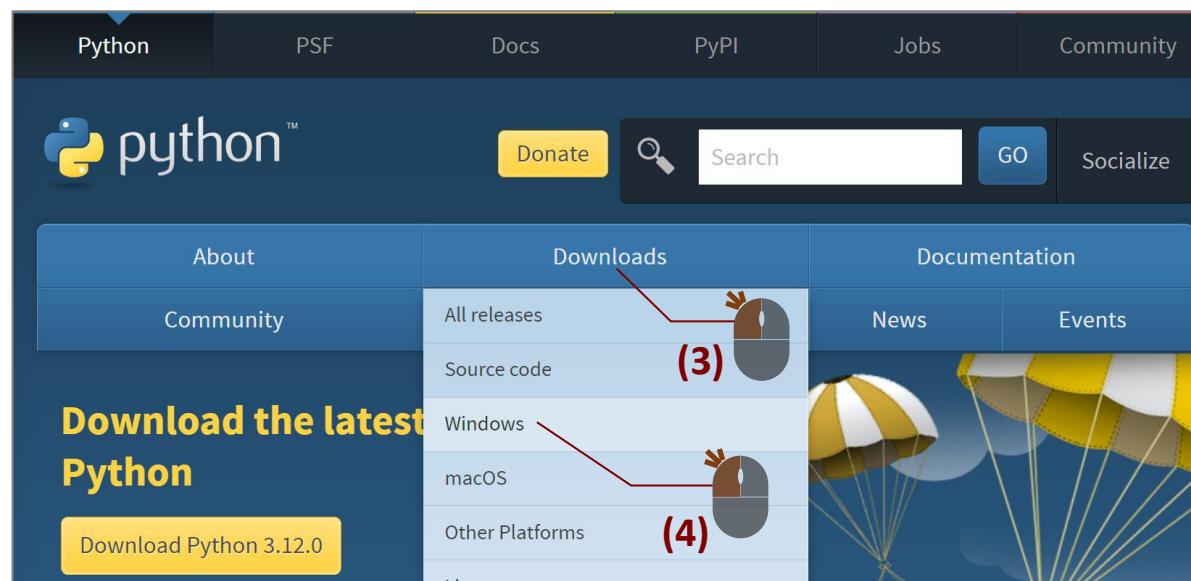
- ✓ Python là một ngôn ngữ lập trình thông dịch (interpreted), hướng đối tượng (object-oriented), và là một ngôn ngữ bậc cao (high-level) ngữ nghĩa động (dynamic semantics).
- ✓ Python hỗ trợ các module và gói (packages), có thể tạo chương trình module hóa và tái sử dụng mã.
- ✓ Trình thông dịch Python và thư viện chuẩn mở rộng có sẵn dưới dạng mã nguồn mở hoặc dạng nhị phân miễn phí cho tất cả các nền tảng chính và có thể được phân phối tự do.

Các điểm nổi bật.

- ✓ Đơn giản và dễ học: Lập trình bằng Python có hình thức sáng sủa, cấu trúc rõ ràng, cú pháp ngắn gọn, có cộng đồng lập trình rất lớn, hệ thống thư viện chuẩn được chia sẻ trên mạng.
- ✓ Là ngôn ngữ mã nguồn mở: Cài đặt Python dùng giấy phép nguồn mở nên được sử dụng và phân tối tự do, ngay cả trong việc thương mại.
- ✓ Là ngôn ngữ có khả năng chạy trên nhiều nền tảng: Python có cho mọi hệ điều hành như Windows, Linux/Unix, Mac, Android...
Với cùng một mã nguồn (Code) sẽ chạy giống nhau trên mọi nền tảng.
- ✓ Dễ dàng kết nối với các ngôn ngữ khác: Python có thể viết, tạo các thư viện để nhúng, liên kết với C, C++, Excel, Autocad... và ngược lại.

2. Hướng dẫn cài đặt.

2.1. Địa chỉ tải và cài đặt phần mềm Python gốc.



Stable Releases

- [Python 3.12.0 - Oct. 2, 2023](#)

Note that Python 3.12.0 cannot be used on Windows

7 or earlier.

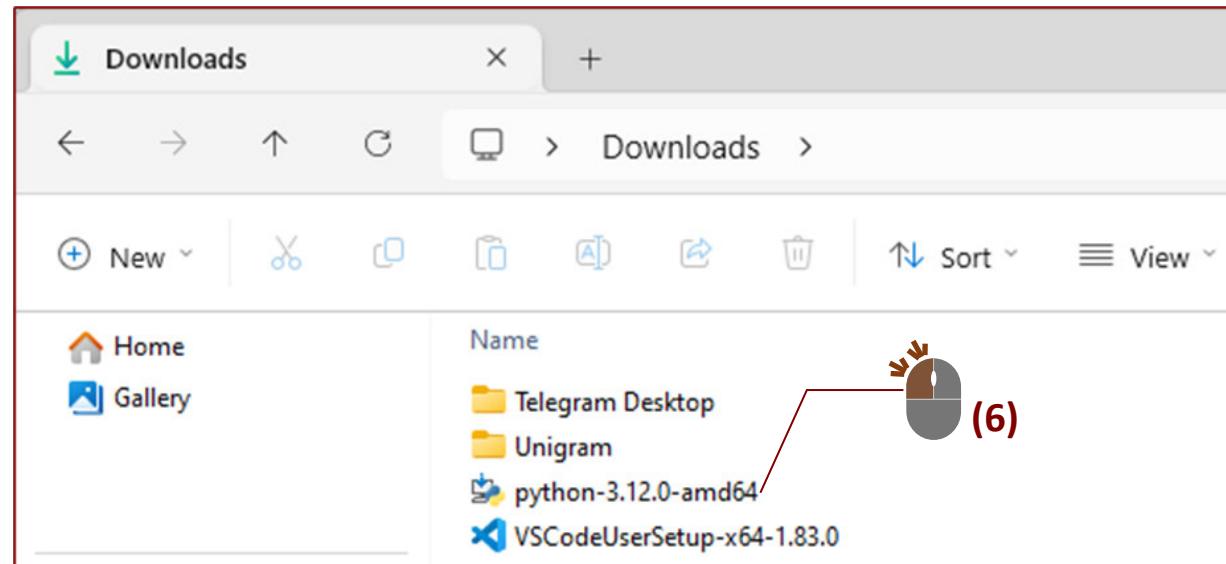
Chọn bản phù hợp với windows định cài

- Download [Windows embeddable package \(32-bit\)](#)
- Download [Windows embeddable package \(64-bit\)](#)
- Download [Windows embeddable package \(ARM64\)](#)
- Download [Windows installer \(32-bit\)](#)
- Download [Windows installer \(64-bit\)](#)
- Download [Windows installer \(ARM64\)](#)

(5)

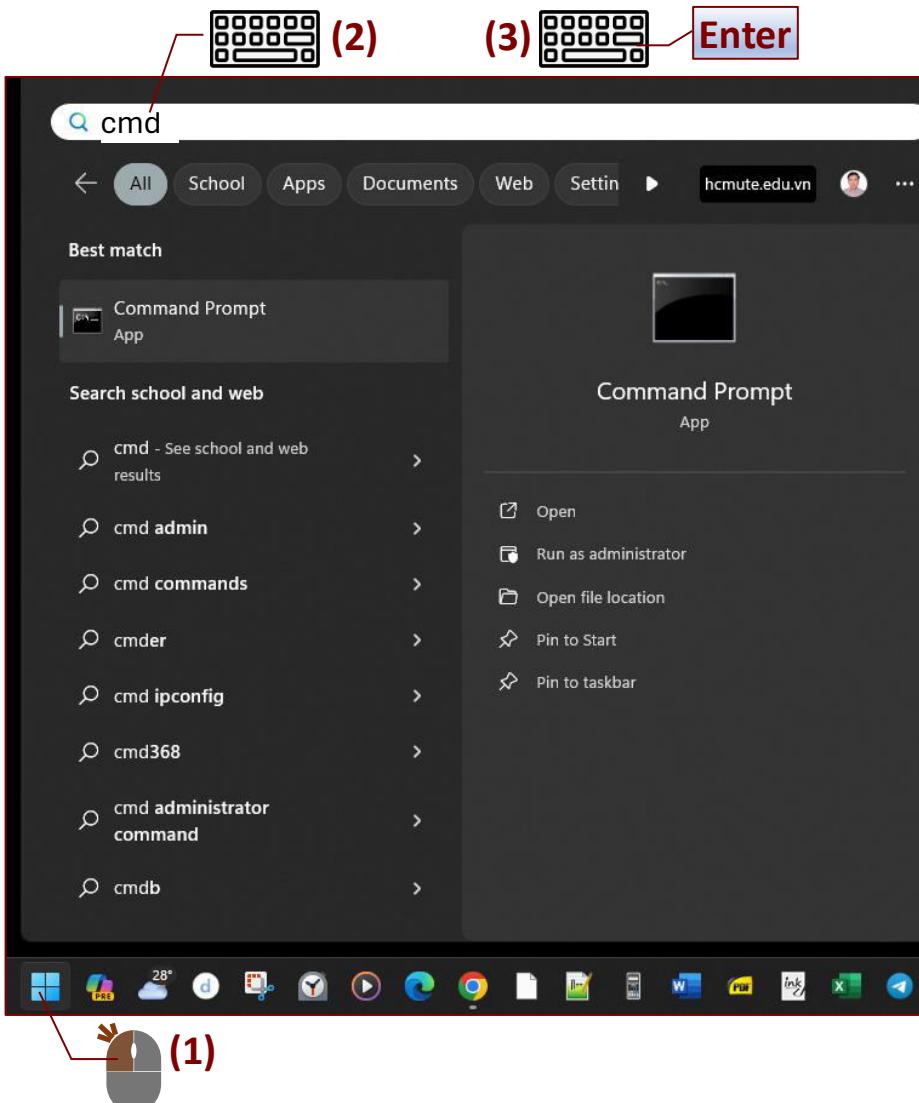
2. Hướng dẫn cài đặt.

2.1. Địa chỉ tải và cài đặt ứng dụng Python gốc.



2. Hướng dẫn cài đặt.

2.2. Cài đặt các gói bổ sung cho Python.



```

C:\ Command Prompt x + v
Microsoft Windows [Version 10.0.22621.2361]
(c) Microsoft Corporation. All rights reserved.

C:\Users\letha> pip install numpy
(4) (5) Enter

pip install matplotlib
(6) (7) Enter

pip install sympy
(8) (9) Enter

pip install tabulate
(10) (11) Enter

python.exe -m pip install --upgrade pip
(12) (13) Enter
  
```

(1), (2), (3): Mở cửa sổ Command Prompt

(4), (5): Cài gói thư viện mảng, đại số Numpy

(6), (7): Cài gói thư viện vẽ Matplotlib

(8), (9): Cài gói thư viện tính toán bằng chữ

(10), (11): Xuất kết quả dạng bảng

(12), (13): Cập nhật phần mềm Python

2. Hướng dẫn cài đặt.

2.2. Cài đặt các gói bổ sung cho Python. (Tiếp theo)

Các gói cài đặt bổ sung (thư viện hàm ứng dụng) rất đa dạng, cho rất nhiều lĩnh vực khoa học khác nhau. Những gói phổ biến trong kỹ thuật bao gồm:

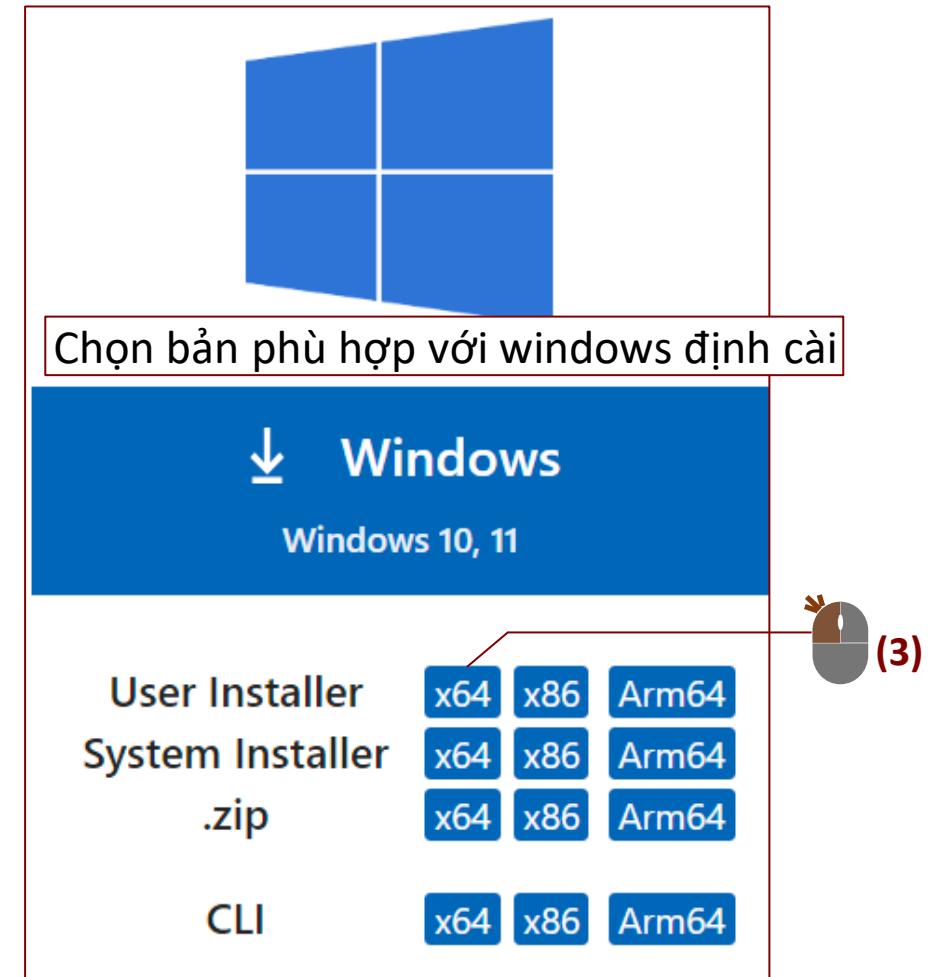
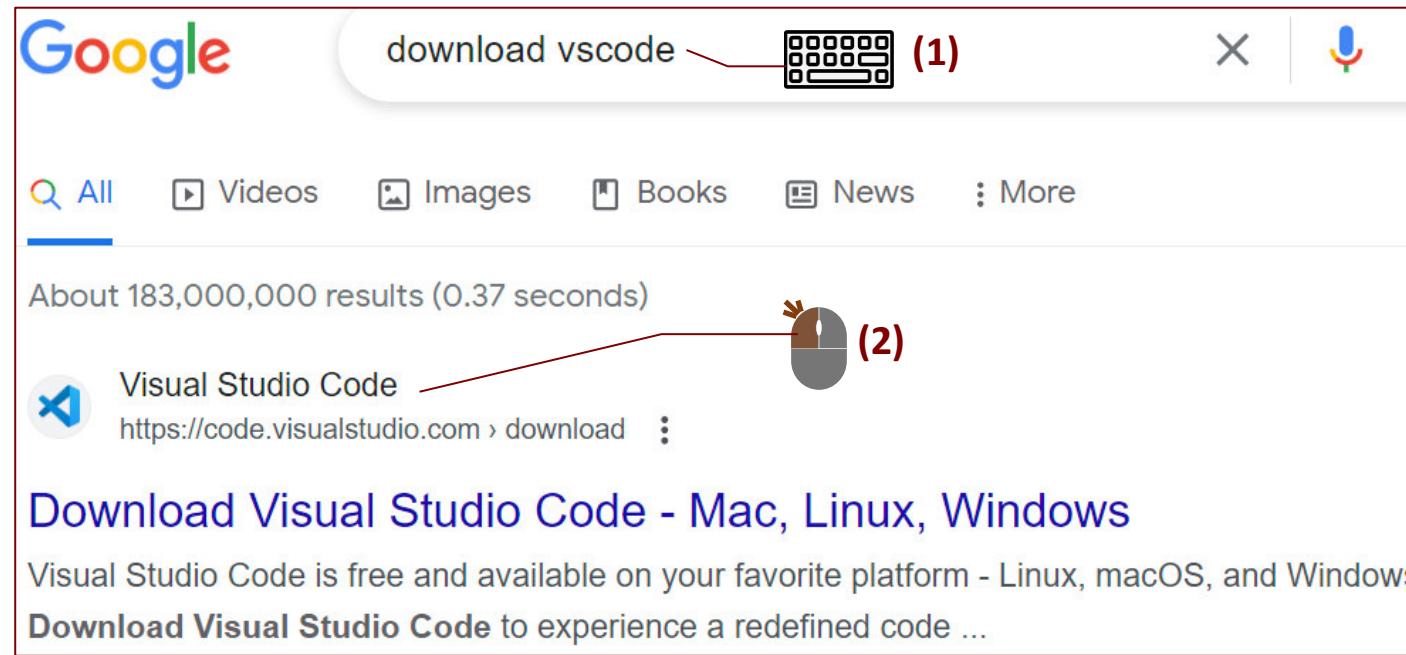
Numpy: (Địa chỉ: <https://numpy.org/>) Là gói cung cấp thư viện với đầy đủ hàm dùng để xử lý các phép toán trên mảng một chiều – véc tơ (vector), mảng hai chiều – ma trận (matrix), và mảng ba chiều. Các phép toán này thường dùng trong đại số tuyến tính và được ứng dụng trong các tính toán kỹ thuật. Cài đặt gói:
pip install numpy

Matplotlib: (Địa chỉ: <https://matplotlib.org/>) Là một trong những gói mở rộng của Python phổ biến nhất được sử dụng để trực quan hóa dữ liệu. Thư viện bao gồm các hàm có thể sử dụng trên đa nền tảng để tạo ra nhiều loại biểu đồ, đồ thị từ dữ liệu trong các mảng. Cài đặt gói:
pip install matplotlib

Sympy: (Địa chỉ: <https://www.sympy.org/>) Là gói mở rộng của Python, cung cấp thư viện bao gồm các hàm dùng để tính toán về số học, đại số, giải tích mà lời giải cho kết quả bằng chữ (symbol). Cài đặt gói:
pip install sympy

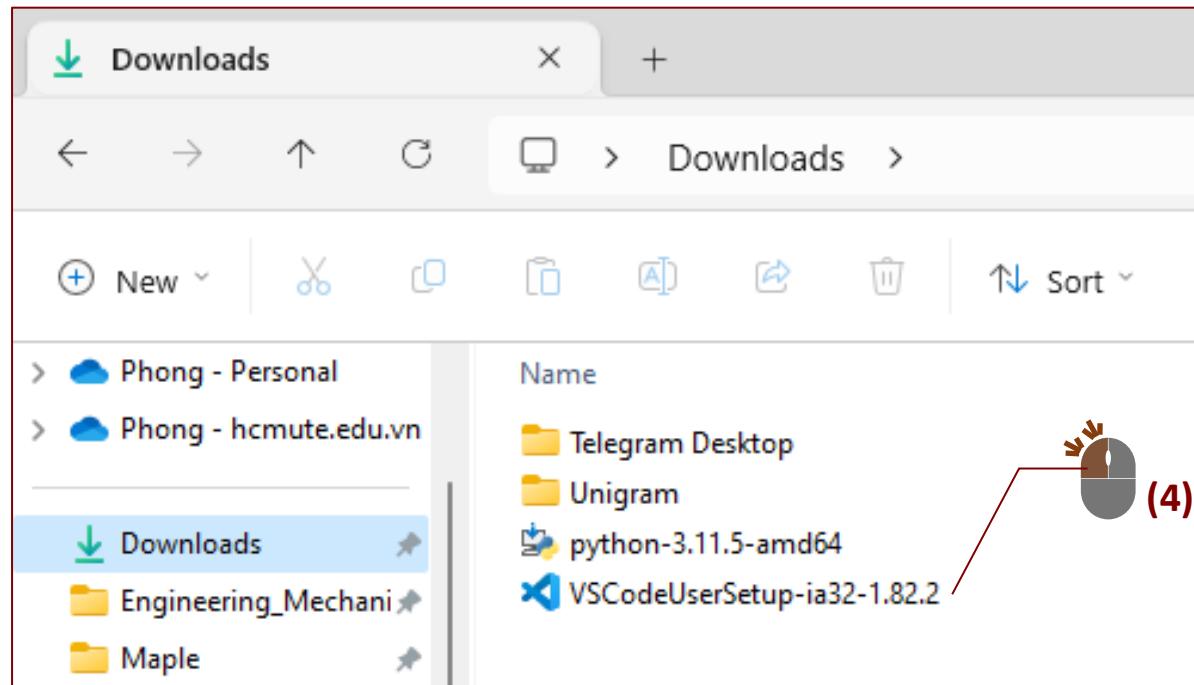
2. Hướng dẫn cài đặt.

2.3. Địa chỉ tải và cài đặt phần mềm VSCode gốc.



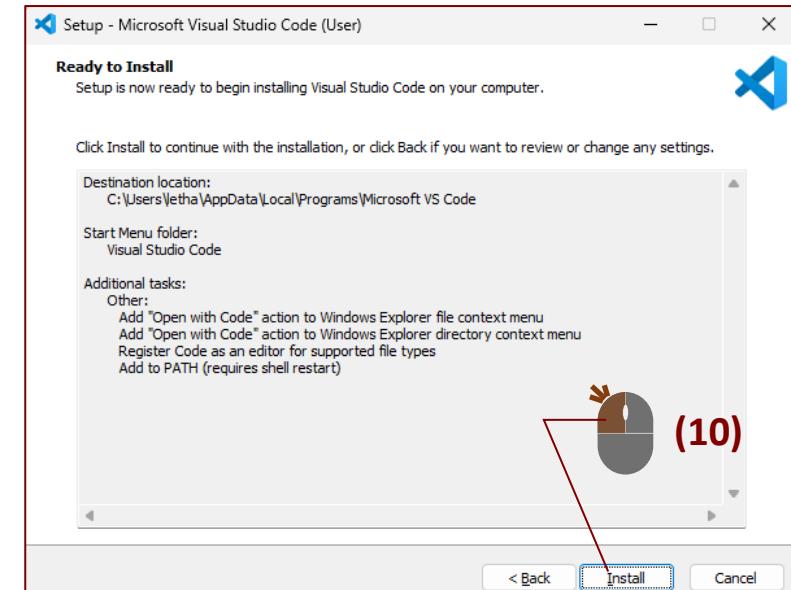
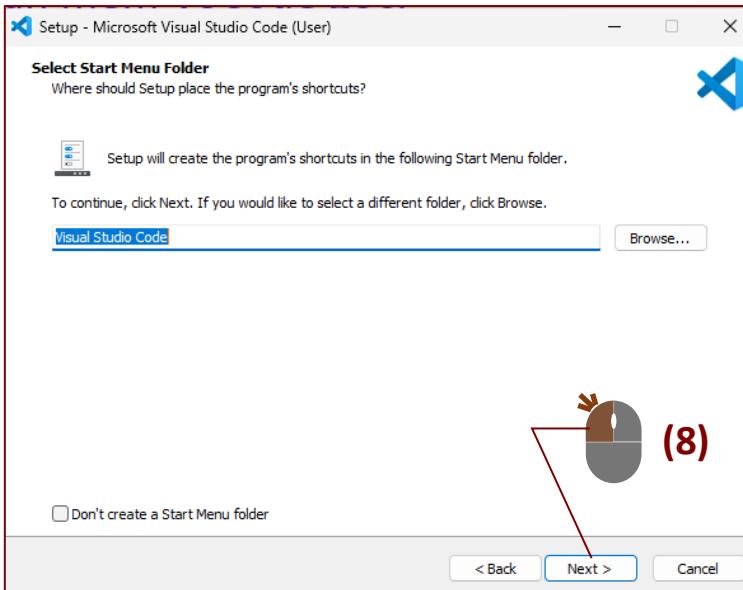
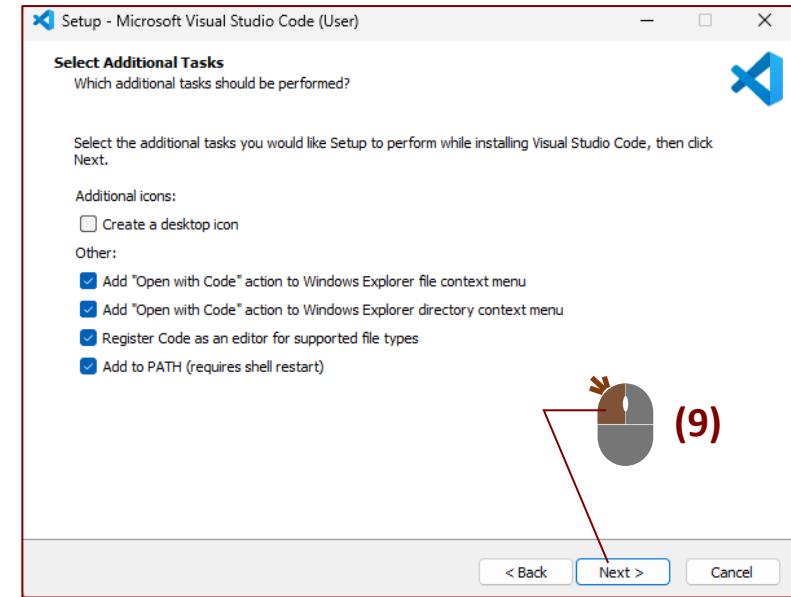
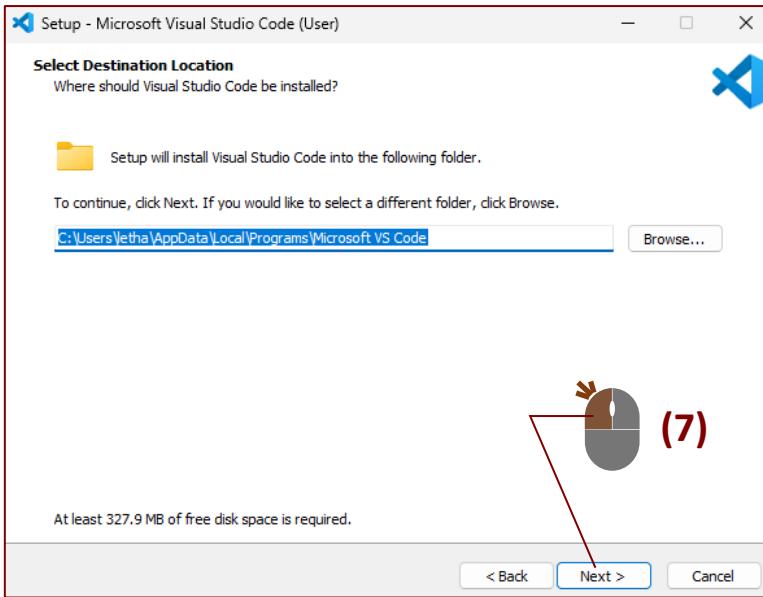
2. Hướng dẫn cài đặt.

2.3. Địa chỉ tải và cài đặt phần mềm VSCode gốc. (Tiếp theo)



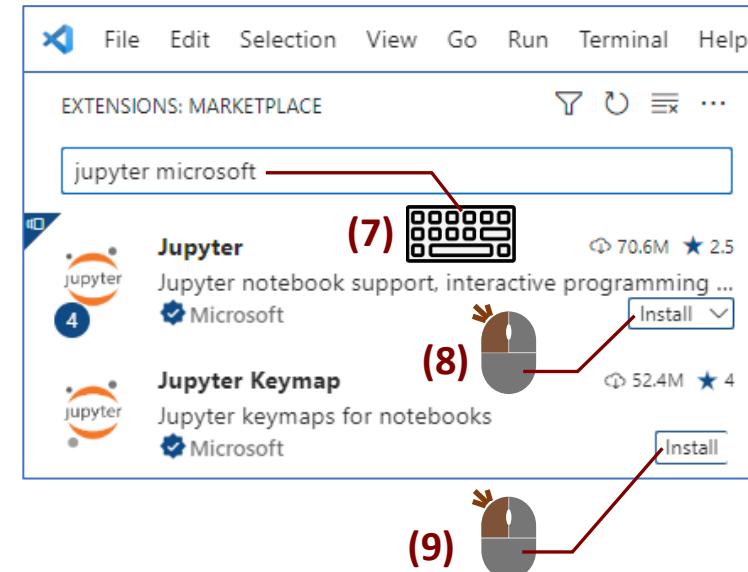
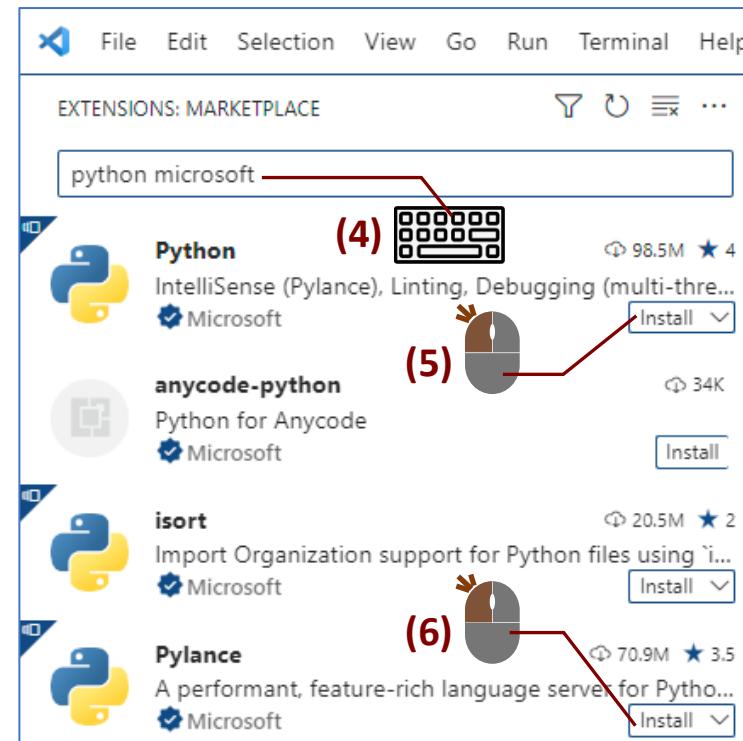
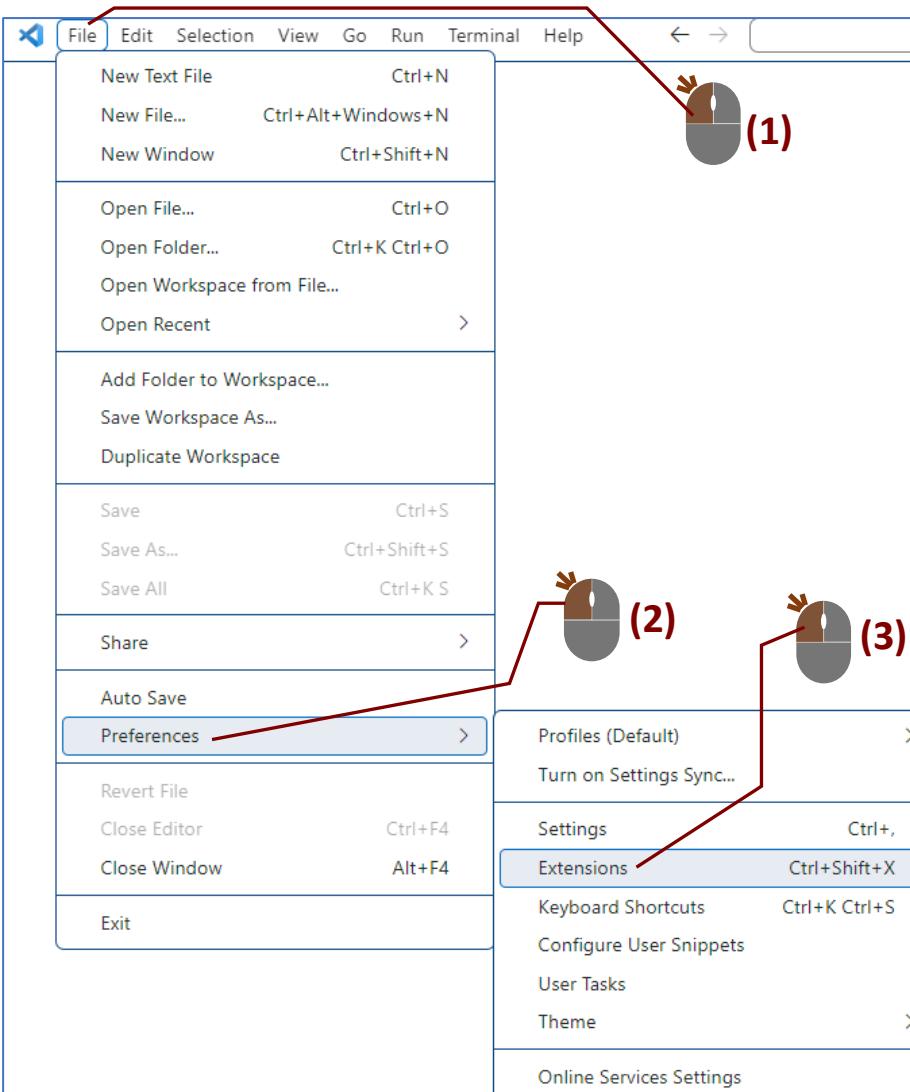
2. Hướng dẫn cài đặt.

2.3. Địa chỉ tải và cài đặt phần mềm VSCode gốc. (Tiếp theo)



2. Hướng dẫn cài đặt.

2.4. Cài đặt các phần mở rộng cho VSCode.

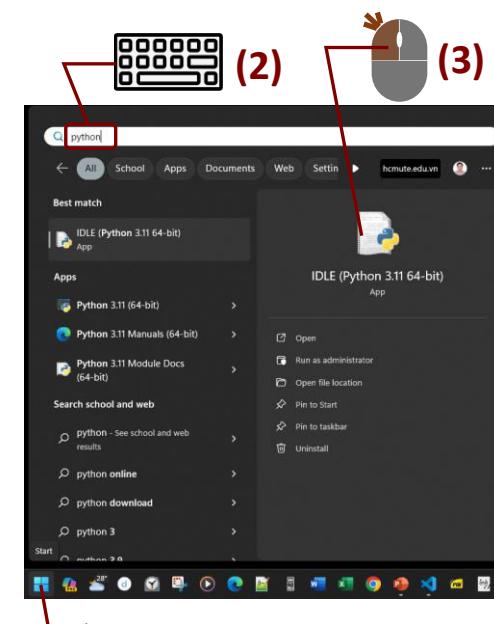


(1), (2), (3), phím tắt **Ctrl + Shift + X**: Vào phần cài đặt mở rộng cho VSCode
 (4), (5), (6): Tìm và cài đặt phần mở rộng Python trên VSCode
 (7), (8), (9): Tìm và cài đặt phần mở rộng Jupyter trên VSCode

3. Giao diện và sử dụng cơ bản.

3.1. Giao diện Python gốc.

3.1.1. Mở ứng dụng Python:



Cửa sổ dòng lệnh đơn và xuất kết quả
(IDLE Sheell)

IDLE Shell 3.10.6

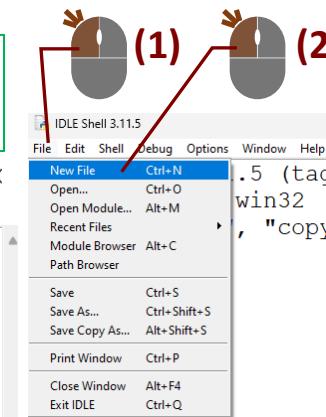
```

Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug 1 2022,
21:53:49) [MSC v.1932 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> 1+2-3*4/5
0.6000000000000001
>>> a=6; b=7
>>> a*b
42
>>> 15/7
2.142857142857143
>>> 15//7
2
>>> 15%7
1
>>> c=a**2+b**3
>>> print('c =',c)
c = 379
>>> 3**1/2
1.5
>>> 3**(1/2)
1.7320508075688772
>>>

```

3.1.2. Mở ứng dụng soạn thảo chương trình:



Phím tắt:
CTrl + N

Cửa sổ soạn thảo lệnh (Edit Window)

MoDau.py - D:\OneDrive\aa_Work\Python\PyCode\TUD\MoDau.py (3.10.6)

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sympy as sp
4 # 1/ Giải hệ phương trình đại số tuyến tính:
5 A=np.array([[2, -3, 5],
6             [4, 7, -2],
7             [-5, 1, 8]])
8 b=np.array([6, 2, 9])
9 x=np.linalg.solve(A,b)
10 print("Nghiệm của hệ A.x=b là: x =",x)
11 # 2/ Vẽ đồ thị hàm số:
12 x=np.linspace(0,10,500)
13 y=np.sin(x)*np.sqrt(5*x**4+x)
14 plt.plot(x,y); plt.show()
15 # 3/ Giải toán bằng chữ: (symbol)
16 a, b, c, x=sp.symbols('a, b, c, x')
17 f=a*x**2+b*x+c
18 x=sp.solve(f,x)
19 print('Nghiệm của phương trình bậc 2 là:',x)
20

```

3. Giao diện và sử dụng cơ bản.

3.1. Giao diện Python gốc. (Tiếp theo)

IDLE Sheel: (Cửa sổ dòng lệnh đơn)

- Là nơi nhập lệnh đơn dòng và chủ yếu dùng để xuất kết quả, nhận các thông báo lỗi khi thực thi lệnh.
- Khi muốn thực thi nhiều lệnh trên một dòng thì sử dụng dấu chấm phẩy “;” để ngăn cách.
- Tại dòng có 3 dấu lớn hơn “>>>” là nơi nhập lệnh. Còn khi xuất kết quả thì sẽ không có 3 dấu này.
- Các câu lệnh nhập trên này không được lưu trữ khi thoát ứng dụng.

Edit Window: (Cửa sổ soạn thảo lệnh)

- Là nơi dùng để soạn thảo chương trình và được lưu lại dưới dạng tập tin (file) có phần mở rộng là .py
- Để thực thi chạy một chương trình: Run → Run Module hoặc phím tắt **F5**
- Khi thoát ứng dụng thì tập tin chương trình vẫn được lưu lại trên máy tính có thể sử dụng, chỉnh sửa cho lần sau.

3. Giao diện và sử dụng cơ bản.

3.2. Giao diện Python trên VSCode.

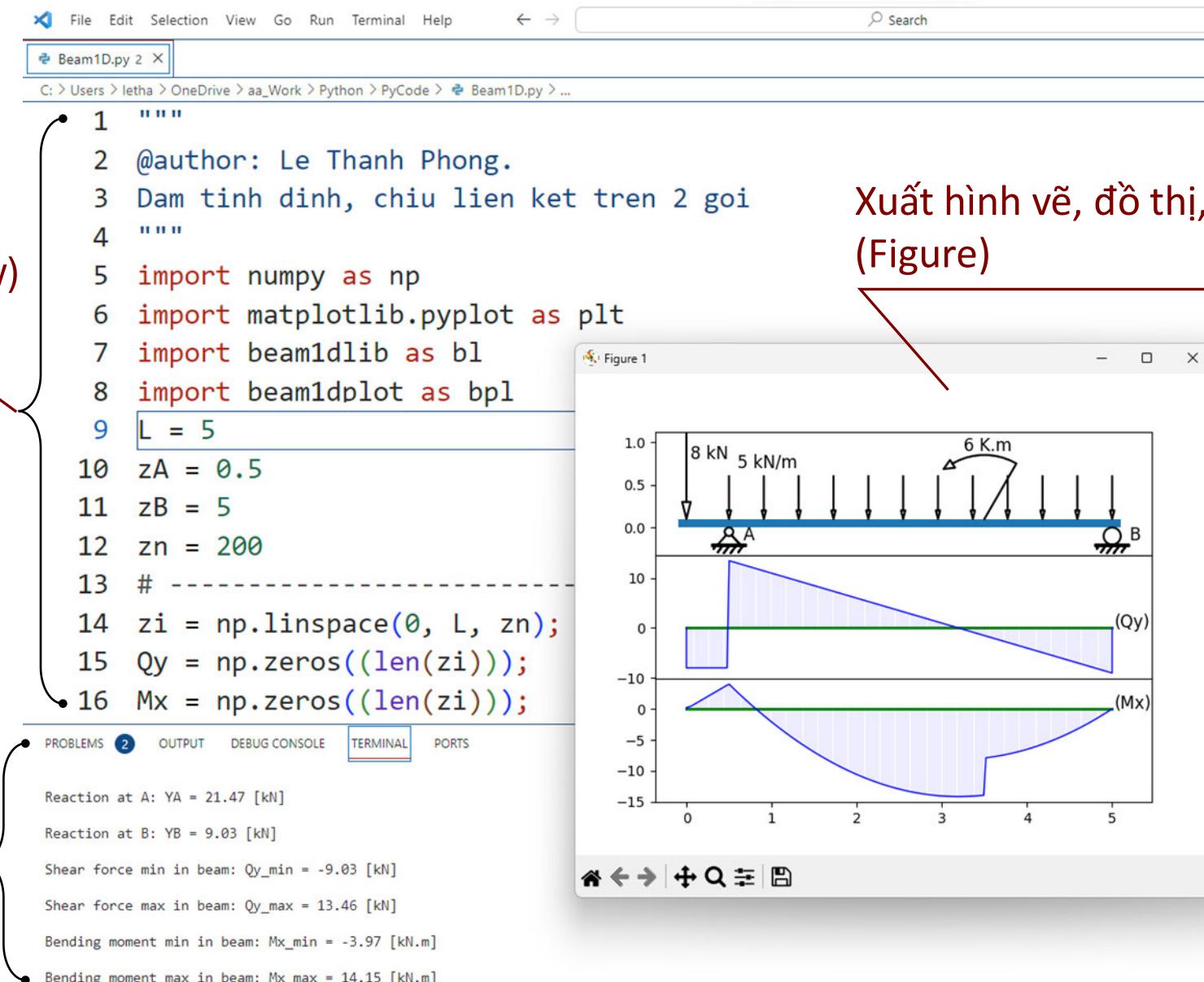
Chạy chương trình:

Run → Start Debugging

Phím tắt: **F5**

Cửa sổ soạn thảo lệnh,
chương trình (Edit Window)

Cửa sổ xuất kết quả
(Terminal Output)



The screenshot shows the VSCode interface with the following components:

- Edit Window:** Displays the Python code for a beam analysis. The code imports numpy, matplotlib.pyplot, beam1dlib, and beam1dplot. It defines constants L=5, zA=0.5, zB=5, and zn=200. It then creates arrays for reaction forces, shear force, and bending moment over the beam's length.
- Terminal Output:** Shows the results of the script execution, including reaction forces at A and B, and minimum and maximum values for shear force (Qy) and bending moment (Mx).
- Figure:** A plot titled "Figure 1" showing a beam of length 5 units. The beam is fixed at A (0,0) and B (5,0). It is subjected to a downward reaction of 8 kN at A, a downward distributed load of 5 kN/m from A to 3, a clockwise moment of 6 K.m at the midpoint (2.5), and a downward reaction of 9.03 kN at B. Below the beam, two graphs are plotted: the Shear Force Diagram (Qy) and the Bending Moment Diagram (Mx). The Qy graph shows a constant negative value of -9.03 kN from A to 3, a linear increase from -9.03 to 13.46 kN from 3 to 4, and a constant positive value of 13.46 kN from 4 to B. The Mx graph shows a parabolic curve starting at 0 at A, reaching a minimum of -3.97 kN.m at x=3, and increasing to 14.15 kN.m at B.

Xuất hình vẽ, đồ thị, biểu đồ...
(Figure)

3. Giao diện và sử dụng cơ bản.

3.3. Giao diện Jupyter trên VSCode.

Chạy chương trình
trong tất cả các
cells trên file

Chạy chương trình
trong cells này

Hoặc đưa con trỏ vào
trong cells muốn chạy
và nhấn tổ hợp phím:

Ctrl + Enter

Bài 4.4: Sử dụng phương pháp tiếp tuyến (Newton - Raphson), tìm nghiệm của phương trình:
 $f(x) = 2 + 4 \cos \frac{2-x}{0.1+x} - (3-x)\sqrt{5x-x^2}$
Với sai số: $err = 10^{-5}$ và nghiệm đề nghị ban đầu: $x_0 = 0.2$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 f = lambda x: 2+4*np.cos((2-x)/(0.1+x))-(3-x)*np.sqrt(5*x-x**2)
4 df = lambda x: (f(x+h)-f(x-h))/2/h; h = 1e-3
5 err, x0, i = 1e-5, 0.2, 0
6 while abs(f(x0)) > err:
7     x1 = x0-f(x0)/df(x0); x0 = x1; i += 1
8     if i > 1000: print('Lặp đến 1000 lần vẫn chưa tìm ra nghiệm'); break
9 print('Nghiệm x =', round(x0, 4), 'sau', i, 'lần lặp.')
10 xv = np.linspace(0.2, 1, 500); yv = f(xv)
11 plt.plot(xv, yv, x0, f(x0), 'bo'); plt.grid(); plt.show()

```

Nghiệm x = 0.252 sau 4 lần lặp.

Ô (Cells) soạn văn bản -
công thức trong môi
trường **MarkDown**

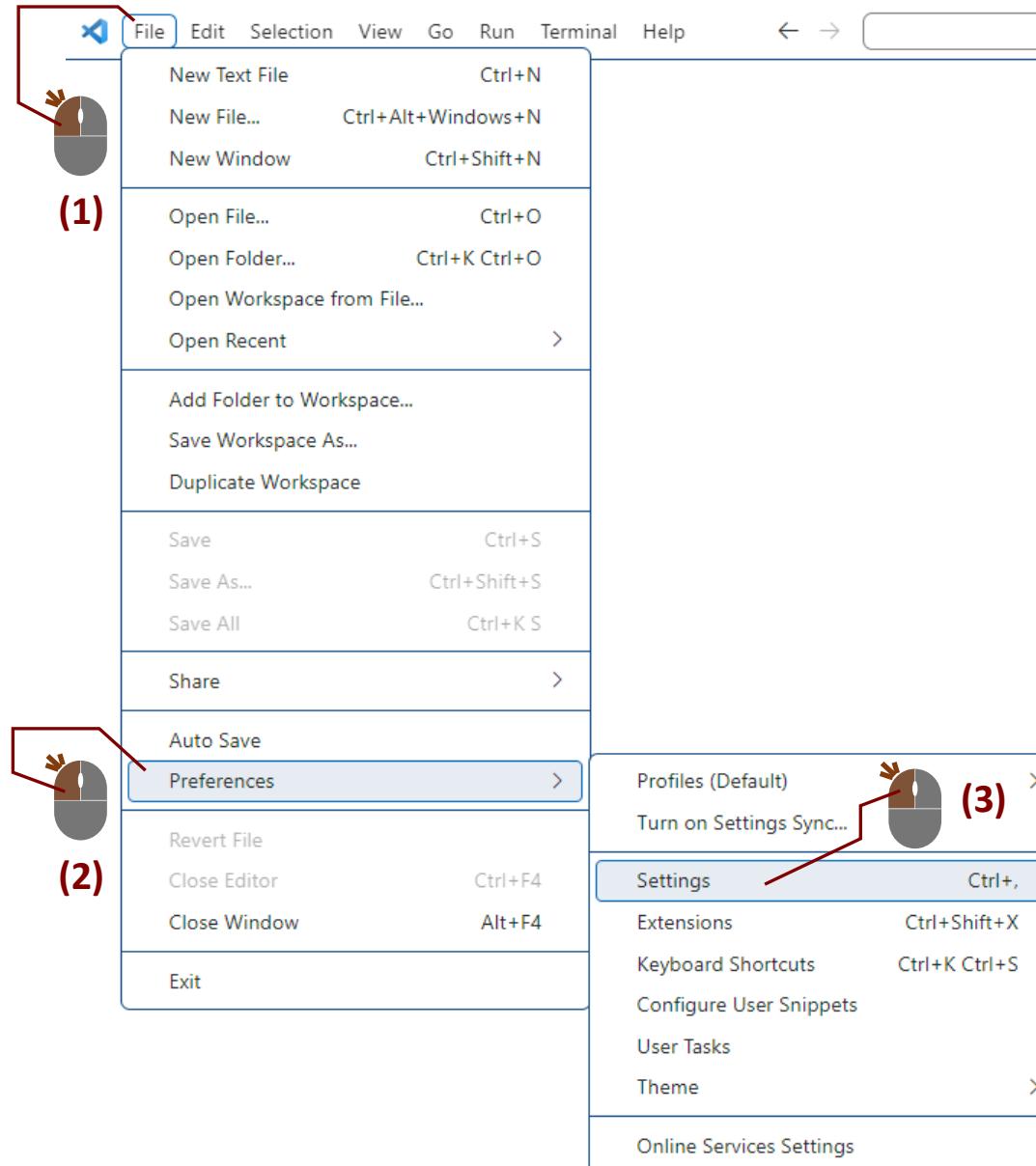
Ô (Cells) viết chương trình
(Code) trong môi trường
Jupyter - Python

Vùng xuất kết quả (Output)

Xuất kết quả đồ thị

3. Giao diện và sử dụng cơ bản.

3.4.1. Trả lại mọi cài đặt trên VSCode về mặc định.



The screenshot shows the 'settings.json' file in the VSCode editor. A red box labeled (4) highlights the file tab. A red box labeled (5) highlights the following JSON code:

```
1 {  
2   "workbench.statusBar.visible": false,  
3   "workbench.activityBar.visible": false,  
4   "workbench.colorTheme": "Default High Contrast Light",  
5   "security.workspace.trust.untrustedFiles": "open",  
6   "editor.fontSize": 24,  
7   "notebook.markup.fontSize": 24,  
8   "notebook.output.fontSize": 24,  
9   "editor.minimap.enabled": false  
10 }
```

The screenshot shows the 'settings.json' file in the VSCode editor. A red box labeled (6) highlights the entire file content, which is currently empty.

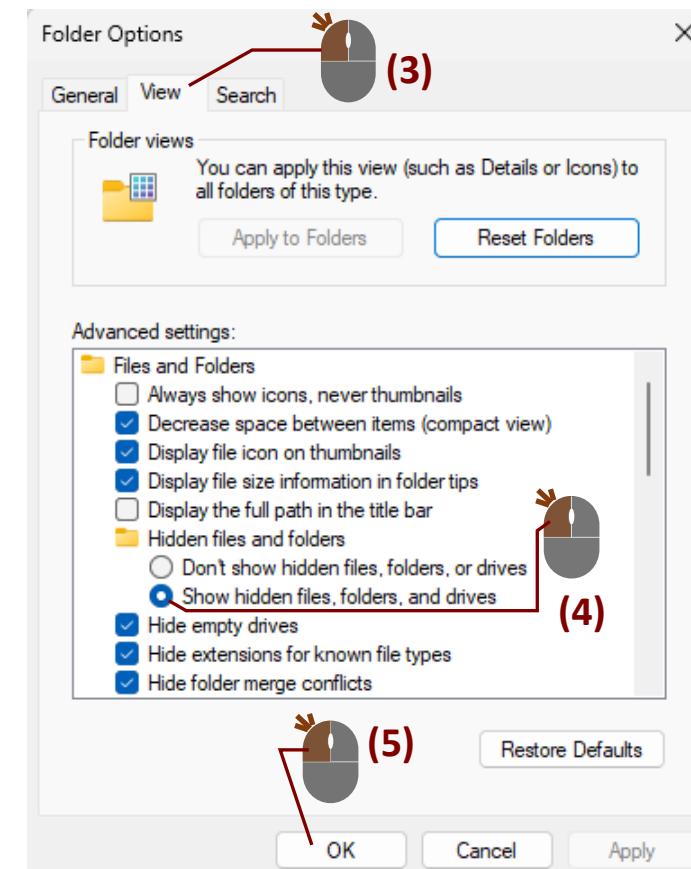
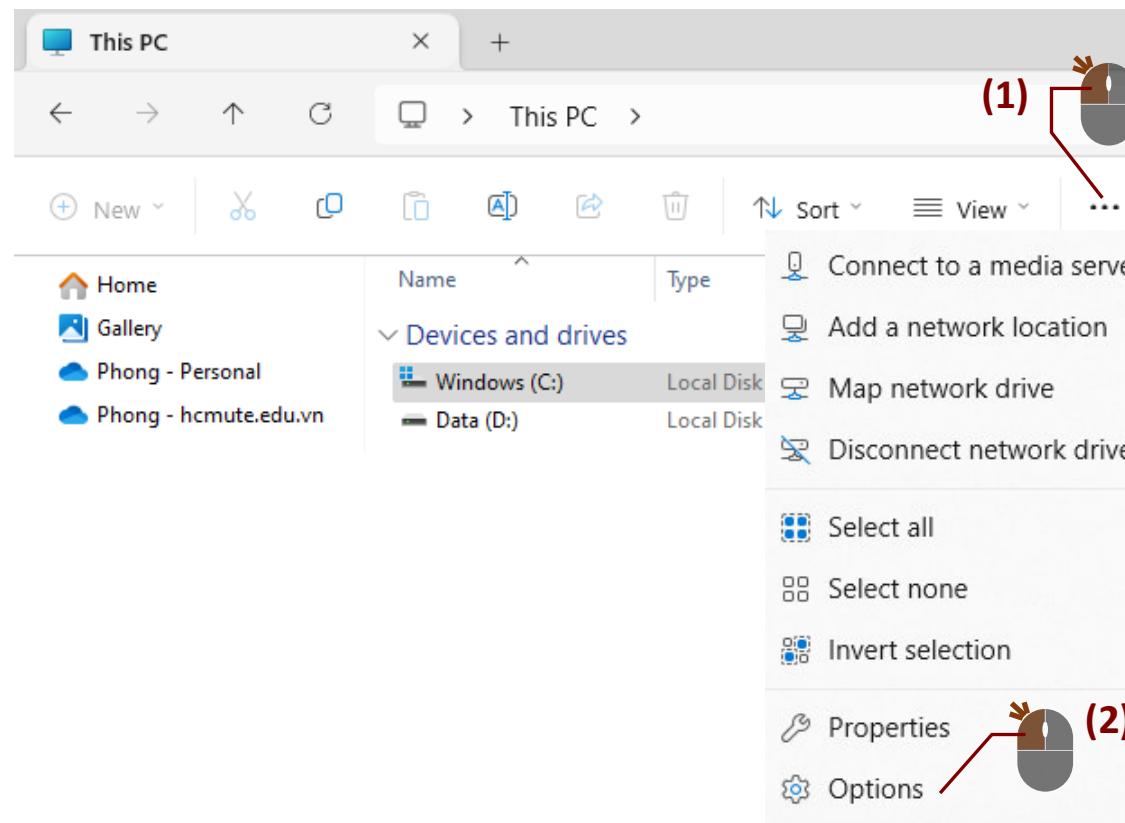
(1), (2), (3), phím tắt **Ctrl** + **,** : Vào phần cài đặt
(4): Mở file đã lưu lại cài đặt (setting) của VSCode
(5), (6): Xóa toàn bộ nội dung lưu những cài đặt trên VSCode

3. Giao diện và sử dụng cơ bản.

3.4.2. Gỡ sạch ứng dụng Python, VSCode để cài mới.

Nhằm gỡ sạch ứng dụng và các gói cài bổ sung, phần mở rộng để cài lại Python hay VSCode mới thì trình tự tiến hành các bước sau:

- 1/ Gỡ cài đặt bình thường (Uninstall) ứng dụng tương ứng.
- 2/ Bật chế độ hiển thị các files, folders đang ẩn:



3. Giao diện và sử dụng cơ bản.

3.4.2. Gỡ sạch ứng dụng Python, VSCode để cài mới. (Tiếp theo)

3/ Tìm đến và xóa các **thư mục** sau. Lưu ý: **Name** là tên của user trên máy tính cần xóa.

Xóa sạch Python:

C:\Users\Name\.ipython

C:\Users\Name\.matplotlib

C:\Users\Name\AppData\Local\pip

C:\Users\Name\AppData\Local\Programs\Python

C:\Users\Name\AppData\Roaming\Python

Xóa sạch VSCode:

C:\Users\Name\.vscode

C:\Users\Name\.jupyter

C:\Users\Name\.templateengine

C:\Users\Name\AppData\Local\Programs\Microsoft VS Code

C:\Users\Name\AppData\Roaming\Code

C:\Users\Name\AppData\Roaming\jupyter

4/ Khởi động lại máy tính.

5/ Cài đặt lại ứng dụng.

3. Giao diện và sử dụng cơ bản.

3.4.3 Cài đặt kích thước của chữ trong VSCode.

(1) Nhấn tổ hợp phím: **Ctrl + ,**

Bài 4.6:

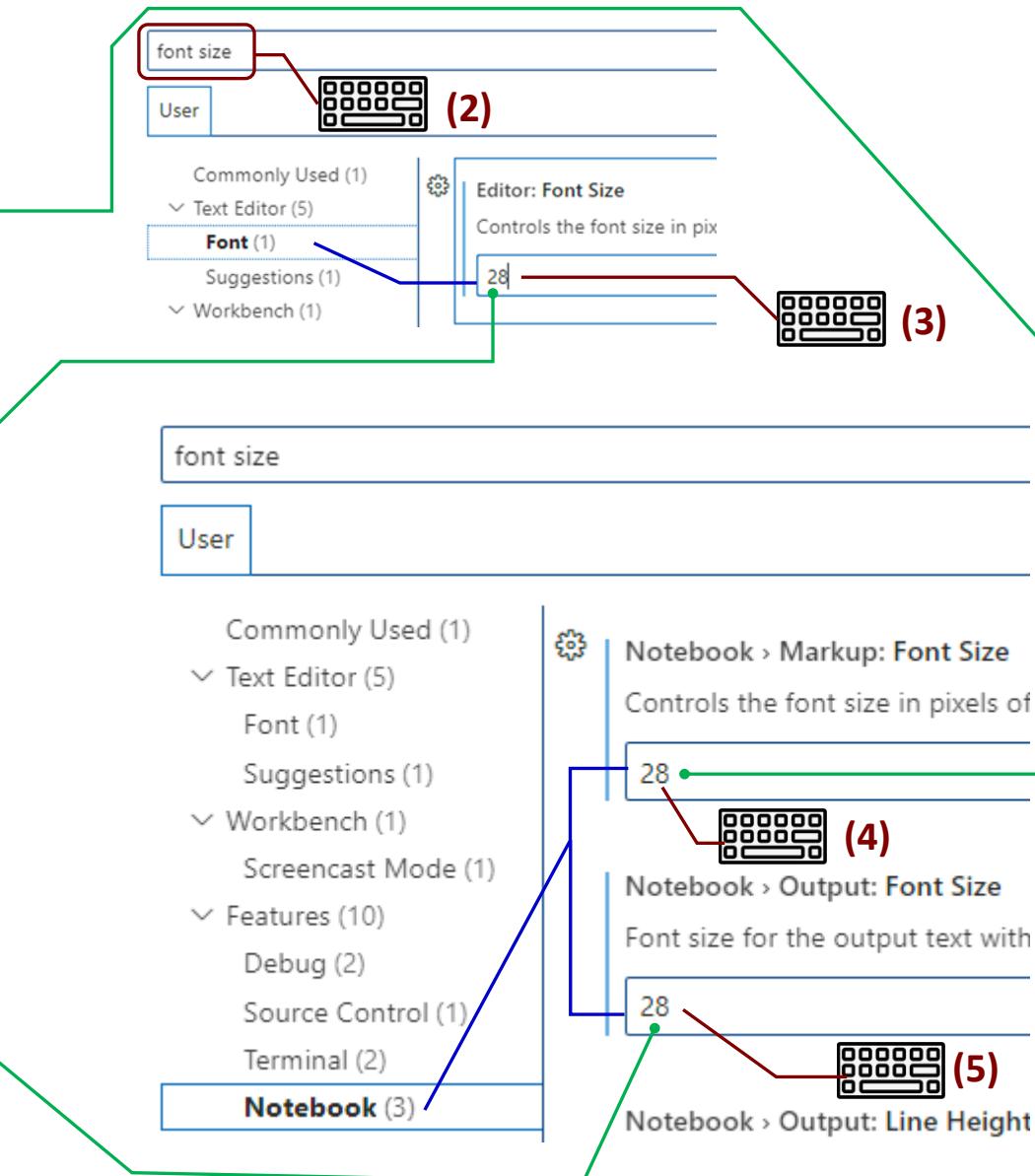
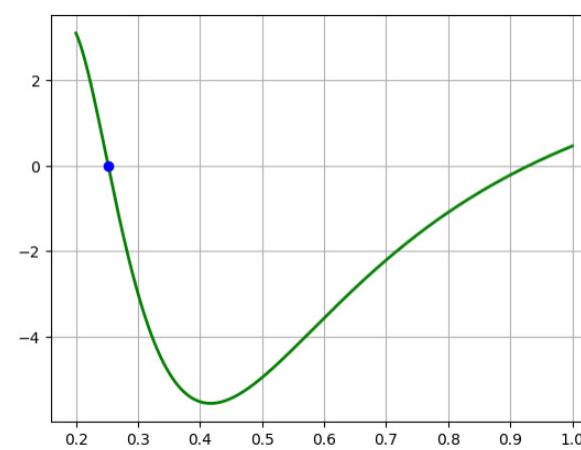
Sử dụng phương pháp cát tuyến (Secant), tìm nghiệm của phương trình:

$$f(x) = 2 + 4 \times \cos \frac{2-x}{0.1+x} - (3-x) \times \sqrt{5x - x^2}$$

Với sai số: $err = 10^{-5}$ và hai nghiệm đề nghị ban đầu: $x_0 = 0.2$; $x_1 = 0.4$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sys
4 f = lambda x: 2+4*np.cos((2-x)/(0.1+x))-(3-x)*np.sqrt(5*x-x**2)
5 err = 1e-5; x0 = 0.2; x1 = 0.4; i = 0
6 while abs(f(x1)) > err:
7     x2 = x1-f(x1)*(x1-x0)/(f(x1)-f(x0))
8     x0 = x1; x1 = x2; i += 1
9     if i > 1000: break; sys.exit('Lặp đến 1000 lần vẫn chưa tìm ra nghiệm')
10 print('Nghiệm x =', round(x1, 4), 'sau', i, 'lần lặp.')
11 xv = np.linspace(0.2, 1, 500); yv = f(xv)
12 plt.plot(xv, yv, 'g', x1, f(x1), 'bo', lw=2); plt.grid(); plt.show()
✓ 1.0s
```

... Nghiệm $x = 0.252$ sau 5 lần lặp.



3. Giao diện và sử dụng cơ bản.

3.5. Các ký tự đặc biệt trong Python và phím tắt trong Jupyter.

3.5.1. Các ký tự đặc biệt trong Python.

- Trong Python khi viết chữ **hoa** khác với chữ **thường**, do đó cần lưu ý khi đặt tên biến.

- Ghi các chú thích trong một khối:

....

Tiêu đề, mô tả...

Trong môi trường Python
xem trong KHỐI này là chú
thích, không biên dịch

....

...

Tiêu đề, mô tả...

Trong môi trường Python
xem trong KHỐI này là chú
thích, không biên dịch

...

- Ghi một chú thích cho dòng lệnh sau dấu thăng **#**: `x = (3 + 9)/4 # Chú thích cho dòng lệnh`

- Viết nhiều dòng lệnh trên một dòng trình bày thì ngăn cách bằng dấu chấm phẩy: `x = 4; y = 7; z = 5`

- Ngắt dòng lệnh (quá dài) thành nhiều dòng trình bày bằng ký tự **** trước khi ngắt:

`b = 5; c = 8; h = 12`

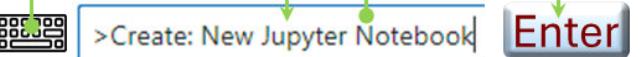
`Iy = (2 * b * h ^ 3 + (2 * b + 3 * c) ^ 2 \`
`* b * h + 3 * c * h ^ 3) / 18`

- Gán một lần cho nhiều biến: `x, y, z = 3, 8, 6`

3. Giao diện và sử dụng cơ bản.

3.5. Các ký tự đặc biệt trong Python và phím tắt trong Jupyter.

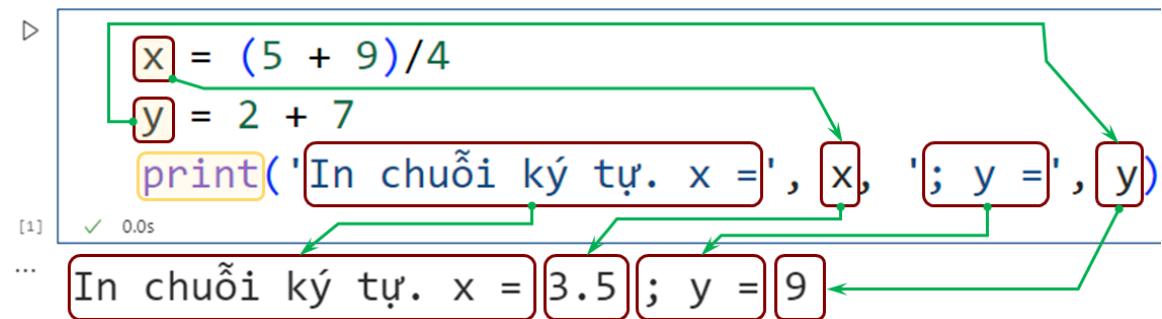
3.5.2. Các tổ hợp phím tắt trong Jupyter.

Ctrl + Shift + P	Hiện cửa sổ dòng lệnh và tất cả các lệnh trên VSCode
	Tạo một file Jupyter Notebook mới
Ctrl + Enter	Chạy Code, chạy MarkDown (Khi con trỏ nằm trong Code hay MarkDown đó)
Ctrl + ,	Mở cửa sổ cài đặt Setting
Ctrl + K → T	Mở cửa sổ dòng lệnh lựa chọn cài đặt giao diện Theme
Ctrl + Shift + X	Mở cửa sổ cài đặt, điều khiển phần mở rộng Extensions
Shift + L	(Hiện / Ẩn) đánh số hàng trên Code, Markdown (Con trỏ không nằm trong Cell)

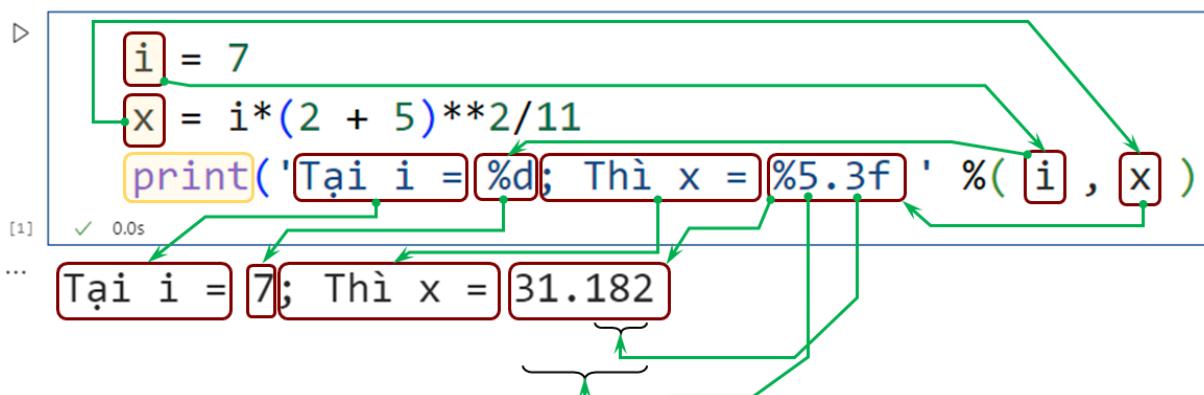
4. Function (hàm) trong Python.

4.1. Hàm cơ bản có sẵn trong Python.

- Hàm **print** trong Python được sử dụng rất nhiều để xuất, trả kết quả, chuỗi, dòng ký tự...

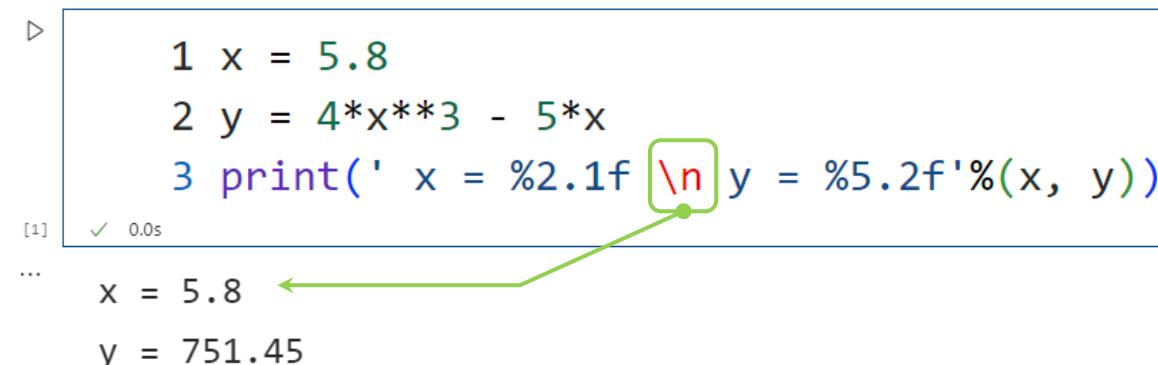


- Định dạng và giữ chỗ của kết quả xen lẫn trong chuỗi (dòng text)



%d : Xất kết quả theo kiểu số nguyên
%T.Sf : Xất kết quả theo kiểu số thập phân,
T là Tổng của số chữ số muốn xuất, **S** là
tổng số chữ số muốn giữ lại sau dấu phẩy

- Kết quả xuống dòng sau cặp ký tự **\n** trong chuỗi (dòng text)



4. Function (hàm) trong Python.

4.1. Hàm cơ bản có sẵn trong Python. (Tiếp theo)

- Hàm **round** dùng để **làm tròn số**, tham số thứ nhất – số muốn làm tròn, tham số thứ 2 – số chữ số muốn giữ lại sau dấu phẩy.

```
1 a = 1.2345; print('In giá trị: a =', a)
2 b = round(a, 1); print('In kết quả làm tròn: b =', b)
[1] ✓ 0.0s
```

... In giá trị: a = 1.2345
In kết quả làm tròn: b = 1.2

- Hàm **abs** dùng để lấy giá trị tuyệt đối của một số.

```
1 c = -2.435; print('In giá trị: c =', c)
2 d = abs(c); print('In kết quả lấy trị tuyệt đối: d =', d)
[2] ✓ 0.0s
```

... In giá trị: c = -2.435
In kết quả lấy trị tuyệt đối: d = 2.435

4. Function (hàm) trong Python.

4.1. Hàm cơ bản có sẵn trong Python. (Tiếp theo)

- Hàm **range** dùng để tạo một tập hợp số có cấu trúc.

```

stop = 4
x = range(stop)
print('x =', x)
print('x =', x[0], x[1], x[2], x[3])
[1] ✓ 0.0s
...
x = range(0, 4)
x = 0 1 2 3

```

• Tạo mảng có: **stop** phần tử:

- Phần tử đầu tiên thứ **0** bằng **0** (mặc định)
- Phần tử thứ cuối cùng (**stop – 1**) bằng **stop – 1**
- Bước (**step**) bằng **1** (mặc định)

```

start, stop = 5, 9
x = range(start, stop)
print('x =', x)
print('x =', x[0], x[1], x[2], x[3])
[1] ✓ 0.0s
...
x = range(5, 9)
x = 5 6 7 8

```

• Tạo mảng có: **stop – start** phần tử:

- Phần tử đầu tiên thứ **0** bằng **start**
- Phần tử thứ cuối cùng (**stop – start – 1**) bằng **stop – 1**
- Bước (**step**) bằng **1** (mặc định)

```

start, stop, step = 2, 14, 3
x = range(start, stop, step)
print('x =', x)
print('x =', x[0], x[1], x[2], x[3])
[1] ✓ 0.0s
...
x = range(2, 14, 3)
x = 2 5 8 11

```

• Tạo mảng có: **n** phần tử:

- Phần tử đầu tiên thứ **0** bằng **start**
- Bước bằng **step**
- Phần tử thứ cuối cùng (**n**) bằng **start + (n – 1)*step <= stop – 1**

$$\text{start} + 0 * \text{step} \quad \text{start} + 1 * \text{step} \quad \text{start} + 2 * \text{step} \quad \dots \quad \text{start} + (n - 1) * \text{step}$$

4. Function (hàm) trong Python.

4.1. Hàm cơ bản có sẵn trong Python. (Tiếp theo)

- Hàm **range** thường kết hợp với vòng lặp **For** và gói **Numpy** để tạo mảng trong Python.

```

import numpy as np
stop = 4
Index = []; X = []
for i in range(stop):
    Index = np.append(Index, i)
    Tinh = i * 4
    X = np.append(X, Tinh)
print('Chỉ số của phần tử =', Index)
print('Mảng X =', X)

```

[1] ✓ 0.1s

... Chỉ số của phần tử = [0. 1. 2. 3.]
Mảng X = [0. 4. 8. 12.]

i lần lượt nhận các giá trị:

0 1 2 ... stop – 1

```

import numpy as np
start, stop = 4, 9
Index = []; X = []
for i in range(start, stop):
    Index = np.append(Index, i)
    Tinh = i * 2.6
    X = np.append(X, Tinh)
print('Chỉ số của phần tử =', Index)
print('Mảng X =', X)

```

[1] ✓ 0.0s

... Chỉ số của phần tử = [4. 5. 6. 7. 8.]
Mảng X = [10.4 13. 15.6 18.2 20.8]

i lần lượt nhận các giá trị:

start start + 1 start + 2 ... stop – 1

4. Function (hàm) trong Python.

4.2. Hàm trong các gói cài đặt bổ sung.

- Để sử dụng được thư viện hàm có trong các package (gói), module (mô-đun) đã cài đặt thông qua lệnh pip thì phải nạp (**import**) package, module đó trước dòng lệnh sử dụng hàm.
- Khi sử dụng hàm thì phải có **tên gói** phía trước, ngăn cách **dấu chấm** rồi đến **tên hàm**.

```
1 import numpy  
2 x = numpy.array([1, 2, 3, 4])  
3 print('x =', x)  
[1] ✓ 0.0s  
... x = [1 2 3 4]
```

Nạp gói thư viện numpy vào chương trình

Sử dụng hàm array (tạo mảng) trong gói numpy

- Thường, tên gói được rút gọn để tiện cho việc sử dụng.

```
1 import numpy as np  
2 x = np.linspace(1, 6, 5)  
3 print('x =', x)  
[1] ✓ 0.0s  
... x = [1. 2.25 3.5 4.75 6. ]
```

Tên gói được rút gọn

4. Function (hàm) trong Python.

4.2. Hàm trong các gói cài đặt bổ sung. (Tiếp theo)

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sympy as sp
4 x = sp.symbols('x')
5 fx = 3*x**2*sp.sin(x); print('fx =', fx)
6 xv = np.linspace(0, 10, 200); fv = []
7 for i in range(len(xv)):
8     fv = np.append(fv, fx.subs(x, xv[i]))
9 plt.plot(xv, fv); plt.show()
[1]   ✓ 0.8s
...
fx = 3*x**2*sin(x)

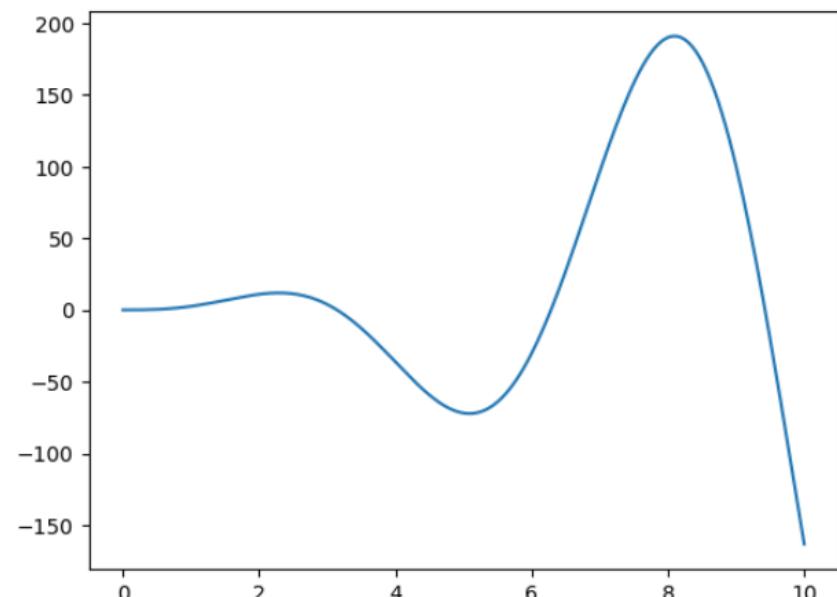
```

Nạp thư viện numpy: Tạo mảng, các phép toán trên mảng, đại số tuyến tính...

Nạp thư viện pyplot: Vẽ, trình bày dữ liệu, đồ thị, biểu đồ...

Nạp thư viện sympy: Tính các phép toán bằng chữ (symbol)

Hiện số hàng (có trong cài đặt chứ không phải nhập vào) để thuận tiện khi lập trình, không liên quan nội dung của chương trình



4. Function (hàm) trong Python.

4.2. Hàm trong các gói cài đặt bổ sung. (Tiếp theo)

```
[1] 1 import numpy as np  
2 print(dir(np))  
✓ 0.0s
```

In danh sách các **hàm** có trong gói numpy

```
▷ v [1] 1 import numpy as np  
2 print(np.array.__doc__)  
✓ 0.0s
```

Hai ký tự gạch chéo (__)

In mô tả, hướng dẫn sử dụng, ví dụ... của hàm array

4. Function (hàm) trong Python.

4.3. Hàm do người dùng tự định nghĩa, tạo ra.

- Hàm là một đoạn code (khối lệnh) được tạo ra theo cú pháp được qui định cụ thể.
- Mục đích của tạo và sử dụng hàm là để **tái sử dụng** nhiều lần trong chương trình.
- Thông thường, sử dụng hàm là truyền vào các tham biến và trả về các giá trị.

4.3.1. Hàm được tạo bằng từ khóa **lambda**. (Chỉ trả về một giá trị)

```

import numpy as np
Wr = lambda d: pi*d**3/16; pi = np.pi
Wx = lambda b, h: b*h**2/6
d, Mz = 6, 9
ta = Mz / Wr(d); print('tau =', ta, 'kN/cm^2')
b, h, Mx = 5, 9, 7
si = Mx / Wx(b,h); print('sigma =', si, 'kN/cm^2')
    
```

Tên hàm

Tham biến truyền vào

Các câu lệnh khác trong hàm

Giá trị trả về của hàm

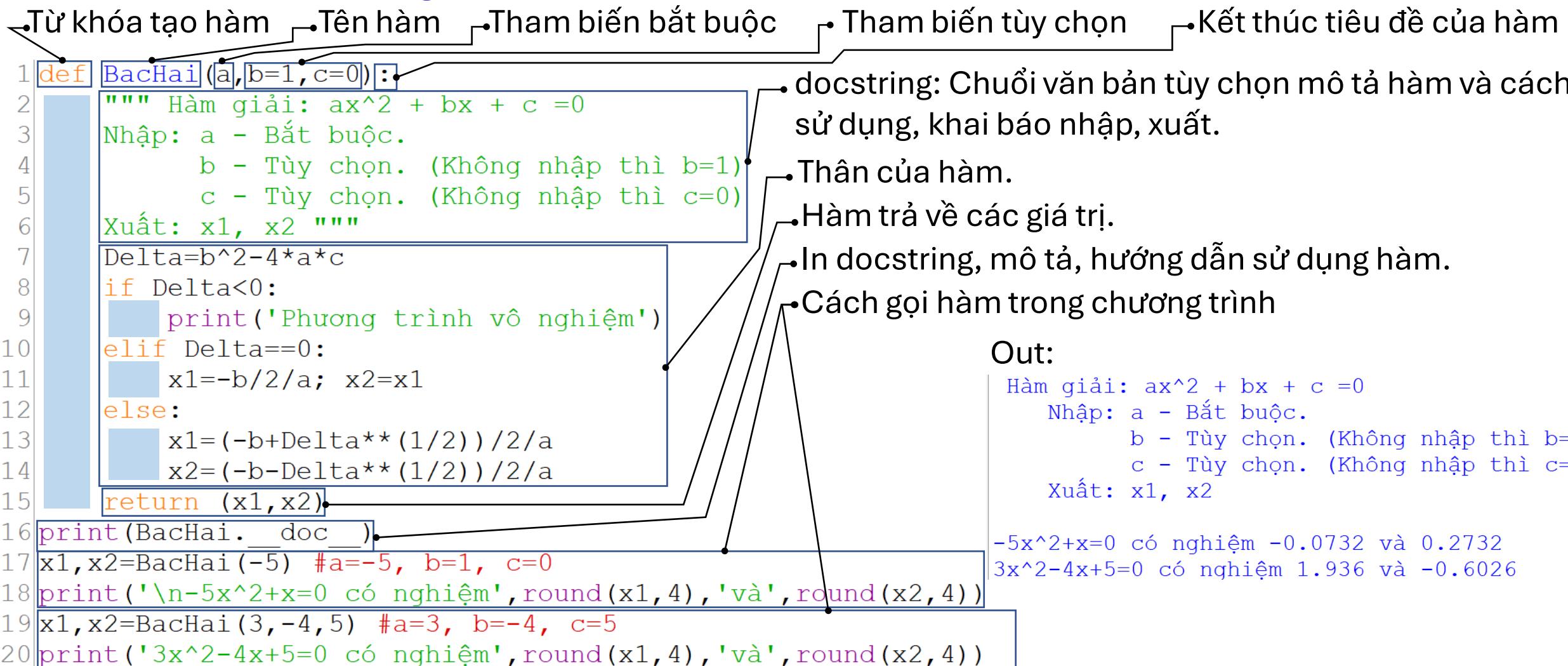
Sử dụng hàm

tau = 0.2122065907891938 kN/cm²

sigma = 0.1037037037037037 kN/cm²

4. Function (hàm) trong Python.

4.3.2. Hàm được tạo bằng từ khóa def.



Bắt buộc thét vào 1 Tab.

print('\n... xuống 1 dòng của văn bản đứng sau cặp từ khóa \n

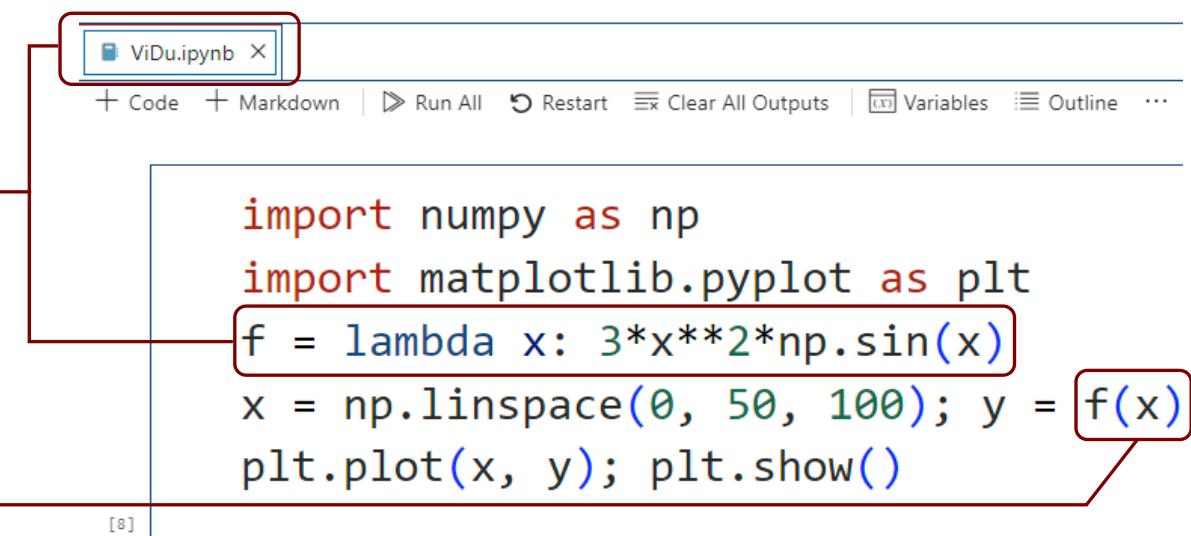
4. Function (hàm) trong Python.

4.4. Vị trí tạo, và nơi lưu trữ hàm.

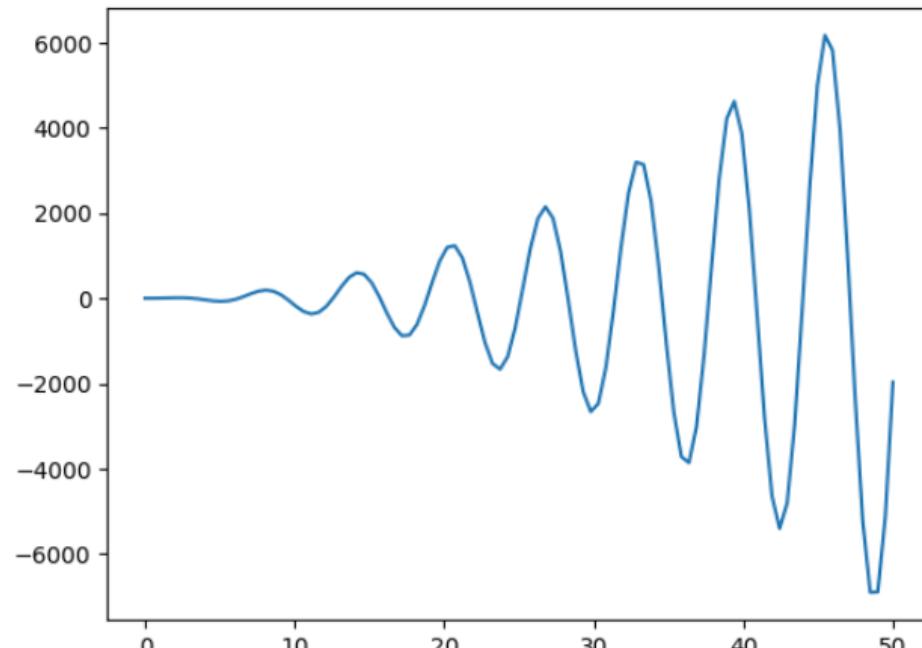
4.4.1. Hàm tạo trong chương trình chính.

- Hàm và chương trình cùng nằm một nơi trên một tập tin (file)

- Hàm phải được tạo trước lệnh gọi sử dụng hàm. Thường là đầu chương trình.



```
import numpy as np
import matplotlib.pyplot as plt
f = lambda x: 3*x**2*np.sin(x)
x = np.linspace(0, 50, 100); y = f(x)
plt.plot(x, y); plt.show()
```



4. Function (hàm) trong Python.

4.4.2. Hàm tạo trên một file (Module) khác.

- Đối với file dùng để tạo hàm thì phải lưu dưới định dạng **.py**
- Trên một file, thường được gọi là module có thể tạo ra nhiều hàm khác nhau.
- Với file dùng để viết chương trình chính thì có thể lưu định dạng **.py** hoặc **.ipynb** đều được.
- Ví dụ: Tạo một **module** có tên **stress**

1/ Tạo folder **My_Python** trong ổ D: D:\My_Python

2/ Tạo file (Module) có tên **stress.py** trong nó tạo các hàm: **doc**, **xoan**, **uon**, **PhucTap** và lưu vào folder **My_Python**

3/ Tạo file (Chương trình chính) có tên **SucBenVatLieu.ipynb** có chứa dòng code: **import stress** và các nội dung khác rồi lưu vào folder **My_Python**

4/ Chạy chương trình chính → sinh ra:

 4.1/ Folder con: **__pycache__** ở trong folder **My_Python**

 4.2/ File: **stress.cpython-311.pyc** ở trong folder **__pycache__**

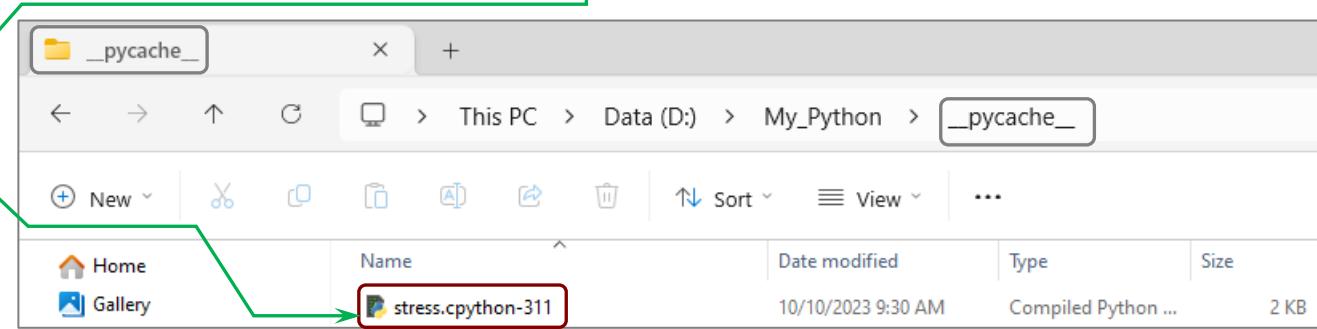
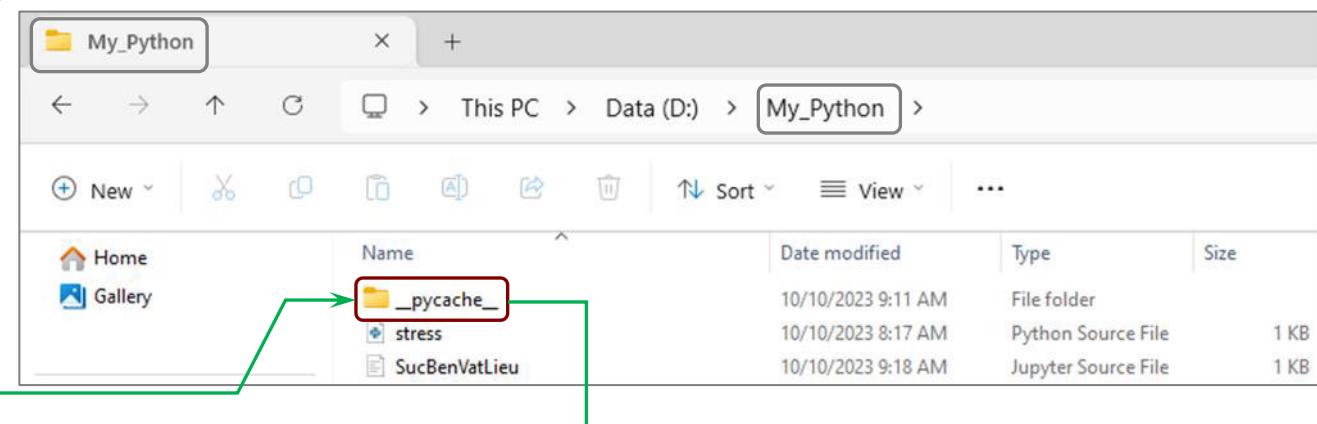
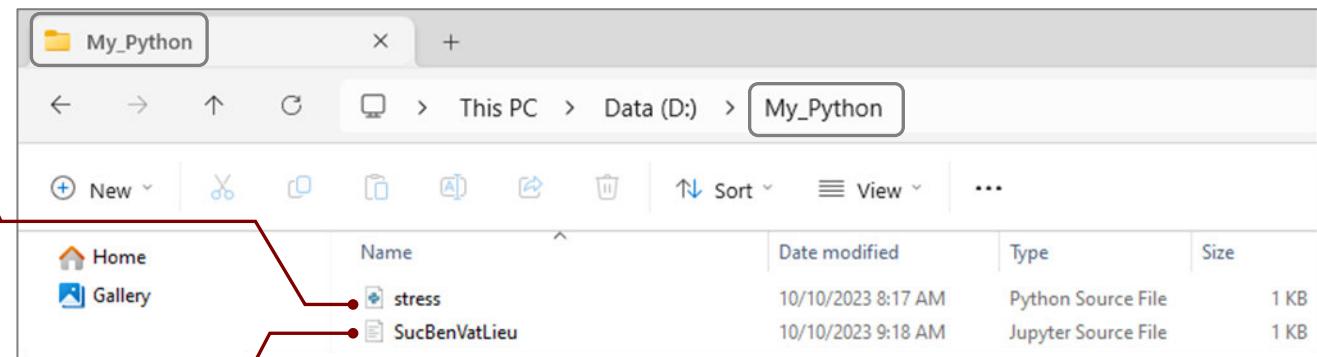
5/ Lúc này, ta có thể sử dụng các hàm tồn tại trong module **stress**:
stress.doc , **stress.xoan** , **stress.uon** , **stress.PhucTap**

4. Function (hàm) trong Python.

4.4.2. Hàm tạo trên một file (Module) khác. (Tiếp tục)

```
stress.py x
1 import numpy as np
2 def doc(Nz, F, scp):
3     s = Nz/F; dkb = 100*s/scp
4     return dkb
5 def xoan(Mz, d, tcp):
6     t = Mz*16/np.pi/d**3
7     dkb = 100*t/tcp
8     return dkb
9 def uon(Mx, Jx, ymax, scp):
10    s = Mx*ymax/Jx
11    dkb = 100*s/scp
12    return dkb
13 def PhucTap(Mx, My, Mz, Wu, scp):
14    s4 = np.sqrt(Mx**2 + My**2 + 0.75*Mz**2)/Wu
15    dkb = 100*s4/scp
16    return dkb
```

```
SucBenVatLieu.ipynb x
+ Code + Markdown > Run All ⏪ Restart ⏪ Clear All Outputs ⏪ Variables ⏪ Outline ...
import stress
Nz, F, scp = 9, 0.4, 15
dkb = stress.doc(Nz, F, scp)
print('sigma =', dkb, '% của sigma cho phép')
[1] 0.0s
... sigma = 150.0 % của sigma cho phép
```



4. Function (hàm) trong Python.

4.4.3. Hàm tạo trên nhiều file (Module) trong một folder (Package) khác.

- Ví dụ: Tạo một **Package** có tên **My_Function**

1/ Tạo folder **Working_Python** trong ổ D: D:\Working_Python

2/ Tạo folder (Package) **My_Function** trong folder **Working_Python**: D:\Working_Python\My_Function

3/ Tạo file trống có tên: **__init__.py** không có nội dung, lưu trong folder **My_Function**

4/ Tạo hai file (Module) có tên **Cong_Tru.py** và **Nhan_Chia.py** được lưu trong folder **My_Function**

Trong module **Cong_Tru.py** tạo các hàm: **cong, tru**

Trong module **Nhan_Chia.py** tạo các hàm: **nhan, chia**

5/ Tạo file (Chương trình chính) có tên **Use_Package.ipynb** có chứa hai dòng code:

import My_Function.Cong_Tru

import My_Function.Nhan_chia

cùng với các nội dung khác rồi lưu vào folder **Working_Python**

6/ Chạy chương trình chính → sinh ra:

6.1/ Folder con: **__pycache__** ở trong folder **My_Function**

6.2/ File: **__init__.cpython-311.pyc ; Cong_Tru.cpython-311.pyc ; Nhan_Chia.cpython-311.pyc**
ở trong thư mục **__pycache__**

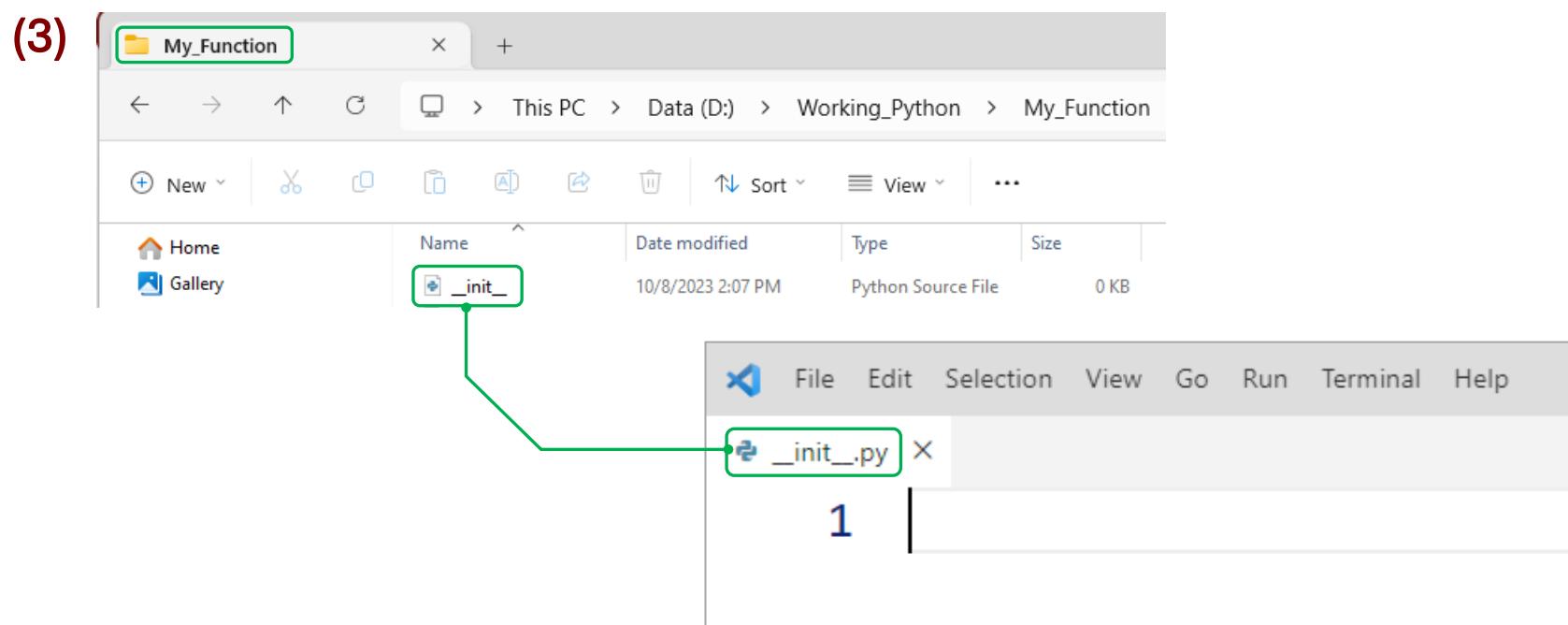
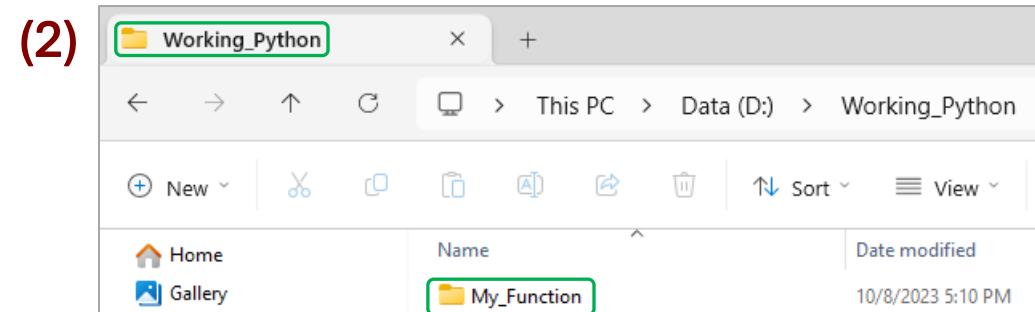
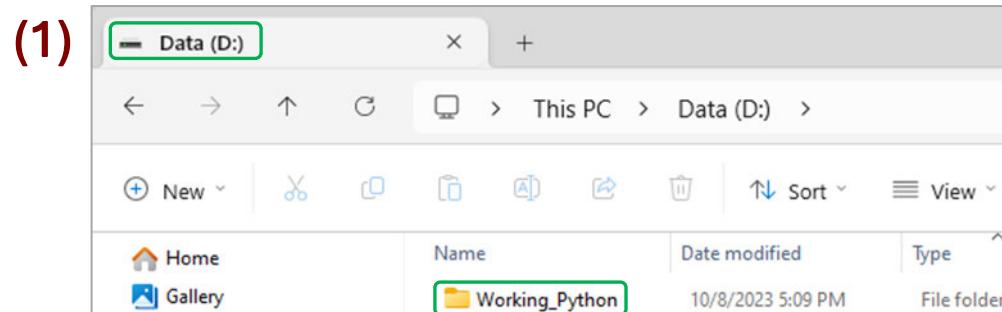
7/ Lúc này, ta có thể sử dụng các hàm tồn tại trong Package **My_Function**:

My_Function.Cong_Tru.cong, My_Function.Cong_Tru .tru

My_Function.Nhan_chia.nhan , My_Function.Nhan_chia.chia

4. Function (hàm) trong Python.

4.4.3. Hàm tạo trên nhiều file (Module) trong một folder (Package) khác. (Tiếp tục)



4. Function (hàm) trong Python.

4.4.3. Hàm tạo trên nhiều file (Module) trong một folder (Package) khác. (Tiếp tục)

(4)

The screenshot illustrates the creation of a Python package named 'My_Function'. The package structure is as follows:

```
My_Function
├── _init_.py
└── Working_Python
    └── My_Function
        ├── Cong_Tru.py
        └── Nhan_Chia.py
```

The 'Cong_Tru.py' file contains the following code:

```
1 def cong(a, b):
2     KQco = a + b
3     return KQco
4 def tru(a, b):
5     KQtr = a - b
6     return KQtr
```

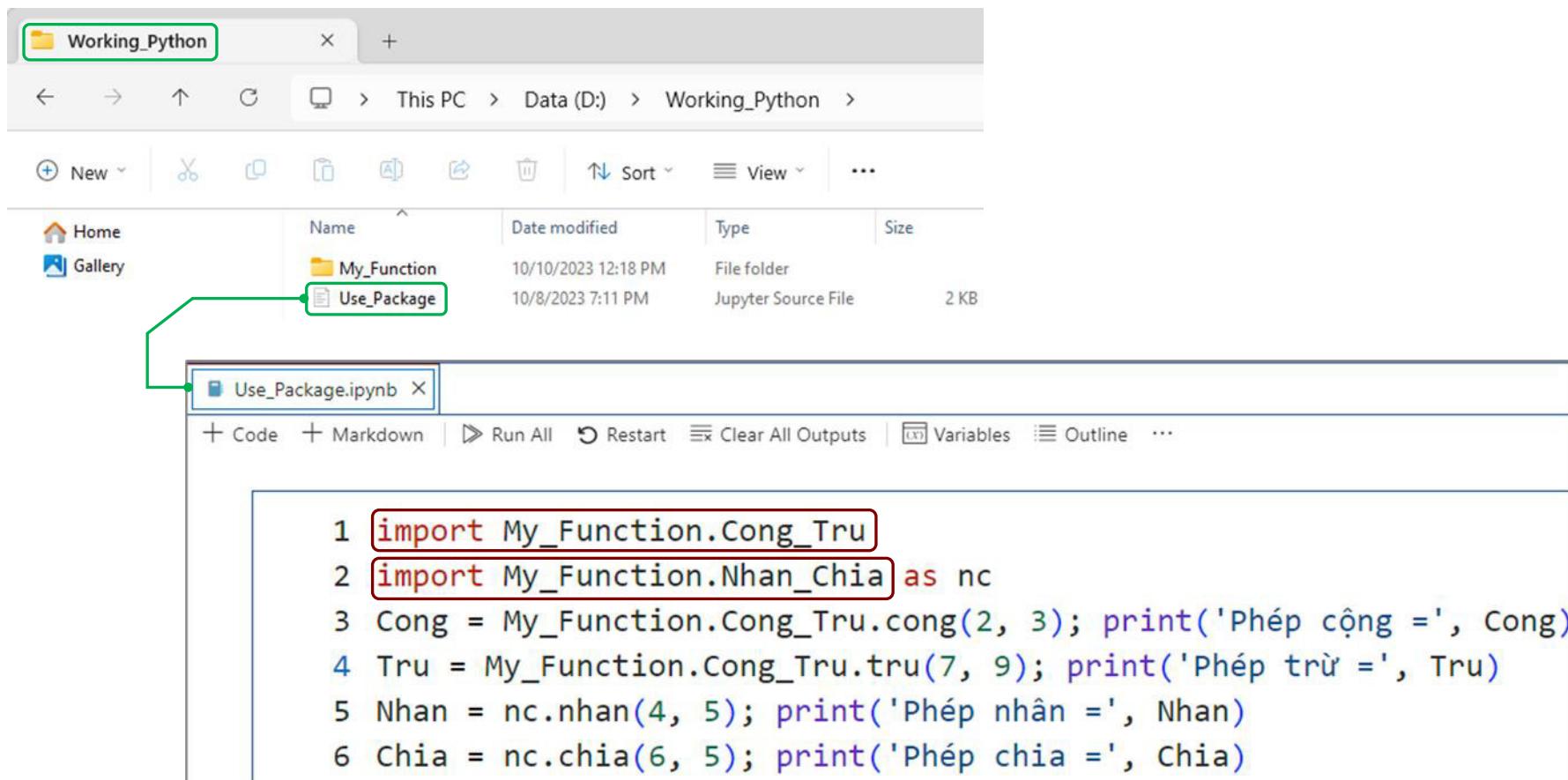
The 'Nhan_Chia.py' file contains the following code:

```
1 def nhan(a, b):
2     KQnh = a * b
3     return KQnh
4 def chia(a, b):
5     KQch = a / b
6     return KQch
```

4. Function (hàm) trong Python.

4.4.3. Hàm tạo trên nhiều file (Module) trong một folder (Package) khác. (Tiếp tục)

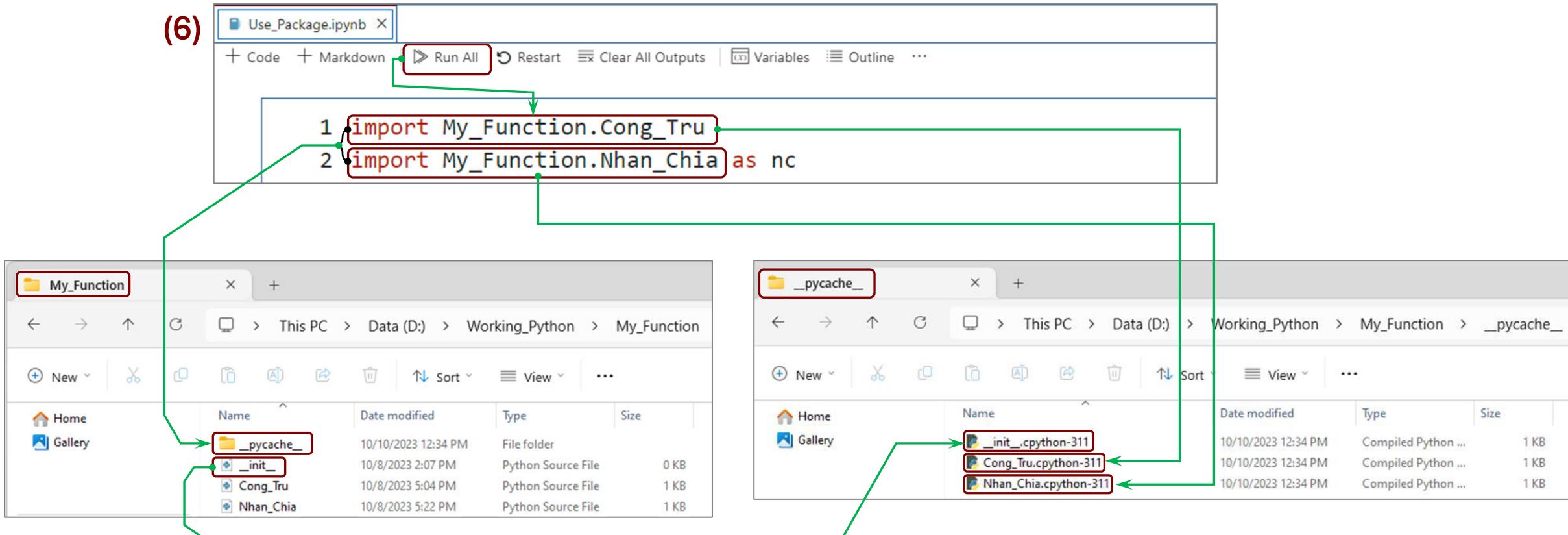
(5)



4. Function (hàm) trong Python.

4.4.3. Hàm tạo trên nhiều file (Module) trong một folder (Package) khác. (Tiếp tục)

(6)



4. Function (hàm) trong Python.

4.4.3. Hàm tạo trên nhiều file (Module) trong một folder (Package) khác. (Tiếp tục)

(7)

The diagram illustrates the execution of a Python script named 'Use_Package.ipynb' in a Jupyter Notebook environment. The code imports functions from a package named 'My_Function'. The package contains three modules: 'Cong_Tru', 'Nhan_Chia', and 'Chia'. The imported module 'Nhan_Chia' is aliased as 'nc'. The code then uses these functions to perform arithmetic operations and print results. The output shows the results of each operation: addition, subtraction, multiplication, and division.

```
1 import My_Function.Cong_Tru
2 import My_Function.Nhan_Chia as nc
3 Cong = My_Function.Cong_Tru.cong(2, 3); print('Phép cộng =', Cong)
4 Tru = My_Function.Cong_Tru.tru(7, 9); print('Phép trừ =', Tru)
5 Nhan = nc.nhan(4, 5); print('Phép nhân =', Nhan)
6 Chia = nc.chia(6, 5); print('Phép chia =', Chia)

[1]    ✓ 0.0s
...
Phép cộng = 5
Phép trừ = -2
Phép nhân = 20
Phép chia = 1.2
```

Annotations with arrows point to specific parts of the code:

- A green bracket on the left points to the first two lines of the code: `1 import My_Function.Cong_Tru` and `2 import My_Function.Nhan_Chia as nc`. It is labeled "Tên của Package (Folder, Gói, Thư mục)" (Name of the Package (Folder, Package, Directory)).
- A purple bracket on the right points to the line `2 import My_Function.Nhan_Chia as nc`. It is labeled "Tên của Package + Module rút gọn" (Name of the Package + Abbreviated Module Name).
- A red bracket at the bottom points to the line `3 Cong = My_Function.Cong_Tru.cong(2, 3);`. It is labeled "Tên của Module (File, Tập tin, Tiệp tin)" (Name of the Module (File, Document, File)).
- A red bracket on the right points to the line `3 Cong = My_Function.Cong_Tru.cong(2, 3);`. It is labeled "Tên của Function (Hàm)" (Name of the Function (Function)).

5. Các toán tử trong Python.

5.1. Toán tử số học.

Toán tử	Mô tả	Ví dụ	Kết quả
+	Phép cộng hai số	2+3	5
-	Phép trừ hai số	4-7	-3
*	Phép nhân hai số	6*5	30
/	Phép chia hai số	15/7	2.142857142857143
//	Phép chia hai số lấy phần nguyên	15//7	2
%	Phép chia hai số lấy phần dư	15%7	1
**	Phép lũy thừa của một số	2**3	8
(1/2)	Phép lấy căn bậc 2 của một số	3^(1/2)	1.7320508075688772

Thứ tự các phép toán được thực hiện:

Trong ngoặc → Lũy thừa → Nhân và chia → Cộng và trừ

5. Các toán tử trong Python.

5.2. Toán tử gán.

Dùng để gán giá trị của một đối tượng cho một đối tượng khác.

Toán tử	Mô tả	Ví dụ	Kết quả
=	Lấy giá trị của biến bên phải gán cho biến bên trái.	a = 4 b = a	a = 4 b = 4
+=	Lấy giá trị của biến bên trái (đã có) cộng với biến bên phải, kết quả gán cho biến bên trái (cập nhật)	a = 4 a += 1 (a = a + 1)	a = 4 a = 5
-=	Lấy giá trị của biến bên trái (đã có) trừ cho biến bên phải, kết quả gán cho biến bên trái (cập nhật)	a = 4 a -= 5 (a = a - 5)	a = 4 a = -1
*=	Lấy giá trị của biến bên trái (đã có) nhân cho biến bên phải, kết quả gán cho biến bên trái (cập nhật)	a = 4 a *= 2 (a = a * 2)	a = 4 a = 8
/=	Lấy giá trị của biến bên trái (đã có) chia cho biến bên phải, kết quả gán cho biến bên trái (cập nhật)	a = 4 a /= 2 (a = a / 2)	a = 4 a = 2

5. Các toán tử trong Python.

5.3. Toán tử quan hệ.

Dùng để so sánh các giá trị với nhau, kết quả trả về là True nếu đúng và False nếu sai. Thường được dùng trong các câu lệnh điều kiện.

Toán tử	Mô tả	Ví dụ	Kết quả
<code>==</code>	So sánh bằng nhau giữa hai đối số	<code>2 == 3</code>	False
<code>!= hoặc <></code>	So sánh khác nhau giữa hai đối số	<code>2 != 3</code>	True
<code><</code>	So sánh liệu số bên trái có nhỏ hơn số bên phải không	<code>2 < 3</code>	True
<code>></code>	So sánh liệu số bên trái có lớn hơn số bên phải không	<code>2 > 3</code>	False
<code><=</code>	So sánh liệu số trước có nhỏ hơn hoặc bằng số sau không	<code>2 <= 2</code>	True
<code>>=</code>	So sánh liệu số trước có lớn hơn hoặc bằng số sau không	<code>3 >= 2</code>	True

5. Các toán tử trong Python.

5.4. Toán tử logic.

Dùng để kết hợp hoặc loại trừ của các kết quả của True, False. Kết quả trả về là True hoặc False. Thường được dùng trong các câu lệnh điều kiện.

Toán tử	Mô tả	Ví dụ	Kết quả
and	True and True = True True and False = False False and True = False False and False = False	a = 4 a \geq 3 and a \leq 5 ~ 3 \leq a \leq 5 a > 4 and a < 5 ~ 4 < a < 5 a > 5 and a < 3 ~ 5 < a < 3	a = 4 True True False
or	True or True = True True or False = True False or True = True False or False = False	a = 4 a \geq 3 or a \leq 5 a > 4 or a < 5 a > 5 or a < 3	a = 4 True True False
not	Phủ định của kết quả	a = 4 not a >< 4	a = 4 True

5. Các toán tử trong Python.

5.5. Toán tử tìm kiếm, xác thực.

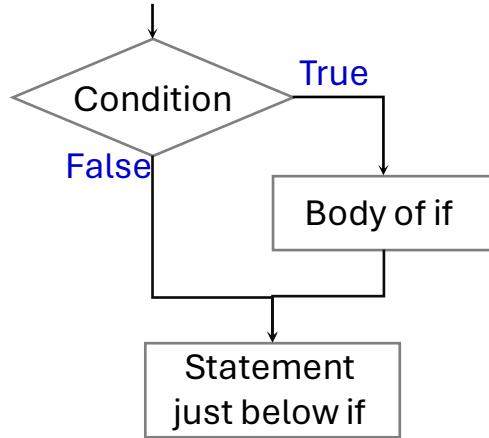
Dùng biến vẽ trái tìm trong một tập hợp. Nếu tìm có thì trả kết quả True và ngược lại.

Toán tử	Mô tả	Ví dụ	Kết quả
in	Tìm giá trị vẽ trái xem có trong tập hợp bên vẽ phải không	a = 4 b = [1, 3, 5] a in b	a = 4 b = [1, 3, 5] False
not in	Tìm giá trị vẽ trái xem liệu không có trong tập hợp bên vẽ phải không	a = 4 b = [1, 3, 5] a not in b	a = 4 b = [1, 3, 5] True
is	Tìm giá trị vẽ trái xem liệu có đúng bằng giá trị bên vẽ phải không	a = 4; b = 5 a is b	a = 4; b = 5 False
not is	Tìm giá trị vẽ trái xem có phải không bằng giá trị vẽ phải không	a = 4; b = 5 a not is b	a = 4; b = 5 True

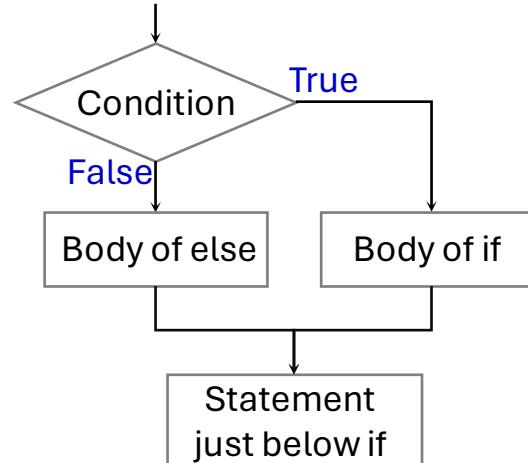
6. Các mệnh đề, vòng lặp điều khiển.

6.1. Điều kiện if. NẾU [điều kiện đặt ra thỏa mãn] THÌ làm ...

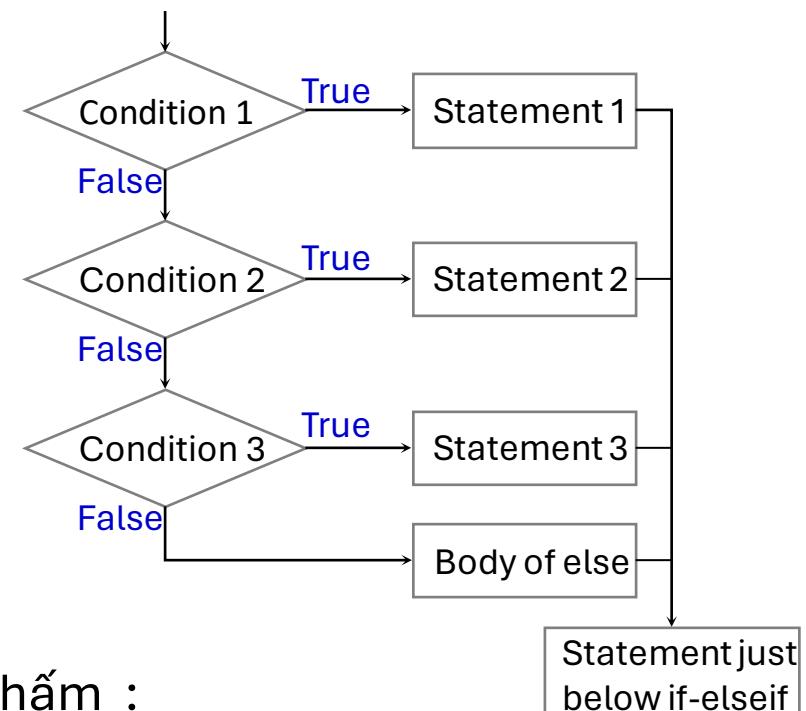
If “ĐK1” (OR, AND) “ĐK2”:
[Tab] Câu lệnh trong if



If “ĐK1” (OR, AND) “ĐK2”:
[Tab] Câu lệnh nếu thỏa ĐK1 (hoặc và) ĐK2
Else:
[Tab] Câu lệnh trong các trường hợp còn lại



If “ĐK1” (OR, AND) “ĐK2”:
[Tab] Câu lệnh nếu thỏa ĐK1 (hoặc và) ĐK2
Elif “ĐK3” (OR, AND) “ĐK4”:
[Tab] Câu lệnh nếu thỏa ĐK3 (hoặc và) ĐK4
Elif “ĐK5” (OR, AND) “ĐK6”:
[Tab] Câu lệnh nếu thỏa ĐK5 (hoặc và) ĐK6
Else:
[Tab] Câu lệnh trong các trường hợp còn lại

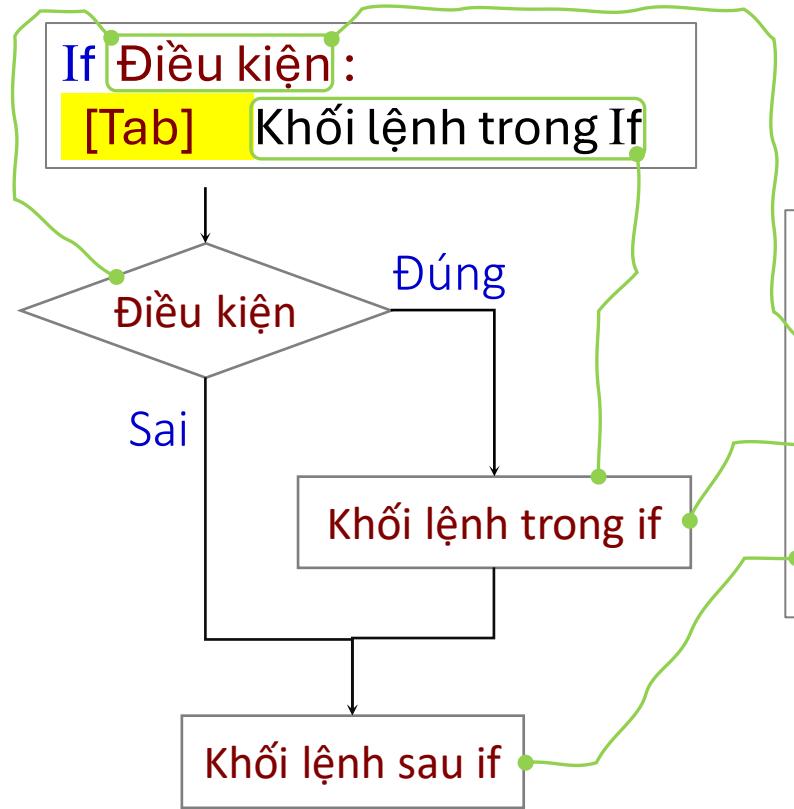


Lưu ý: 1. Sau khối điều kiện **if**, **elif** hoặc **else** phải khóa bằng dấu hai chấm :

2. Tất cả các khối lệnh thực thi sau **if** phải thụt vào 1 **tab** hoặc nằm cùng hàng với **if**

6. Các mệnh đề, vòng lặp điều khiển.

6.1.1. Điều kiện if một nhánh rẽ.



```

x = 5
den = 2*x**3 - 7*x**2 + 4*x
if den == 0:
    den = 1e6 # Tránh lỗi chia 0
    f = (6*x - 7)/den
    print('f = ', f)
    
```

Khi viết trên một dòng:

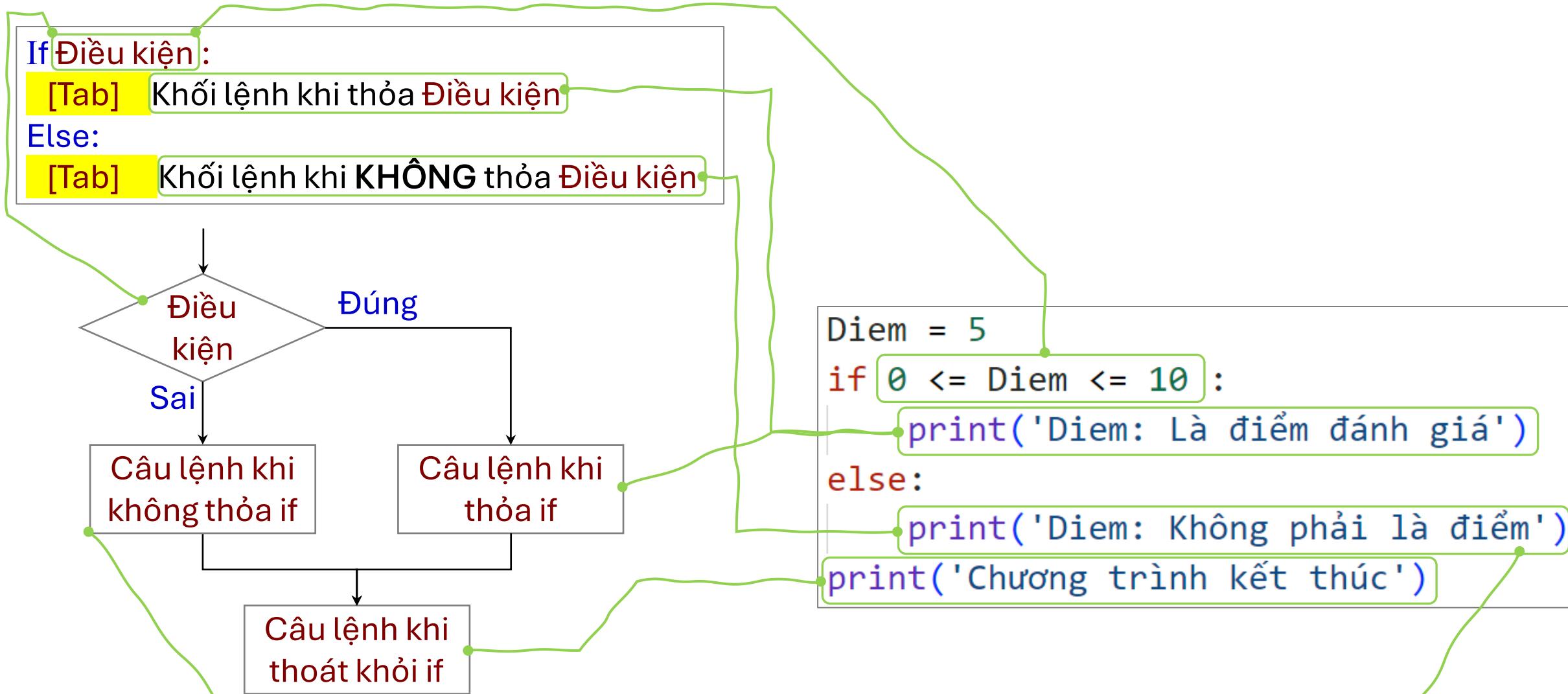
If **Điều kiện**: **Khối lệnh của If**

```

x = 3
den = 2*x**4 - 6*x**3 - 5*x
if den == 0: den = 1e6
    f = (6*x - 7)/den
    print('f = ', f)
    
```

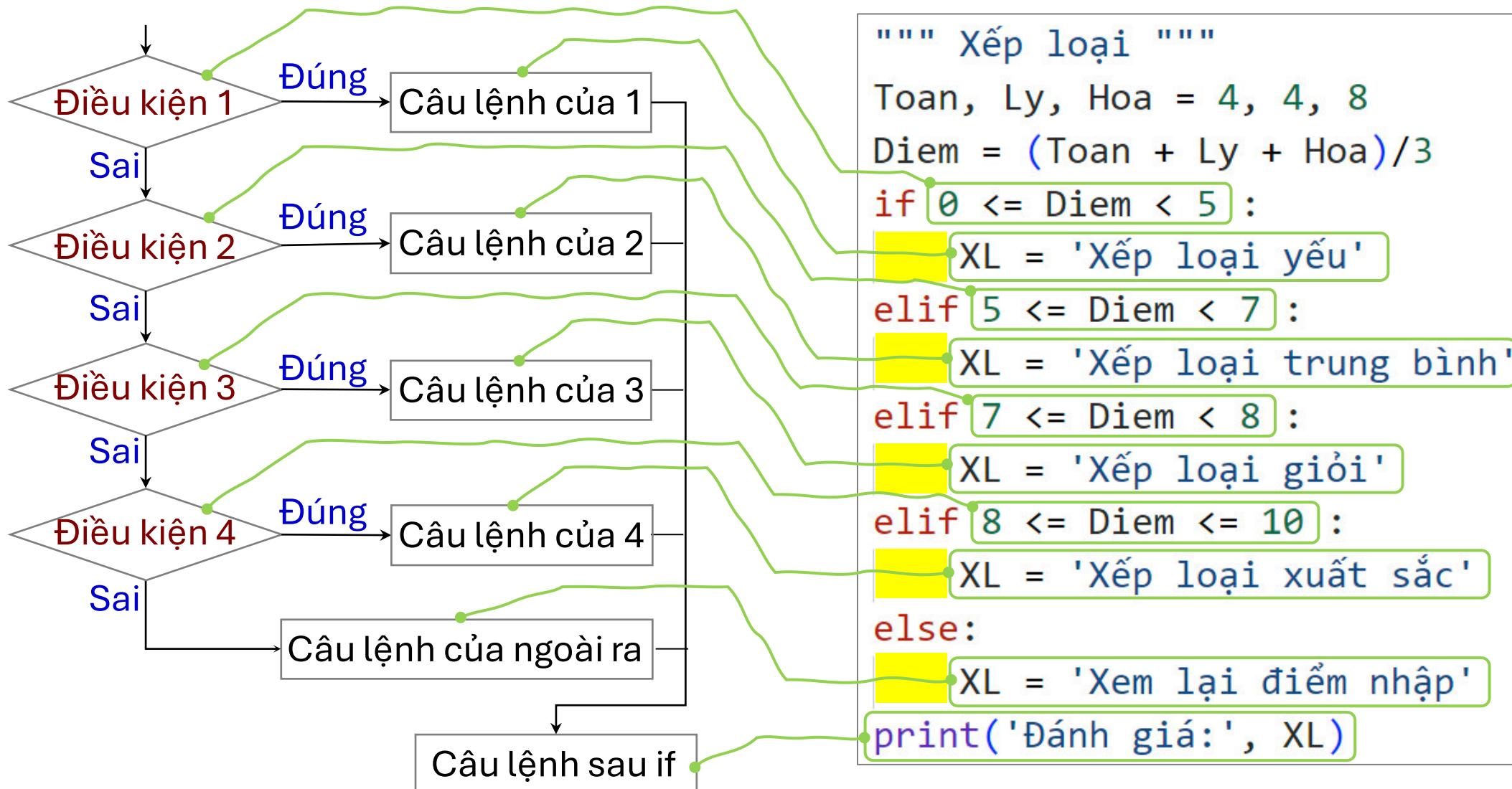
6. Các mệnh đề, vòng lặp điều khiển.

6.1.2. Điều kiện if hai nhánh rẽ.



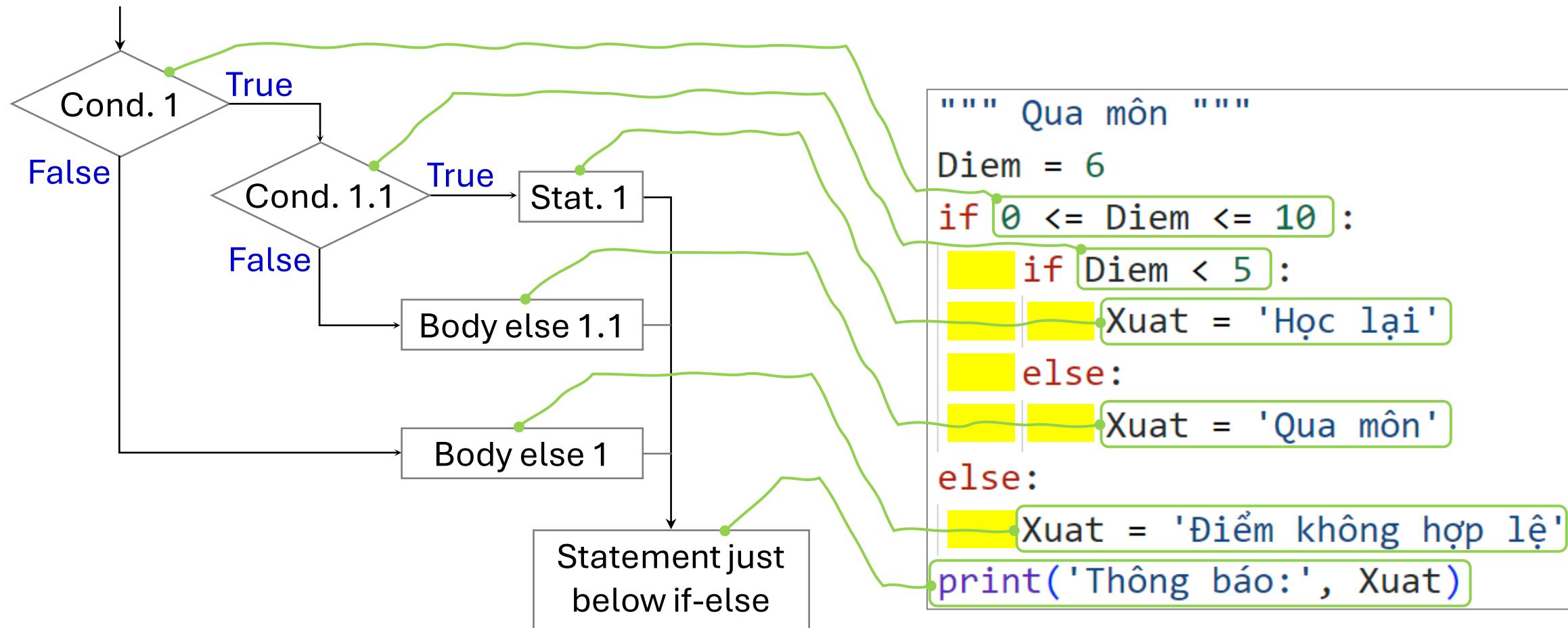
6. Các mệnh đề, vòng lặp điều khiển.

6.1.3. Điều kiện if nhiều hơn hai nhánh rẽ.



6. Các mệnh đề, vòng lặp điều khiển.

6.1.4. Điều kiện if nhiều nhánh rẽ lồng nhau.



6. Các mệnh đề, vòng lặp điều khiển.

6.2. Vòng lặp For. Đã biết trước số lần lặp.

For Biến lặp in Vùng lặp:

[Tab] Nếu Biến lặp còn trong Vùng lặp thì làm khôi lệnh trong For

Khởi tạo giá trị lặp

Điều kiện lặp

Đúng

Khối lệnh trong For

Cập nhật giá trị lặp (+1)

Câu lệnh sau For

""" Tạo dãy số Fibonacci """

```
import numpy as np
n = 12; F = []
for i in range(n):
    if i == 0 or i == 1:
        F = np.append(F, 1)
    else:
        F = np.append(F, F[i-2]+F[i-1])
print('F = ', F)
```

range(n) = 0 1 2 ... n-1

...

F = [1. 1. 2. 3. 5. 8. 13. 21. 34. 55. 89. 144.]

6. Các mệnh đề, vòng lặp điều khiển.

6.3. Vòng lặp While.

Khởi tạo giá trị lặp

While Điều kiện lặp :

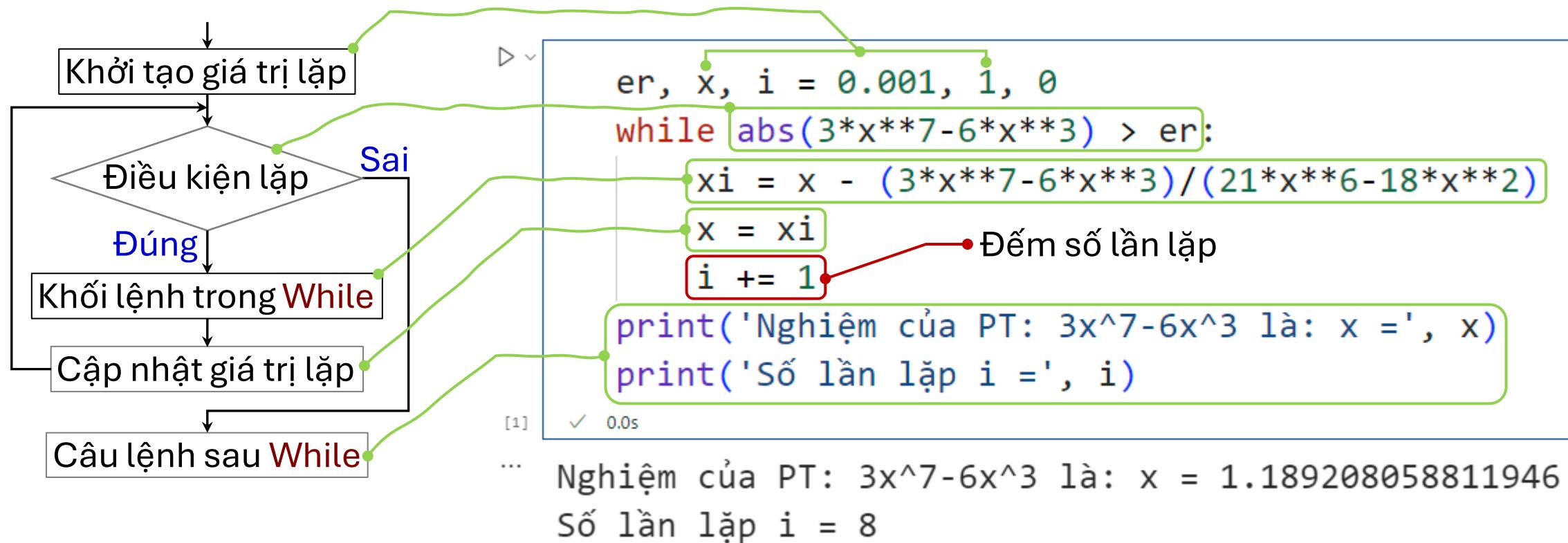
[Tab] Khối lệnh trong While

[Tab] Cập nhật giá trị lặp

Chưa biết số lần lặp, kiểm tra điều kiện trước khi làm.

Trong khi điều kiện còn đúng thì cứ làm.

- Nếu “Điều kiện còn đúng” → Thì làm.
- Nếu “Điều kiện không còn đúng” → Thì dừng.



7. Sử dụng gói Numpy cơ bản.

`import numpy as np` Định dạng số chữ số khi xuất kết quả: `np.set_printoptions(precision=4)`

7.1. Mảng trong Numpy.

Hàm `array` dùng để tạo mảng trong `numpy` mà ma trận (matrix) hay véc tơ (vector) thuộc tập con của nó.

7.1.1. Tạo mảng `np.array` 1-D.

`A = np.array([a0, a1, a2, … , an])` → Tạo mảng A: 1 chiều (1-D) với a_i là các phần tử được gán cho mảng.

`A.ndim` = 1 → Số chiều của A

`A.size` = n+1 → Số phần tử của A

`A.shape` = (n+1,) → Dạng của A (số phần tử trong chiều thứ nhất = n+1)

`A.dtype` → Kiểu dữ liệu của A (phụ thuộc dữ liệu khi tạo A): `int32` – kiểu số nguyên, `float64` – kiểu số thực

```
▶ 
    import numpy as np
    A = np.array([4, 5, 6]);          B = np.array([7, 8, 9], dtype='float64');      Anew = A.astype('float')
    print('A =', A, '; ndim_A =', A.ndim, '; size_A =', A.size, '; shape_A =', A.shape, '; dtype_A =', A.dtype)
    print('B =', B, '; dtype_B =', B.dtype, '; Anew =', Anew, '; dtype_Anew =', Anew.dtype)
[1]   ✓ 0.1s
```

... A = [4 5 6] ; ndim_A = 1 ; size_A = 3 ; shape_A = (3,) ; dtype_A = int32
B = [7. 8. 9.] ; dtype_B = float64 ; Anew = [4. 5. 6.] ; dtype_Anew = float64

Khai báo kiểu dữ liệu khi tạo mảng thì truyền thêm tham biến: `dtype='KieuDuLieu'` trong hàm `array`.

Chuyển qua kiểu dữ liệu mới cho mảng A: `A.astype(np.float64) ~ A.astype('float64')`

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.2. Tạo mảng np.array 2-D.

`A = np.array([[a00, a01, … , a0n], [a10, a11, … , a1n], … , [am0, am1, … , amn]])`

→ Tạo mảng A: 2 chiều (2-D) với a_{ij} là các phần tử được gán cho mảng.

`A.ndim` = 2 → Số chiều của A

`A.size` = $m \times n$ → Số phần tử của A

`A.shape` = ($m + 1, n + 1$) → Dạng của A (số phần tử trên chiều thứ nhất (hàng) $m+1$ và chiều thứ hai (cột) $n+1$)

`A.dtype` → Kiểu dữ liệu của A (phụ thuộc dữ liệu khi tạo A): `int32` – kiểu số nguyên, `float64` – kiểu số thực

```
▶ 
    import numpy as np
    A = np.array([[2., 3, 4], [5, 6, 7]])
    print('A =\n', A, '\nA.ndim =', A.ndim, '; A.size =', A.size)
    print('A.shape =', A.shape, '; A.dtype =', A.dtype)
[1] ✓ 0.1s
```

```
...
    A =
        [[2. 3. 4.]
         [5. 6. 7.]]
    A.ndim = 2 ; A.size = 6
    A.shape = (2, 3) ; A.dtype = float64
```

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.3. Tạo mảng np.array 3-D.

```
> 
    import numpy as np
    A = np.array([[ [1, 2, 3, 4], [5, 6, 7, 8], [8, 7, 6, 5] ],
                  |   |   |   |   [[9, 8, 7, 6], [5, 4, 3, 2], [4, 5, 6, 7]]])
    print('A =\n', A)
    print('A.ndim =', A.ndim, '; A.size =', A.size, '; A.shape =', A.shape)
[1] ✓ 0.1s
```

... A =

Lớp {

A =

1	2	3	4
5	6	7	8
8	7	6	5

Hàng } Hàng

9	8	7	6
5	4	3	2
4	5	6	7

Cột }

A.ndim = 3 ; A.size = 24 ; A.shape = (2, 3, 4)

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.4. Các phép toán trên mảng.

7.1.4.1. Phép toán cộng, trừ, nhân, chia từng cặp phần tử tương ứng giữa hai mảng.

Với các phép toán này thì dạng (shape) của hai mảng phải giống nhau

```
▶ 
    import numpy as np
    A = np.array([7, 5, 6])
    B = np.array([1, 2, 3])
    print('A+B=', A+B, '\nA-B=', A-B)
    print('A*B=', A*B, '\nA/B=', A/B)
[1] ✓ 0.1s
```

```
... A+B= [8 7 9]
      A-B= [6 3 3]
      A*B= [ 7 10 18]
      A/B= [7. 2.5 2. ]
```

```
▶ 
    import numpy as np
    C = np.array([[7, 5, 6], [5, 9, 8]])
    D = np.array([[4, 2, 3], [4, 1, 2]])
    print('C+D=\n', C+D, '\nC-D=\n', C-D)
    print('C*D=\n', C*D, '\nC/D=\n', C/D)
[1] ✓ 0.1s
```

```
... C+D=
[[11 7 9]
 [ 9 10 10]]
C-D=
[[3 3 3]
 [1 8 6]]
C*D=
[[28 10 18]
 [20 9 16]]
C/D=
[[1.75 2.5 2. ]
 [1.25 9. 4. ]]
```

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.4.2. Phép toán nhân ma trận (matrix), véc tơ (vector), cú pháp: $A @ B$ hoặc $np.dot(A, B)$

Chỉ áp dụng trong 2-D. Với A nhân B là hai ma trận hoặc véc tơ thì phép nhân chỉ có thể thực hiện khi:

- A và B phải cùng số chiều là 2 (ndim = 2).
- Chiều thứ hai (cột) của A phải bằng chiều thứ nhất (hàng) của B.

```
▶ 
    import numpy as np
    A = np.array([[4, 5, 6]])
    B = np.array([[7], [8], [9]])
    print('A =\n', A, '\nB =\n', B)
    print('A@B=\n', A@B, '\nB@A=\n', B@A)
```

```
[1] ✓ 0.1s
```

... A =
[[4 5 6]] → [1, 3]

B =
[[7]
[8]
[9]] } → [3, 1]

A@B= [[122]] → [1, 1]

B@A= [[28 35 42]
[32 40 48]
[36 45 54]] } → [3, 3]

```
▶ 
    import numpy as np
    C = np.array([[4, 5, 6], [9, 7, 8]])
    D = np.array([[1, 2], [3, 4], [5, 6]])
    print('C =\n', C, '\nD =\n', D)
    print('C@D=\n', C@D)
```

```
[1] ✓ 0.1s
```

... C =
[[4 5 6]
[9 7 8]] } → [2, 3]

D =
[[1 2]
[3 4]
[5 6]] } → [3, 2]

C@D= [[49 64]
[70 94]] } → [2, 2]

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.5. Chuyển đổi chiều và dạng của mảng.

Chuyển sang mảng 2-D: Mảng A cần chuyển phải được tạo trong gói numpy và có thể là 1-D hoặc 2-D:

`A.reshape(m, n)` với m số hàng, n số cột của mảng mới. Có thể dùng:

`A[np.newaxis, :]` → chuyển thành 1 cột

`A[:, np.newaxis]` → chuyển thành 1 hàng

```
import numpy as np
A = np.array([1, 2, 3, 4, 5, 6, 7, 8]); print('A =', A)
B = A.reshape(1, 8); print('B =', B) # ~ A[np.newaxis, :]
C = A.reshape(2, 4); print('C =\n', C)
[1] ✓ 0.2s
```

```
... A = [1 2 3 4 5 6 7 8]
B = [[1 2 3 4 5 6 7 8]]
C =
[[1 2 3 4]
 [5 6 7 8]]
```

7.1.6. Chuyển trí (chuyển vị) của mảng.

Chuyển trí của mảng A (2-D) là phép biến đổi hàng thành cột và ngược lại của mảng A.

`B = A.T`

```
import numpy as np
A = np.array([[1, 2, 3], [4, 5, 6]])
print('A =\n', A, '\nA.T =\n', A.T)
[1] ✓ 0.1s
```

```
... A =
[[1 2 3]
 [4 5 6]]
A.T =
[[1 4]
 [2 5]
 [3 6]]
```

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.7. Hàm tạo mảng đặc biệt.

- ✓ Hàm `arange`:

`np.arange(stop)`

`np.arange(start, stop)`

`np.arange(start, stop, step)`

start: điểm đầu (**mặc định bằng 0**)

stop: điểm cuối (nhỏ hơn điểm cuối step đơn vị)

step: bước (**mặc định bằng 1**)

```
import numpy as np
A = np.arange(5); print('A =', A)
B = np.arange(3, 8); print('B =', B)
C = np.arange(4, 14, 2); print('C =', C)
[1] ✓ 0.1s
...
A = [0 1 2 3 4]
B = [3 4 5 6 7]
C = [ 4   6   8 10 12]
```

- ✓ Hàm `linspace`:

`np.linspace(start, stop)`

`np.linspace(start, stop, num)`

start: điểm đầu

stop: điểm cuối

num: số điểm (**mặc định bằng 50**)

```
import numpy as np
A = np.linspace(1, 5, 6); print('A =', A)
[13] ✓ 0.0s
...
A = [1.  1.8 2.6 3.4 4.2 5. ]
```

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.7. Hàm tạo mảng đặc biệt. (Tiếp theo)

- ✓ Hàm **zeros**: phần tử bằng 0

np.zeros(num)

np.zeros(m, n)

num: mảng 1-D có num phần tử

m: Số hàng (mảng 2-D)

n: Số cột

```
import numpy as np
Z1 = np.zeros(3); print('Z1 =', Z1)
Z2 = np.zeros((2, 4)); print('Z2 =\n', Z2)
[1] ✓ 0.1s
...
Z1 = [0. 0. 0.]
Z2 =
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

- ✓ Hàm **ones**: phần tử bằng 1

np.ones(num)

np.ones(m, n)

num: mảng 1-D có num phần tử

m: Số hàng (mảng 2-D)

n: Số cột

```
import numpy as np
O1 = np.ones(3); print('O1 =', O1)
O2 = np.ones((2, 4)); print('O2 =\n', O2)
[1] ✓ 0.1s
...
O1 = [1. 1. 1.]
O2 =
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.7. Hàm tạo mảng đặc biệt. (Tiếp theo)

- ✓ Hàm `eye`: mảng đơn vị, vuông

`np.eye(num)`

num: số hàng = số cột

```
import numpy as np
I = np.eye(3); print('I=\n', I)
[1] ✓ 0.1s
```

```
... I=
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

- ✓ Hàm `tri`: mảng đơn vị, vuông, tam giác dưới

`np.tri(num)`

num: số hàng = số cột

```
import numpy as np
Tri = np.tri(3); print('Tri=\n', Tri)
[1] ✓ 0.7s
```

```
... Tri=
[[1. 0. 0.]
 [1. 1. 0.]
 [1. 1. 1.]]
```

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.7. Hàm tạo mảng đặc biệt. (Tiếp theo)

- ✓ Hàm `tril`: Tạo mảng tam giác dưới, được suy ra từ mảng gốc.

A: Mảng gốc

`np.tril(A)` → Các phần tử phía bên trên đường chéo chính bằng 0

- ✓ Hàm `triu`: Tạo mảng tam giác trên, được suy ra từ mảng gốc.

A: Mảng gốc

`np.triu(A)` → Các phần tử phía bên dưới đường chéo chính bằng 0

```
import numpy as np
A = np.arange(1, 10).reshape(3,3); print('A =\n', A)
TrilA = np.tril(A); print('TrilA =\n', TrilA)
TriuA = np.triu(A); print('TriuA =\n', TriuA)
[1] ✓ 0.1s
...
A =
[[1 2 3]
 [4 5 6]
 [7 8 9]]
TrilA =
[[1 0 0]
 [4 5 0]
 [7 8 9]]
TriuA =
[[1 2 3]
 [0 5 6]
 [0 0 9]]
```

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.8. Tạo mảng bằng cách ghép 2 mảng đã có.

- ✓ Ghép vào hàng: Số cột của 2 mảng phải bằng nhau.

`np.r_[A, B]`

- ✓ Ghép vào cột: Số hàng của 2 mảng phải bằng nhau.

`np.c_[A, B]`

```
import numpy as np
A = np.arange(6).reshape(2, 3); print('A =\n', A)
B = np.arange(6, 12).reshape(2, 3); print('B =\n', B)
ArB = np.r_[A, B]; print('ArB =\n', ArB)
AcB = np.c_[A, B]; print('AcB =\n', AcB)
```

[1] ✓ 0.1s

```
... A =
[[0 1 2]
 [3 4 5]]
B =
[[ 6  7  8]
 [ 9 10 11]]
ArB =
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
AcB =
[[ 0  1  2  6  7  8]
 [ 3  4  5  9 10 11]]
```

7. Sử dụng gói Numpy cơ bản.

```
import numpy as np
```

7.1.9. Mở rộng mảng, thêm phần tử vào mảng đã có.

- ✓ Khi không cần biết trước kích thước của mảng, sử dụng phương thức thêm và gán giá trị cho phần tử.

`A = np.append(A, x)` → Mảng A được thêm một phần tử và gán giá trị x vào đó

```
▶ v
import numpy as np
A = []
for i in range(4):
    x = 3**i
    A = np.append(A, x)
print('A%d =' % (i+1), A)
```

[1] ✓ 0.1s

```
... A0 = []
A1 = [1.]
A2 = [1. 3.]
A3 = [1. 3. 9.]
A4 = [ 1. 3. 9. 27.]
```

7. Sử dụng gói Numpy cơ bản.

`import numpy as np`

7.1.10. Cách tham chiếu đến phần tử của mảng.

✓ Dùng dấu ngoặc vuông `[]` để tham chiếu đến phần tử của mảng

✓ Truyền một tham số → Tham chiếu đến hàng.

`A[n]` → Tham chiếu đến hàng n của mảng A

✓ Truyền hai tham số → Tham chiếu hàng trước, cột sau.

`A[m, n]` → Tham chiếu đến hàng m cột n của mảng A

✓ Dấu hai chấm : đại diện cho mọi hàng (cột).

`A[:, n]` → Tham chiếu đến mọi hàng, cột n

✓ Đếm xuôi: Bắt đầu từ `0`

✓ Đếm ngược: Bắt đầu từ `-1`



```
import numpy as np
A = np.arange(16).reshape(4,4); print('A =\n', A)
print('\nA[1, 0] =', A[1, 0], '\nA[-1, -2] =', A[-1, -2])
print('A[0] =', A[0], '\nA[0, :] =', A[0, :], '\nA[:, -2] =', A[:, -2])
```

[1] ✓ 0.1s

... A =

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

```
A[1, 0] = 4
A[-1, -2] = 14
A[0] = [0 1 2 3]
A[0, :] = [0 1 2 3]
A[:, -2] = [ 2  6 10 14]
```

$$\begin{matrix} & \color{blue}{0} & \color{blue}{1} & \cdots & \color{blue}{-2} & \color{blue}{-1} \\ \color{blue}{0} & a_{0,0} & a_{0,1} & \cdots & a_{0,-2} & a_{0,-1} \\ \color{blue}{1} & a_{1,0} & a_{1,1} & \cdots & a_{1,-2} & a_{1,-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \color{blue}{-2} & a_{-2,0} & a_{-2,1} & \cdots & a_{-2,-2} & a_{-2,-1} \\ \color{blue}{-1} & a_{-1,0} & a_{-1,1} & \cdots & a_{-1,-2} & a_{-1,-1} \end{matrix}$$

7. Sử dụng gói Numpy cơ bản.

```
import numpy.linalg as la
```

7.2. Đại số tuyến tính (Linear algebra) trong Numpy.

- ✓ Hàm `eig` dùng để tính các giá trị riêng và vector riêng bên phải của một mảng vuông.

`la.eig(A)` → Trả kết quả của trị riêng và vector riêng của mảng vuông A



[1]

```
import numpy as np
import numpy.linalg as la
A = np.arange(9).reshape(3,3); print('A =\n', A)
eig_val, eig_vec = la.eig(A)
print('eig_val =', eig_val)
print('eig_vec =\n', eig_vec)
```

✓ 0.1s

... A =
[[0 1 2]
 [3 4 5]
 [6 7 8]]
eig_val = [1.33484692e+01 -1.34846923e+00 -2.48477279e-16]
eig_vec =
[[0.16476382 0.79969966 0.40824829]
 [0.50577448 0.10420579 -0.81649658]
 [0.84678513 -0.59128809 0.40824829]]

7. Sử dụng gói Numpy cơ bản.

```
import numpy.linalg as la
```

7.2. Đại số tuyến tính (Linear algebra) trong Numpy. (Tiếp theo)

- ✓ Hàm `norm` dùng để tính chuẩn của matrix hoặc vector.

`la.norm(X, 1)` → Chuẩn cột

`la.norm(X)` hoặc `la.norm(X, 2)`
→ Chuẩn Euclide

`la.norm(X, np.inf)` → Chuẩn hàng



[1]

```
import numpy as np
import numpy.linalg as la
X = np.arange(9); print('X =', X)
X_Norm1 = la.norm(X, 1); print('X_Norm1 =', X_Norm1)
X_Norm2 = la.norm(X); print('X_Norm2 =', X_Norm2)
X_Norm3 = la.norm(X, np.inf); print('X_Norm3 =', X_Norm3)
```

✓ 0.8s

...

```
X = [0 1 2 3 4 5 6 7 8]
X_Norm1 = 36.0
X_Norm2 = 14.2828568570857
X_Norm3 = 8.0
```

7. Sử dụng gói Numpy cơ bản.

```
import numpy.linalg as la
```

7.2. Đại số tuyến tính (Linear algebra) trong Numpy. (Tiếp theo)

- ✓ Hàm `det` dùng để tính định thức của matrix vuông.

`la.det(A)` → Định thức của A

- ✓ Hàm `inv` dùng để tính nghịch đảo của matrix vuông.

`la.inv(A)` → Nghịch đảo của A

```
import numpy as np
import numpy.linalg as la
A = np.array([[-5, 2, 1], [7, 3, -6], [4, 8, -9]])
print('A =\n', A)
A_det = la.det(A); print('A_det =', A_det)
A_inv = la.inv(A); print('A_inv =\n', A_inv)
```

[1] ✓ 0.1s

... A =
[[-5 2 1]
[7 3 -6]
[4 8 -9]]
A_det = 16.99999999999999
A_inv =
[[1.23529412 1.52941176 -0.88235294]
[2.29411765 2.41176471 -1.35294118]
[2.58823529 2.82352941 -1.70588235]]

7. Sử dụng gói Numpy cơ bản.

```
import numpy.linalg as la
```

7.2. Đại số tuyến tính (Linear algebra) trong Numpy. (Tiếp theo)

- ✓ Hàm `solve` dùng để giải phương trình matrix tuyến tính (hệ phương trình vô hướng tuyến tính)
`la.solve(A, b)` → Nghiệm X của hệ phương trình: $A \times X = b$

```
import numpy as np
import numpy.linalg as la
A = np.array([[-3, 2, 1], [7, 4, -6], [4, 8, -9]])
b = np.array([8, -3, 6])
print('A =\n', A, '\nb =', b)
X = la.solve(A, b); print('=> X = inv(A)*b =', X)
```

[1] ✓ 0.1s

```
... A =
[[ -3  2  1]
 [ 7  4 -6]
 [ 4  8 -9]]
b = [ 8 -3  6]
=> X = inv(A)*b = [-0.95121951  2.15853659  0.82926829]
```

8. Sử dụng gói Matplotlib cơ bản.

```
import matplotlib.pyplot as plt
```

✓ Hàm `plot` dùng để vẽ điểm, đường trong mặt phẳng 2-D.

`plt.plot(x, y)` → Vẽ 1 đường với 2 mảng x, y có cùng dạng (shape)

`plt.plot(x1, y1, x2, y2)` → Vẽ 2 đường với 2 cặp mảng (x1, y1) và (x2, y2) tương ứng có cùng dạng (shape)

((Nếu không muốn kiểu vẽ mặc định thì phải khai báo thêm:

‘DạngĐườngMàuĐánhDấu’

DạngĐường: - nét liền, -. nét gạch chấm ...

Màu: k đen, b xanh nước biển, g xanh lá cây...

ĐánhDấu: o dấu tròn, ^ tam giác thuận, v tam giác ngược...

‘-bs’ nét liền, màu xanh nước biển, đánh dấu ô vuông

`label='Dòng text'` → Tạo nhãn cho đồ thị

`r'$` Ghi công thức, ký tự latin ... theo kiểu latex `$`

`lw=2` → Bề rộng nét vẽ

`plt.xlabel('Dòng text')` → Tạo nhãn cho trục x

`plt.ylabel('Dòng text')` → Tạo nhãn cho trục y

`plt.title('Dòng text')` → Tạo tiêu đề cho hình vẽ

`plt.legend()` → Hiện nhãn của đồ thị

`plt.grid()` → Hiện lưới của đồ thị

`plt.text(x, y, 'Dòng text')` → In dòng text tại vị trí x, y

`plt.axis([x1, x2, y1, y2])` → Vùng hiển thị trên trục x và y))

`plt.show()` → Hiện đồ thị

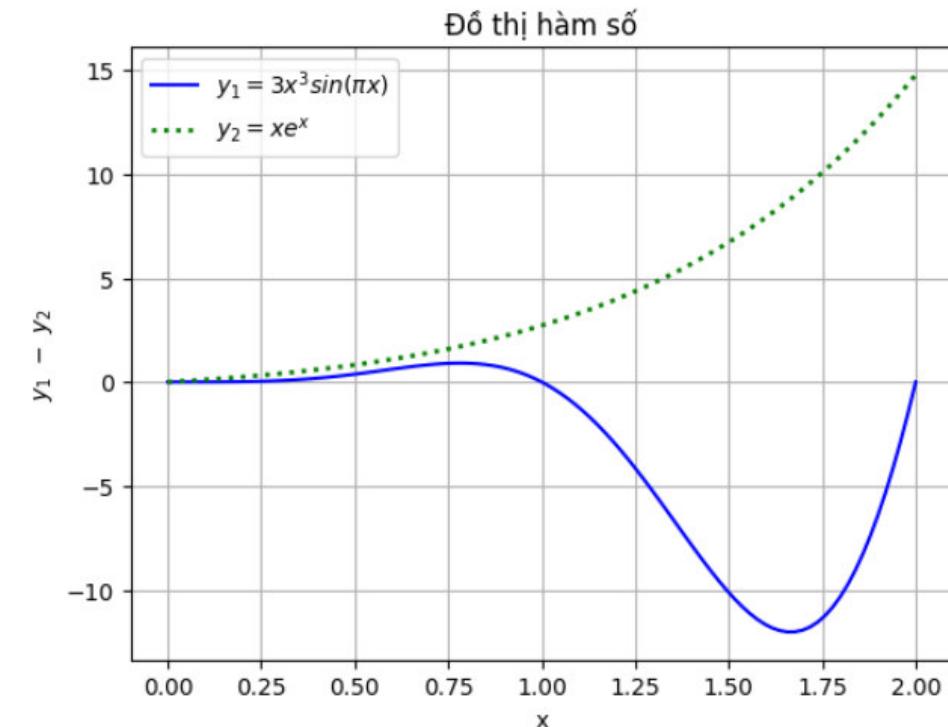
▷ ▾

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2, 100) # Tạo biến x
y1 = 3*x**3*np.sin(np.pi * x) # Tạo hàm y1 = f1(x)
y2 = x*np.exp(x) # Tạo hàm y2 = f2(x)
plt.plot(x,y1,'b', label=r'$y_1 = 3x^3\sin(\pi x)$')
plt.plot(x,y2,:g', lw=2, label=r'$y_2 = xe^x$')
plt.xlabel('x'); plt.ylabel(r'$y_1 \ - \ y_2$')
plt.title('Đồ thị hàm số')
plt.legend(); plt.grid(); plt.show()
```

[1]

✓ 0.6s

...



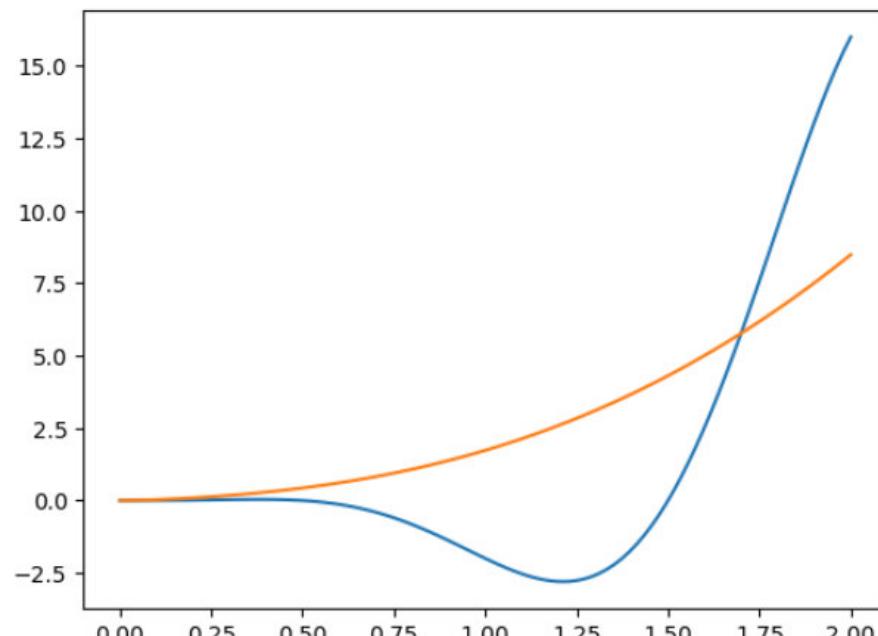
8. Sử dụng gói Matplotlib cơ bản.

```
import matplotlib.pyplot as plt
```

✓ **Hàm figure.**

- Một figure tương ứng với một hình ảnh có thể xuất ra một file ảnh.
- Mặc định khi chỉ vẽ trên một figure thì không cần sử dụng hàm figure.
- Khi muốn sử dụng nhiều hơn một figure thì phải dùng hàm figure.

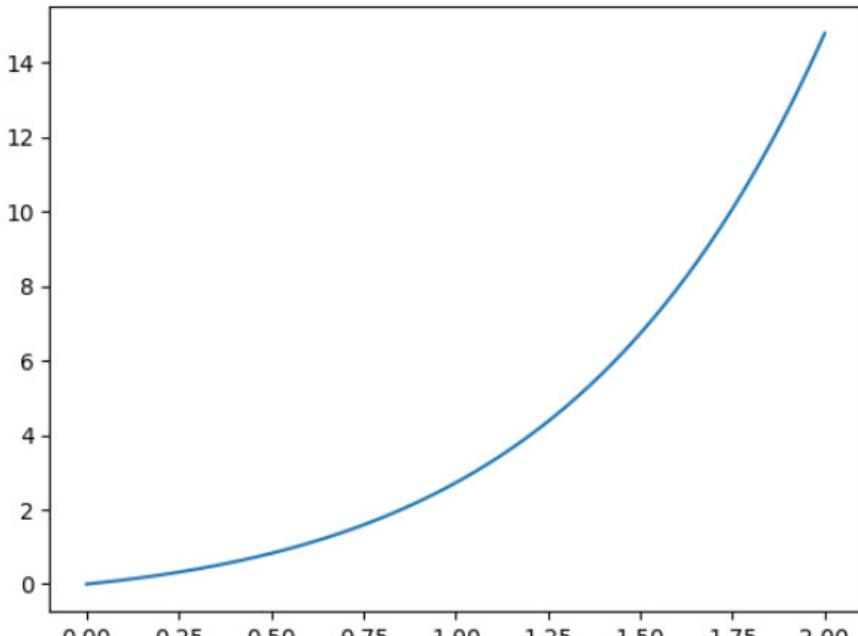
`plt.figure(n)` → Tạo ra một figure thứ n (bắt đầu từ 1).



```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2, 100)
y1 = 2*x**3*np.cos(np.pi * x)
y2 = x*np.exp(x)
y3 = x*np.sqrt(2*x**3+x)
plt.figure(1); plt.plot(x, y1)
plt.figure(2); plt.plot(x, y2)
plt.figure(1); plt.plot(x, y3)
plt.figure(2); plt.show()
```

[1]

✓ 0.5s



8. Sử dụng gói Matplotlib cơ bản.

```
import matplotlib.pyplot as plt
```

- ✓ Hàm `subplot`. Trên một figure, muốn chia làm nhiều vùng con để vẽ thì dùng hàm `subplot`.

`plt.subplot(hck)` ~ `plt.subplot(h, c, k)` → Tạo ra ($h \times c$) vùng con để vẽ và kích hoạt vùng thứ k chuẩn bị vẽ (h, c, k phải là số nguyên).

h là số hàng của subplot.

c là số cột của subplot.

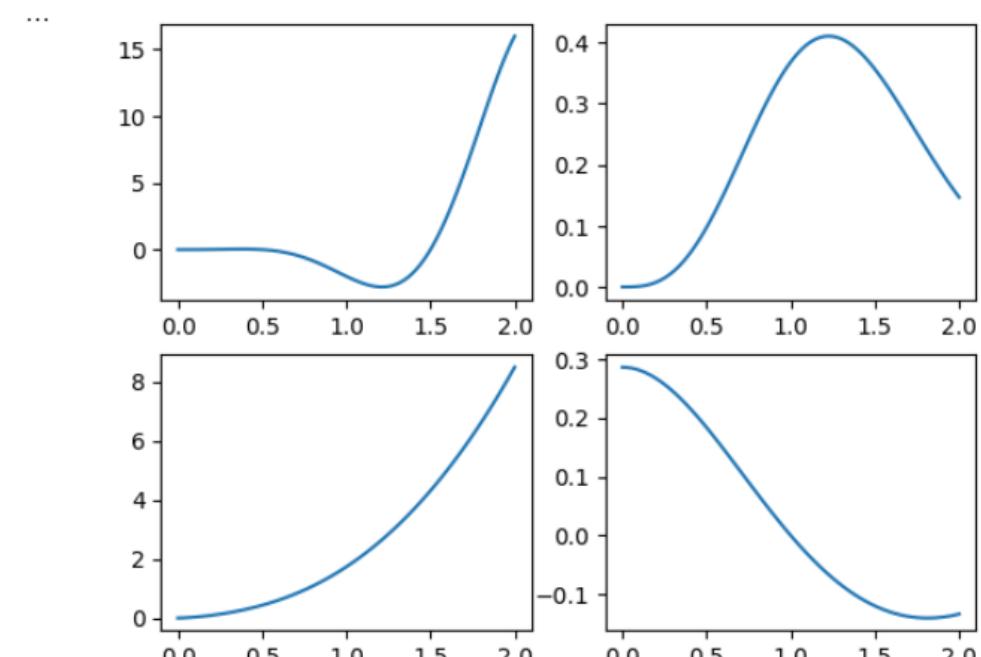
k là chỉ số của vùng muốn kích hoạt: $1 \leq k \leq h \times c$, việc đếm chỉ số k theo thứ tự từ trái qua phải và trên xuống dưới.



```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2, 100)
y1 = 2*x**3*np.cos(np.pi * x)
y2 = x**3*np.exp(-x**2)
y3 = x*np.sqrt(2*x**3+x)
y4 = (x**3-3*x**2+2)/(2*x**2+7)
plt.subplot(221); plt.plot(x, y1)
plt.subplot(222); plt.plot(x, y2)
plt.subplot(223); plt.plot(x, y3)
plt.subplot(224); plt.plot(x, y4); plt.show()
```

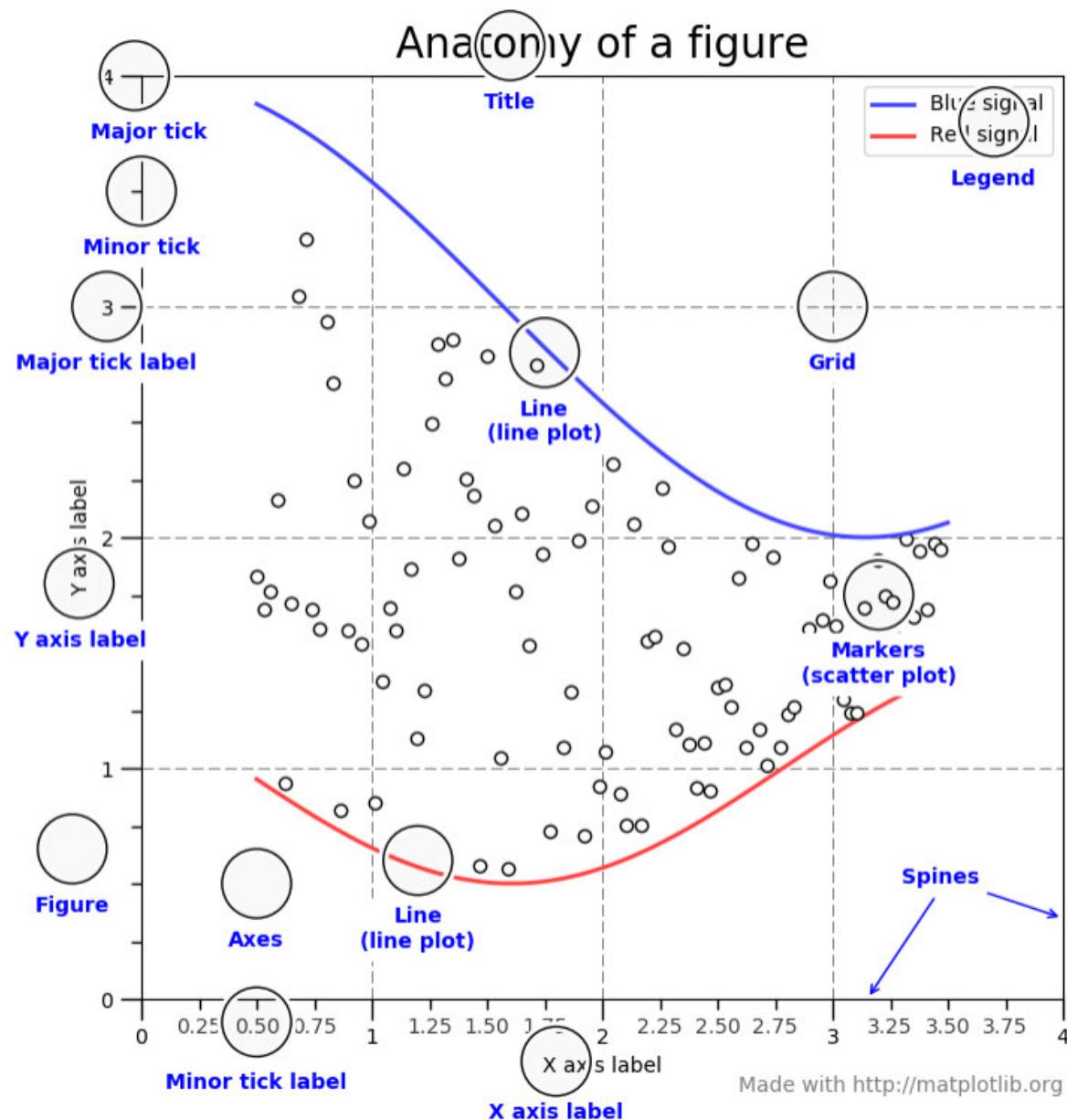
[1]

✓ 0.7s



8. Sử dụng gói Matplotlib cơ bản.

```
import matplotlib.pyplot as plt
```



9. Sử dụng gói Sympy cơ bản.

```
import sympy as sp
```

- ✓ Hàm `Symbol` và `symbols` tạo biến để sử dụng trong biểu thức

Symbol ký tự S viết hoa để tạo một biến đơn
`symbols` số nhiều để tạo nhiều biến cùng lúc

`x = sp.Symbol('x')` → Tạo biến x

`y, z = sp.symbols('y z')` → Tạo hai biến y, z

- ✓ Hàm `subs` thay thế biến và tính biểu thức của hàm
`TenHam.subs([(bienv1, Thay1), ..., (bienn, Thayn)])`

- ✓ Hàm `evalf` tính một biểu thức của hàm sau khi thay thế,
đạo hàm, tích phân...

`TenHam.subs([(bienv1, Thay1), ..., (bienn, Thayn)]).evalf(n)`
n là số chữ số muốn giữ lại trong kết quả, mặc định thì
không cần truyền

- ✓ Hàm `diff` đạo hàm hàm số

`TenHam.diff((bienv1, c1), ..., (bienn, cn))` các cặp tham số
truyền vào là: Tên biến, cấp muốn đạo hàm

- ✓ Hàm `integrate` tích phân hàm số

`TenHam.integrate((b1, cd1, ct1), ..., (bn, cdn, ctn))` từng bộ
ba truyền vào lần lượt là: Tên biến, cận dưới, cận trên

▷ v

```
import sympy as sp
x = sp.Symbol('x')
y, z = sp.symbols('y, z')

f = 2*x**2 * y**2 - 5*x**3*y*z**5 + 7
print('f =', f)

f976 = f.subs([(x, 9), (y, 7), (z, 6)]).evalf()
print('f976 =', f976)

d1fx = f.diff(x)
print('d1fx =', d1fx)

d2fx_d1fy_d3 fz = f.diff((x, 2), (y, 1), (z, 3))
print('d2fx_d1fy_d3 fz =', d2fx_d1fy_d3 fz)

inte_f = f.integrate((x, 0, 2), (y, 1, 3), (z, 3, 7)).evalf(9)
print('inte_f =', inte_f)

[1] ✓ 0.6s
...
f = -5*x**3*y*z**5 + 2*x**2*y**2 + 7
f976 = -198396695.000000
d1fx = -15*x**2*y*z**5 + 4*x*y**2
d2fx_d1fy_d3 fz = -1800*x*z**2
inte_f = -1558636.44
```

9. Sử dụng gói Sympy cơ bản.

```
import sympy as sp
```

- ✓ Hàm `solve` giải phương trình và hệ phương trình đại số tuyến tính.

```
from sympy import linsolve
from sympy.abc import x, y, z
f1 = 3*x + 2*y - z - 1
f2 = 2*x - 2*y + 4*z + 2
f3 = -x + y/2 - z
N0 = linsolve([f1, f2, f3], x, y, z)
print('(x, y, z) =', N0)
```

[1] ✓ 0.4s

... (x, y, z) = {(1, -2, -2)}

- ✓ Hàm `nonlinsolve` giải hệ phương trình phi tuyến.

```
from sympy import Symbol, solve
x = Symbol('x', real=True)
f = x**3 - 3*x**2 + 2
N0 = solve(f, x); print('x =', N0)
```

[1] ✓ 0.4s

... x = [1, 1 - sqrt(3), 1 + sqrt(3)]

- ✓ Hàm `linsolve` giải hệ phương trình đại số tuyến tính.

```
from sympy import symbols, nonlinsolve
x, y, z = symbols('x, y, z', real=True)
f1 = x*y - 1
f2 = 4*x**2 + y**2 - 5
N0 = nonlinsolve([f1, f2], [x, y])
print('(x, y) =', N0)
```

[1] ✓ 0.5s

... (x, y) = {(-1, -1), (-1/2, -2), (1/2, 2), (1, 1)}

10. Hàm xuất định dạng bảng trong gói Tabulate.

```
from tabulate import tabulate
```

- ✓ Hàm `tabulate` dùng để xuất kết quả dạng bảng.

Khi cần in ra kết quả tính toán là mảng 2_D mô tả các đại lượng khác nhau giữa các cột.

Để rõ ràng hơn thì hàng trên cùng cần phải thêm tiêu đề (headers) và cột bên trái đánh số thứ tự của hàng.

Việc đầu tiên là tạo 2 mảng 2-D cho tiêu đề và index rồi ghép cột index và ghép hàng tiêu đề vào mảng sau đó thực hiện in với hàm `tabulate`



```
import numpy as np
from tabulate import tabulate
Bang = np.arange(6, 22).reshape(4, 4)
TieuDe = np.array([['Cột 0', 'Cột 1', 'Cột 2', 'Cột 3', 'Cột 4']])
Index = range(4)
Bang1 = np.c_[Index, Bang]
Bang2 = np.r_[TieuDe, Bang1]
print(tabulate(Bang2, headers='firstrow', tablefmt='rounded_outline'))
```

[1]

...

Cột 0	Cột 1	Cột 2	Cột 3	Cột 4
0	6	7	8	9
1	10	11	12	13
2	14	15	16	17
3	18	19	20	21

```
print(tabulate(Table, headers='firstrow', tablefmt='Kiểu_Đường'))
```

`Table` là tên của bảng đã ghép tiêu đề và index

`headers='firstrow'` lấy hàng đầu tiên làm tiêu đề

`tablefmt='Kiểu_Đường'` có thể chọn một trong số kiểu đường sau:

`rounded_outline` `mixed_grid` `double_grid` `fancy_grid` `simple_outline`
`heavy_outline` `mixed_outline` `double_outline` `fancy_outline` `outline`

Chương 1:

SAI SỐ

Nội dung của chương.

1.1. Khái niệm.

1.2. Các loại sai số.

 1.2.1 Sai số thực.

 1.2.2 Sai số tuyệt đối. (Sai số tuyệt đối giới hạn)

 1.2.3 Sai số tương đối.

 1.2.4. Sai số làm tròn.

 1.2.4.1. Quy tắc quá bán.

 1.2.4.2. Quy tắc làm tròn theo bất đẳng thức.

 1.2.4.3. Chữ số có nghĩa.

 1.2.4.4. Chữ số tin cậy. (Số chắc chắn)

 1.2.5. Sai số tính toán.

 1.2.6. Sai số phương pháp.

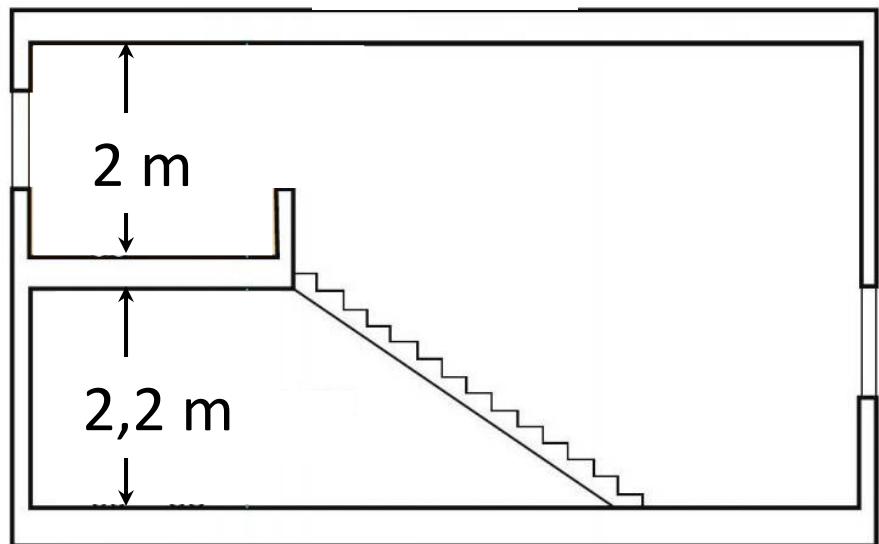
 1.2.7. Sai số hàm số.

1.1. Khái niệm.

Là sự chênh lệch về giá trị của một đại lượng nào đó so với giá trị chính xác, thực tế hay tính toán thông qua quá trình đo đạc, thu thập dữ liệu.

Ví dụ 1.1.

Thiết kế



Hoàn thiện



→ Sự khác biệt kích thước 2m với 1,95m và 2,2m với 2,28m khi thiết kế so với khi hoàn thiện gọi là sai số.

1.2. Các loại sai số.

1.2.1 Sai số thực.

Sai số thực Δ là giá trị chênh lệch của phép đo thực tế so với giá trị đúng của bài toán.

Gọi A : Giá trị đúng của bài toán.

a^* : Giá trị đo thực tế và đúng một cách tuyệt đối.

Sai số thực sẽ là: $\Delta = |A - a^*|$

Lưu ý: Phép đo về số lượng nguyên ($1, 2 \dots n$ cái, quả,...) sẽ cho giá trị đo đúng tuyệt đối. Các phép đo khác, giá trị Δ chỉ mang ý nghĩa về mặc lí thuyết, giá trị đo được chỉ thể hiện kết quả gần đúng.

1.2.2 Sai số tuyệt đối. (Sai số tuyệt đối giới hạn)

Sai số tuyệt đối Δ_a là giá trị giới hạn mức độ sai lệch của giá trị thực tế so với tính toán.

Gọi A : Giá trị đúng của bài toán.

a : Là giá trị gần đúng hay giá trị xấp xỉ của A , nếu a khá gần A ($a < A$ hoặc $a > A$).

Sai số tuyệt đối sẽ là: $\Delta_a = |A - a|$ hay $A = a \pm \Delta_a$

1.2.3 Sai số tương đối.

Sai số tương đối ε_a của số xấp xỉ a là tỉ số giữa sai số tuyệt đối của số xấp xỉ với trị tuyệt đối của nó. $\varepsilon_a = \frac{\Delta a}{|a|}$

Ví dụ 1.2.

Chiều dài của cây cầu và đinh tán có giá trị đúng lần lượt là 50000 cm và 10 cm. Khi dùng thước để đo thì nhận được kết quả lần lượt là 49999 cm và 9 cm. Xác định sai số thực và sai số tương đối của hai đại lượng trên.

Giải.

1/ Sai số thực.

$$\text{Đối với cầu: } \Delta_{\text{cầu}} = 50000 \text{ cm} - 49999 \text{ cm} = 1 \text{ cm}$$

$$\text{Đối với đinh tán: } \Delta_{\text{đinh}} = 10 \text{ cm} - 9 \text{ cm} = 1 \text{ cm}$$

2/ Sai số tương đối.

$$\text{Đối với cầu: } \varepsilon_{\text{cầu}} = \frac{1 \text{ cm}}{50000 \text{ cm}} \cdot 100\% = 0,002\%$$

$$\text{Đối với đinh tán: } \varepsilon_{\text{đinh}} = \frac{1 \text{ cm}}{10 \text{ cm}} \cdot 100\% = 0,1\%$$

1.2.4. Sai số làm tròn.

Là giá trị chênh lệch giữa giá trị trước và sau khi làm tròn của giá trị đo.

Qui tắc làm tròn:

1.2.4.1. Quy tắc quá bán. Là làm tròn số dựa trên chữ số đứng bên phải theo nguyên tắc:

Nếu chữ số bỏ đi đầu tiên ≥ 5 thì cộng thêm 1 vào chữ số trước đó.

Nếu chữ số bỏ đi đầu tiên < 5 thì giữ nguyên chữ số trước đó.

Ví dụ 1.3.

Cần làm tròn số $\pi = 3.141592653589793 \dots$

Giải.

Làm tròn đến phần chục: $\pi \approx 3.1$

Làm tròn đến phần trăm: $\pi \approx 3.14$

Làm tròn đến phần nghìn: $\pi \approx 3.142$

Làm tròn đến phần triệu: $\pi \approx 3.1416$

...

Hàm làm tròn trong Python: `Round(A,b)`

Trong đó A là số cần làm tròn, b là số chữ số sau dấu phẩy được giữ lại.

```
>>> A=15/7; print ('A = ',A)
A = 2.142857142857143
>>> b=2; print ('b = ',b)
b = 2
>>> C=round(A,b); print ('C = ',C)
C = 2.14
```

1.2.4.2. Quy tắc làm tròn theo bất đẳng thức.

Vẽ trái (nhỏ) làm tròn xuống, vẽ phải (lớn) làm tròn lên.

Ví dụ 1.4.

Cho bất đẳng thức $-2,43157\dots \leq x \leq 4,21536\dots$

Viết lại bất đẳng thức sau khi làm tròn.

Giải.

Làm tròn đến phần chục: $-2,4 \leq x \leq 4,3$

Làm tròn đến phần trăm: $-2,43 \leq x \leq 4,22$

Làm tròn đến phần nghìn: $-2,431 \leq x \leq 4,216$

Làm tròn đến phần triệu: $-2,4315 \leq x \leq 4,2154$

1.2.4.3. Chữ số có nghĩa.

Bao gồm tất cả các chữ số tính từ chữ số đầu tiên khác 0 từ trái sang.

Ví dụ 1.5.

Số: 0.0**1534** có 4 chữ số có nghĩa.

Số: 0.00**127900** có 6 chữ số có nghĩa.

Số: **354.8679** có 7 chữ số có nghĩa.

1.2.4.4. Chữ số tin cậy. (Số chắc chắn)

Là chữ số không bị ảnh hưởng bởi sai số. Một số a với sai số Δa , nếu $\Delta a \leq 0.5 \times 10^{-k}$ thì từ chữ số thứ k (sau dấu phẩy) sang trái là chữ số tin cậy.

Ví dụ 1.6.

Số $a = 3.0543217$ với sai số $\Delta a = 0.6 \times 10^{-3}$. Do $\Delta a = 0.6 \times 10^{-3} \leq 0.5 \times 10^{-2}$

Vậy, từ chữ số đầu tiên sau dấu phẩy đến chữ số thứ 2 là chữ số đáng tin: 3, 0, 5

1.2.5. Sai số tính toán.

Sai số tính toán là sai số được tạo ra do tất cả các lần xấp xỉ và làm tròn.

1.2.6. Sai số phương pháp.

Sai số phương pháp là sai số tạo ra do việc thay bài toán phức tạp thành bài toán đơn giản.

1.2.7. Sai số hàm số.

Là sai số đến từ sai số các biến sử dụng trong hàm do các số được xấp xỉ hoặc số đã được làm tròn. Hàm số n biến $f(x_1, x_2, \dots, x_n)$ với các biến có sai số riêng $\Delta x_1, \Delta x_2, \dots, \Delta x_n$.

Thì sai số hàm số sẽ là: $\Delta f = \sum_{k=1}^n \left| \frac{\partial f}{\partial x_k} \right| \cdot \Delta x_k$

Cho hàm số $f(x, y, z) = 4x^3 + 2y^2 + 3z$. Xác định sai số của hàm $f_{(x=1, y=5, z=2)}$

Giải.

$$\left. \begin{array}{l} \Delta f_x = \frac{\partial f}{\partial x} \cdot \Delta x = 12x^2 \cdot \Delta x \\ \Delta f_y = \frac{\partial f}{\partial y} \cdot \Delta y = 4y \cdot \Delta y \\ \Delta f_z = \frac{\partial f}{\partial z} \cdot \Delta z = 3 \cdot \Delta z \end{array} \right\} \Rightarrow \Delta f = \Delta f_x + \Delta f_y + \Delta f_z = 12x^2 \cdot \Delta x + 4y \cdot \Delta y + 3 \cdot \Delta z$$

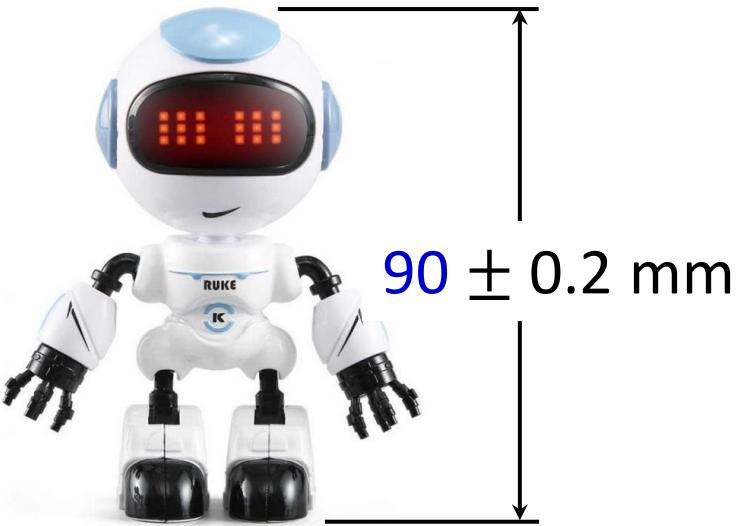
Thay $x = 1, y = 5, z = 2$ vào:

$$\Rightarrow \Delta f = 12 \cdot \Delta x + 20 \cdot \Delta y + 3 \cdot \Delta z$$

Ví dụ 1.8.

Sản phẩm Robot sau khi hoàn thành thì kích thước chiều cao phải thỏa mãn $90 \pm 0.2\text{mm}$ như hình dưới. Nhân viên đảm nhận công việc kiểm tra trước khi đóng gói bằng cách đo một lô gồm 6 mẫu và kết quả thu được như bảng dưới.

1. Tính các sai số tuyệt đối, tương đối, làm tròn cho tất cả các mẫu.
2. Một lô hàng không đạt khi có ít nhất một sản phẩm không đạt. Lô này có đạt không?



Mẫu thứ	1	2	3	4	5	6
Kết quả đo	90.04	89.94	90.23	90.17	89.91	90.08
Làm tròn	90.00	89.90	90.20	90.20	89.90	90.10

Giải.

Ví dụ 1.8.

Mẫu thứ	1
Kết quả đo	90.04
Làm tròn	90.00

Sai số tuyệt đối: $\Delta_a = |A - a| = |90 - 90.04| = 0.04\text{mm}$

Sai số tương đối: $\varepsilon_a = \frac{\Delta_a}{|a|} = \frac{0.04}{90.04} = 0.044\%$

Sai số làm tròn: $\varepsilon_a^* = |90 - 90.00| = 0.00\text{mm}$

Mẫu thứ	2
Kết quả đo	89.94
Làm tròn	89.90

Sai số tuyệt đối: $\Delta_a = |A - a| = |90 - 89.94| = 0.06\text{mm}$

Sai số tương đối: $\varepsilon_a = \frac{\Delta_a}{|a|} = \frac{0.06}{89.94} = 0.067\%$

Sai số làm tròn: $\varepsilon_a^* = |90 - 89.90| = 0.10\text{mm}$

Mẫu thứ	3
Kết quả đo	90.23
Làm tròn	90.20

Sai số tuyệt đối: $\Delta_a = |A - a| = |90 - 90.23| = 0.23\text{mm}$

Sai số tương đối: $\varepsilon_a = \frac{\Delta_a}{|a|} = \frac{0.23}{90.23} = 0.255\%$

Sai số làm tròn: $\varepsilon_a^* = |90 - 90.20| = 0.20\text{mm}$

Mẫu thứ	4
Kết quả đo	90.17
Làm tròn	90.20

Sai số tuyệt đối: $\Delta_a = |A - a| = |90 - 90.17| = 0.17\text{mm}$

Sai số tương đối: $\varepsilon_a = \frac{\Delta_a}{|a|} = \frac{0.17}{90.17} = 0.189\%$

Sai số làm tròn: $\varepsilon_a^* = |90 - 90.20| = 0.20\text{mm}$

Mẫu thứ	5
Kết quả đo	89.91
Làm tròn	89.90

Sai số tuyệt đối: $\Delta_a = |A - a| = |90 - 89.91| = 0.09\text{mm}$

Sai số tương đối: $\varepsilon_a = \frac{\Delta_a}{|a|} = \frac{0.09}{89.91} = 0.100\%$

Sai số làm tròn: $\varepsilon_a^* = |90 - 89.90| = 0.10\text{mm}$

Mẫu thứ	6
Kết quả đo	90.08
Làm tròn	90.10

Sai số tuyệt đối: $\Delta_a = |A - a| = |90 - 90.08| = 0.08\text{mm}$

Sai số tương đối: $\varepsilon_a = \frac{\Delta_a}{|a|} = \frac{0.08}{90.08} = 0.089\%$

Sai số làm tròn: $\varepsilon_a^* = |90 - 90.10| = 0.10\text{mm}$

Yêu cầu đặt ra về kích thước chiều cao của Robot là $90 \pm 0.2\text{ mm}$, có nghĩa là sai số tuyệt đối không được vượt quá **0.2 mm**. Trong khi đó, mẫu thứ 3 có $\Delta_a = 0.23\text{mm}$ không đạt. Vậy, lô hàng không đạt.

Ví dụ 1.9.

Bình lập phương đựng đầy nước có kích thước đo được $15 \times 15 \times 15\text{mm}$, phép đo có sai số: $\Delta_1 = 0.2\text{mm}$ trên cả 3 chiều đo. Bình trụ có khắc vạch chia với sai số $\Delta_2 = 0.1\text{ml}$ chứa 800ml đọc được trên vạch chia. Rót cả hai bình chứa nước vào ấm đun, hỏi ấm đun chứa bao nhiêu ml nước (tính luôn sai số).



+



=



Giải.

Thể tích nước trong ấm đun:

$$V = V_1 + V_2 = x^3\text{ml} + V_2 = 15^3\text{ml} + 800\text{ml} = 4175\text{ml}$$

Sai số của thể tích nước trong ấm đun:

$$\Delta V = 3 \times x^2 \times \Delta x + \Delta V_2 = 3 \times 15^2 \times 0.2 + 0.1 = 135.1\text{ml}$$

Vậy, thể tích nước chứa trong ấm đun sẽ là: $V = 4175 \pm 135.1\text{ml}$

Xác định và làm tròn điều kiện của x theo hàng phần chục để biểu thức thỏa mãn điều kiện: $y = \sqrt{(2.0619814 - x)(x - 5.2147094)}$

Giải.

Để biểu thức có nghĩa thì biểu thức trong căn bậc hai phải dương:

$$(2.0619814 - x)(x - 5.2147094) \geq 0 \Rightarrow 2.0619814 \leq x \leq 5.2147094$$

Làm tròn đến hàng chục: $2.0 \leq x \leq 5.3$, nhưng khi $x = 2$ thì biểu thức không có nghĩa.
Vậy, kết quả làm tròn theo điều kiện sẽ là: $2.1 \leq x \leq 5.3$

Ví dụ 1.11.

Cho bảng số liệu gồm kết quả đo và sai số của 4 mẫu như bảng dưới.

- Làm tròn đến hàng chục của các giá trị đo theo qui tắc quá bán.
- Xác định chữ số có nghĩa, chữ số tin cậy của các mẫu.

Mẫu thứ	1	2	3	4
Kết quả đo	17.0134	0.07254	2754.4	1578
Sai số	0.014	0.0003	0.1	0.5

Giải.

Mẫu thứ	1
Kết quả đo	17.0134
Sai số	0.014

Chữ số sau khi làm tròn đến hàng chục: 17.0

$$0.014 \leq 0.5 \times 10^{-k} = 0.5 \times 10^{-1} \text{ nên chữ số tin cậy: } 1, 7, 0$$

Chữ số có nghĩa: 17.0134

Ví dụ 1.11.

Mẫu thử	2
Kết quả đo	0.07254
Sai số	0.0003

Chữ số sau khi làm tròn đến hàng chục: 0.1

$$0.0003 \leq 0.5 \times 10^{-k} = 0.5 \times 10^{-3} \text{ nên chữ số tin cậy: } 0, 0, 7, 2$$

Chữ số có nghĩa: 7, 2, 5, 4

Mẫu thử	3
Kết quả đo	2754.4
Sai số	0.1

Chữ số sau khi làm tròn đến hàng chục: 2754.4

$$0.1 \leq 0.5 \times 10^{-k} = 0.5 \times 10^1 \text{ nên chữ số tin cậy: } 2, 7, 5, 4$$

Chữ số có nghĩa: 2, 7, 5, 4, 5

Mẫu thử	4
Kết quả đo	1578
Sai số	0.5

Chữ số sau khi làm tròn đến hàng chục: 1578

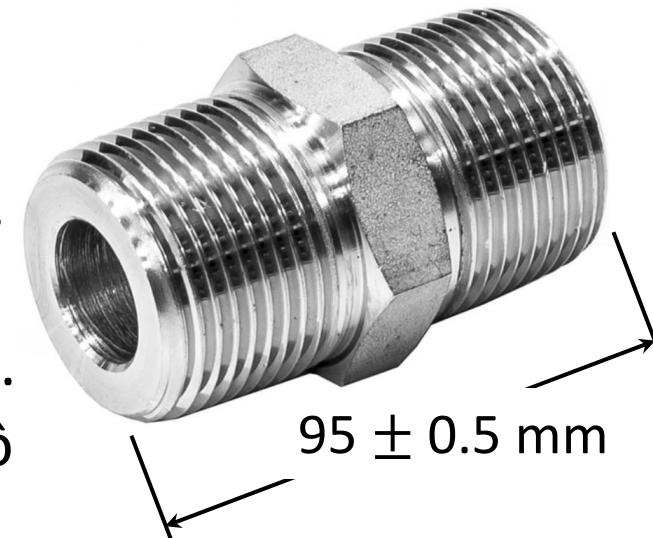
$$0.5 \leq 0.5 \times 10^{-k} = 0.5 \times 10^1 \text{ nên chữ số tin cậy: } 1, 5, 7, 8$$

Chữ số có nghĩa: 1, 5, 7, 8

Bài tập 1.1.

Bộ phận KCS (kiểm tra chất lượng sản phẩm) trong một nhà máy, tiến hành đo chiều dài của 10 mẫu ren nối tiện trong 1 lô sản phẩm. Với những mẫu có chiều dài thỏa mãn $95 \pm 0.5\text{mm}$ thì đạt. Kết quả thu được như bảng dưới.

- Tính các sai số tuyệt đối, tương đối, làm tròn cho tất cả các mẫu.
- Một lô hàng không đạt khi có ít nhất hai sản phẩm không đạt. Lô này có đạt không?



Mẫu thứ	1	2	3	4	5	6	7	8	9	10
Kết quả đo	95.12	95.09	94.85	95.04	94.78	95.16	94.70	94.82	95.02	95.07
Làm tròn	95.10	95.10	94.90	95.00	94.80	95.10	94.70	94.80	95.00	95.10

Bài tập 1.2.

Xác định và làm tròn điều kiện của x theo hàng phần chục để biểu thức thỏa mãn điều kiện: $y = \frac{1}{\sqrt{x^2 - 3.45x + 7.546793228}}$

Bài tập.

Bài tập 1.3.

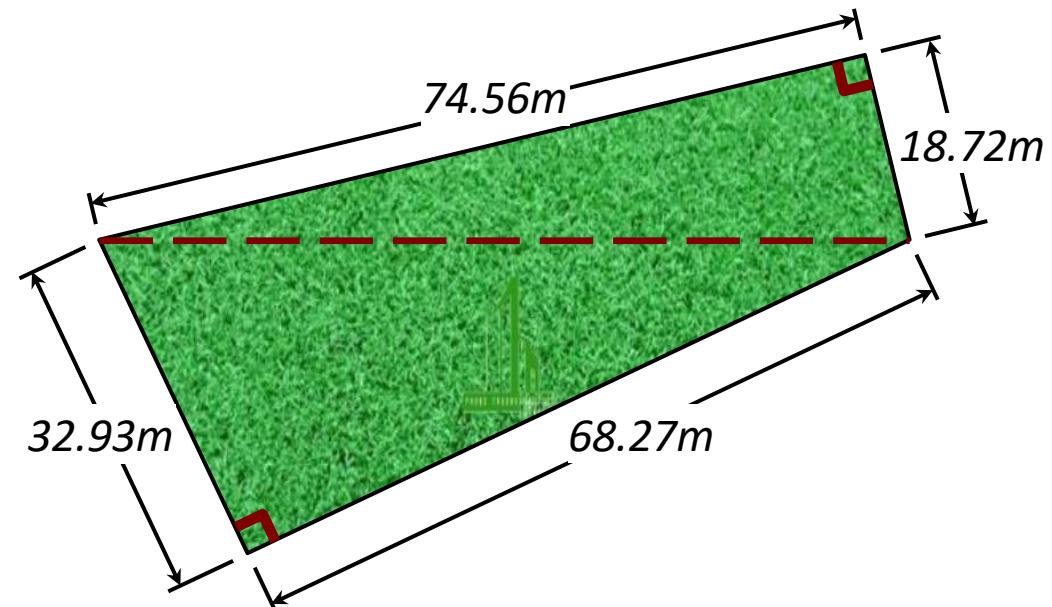
Cho bảng số liệu gồm kết quả đo và sai số của 5 mẫu như bảng dưới.

- Làm tròn đến hàng trăm của các giá trị đo theo qui tắc quá bán.
- Xác định chữ số có nghĩa, chữ số tin cậy của các mẫu.

Mẫu thứ	1	2	3	4	5
Kết quả đo	11.0107	0.04754	3574.8	75988	1549.32
Sai số	0.0124	0.0004	0.2	0.5	0.1

Bài tập 1.4.

Một thửa đất hình tứ diện có hai góc vuông. Chiều dài các cạnh đo được như trên hình vẽ, với sai số của phép đo là $\Delta = 0.5\text{cm}$. Xác định diện tích của thửa đất. (tính luôn sai số)



Chương 2:

XÂP XỈ CỦA ĐẠO HÀM VÀ TÍCH PHÂN

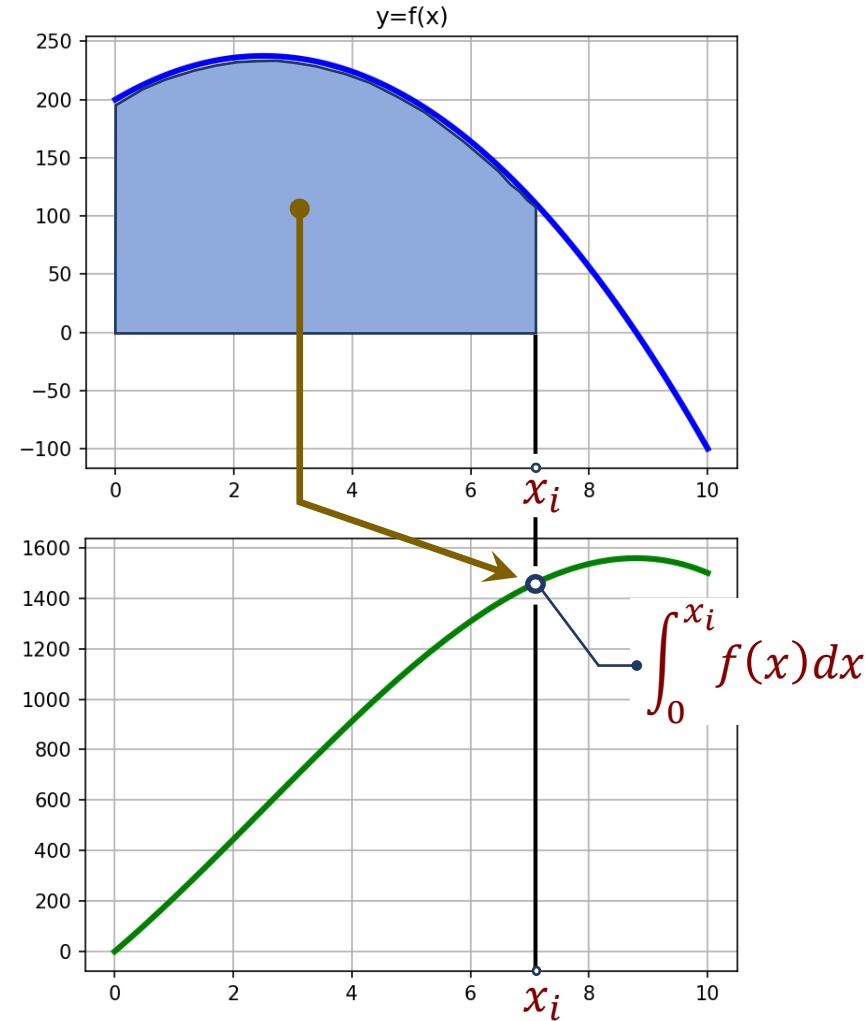
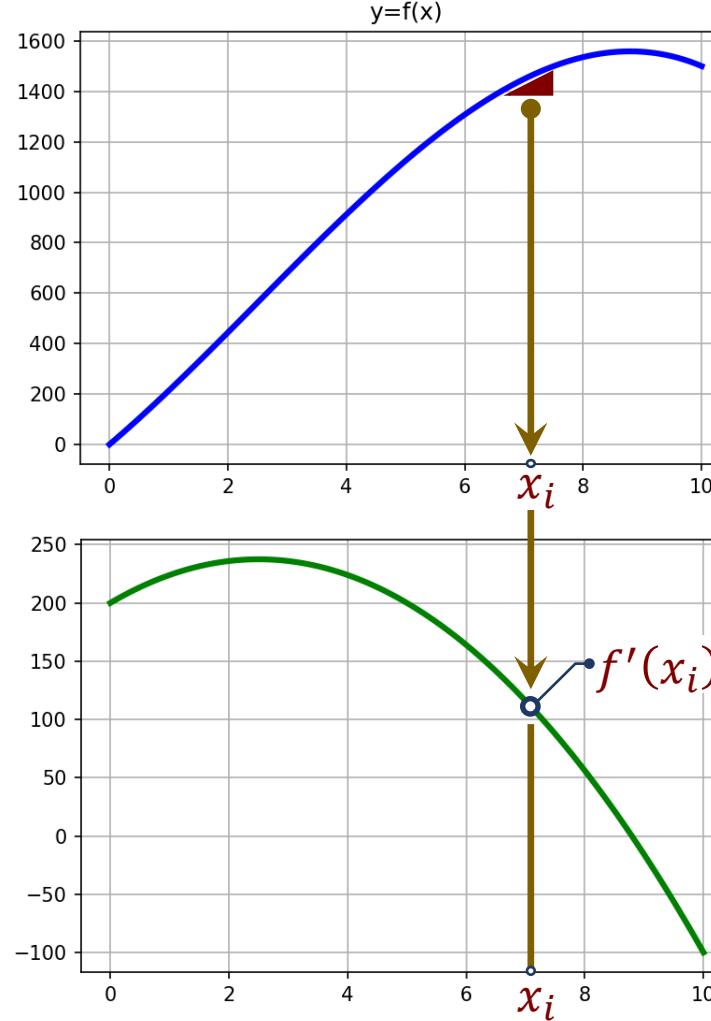
Nội dung của chương.

- 2.1. Khái niệm.
- 2.2. Xấp xỉ của đạo hàm.
 - 2.2.1. Xấp xỉ của đạo hàm cấp 1.
 - 2.2.2. Xấp xỉ của đạo hàm cấp cao.
- 2.3. Xấp xỉ của tích phân xác định.
 - 2.3.1. Công thức tích phân Newton-Leibnitz.
 - 2.3.2. Công thức tích phân Newton-Cotes.
 - 2.3.3. Công thức hình thang (Trapezoidal).
 - 2.3.4. Công thức Simpson.
 - 2.3.5. Công thức Gauss cầu phương.

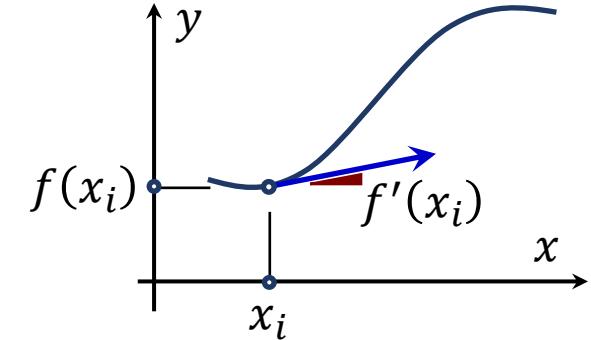
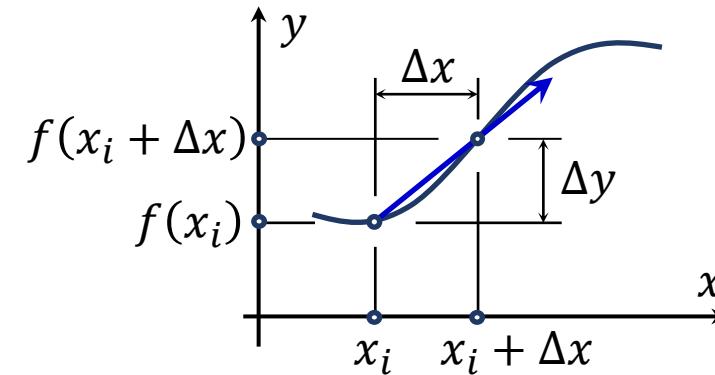
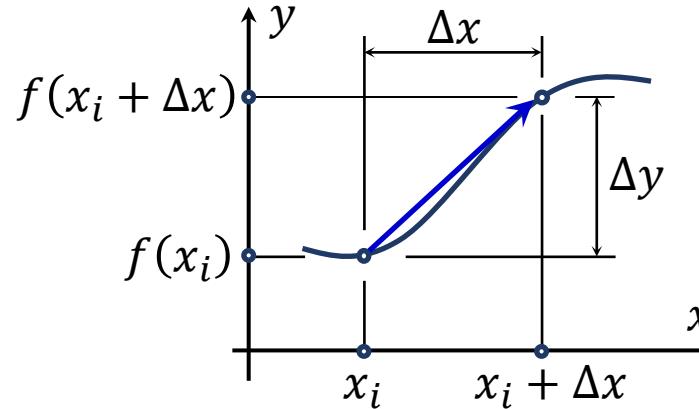
2.1. Khái niệm.

Đạo hàm và tích phân là 2 đại lượng phổ biến trong kỹ thuật.

Đạo hàm là hệ số góc, còn tích phân là diện tích hình thang cong của hàm số.



2.2. Xấp xỉ của đạo hàm.



Xét sai phân: $\frac{\Delta y}{\Delta x} = \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$

Khi $\Delta x \rightarrow 0$ thì sai phân sẽ trở thành đạo hàm của hàm số:

$$y'_i \equiv f'(x_i) \equiv \frac{dy}{dx} = \lim_{\Delta x \rightarrow 0} \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$$

2.2. Xấp xỉ của đạo hàm.

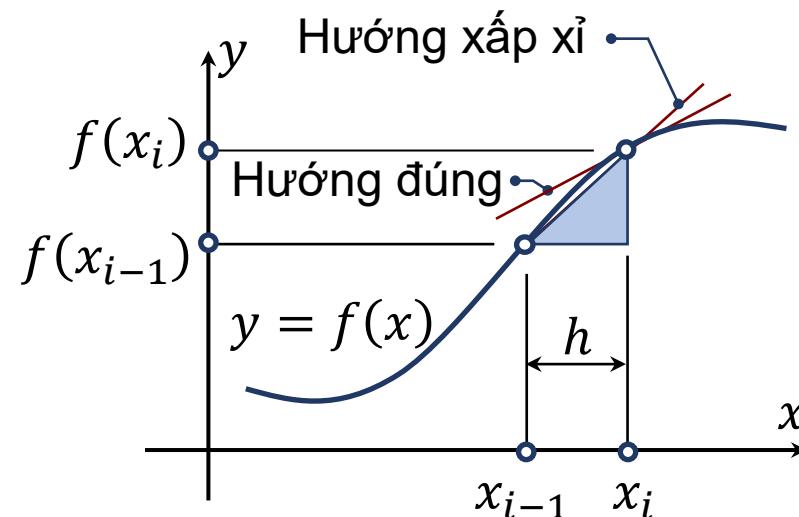
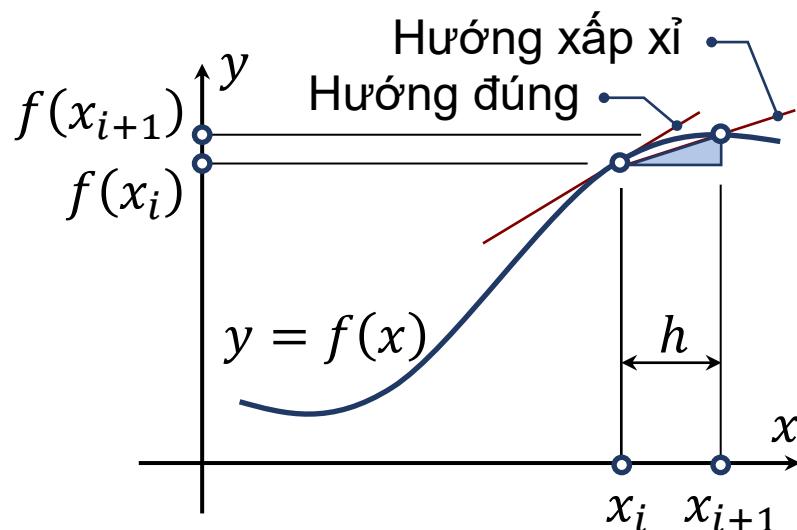
2.2.1. Xấp xỉ của đạo hàm cấp 1.

Khai triển chuỗi Taylor cho hàm $f(x)$ tại vị trí x_i với $h = x_{i+1} - x_i$:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(x_i)}{3!}h^3 + \dots + \frac{f^{(n)}(x_i)}{n!}h^n$$

Chỉ xét đến đạo hàm cấp 1, ta có sai phân tiến: $f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)$

Tương tự cho sai phân lùi: $f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + O(h)$

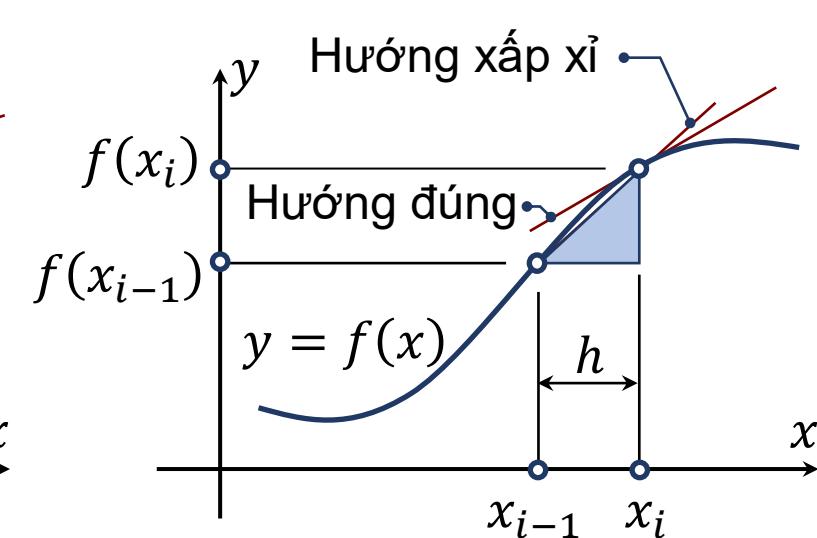
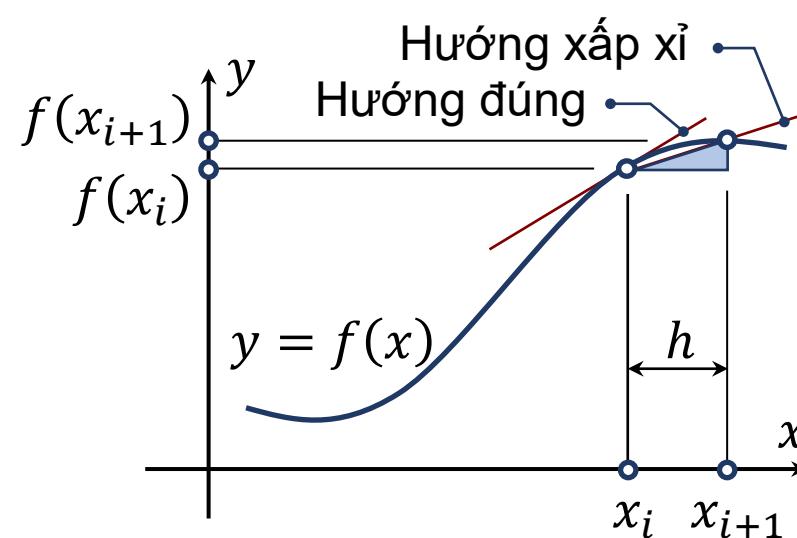
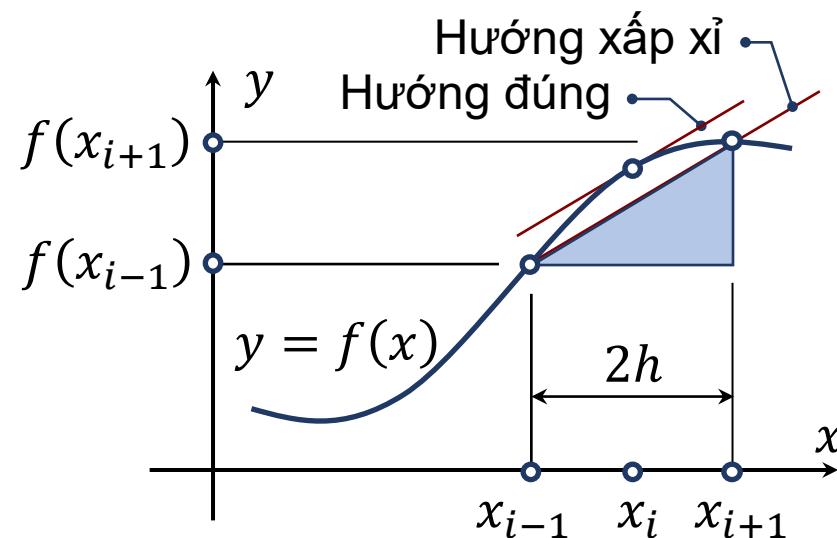


2.2.1. Xấp xỉ của đạo hàm cấp 1.

Để tăng độ chính xác của đạo hàm lên cấp 2 \rightarrow tăng thêm 1 điểm xấp xỉ để tạo ra đối xứng qua điểm cần tính đạo hàm x_i .

Lúc này ta có công thức sai phân hướng tâm (xấp xỉ đạo hàm cấp 1):

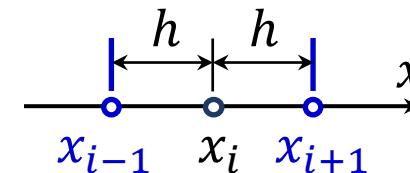
$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}} + O(h^2) = \frac{f(x_i + h) - f(x_i - h)}{2h} + O(h^2)$$



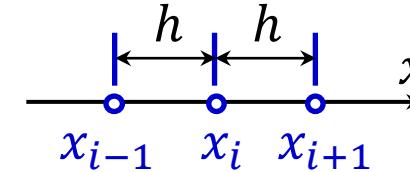
2.2.2. Xấp xỉ của đạo hàm cấp cao.

Tiến hành rút các số hạng đạo hàm cần tìm trong chuỗi Taylor ta nhận được xấp xỉ đạo hàm từ cấp 1 đến cấp 4 với sai số cấp 2:

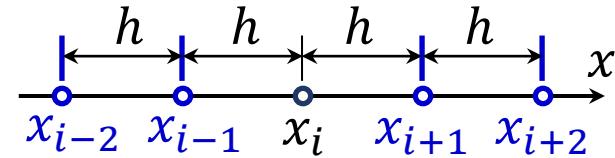
Cấp 1:
$$f^I(x_i) = \frac{f(x_i + h) - f(x_i - h)}{2h} + O(h^2)$$



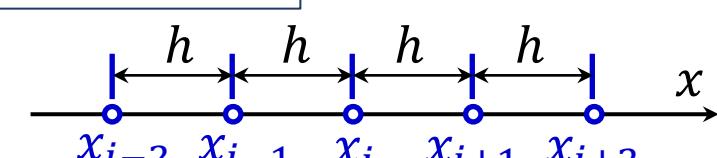
Cấp 2:
$$f^{II}(x_i) = \frac{f(x_i + h) - 2f(x_i) + f(x_i - h)}{h^2} + O(h^2)$$



Cấp 3:
$$f^{III}(x_i) = \frac{f(x_i + 2h) - 2f(x_i + h) + 2f(x_i - h) - f(x_i - 2h)}{2h^3} + O(h^2)$$



Cấp 4:
$$f^{IV}(x_i) = \frac{f(x_i + 2h) - 4f(x_i + h) + 6f(x_i) - 4f(x_i - h) + f(x_i - 2h)}{h^4} + O(h^2)$$



2.2.2. Xấp xỉ của đạo hàm cấp cao.

Ví dụ 2.1. Tính xấp xỉ đạo hàm từ cấp 1 đến cấp 4 bằng sai phân hướng tâm với bước $h = 0.01$ tại vị trí $x = 2$ của hàm số: $f(x) = 2x^4 - 3x^5 + x^6$, đánh giá sai số.

Giải. Đạo hàm chính xác:

$$\begin{aligned} f_{cx}^I \Big|_{x_i=2} &= 8x^3 - 15x^4 + 6x^5 = 16; \quad f_{cx}^{II} \Big|_{x_i=2} = 24x^2 - 60x^3 + 30x^4 = 96 \\ f_{cx}^{III} \Big|_{x_i=2} &= 48x - 180x^2 + 120x^3 = 336; \quad f_{cx}^{IV} \Big|_{x_i=2} = 48 - 360x + 360x^2 = 768 \\ f^I \Big|_{x_i=2, h=0.01} &= \frac{f(x_i + h) - f(x_i - h)}{2h} = 16.01; \text{ Sai số: } \varepsilon_{(I)} = \left| \frac{f_{cx}^I - f^I}{f_{cx}^I} \right| \times 100 = 3.5 \times 10^{-2}\% \\ f^{II} \Big|_{x_i=2, h=0.01} &= \frac{f(x_i + h) - 2f(x_i) + f(x_i - h)}{h^2} = 96.01; \text{ Sai số: } \varepsilon_{(II)} = \left| \frac{f_{cx}^{II} - f^{II}}{f_{cx}^{II}} \right| \times 100 = 6.7 \times 10^{-3}\% \\ f^{III} \Big|_{x_i=2, h=0.01} &= \frac{f(x_i + 2h) - 2f(x_i + h) + 2f(x_i - h) - f(x_i - 2h)}{2h^3} = 336.03 \\ &\qquad\qquad\qquad \text{Sai số: } \varepsilon_{(III)} = \left| \frac{f_{cx}^{III} - f^{III}}{f_{cx}^{III}} \right| \times 100 = 8 \times 10^{-3}\% \\ f^{IV} \Big|_{x_i=2, h=0.01} &= \frac{f(x_i + 2h) - 4f(x_i + h) + 6f(x_i) - 4f(x_i - h) + f(x_i - 2h)}{h^4} = 768.01 \\ &\qquad\qquad\qquad \text{Sai số: } \varepsilon_{(IV)} = \left| \frac{f_{cx}^{IV} - f^{IV}}{f_{cx}^{IV}} \right| \times 100 = 1.6 \times 10^{-3}\% \end{aligned}$$

Ví dụ 2.1.

```
1 def f(x): y=2*x**4-3*x**5+x**6; return y           d1=16.00
2 def diff(x, order):                                d2=96.00
3     h=0.01                                         d3=336.00
4     d1y=(f(x+h)-f(x-h))/2/h                         d4=768.00
5     d2y=(f(x+h)-2*f(x)+f(x-h))/h**2                  df/dx = 16.01; err=0.03500056 o/o
6     d3y=(f(x+2*h)-2*f(x+h)+2*f(x-h)-f(x-2*h))/2/h**3 d2f/dx2 = 96.01; err=0.00666669 o/o
7     d4y=(f(x+2*h)-4*f(x+h)+6*f(x)-4*f(x-h)+f(x-2*h))/h**4 d3f/dx3 = 336.03; err=0.00803571 o/o
8     if order==1: dif=d1y                            d4f/dx4 = 768.01; err=0.00156222 o/o
9     elif order==2: dif=d2y
10    elif order==3: dif=d3y
11    elif order==4: dif=d4y
12    else: print('Chỉ xét đạo hàm từ cấp 1 đến cấp 4'); exit
13    return dif
14 x=2; d1=8*x**3-15*x**4+6*x**5; d2=24*x**2-60*x**3+30*x**4
15 d3=48*x-180*x**2+120*x**3; d4=48-360*x+360*x**2
16 print('d1=%5.2f\nd2=%5.2f\nd3=%5.2f\nd4=%5.2f\n'% (d1,d2,d3,d4))
17 print('df/dx =%8.2f; err=%9.8f o/o'%(diff(x,1),abs((d1-diff(x,1))/d1*100)))
18 print('d2f/dx2 =%8.2f; err=%9.8f o/o'%(diff(x,2),abs((d2-diff(x,2))/d2*100)))
19 print('d3f/dx3 =%8.2f; err=%9.8f o/o'%(diff(x,3),abs((d3-diff(x,3))/d3*100)))
20 print('d4f/dx4 =%8.2f; err=%9.8f o/o'%(diff(x,4),abs((d4-diff(x,4))/d4*100)))
```

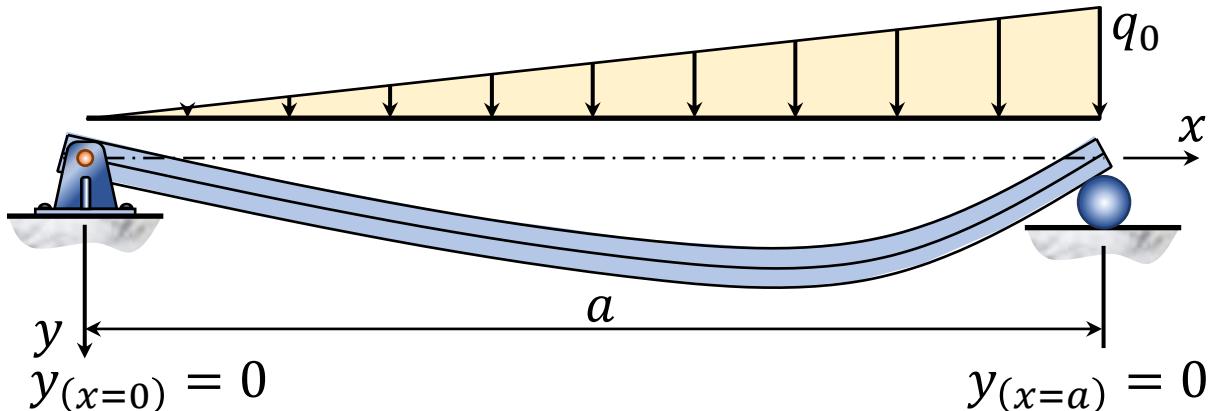
2.2.2. Xấp xỉ của đạo hàm cấp cao.

Ví dụ 2.2. Độ võng của dầm có phương trình: $y = \frac{q_0}{360E.I.a} (-3x^5 + 10a^2x^3 - 7a^4x)$.

Biết: $a = 5m; E = 200GPa; I = 3 \times 10^{-7}m^4; q_0 = 1 \frac{kN}{m}$. Sự liên hệ giữa các đại lượng

trên dầm: $\varphi = y^I$: góc xoay; $M = EI \times y^{II}$: moment uốn; $Q = EI \times y^{III}$: lực cắt.

Xác định φ, M, Q tại vị trí $x = 4m$ bằng xấp xỉ của đạo hàm với bước $h = 0.1$ và đánh giá sai số tương đối so với kết quả chính xác.



Giải.

$$\text{Đặt } y_0 = \frac{q_0}{360E.I.a} = 9.26 \times 10^{-6}$$

$$f = -3x^5 + 10a^2x^3 - 7a^4x = -3x^5 + 10 \times 5^2x^3 - 7 \times 5^4x$$

$$y = y_0 \times f$$

Ví dụ 2.2.

$$y_0 = 9.25 \times 10^{-6}; f = -3x^5 + 10 \times 5^2 x^3 - 7 \times 5^4 x; y = y_0 \times f; x_i = 4; h = 0.1$$

Đạo hàm chính xác: $\varphi_{cx} = y_0 f^I = y_0 (-15x^4 + 30 \times 5^2 x^2 - 7 \times 5^4) = 0.03505$

$$M = EIy_0 f^{II} = EIy_0 (-60x^3 + 60 \times 5^2 x) = 1.2$$

$$Q = EIy_0 f^{III} = EIy_0 (-180x^2 + 60 \times 5^2) = -0.76667$$

$$\varphi \Big|_{x_i} = y_0 \frac{f(x_i + h) - f(x_i - h)}{2h} = 0.03502; \text{ Sai số: } \varepsilon_\varphi = \left| \frac{\varphi_{cx} - \varphi}{\varphi_{cx}} \right| \times 100 = 0.06\%$$

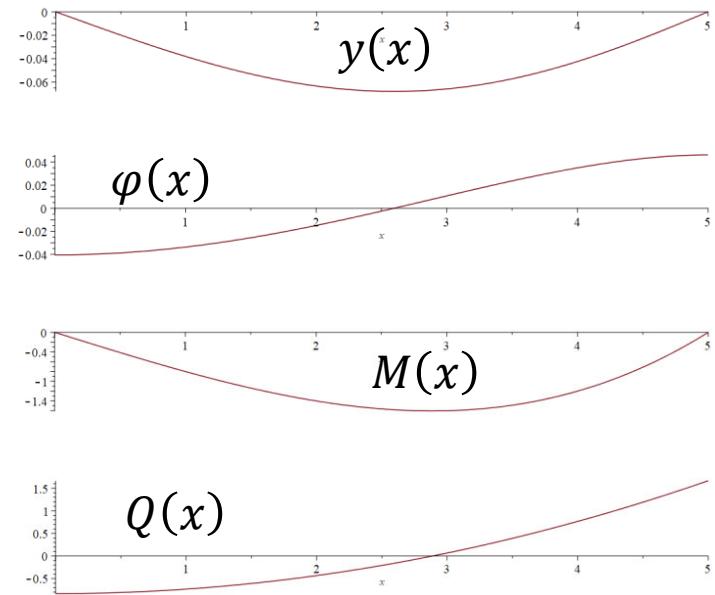
$$M \Big|_{x_i} = EIy_0 \frac{f(x_i + h) - 2f(x_i) + f(x_i - h)}{h^2} = 1.19933; \text{ Sai số: } \varepsilon_M = \left| \frac{M_{cx} - M}{M_{cx}} \right| \times 100 = 0.06\%$$

$$Q \Big|_{x_i} = EIy_0 \frac{f(x_i + 2h) - 2f(x_i + h) + 2f(x_i - h) - f(x_i - 2h)}{2h^3} = -0.76717;$$

$$\text{Sai số: } \varepsilon_Q = \left| \frac{Q_{cx} - Q}{Q_{cx}} \right| \times 100 = 0.07\%$$

Ví dụ 2.2.

```
1 import numpy as np
2 def f(x): a=5; y=-3*x**5+10*a**2*x**3-7*a**4*x; return y
3 def diff(x, h, order):
4     d1y=(f(x+h)-f(x-h))/2/h
5     d2y=(f(x+h)-2*f(x)+f(x-h))/h**2
6     d3y=(f(x+2*h)-2*f(x+h)+2*f(x-h)-f(x-2*h))/2/h**3
7     d4y=(f(x+2*h)-4*f(x+h)+6*f(x)-4*f(x-h)+f(x-2*h))/h**4
8     if order==1: dif=d1y
9     elif order==2: dif=d2y
10    elif order==3: dif=d3y
11    elif order==4: dif=d4y
12    else: print('Chỉ xét đạo hàm từ cấp 1 đến cấp 4'); exit
13    return dif
14 q0=1; E=2e8; I=3e-7; a=5; y0=q0/(360*E*I*a); xi=4; h=0.1
15 d1=y0*(-15*xi**4+30*a**2*xi**2-7*a**4);
16 d2=E*I*y0*(-60*xi**3+60*a**2*xi);
17 d3=E*I*y0*(-180*xi**2+60*a**2)
18 phi=y0*diff(xi, h, 1); M=y0*E*I*diff(xi,h,2)
19 Q=y0*E*I*diff(xi,h,3)
20 print('phi=%6.5f; sai số eph=%6.5f'%(phi,abs((d1-phi)/d1)*100))
21 print('M=%6.5f; sai số eM=%6.5f'%(M,abs((d2-M)/d2)*100))
22 print('Q=%6.5f; sai số eQ=%6.5f'%(Q,abs((d3-Q)/d3)*100))
```



2.2.2. Xấp xỉ của đạo hàm cấp cao.

Ví dụ 2.3. Cho hàm số: $f(x) = \frac{e^{-x^2}}{\sqrt{x}}$

Tính xấp xỉ của đạo hàm từ cấp 1 đến cấp 4 tại $x = 1$ với bước $h = 0.01$

Giải.

$$f^I \Big|_{x_i=1, h=0.01} = \frac{f(x_i + h) - f(x_i - h)}{2h} = -0.9197$$

$$f^{II} \Big|_{x_i=1, h=0.01} = \frac{f(x_i + h) - 2f(x_i) + f(x_i - h)}{h^2} = 1.7474$$

$$f^{III} \Big|_{x_i=1, h=0.01} = \frac{f(x_i + 2h) - 2f(x_i + h) + 2f(x_i - h) - f(x_i - 2h)}{2h^3} = -1.9774$$

$$f^{IV} \Big|_{x_i=1, h=0.01} = \frac{f(x_i + 2h) - 4f(x_i + h) + 6f(x_i) - 4f(x_i - h) + f(x_i - 2h)}{h^4} = 0.9425$$

Ví dụ 2.3.

$$f(x) = \frac{e^{-x^2}}{\sqrt{x}}; x = 1; h = 0.01; f^I = \frac{f(x_i + h) - f(x_i - h)}{2h}; f^{II} = \frac{f(x_i + h) - 2f(x_i) + f(x_i - h)}{h^2}$$

```

1 import numpy as np
2 def f(x):
3     y=np.exp(-x**2)/(x)**(1/2)
4     return y
5 def diff(x, order):
6     h=0.01
7     d1y=(f(x+h)-f(x-h))/2/h
8     d2y=(f(x+h)-2*f(x)+f(x-h))/h**2
9     d3y=(f(x+2*h)-2*f(x+h)+2*f(x-h)-f(x-2*h))/2/h**3
10    d4y=(f(x+2*h)-4*f(x+h)+6*f(x)-4*f(x-h)+f(x-2*h))/h**4
11    if order==1: dif=d1y
12    elif order==2: dif=d2y
13    elif order==3: dif=d3y
14    elif order==4: dif=d4y
15    else: print('Chỉ xét đạo hàm từ cấp 1 đến cấp 4'); exit
16    return dif
17 x=1
18 print('\ndf =%8.4f'%diff(x,1))
19 print('d2f =%8.4f'%diff(x,2))
20 print('d3f =%8.4f'%diff(x,3))
21 print('d4f =%8.4f'%diff(x,4))

```

$$f^{III} = \frac{f(x_i + 2h) - 2f(x_i + h) + 2f(x_i - h) - f(x_i - 2h)}{2h^3}$$

$$f^{IV} = \frac{f(x_i + 2h) - 4f(x_i + h) + 6f(x_i) - 4f(x_i - h) + f(x_i - 2h)}{h^4}$$

df = -0.9197 d2f = 1.7474 d3f = -1.9778 d4f = 0.9457

2.2.2. Xấp xỉ của đạo hàm cấp cao.

Ví dụ 2.4. Bằng thực nghiệm đo được độ võng theo phương y dọc theo phương x của dầm như bảng dưới. Biết: $E = 200GPa$; $I = 3 \times 10^{-10}m^4$



x[m]	0	0.5	1	1.5	2	2.5	3	3.5	4
y[cm]	0	-1.01	-1.89	-2.51	-2.78	-2.63	-2.07	-1.14	0

Dùng xấp xỉ của đạo hàm để tìm các giá trị moment uốn và lực cắt dọc theo dầm.

Liên hệ giữa moment uốn, lực cắt với độ võng của dầm là: $M = EI \frac{d^2y}{dx^2}$; $Q = EI \frac{d^3y}{dx^3}$

Hàm độ võng chính xác của dầm dùng để kiểm chứng kết quả là:

$$y_{cx} = 10^{-5}(-3.47x^5 + 185.18x^3 - 2074.07x)$$

Giải.

Ví dụ 2.4. $E = 200GPa; I = 3 \times 10^{-10}m^4; M = EIy^{II}; Q = EIy^{III}$

x[m]	0	0.5	1	1.5	2	2.5	3	3.5	4
y[cm]	0	-1.01	-1.89	-2.51	-2.78	-2.63	-2.07	-1.14	0

$$M(x_i) = EI \frac{y_{i+1} - 2y_i + y_{i-1}}{(x_{i+1} - x_i)^2}$$

$$M(x_i = 0.5) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-1.89) - 2(-1.01) + (0)}{0.5^2} = 0.312$$

$$M(x_i = 1) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-2.51) - 2(-1.89) + (-1.01)}{0.5^2} = 0.624$$

$$M(x_i = 1.5) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-2.78) - 2(-2.51) + (-1.89)}{0.5^2} = 0.84$$

$$M(x_i = 2) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-2.63) - 2(-2.78) + (-2.51)}{0.5^2} = 1.008$$

$$M(x_i = 2.5) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-2.07) - 2(-2.63) + (-2.78)}{0.5^2} = 0.984$$

$$M(x_i = 3) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-1.14) - 2(-2.07) + (-2.63)}{0.5^2} = 0.888$$

$$M(x_i = 3.5) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(0) - 2(-1.14) + (-2.07)}{0.5^2} = 0.504$$

Ví dụ 2.4. $E = 200GPa$; $I = 3 \times 10^{-10}m^4$; $M = EIy^{II}$; $Q = EIy^{III}$

x[m]	0	0.5	1	1.5	2	2.5	3	3.5	4
y[cm]	0	-1.01	-1.89	-2.51	-2.78	-2.63	-2.07	-1.14	0

$$Q(x_i) = \frac{y_{i+2} - 2y_{i+1} + 2y_{i-1} - y_{i-2}}{2(x_{i+1} - x_i)^3}$$

$$Q(x_i = 1) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-2.78) - 2(-2.51) + 2(-1.01) - (0)}{2 \times 0.5^3} = 0.528$$

$$Q(x_i = 1.5) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-2.63) - 2(-2.78) + 2(-1.89) - (-1.01)}{2 \times 0.5^3} = 0.384$$

$$Q(x_i = 2) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-2.07) - 2(-2.63) + 2(-2.51) - (-1.89)}{2 \times 0.5^3} = 0.144$$

$$Q(x_i = 2.5) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(-1.14) - 2(-2.07) + 2(-2.78) - (-2.51)}{0.5^2} = -0.12$$

$$Q(x_i = 3) = (2 \times 10^{11})(3 \times 10^{-10})(0.01) \frac{(0) - 2(-1.14) + 2(-2.63) - (-2.78)}{2 \times 0.5^3} = -0.48$$

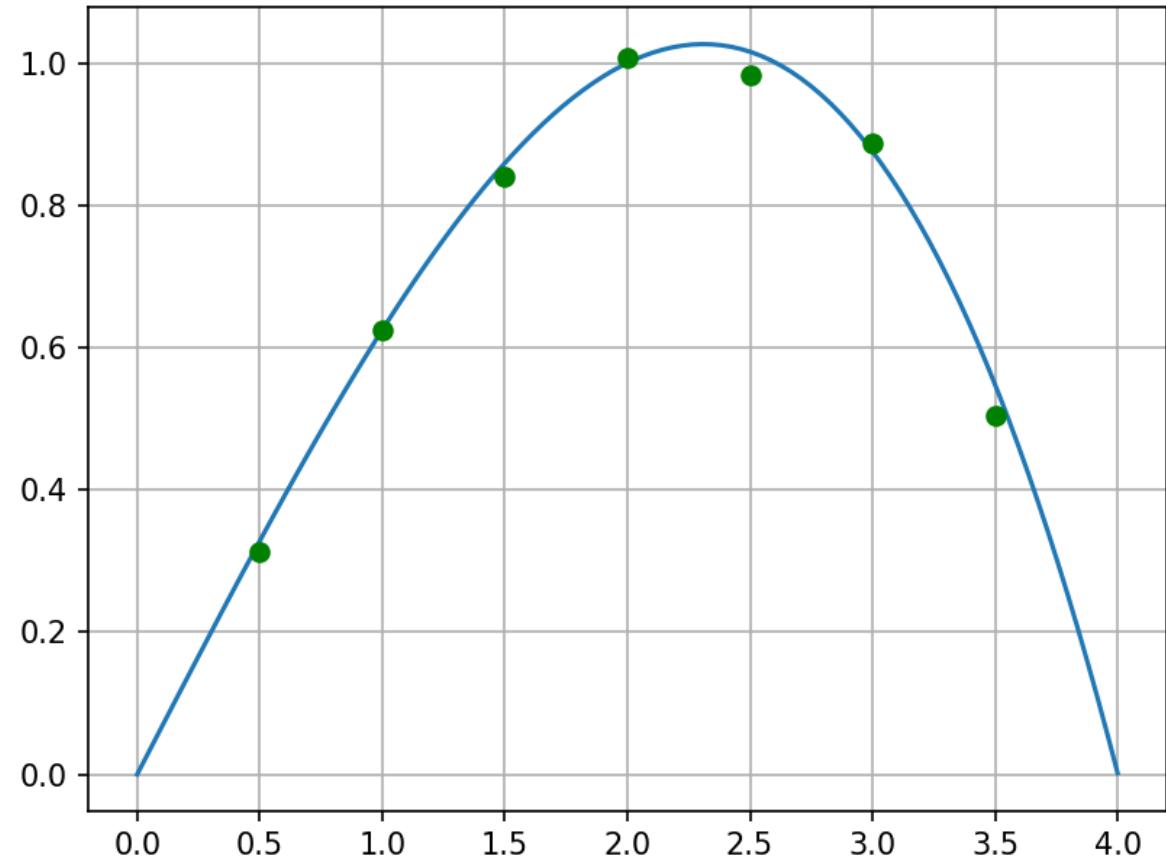
Ví dụ 2.4. $E = 200GPa$; $I = 3 \times 10^{-10}m^4$; $M = EIy^{II}$; $Q = EIy^{III}$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 x=np.array([0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4])
4 y=np.array([0, -1.01, -1.89, -2.51, -2.78, -2.63, -2.07, -1.14, 0])
5 y=y*0.01; E=2e11; I=3e-10; M=[]; Q=[];
6 for i in range(len(x)):
7     if i>0 and i<len(x)-1:
8         Mi=E*I*(y[i+1]-2*y[i]+y[i-1])/(x[i+1]-x[i])**2
9         M.append(Mi)
10    if i>1 and i<len(x)-2:
11        Qi=E*I*(y[i+2]-2*y[i+1]+2*y[i-1]-y[i-2])/2/(x[i+1]-x[i])**3
12        Q.append(Qi)
13 np.set_printoptions(precision=3)
14 print('M=\n',np.c_[x[1:-1],M].T); print('Q=\n',np.c_[x[2:-2],Q].T)
15 def f(x): y=-3.47e-5*x**5+185.19e-5*x**3-2074.07e-5*x; return y
16 xi=np.linspace(x[0],x[-1],100)
17 d1y=-5*3.47e-5*xi**4+3*185.19e-5*xi**2-2074.07e-5
18 d2y=-4*5*3.47e-5*xi**3+2*3*185.19e-5*xi; d2y=E*I*d2y
19 d3y=-3*4*5*3.47e-5*xi**2+2*3*185.19e-5; d3y=E*I*d3y
20 plt.figure(1); plt.grid(); plt.plot(xi,d2y); plt.plot(x[1:-1],M,'go')
21 plt.figure(2); plt.grid(); plt.plot(xi,d3y); plt.plot(x[2:-2],Q,'go')
22 plt.show()
```

Ví dụ 2.4. $E = 200GPa$; $I = 3 \times 10^{-10}m^4$; $M = EIy^{II}$; $Q = EIy^{III}$

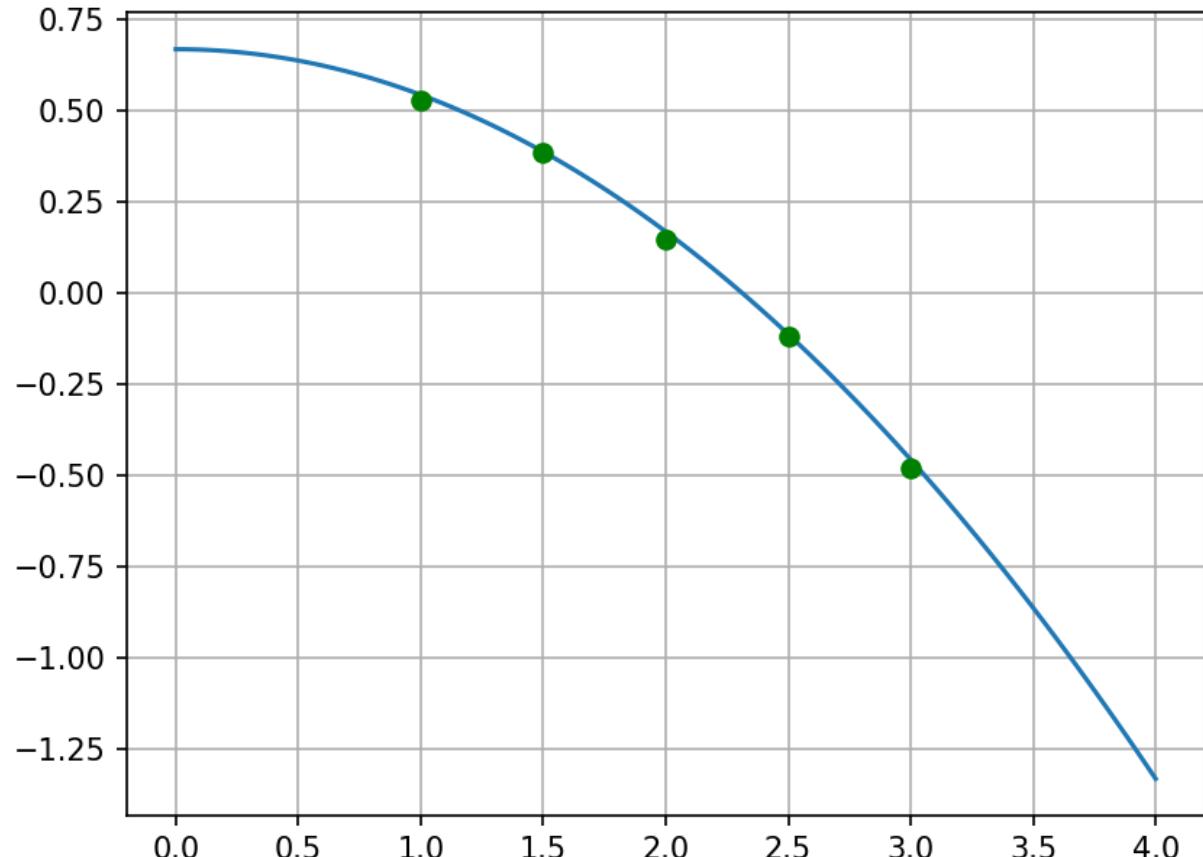
$M =$

```
[[ 0.5   1.    1.5   2.    2.5   3.    3.5 ]
 [ 0.312  0.624  0.84   1.008  0.984  0.888  0.504 ]]
```



$Q =$

```
[[ 1.    1.5   2.    2.5   3.
 [ 0.528  0.384  0.144 -0.12  -0.48 ]]
```



2.3. Xấp xỉ của tích phân xác định.

2.3.1. Công thức tích phân Newton - Leibnitz.

$$I = \int_a^b f(x)dx = F(x) \Big|_a^b = F(b) - F(a)$$

Trong đó: $F'(x) = f(x)$. $F(x)$ là nguyên hàm của $f(x)$.

Trường hợp không tìm được nguyên hàm do hàm $f(x)$ được cho bằng bảng số → Sử dụng tích phân số (gần đúng).

2.3.2. Công thức tích phân Newton - Cotes.

Thay hàm cần tính tích phân $f(x)$ bằng đa thức nội suy $f_n(x)$:

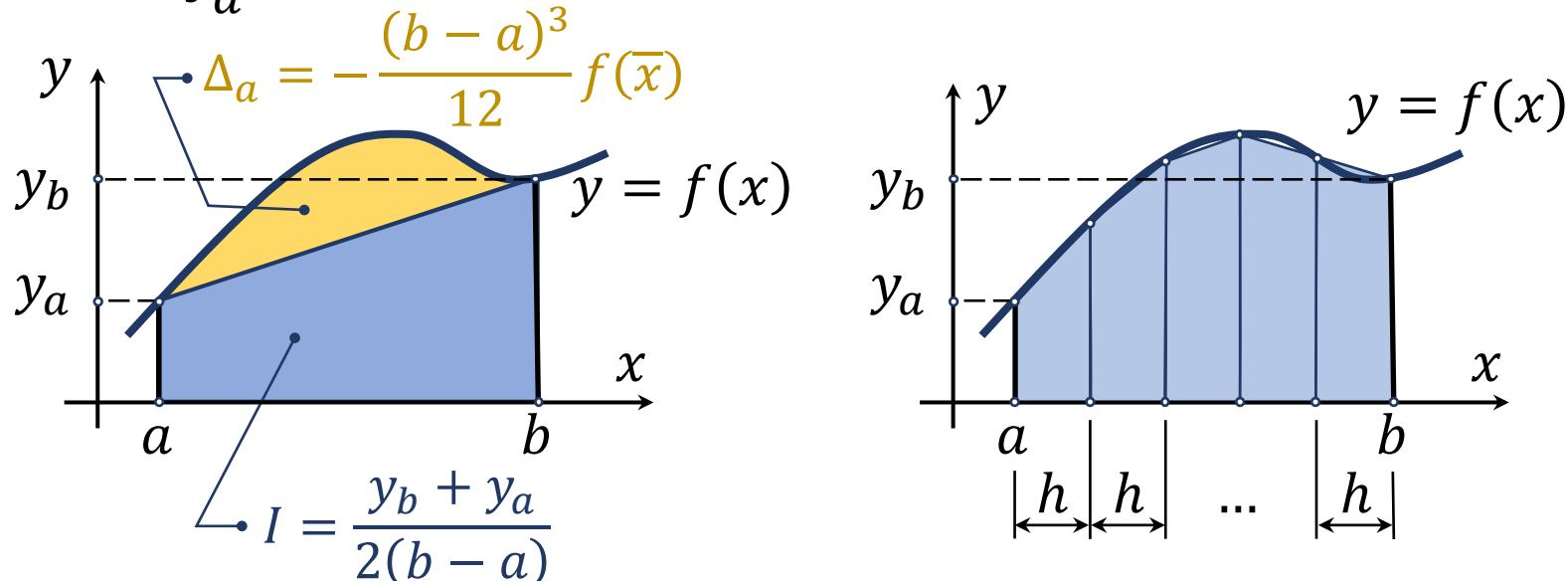
$$f(x) \sim f_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \quad \Rightarrow I = \int_a^b f(x)dx \approx \int_a^b f_n(x)dx$$

2.3. Xấp xỉ của tích phân xác định.

2.3.3. Công thức tích phân hình thang.

Hàm $f(x)$ xấp xỉ tuyến tính qua 2 điểm a, b : $y = f(x) \approx y_a + \frac{y_b - y_a}{b - a}(x - a)$

$$\Rightarrow I = \int_a^b f(x)dx \approx \int_a^b \left[y_a + \frac{y_b - y_a}{b - a}(\underline{x} - a) \right] dx = (b - a) \frac{y_a + y_b}{2}$$



Để giảm sai số \rightarrow Chia nhỏ đoạn a, b thành n đoạn có chiều dài h .

$$\Rightarrow I = \frac{(y_a + y_h)h}{2} + \frac{(y_h + y_{2h})h}{2} + \dots + \frac{(y_{(n-1)h} + y_b)h}{2}$$

$$= \frac{(y_a + y_b)h}{2} + \sum_{i=1}^{n-1} y_{(a+ih)} \times h$$

2.3.3. Công thức tích phân hình thang.

Đánh giá sai số. Theo khai triển Taylor:

$$\Delta_a = -\frac{(b-a)^3}{12 \times n^3} \times \sum_{i=1}^n y''_{\bar{x}_i}, \text{ với } \bar{x}_i = \frac{x_{i-1} + x_{i+1}}{2}, n \text{ là số đoạn chia.}$$

Ví dụ 2.5. Sử dụng công thức hình thang tính tích phân: $I = \int_1^4 (2 + 3x)dx$

Giải.

Lời giải chính xác: $I = \int_1^4 (2 + 3x)dx = \left(2x + \frac{3x^2}{2} \right) \Big|_1^4 = 28.5$

$$I = (b-a) \frac{y_a + y_b}{2} = (4-1) \frac{(2 + 3 \times 1) + (2 + 3 \times 4)}{2} = 28.5$$

→ Cho kết quả chính xác. Do hàm cần tính tích phân là tuyến tính.

2.3.3. Công thức tích phân hình thang.

Ví dụ 2.6. Sử dụng công thức hình thang $n = 3$ đoạn và $n = 6$ đoạn, tính tích phân:

$$I = \int_{-2}^1 (1 - 3x + 4x^2) dx$$

$$I = \frac{(y_a + y_b)h}{2} + \sum_{i=1}^{n-1} y_{(a+ih)} \times h$$

$$\Delta_a = -\frac{(b-a)^3}{12 \times n^3} \times \sum_{i=1}^n y''_{x_i}$$

Giải.

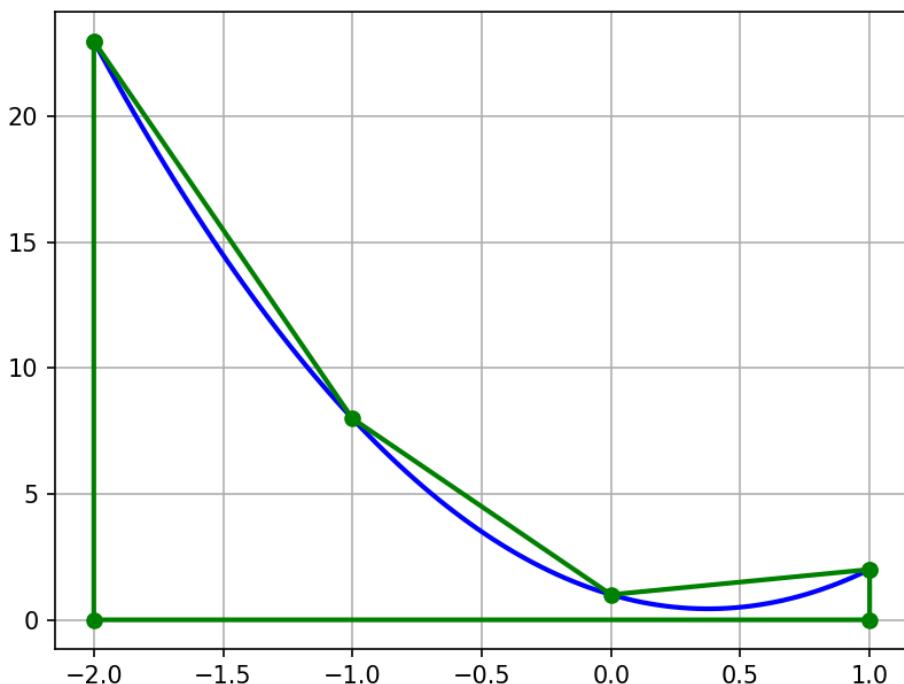
Lời giải chính xác: $I = \left(x - \frac{3x^2}{2} + \frac{4x^3}{3} \right) \Big|_{-2}^1 = 19.5$

$$n = 3: \Rightarrow h = \frac{1 + 2}{3} = 1$$

$$\Rightarrow I_3 = \frac{f(-2) + f(1)}{2} \times 1 + f(-1) \times 1 + f(0) \times 1 = 21.5$$

Sai số tuyệt đối tính theo kết quả chính xác:

$$\Delta_3 = 19.5 - 21.5 = -2$$



Ví dụ 2.6.

$$I = \int_{-2}^1 (1 - 3x + 4x^2) dx$$

$$I = \frac{(y_a + y_b)h}{2} + \sum_{i=1}^{n-1} y_{(a+ih)} \times h$$

$$\Delta_a = -\frac{(b-a)^3}{12 \times n^3} \times \sum_{i=1}^n y''_{\bar{x}_i}$$

Sai số tuyệt đối tính theo công thức:

$$y'' = 8 \Rightarrow \sum_{i=1}^3 y''_{\bar{x}_i} = \sum_{i=1}^3 8 = 24 \Rightarrow \Delta_a = -\frac{(2+1)^3}{12 \times 3^3} \times 24 = -2$$

Sai số tương đối: $\varepsilon_3 = \frac{|-2|}{19.5} \times 100\% = 10.26\%$

Ví dụ 2.6.

$$I = \int_{-2}^1 (1 - 3x + 4x^2) dx$$

$$I = \frac{(y_a + y_b)h}{2} + \sum_{i=1}^{n-1} y_{(a+ih)} \times h$$

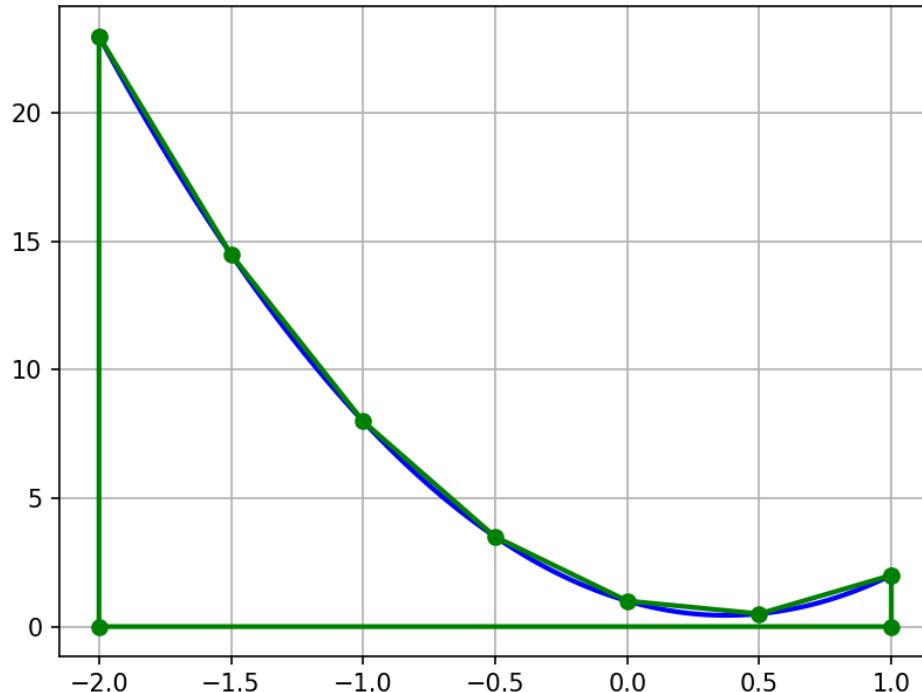
$$\Delta_a = -\frac{(b-a)^3}{12 \times n^3} \times \sum_{i=1}^n y''_{\bar{x}_i}$$

$n = 6$:

$$\Rightarrow h = \frac{b-a}{n} = \frac{1+2}{6} = 0.5$$

$$\begin{aligned} \Rightarrow I_6 &= \frac{f(-2) + f(1)}{2}(0.5) + \\ &+[f(-1.5) + f(-1) + f(-0.5) + f(0) + f(0.5)](0.5) = 20.0 \end{aligned}$$

Sai số tương đối: $\varepsilon_6 = \frac{|19.5 - 20.0|}{19.5} \times 100\% = 2.56\%$



Ví dụ 2.6.

$$I = \int_{-2}^1 (1 - 3x + 4x^2) dx$$

$$I = \frac{(y_a + y_b)h}{2} + \sum_{i=1}^{n-1} y_{(a+ih)} \times h$$

$$\Delta_a = -\frac{(b-a)^3}{12 \times n^3} \times \sum_{i=1}^n y''_{x_i}$$

```
1| import numpy as np
2| a=1; b=9; n=40
3| def f(x): y=np.pi*(2+np.sin(x)*x** (2/3))**2; return y
4| def d2f(x): h=1e-3; d2f=(f(x+h)-2*f(x)+f(x-h))/h**2; return d2f
5| h=(b-a)/n; x=np.linspace(a,b,n+1)
6| V=0; D=0
7| for i in range(n+1):
8|     if i==0 or i==n: V += f(x[i])*h/2
9|     else: V += f(x[i])*h
10|    if i < n: D += -(b-a)**3/12/n**3*d2f((x[i]+x[i+1])/2)
11| print('Tích phân hình thang n=%d đoạn:'%n)
12| print('V(%d)=%6.4f'%(n, V))
13| print('Sai số tuyệt đối: D_a(%d)=%6.4f'%(n, D))
14| e = abs(D)/(V+D)*100
15| print('Sai số tương đối: e_a(%d)=%4.2f'%(n, e))
```

Ví dụ 2.6.

$$I = \int_{-2}^1 (1 - 3x + 4x^2) dx$$

$$I = \frac{(y_a + y_b)h}{2} + \sum_{i=1}^{n-1} y_{(a+ih)} \times h$$

$$\Delta_a = -\frac{(b-a)^3}{12 \times n^3} \times \sum_{i=1}^n y''_{\bar{x}_i}$$

```

18 a = -2; b = 1; n3 = 3; n6=6; n100=100
19 trap(a,b,n3); trap(a,b,n6); trap(a,b,n100)
20 x=np.linspace(a,b,100); y=f(x);
21 x3=np.linspace(a,b,n3+1); xx=np.array([b,b,a,a])
22 x3a=np.r_[x3,xx]
23 y3=f(x3); yy=np.array([f(b),0,0,f(a)])
24 y3a=np.r_[y3,yy]
25 plt.figure(1); plt.grid();
26 plt.plot(x,y,'b',lw=2)
27 plt.plot(x3a,y3a,'-go',lw=2)
28
29 x6=np.linspace(a,b,n6+1); x6a=np.r_[x6,xx]
30 y6=f(x6); y6a=np.r_[y6,yy]
31 plt.figure(2); plt.grid()
32 plt.plot(x,y,'b',lw=2)
33 plt.plot(x6a,y6a,'-go',lw=2)
34 plt.show()

```

Tích phân hình thang n=3 đoạn:

Itrap(3)=21.5000

Sai số tuyệt đối: Dtr(3)=-2.0000

Sai số tương đối: etr(3)=10.26

Tích phân hình thang n=6 đoạn:

Itrap(6)=20.0000

Sai số tuyệt đối: Dtr(6)=-0.5000

Sai số tương đối: etr(6)=2.56

Tích phân hình thang n=100 đoạn:

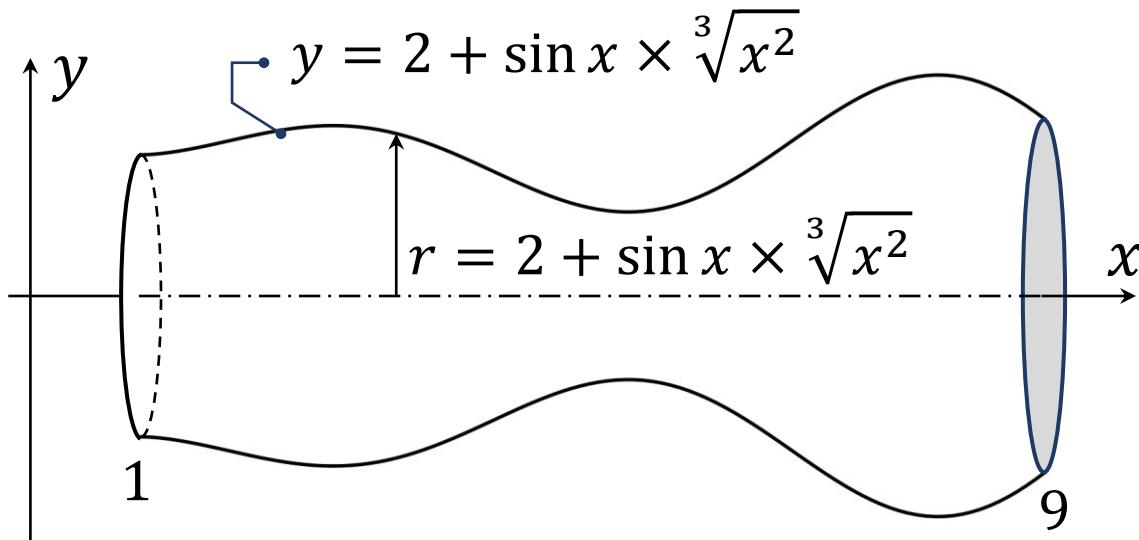
Itrap(100)=19.5018

Sai số tuyệt đối: Dtr(100)=-0.0018

Sai số tương đối: etr(100)=0.01

2.3.3. Công thức tích phân hình thang.

Ví dụ 2.7. Đường cong có phương trình $y = 2 + \sin x \times \sqrt[3]{x^2}$, ($1 \leq x \leq 9$) xoay quanh trục x tạo thành vật thể tròn xoay như hình vẽ. Tính thể tích của vật.



Giải.

Công thức tính thể tích của vật thể tròn xoay:

$$V = \int_1^9 \pi \times r^2 dx = \int_1^9 \pi \left(2 + \sin x \times \sqrt[3]{x^2}\right)^2 dx$$

Ví dụ 2.7.

$$V = \int_1^9 \pi \left(2 + \sin x \times \sqrt[3]{x^2} \right)^2 dx$$

$$I = \frac{(y_a + y_b)h}{2} + \sum_{i=1}^{n-1} y_{(a+ih)} \times h$$

$$\Delta_a = -\frac{(b-a)^3}{12 \times n^3} \times \sum_{i=1}^n y''_{\bar{x}_i}$$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 def f(x):
4     y=np.pi*(2+np.sin(x)*x** (2/3))**2
5     return y
6 def df1(x):
7     h=1e-3; d1f=(f(x+h)-f(x-h))/2/h
8     return d1f
9 def df2(x):
10    h=1e-3; d2f=(df1(x+h)-df1(x-h))/2/h
11    return d2f
12 a = 1; b = 9; n = 40
13 h=(b-a)/n; x=np.linspace(a,b,n+1)
14 V=0; D=0
15 for i in range(n+1):
16     if i==0 or i==n: V += f(x[i])*h/2
17     else: V += f(x[i])*h
18     if i < n: D += -(b-a)**3/12/n**3*df2((x[i]+x[i+1])/2)
19 print('Tích phân hình thang n=%d đoạn:' %n)
20 print('V(%d)=%6.4f' %(n, V))
21 print('Sai số tuyệt đối: D_a(%d)=%6.4f' %(n, D))
22 e = abs(D)/(V+D)*100
23 print('Sai số tương đối: e_a(%d)=%4.2f' %(n, e))

```

Tích phân hình thang n=40 đoạn:

$$V(40)=276.3565$$

Sai số tuyệt đối: $D_a(40)=0.3691$

Sai số tương đối: $e_a(40)=0.13$

2.3. Xấp xỉ của tích phân xác định.

2.3.4. Công thức Simpson.

Công thức Simpson 1/3 theo luật nội suy đa thức bậc 2 nhằm làm tăng tốc độ hội tụ kết quả. (Giảm số khoảng chia)

Với số khoảng chia n (n phải **chẵn** và ít nhất 4 khoảng). Chiều dài của mỗi khoảng:

$$h = \frac{b - a}{n}. \text{ Luật Simpson như sau:}$$

$$I = \int_a^b f(x)dx = \frac{h}{3} (y_{x_0} + 4y_{x_1} + 2y_{x_2} + 4y_{x_3} + \dots + 4y_{x_{n-1}} + y_{x_n}) + \Delta_a$$

Đặt $m = n/2 \Rightarrow I = \frac{h}{3} \left(y_a + 4 \sum_{i=1}^m y_{a+(2i-1)h} + 2 \sum_{i=1}^{m-1} y_{a+2ih} + y_b \right) + \Delta_a$

Sai số: $\Delta_a = -\frac{(b-a)^5}{180n^4} \times \sum_{i=1}^n y_{\bar{x}_i}^{(IV)}$, với $\bar{x}_i = \frac{x_{i-1} + x_{i+1}}{2}$

2.3.4. Công thức Simpson.

$$I \approx \frac{h}{3} \left(y_a + 4 \sum_{i=1}^m y_{a+(2i-1)h} + 2 \sum_{i=1}^{m-1} y_{a+2ih} + y_b \right)$$

$$\Delta_a = -\frac{(b-a)^5}{180n^4} \times \sum_{i=1}^n y_{\bar{x}_i}^{(IV)}$$

Ví dụ 2.8. Sử dụng công thức Simpson với $n = 4$ đoạn, tính tích phân:

$$I = \int_{-2}^1 (1 - 3x + 4x^2) dx$$

Giải.

$$h = \frac{1 + 2}{4} = 0.75$$

$$\Rightarrow I_4 = \frac{0.75}{3} [f(-2) + 4 \times f(-1.25) + 2 \times f(-0.5) + 4 \times f(0.25) + f(1)] = 19.5$$

Vì công thức Simpson nội suy bậc 2 nên khi tính tích phân hàm bậc 2 cho kết quả chính xác.

2.3.4. Công thức Simpson.

$$I \approx \frac{h}{3} \left(y_a + 4 \sum_{i=1}^m y_{a+(2i-1)h} + 2 \sum_{i=1}^{m-1} y_{a+2ih} + y_b \right)$$

$$\Delta_a = -\frac{(b-a)^5}{180n^4} \times \sum_{i=1}^n y_{\bar{x}_i}^{(IV)}$$

Ví dụ 2.9. Sử dụng công thức Simpson với $n = 10$ đoạn, tính tích phân: $I = \int_0^1 \frac{dx}{x+1}$

Giải.

$$h = \frac{1-0}{10} = 0.1$$

x_i	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
y_i	1	$\frac{10}{11}$	$\frac{5}{6}$	$\frac{10}{13}$	$\frac{5}{7}$	$\frac{2}{3}$	$\frac{5}{8}$	$\frac{10}{17}$	$\frac{5}{9}$	$\frac{10}{19}$	$\frac{1}{2}$

$$y_a + y_b = y_0 + y_{10} = 1 + 0.5 = 1.5$$

$$4(y_1 + y_3 + y_5 + y_7 + y_9) = 13.838$$

$$2(y_2 + y_4 + y_6 + y_8) = 5.4564$$

$$\Rightarrow I = \frac{0.1}{3} \times (1.5 + 13.838 + 5.4564) = 0.69315$$

Ví dụ 2.9.

$$I = \int_0^1 \frac{dx}{x+1}$$

$$I \approx \frac{h}{3} \left(y_a + 4 \sum_{i=1}^m y_{a+(2i-1)h} + 2 \sum_{i=1}^{m-1} y_{a+2ih} + y_b \right)$$

$$\Delta_a = -\frac{(b-a)^5}{180n^4} \times \sum_{i=1}^n y_{\bar{x}_i}^{(IV)}$$

Đánh giá sai số:

$$y^{(4)} = \frac{24}{(x+1)^5} \Rightarrow \sum_{i=1}^{10} y_{\bar{x}_i}^{(4)} = \sum_{i=1}^{10} \frac{24}{(\bar{x}_i + 1)^5} = 55.7638$$

i	1	2	3	4	5	6	7	8	9	10
$x_0 + x_1$	$x_1 + x_2$	$x_2 + x_3$	$x_3 + x_4$	$x_4 + x_5$	$x_5 + x_6$	$x_6 + x_7$	$x_7 + x_8$	$x_8 + x_9$	$x_9 + x_{10}$	
2	2	2	2	2	2	2	2	2	2	

Sai số tuyệt đối: $\Delta_{10} = -\frac{(1-0)^5}{180 \times 10^4} \times 55.7638 = -3 \times 10^{-5}$

Sai số tương đối: $\varepsilon_{10} = \frac{|-3 \times 10^{-5}|}{0.69315 - 3 \times 10^{-5}} \times 100 = 4.5 \times 10^{-3}$

2.3.4. Công thức Simpson.

$$I \approx \frac{h}{3} \left(y_a + 4 \sum_{i=1}^m y_{a+(2i-1)h} + 2 \sum_{i=1}^{m-1} y_{a+2ih} + y_b \right)$$

$$\Delta_a = -\frac{(b-a)^5}{180n^4} \times \sum_{i=1}^n y_{\bar{x}_i}^{(IV)}$$

Ví dụ 2.10. Sử dụng công thức Simpson với $n = 6$ đoạn, tính tích phân:

$$I = \int_1^3 \sin(\sqrt{x}) \times e^x \, dx$$

Giải.

- 1/ Tạo hàm $f(x)$ để sử dụng nhiều lần
- 2/ Tạo hàm $d4f(x)$ tính sai phân cấp 4
- 3/ Khai báo: a, b, n
- 4/ Gán $h = (b - a)/n$
- 5/ Tạo mảng $x = (a, a + h, \dots, a + nh)$
- 6/ Khởi tạo $I = 0, \Delta = 0$ ban đầu

7/ Cho vòng lặp $i = 0 \dots n$

7.1/ Nếu $i = 0$ hoặc $i = n$:

$$I = I + f(x_i)$$

7.2/ Nếu ngược với 7.1/:

7.2.1/ Nếu i lẻ (chia 2 dư 1):

$$I = I + 4 \times f(x_i)$$

7.2.1/ Nếu i chẵn (chia 2 dư 0)

$$I = I + 2 \times f(x_i)$$

7.3/ Nếu $i < n$:

$$\Delta = \Delta + df4[(x_i + x_{i+1})/2]$$

Ví dụ 2.10.

$$I \approx \frac{h}{3} \left(y_a + 4 \sum_{i=1}^m y_{a+(2i-1)h} + 2 \sum_{i=1}^{m-1} y_{a+2ih} + y_b \right)$$

$$\Delta_a = -\frac{(b-a)^5}{180n^4} \times \sum_{i=1}^n y_{\bar{x}_i}^{(IV)}$$

```

1 import numpy as np
2 a=1; b=3; n=6
3 def f(x): y=np.sin(x** (1/2))*np.exp(x); return y
4 def d4f(x):
5     h=1e-3
6     dy4=(f(x+2*h)-4*f(x+h)+6*f(x)-4*f(x-h)+f(x-2*h))/h**4
7     return dy4
8 h=(b-a)/n; x=np.linspace(a,b,n+1)
9 Int=0; D=0
10 for i in range(n+1):
11     if i==0 or i==n: Int += f(x[i])*h/3
12     else:
13         if i%2==1: Int += 4*f(x[i])*h/3
14         else: Int += 2*f(x[i])*h/3
15     if i<n: D += (b-a)**5/180/n**4*d4f((x[i]+x[i+1])/2)
16 e=abs(D)/(Int+D)*100
17 print('Tích phân bằng phương pháp Simpson %d khoảng.'%n)
18 print('Kết quả: I(%d)=%6.4f'%(n,Int))
19 print('Sai số tuyệt đối: D(%d)=%8.7f'%(n,D))
20 print('Sai số tương đối: e(%d)=%3.2f'%(n,e))

```

Tích phân bằng phương pháp Simpson 6 khoảng.
 Kết quả: $I(6)=17.0473$
 Sai số tuyệt đối: $D(6)=0.0043038$
 Sai số tương đối: $e(6)=0.03$

2.3. Xấp xỉ của tích phân xác định.

2.3.5. Công thức Gauss cầu phương.

Cần tính $I = \int_a^b f(x)dx$. Đổi qua biến t bằng cách đặt: $x = \frac{b+a}{2} + \frac{b-a}{2} \times t$

$$dx = \frac{b-a}{2} \times dt \Rightarrow \int_a^b f(x)dx = \int_{-1}^1 f\left(\frac{b+a}{2} + \frac{b-a}{2} \times t\right) \times \frac{b-a}{2} \times dt$$

Thế nên, ta chỉ đề cập đến tích phân dạng: $I = \int_{-1}^1 f(x)dx$

Luật Gauss là thay thế tích phân của hàm $f(x)$ bằng xấp xỉ của hàm nội suy:

$$I = \int_{-1}^1 f(x)dx \approx w_1 \times f(x_1) + w_2 \times f(x_2) + \cdots + w_n \times f(x_n) = \sum_{i=1}^n w_i \times f(x_i)$$

Với x_1, x_2, \dots, x_n là tọa độ Gauss, $(-1 < x_i < 1)$

Và w_1, w_2, \dots, w_n là các trọng số tương ứng.

2.3.5. Công thức Gauss cầu phương.

Các tọa độ và trọng số được tìm thông qua ràng buộc:

Nếu số điểm Gauss là n , thì cần phải tìm $2n$ ẩn số (x_i, w_i) .

Xét đa thức **đủ** bậc p có các hệ số bằng 1: $g(x) = \mathbf{1} + \mathbf{x} + \mathbf{x}^2 + \mathbf{x}^3 + \cdots + \mathbf{x}^p$

$2n$ ẩn số được xác định thông qua phép tính tích phân lần lượt trong khoảng $(-1, 1)$ của $2n$ số hạng đầu tiên của đa thức $g(x)$, bậc của số hạng phải tương ứng với bậc của tọa độ cầu phương:

$n = 1$ điểm Gauss $\rightarrow 2$ ẩn \rightarrow thiết lập 2 phương trình:

$$\begin{cases} \int_{-1}^1 \mathbf{1} dx = w_1 \\ \int_{-1}^1 \mathbf{x} dx = w_1 x_1 \end{cases} \Rightarrow \begin{cases} 2 = w_1 \\ 0 = w_1 x_1 \end{cases} \Rightarrow \begin{cases} w_1 = 2 \\ x_1 = 0 \end{cases}$$

2.3.5. Công thức Gauss cầu phương.

$$g(x) = \mathbf{1} + \mathbf{x} + \mathbf{x^2} + \mathbf{x^3} + \cdots + \mathbf{x^p}$$

$n = 2$ điểm Gauss \rightarrow 4 ẩn \rightarrow thiết lập 4 phương trình:

$$\begin{cases} \int_{-1}^1 \mathbf{1} dx = w_1 + w_2 \\ \int_{-1}^1 \mathbf{x} dx = w_1 x_1 + w_2 x_2 \\ \int_{-1}^1 \mathbf{x^2} dx = w_1 x_1^2 + w_2 x_2^2 \\ \int_{-1}^1 \mathbf{x^3} dx = w_1 x_1^3 + w_2 x_2^3 \end{cases} \Rightarrow \begin{cases} 2 = w_1 + w_2 \\ 0 = w_1 x_1 + w_2 x_2 \\ \frac{2}{3} = w_1 x_1^2 + w_2 x_2^2 \\ 0 = w_1 x_1^3 + w_2 x_2^3 \end{cases} \Rightarrow \begin{cases} w_1 = 1 \\ x_1 = -\frac{1}{\sqrt{3}} \\ w_2 = 1 \\ x_2 = \frac{1}{\sqrt{3}} \end{cases}$$

2.3.5. Công thức Gauss cầu phương.

$$g(x) = \mathbf{1} + \mathbf{x} + \mathbf{x^2} + \mathbf{x^3} + \cdots + \mathbf{x^p}$$

$n = 3$ điểm Gauss $\rightarrow 6$ ẩn \rightarrow thiết lập 6 phương trình:

$$\begin{cases} \int_{-1}^1 \mathbf{1} dx = w_1 + w_2 + w_3 \\ \int_{-1}^1 \mathbf{x} dx = w_1 x_1 + w_2 x_2 + w_3 x_3 \\ \int_{-1}^1 \mathbf{x^2} dx = w_1 x_1^2 + w_2 x_2^2 + w_3 x_3^2 \\ \int_{-1}^1 \mathbf{x^3} dx = w_1 x_1^3 + w_2 x_2^3 + w_3 x_3^3 \\ \int_{-1}^1 \mathbf{x^4} dx = w_1 x_1^4 + w_2 x_2^4 + w_3 x_3^4 \\ \int_{-1}^1 \mathbf{x^5} dx = w_1 x_1^5 + w_2 x_2^5 + w_3 x_3^5 \end{cases}$$

$$\Rightarrow \begin{cases} 2 = w_1 + w_2 + w_3 \\ 0 = w_1 x_1 + w_2 x_2 + w_3 x_3 \\ \frac{2}{3} = w_1 x_1^2 + w_2 x_2^2 + w_3 x_3^2 \\ 0 = w_1 x_1^3 + w_2 x_2^3 + w_3 x_3^3 \\ \frac{2}{5} = w_1 x_1^4 + w_2 x_2^4 + w_3 x_3^4 \\ 0 = w_1 x_1^5 + w_2 x_2^5 + w_3 x_3^5 \end{cases}$$

$$\Rightarrow \begin{cases} w_1 = \frac{5}{9} \\ x_1 = -\sqrt{\frac{3}{5}} \\ w_2 = \frac{8}{9} \\ x_2 = 0 \\ w_3 = \frac{5}{9} \\ x_3 = \sqrt{\frac{3}{5}} \end{cases}$$

2.3.5. Công thức Gauss cầu phương.

Cách thực hiện tương tự, ta sẽ liệt kê tọa độ và trọng số từ 1 đến 6 điểm Gauss:

Số điểm	Tọa độ	Trọng số	Chính xác đến bậc
$n = 1$	0	2.000000000000000	$2n - 1 = 1$

Số điểm	Tọa độ	Trọng số	Chính xác đến bậc
$n = 2$	-0.577350269189626	1.000000000000000	$2n - 1 = 3$
	0.577350269189626	1.000000000000000	

Số điểm	Tọa độ	Trọng số	Chính xác đến bậc
$n = 3$	-0.774596669241483	0.5555555555555556	$2n - 1 = 5$
	0	0.8888888888888889	
	0.774596669241483	0.5555555555555556	

Số điểm	Tọa độ	Trọng số	Chính xác đến bậc
$n = 4$	-0.861136311594053	0.347854845137454	$2n - 1 = 7$
	-0.339981043584856	0.652145154862546	
	0.339981043584856	0.652145154862546	
	0.861136311594053	0.347854845137454	

2.3.5. Công thức Gauss cầu phương.

Số điểm	Tọa độ	Trọng số	Chính xác đến bậc
$n = 5$	-0.906179845938664	0.236926885056189	$2n - 1 = 9$
	-0.538469310105683	0.478628670499366	
	0	0.5688888888888889	
	0.538469310105683	0.478628670499366	
	0.906179845938664	0.236926885056189	

Số điểm	Tọa độ	Trọng số	Chính xác đến bậc
$n = 6$	-0.932469514203152	0.171324492379170	$2n - 1 = 11$
	-0.661209386466264	0.360761573048139	
	-0.238619186083197	0.467913934572691	
	0.238619186083197	0.467913934572691	
	0.661209386466264	0.360761573048139	
	0.932469514203152	0.171324492379170	

Lưu ý: Chọn số điểm n cần thiết để cho kết quả chính xác.

(Chọn lớn hơn số điểm cần thiết thì vẫn cho kết quả chính xác)

$n \geq \text{round}((\text{Bậc} + 1)/2, 0) \rightarrow n$ làm tròn lên để trở thành số nguyên.

2.3.5. Công thức Gauss cầu phương.

Ví dụ 2.11. Sử dụng công thức Gauss tính tích phân: $I = \int_{-2}^1 (1 - 3x + 4x^2) dx$

Giải.

$$x = \frac{b+a}{2} + \frac{b-a}{2}t = -\frac{1}{2} + \frac{3}{2}t; dx = \frac{b-a}{2}dt = \frac{3}{2}dt; x = a \Rightarrow t = -1; x = b \Rightarrow t = 1$$

$$\Rightarrow I = \int_{-1}^1 \left[1 - 3\left(-\frac{1}{2} + \frac{3}{2}t\right) + 4\left(-\frac{1}{2} + \frac{3}{2}t\right)^2 \right] \left(\frac{3}{2}dt\right)$$

$$\text{Hàm cần tính tích phân là: } f(t) = \frac{3}{2} \left[1 - 3\left(-\frac{1}{2} + \frac{3}{2}t\right) + 4\left(-\frac{1}{2} + \frac{3}{2}t\right)^2 \right]$$

$$f(t) \text{ bậc 2 nên cần sử dụng tối thiểu số điểm Gauss: } n = round\left(\frac{2+1}{2}, 0\right) = 2$$

$$\begin{aligned} \Rightarrow I &= w_1 \times f(t_1) + w_2 \times f(t_2) = 1 \times f(-0.577350269189626) + \\ &\quad + 1 \times f(0.577350269189626) = 19.5000 \end{aligned}$$

Ví dụ 2.11.

```
1 import numpy as np
2 xw=np.array([[-0.577350269189626, 1],
3                 [0.577350269189626, 1]])
4 def f(t,a,b):
5     x=(b+a)/2+t*(b-a)/2;
6     dx=(b-a)/2
7     y=1-3*x+4*x**2; #Hàm cần tính tích phân
8     y=y*dx
9     return y
10 a,b,n=-2,1,2 #Cận dưới, cận trên và số điểm Gauss
11 Int=0;
12 for i in range(xw.shape[0]):
13     Int += f(xw[i,0],a,b)*xw[i,1]
14 print('I(%d)=%6.4f'%(n,Int))
| I(2)=19.5000
```

2.3.5. Công thức Gauss cầu phương.

Ví dụ 2.12. Sử dụng công thức Gauss tính tích phân: $I = \int_1^3 \sin(\sqrt{x}) \times e^x dx$

Giải.

```
1 import numpy as np
2 def gauss(n):
3     if n==1:
4         xw=np.array([[0,2]])
5     elif n==2:
6         xw=np.array([-0.577350269189626,1],
7                     [0.577350269189626,1])
8     elif n==3:
9         xw=np.array([-0.774596669241483,0.555555555555556],
10                    [0,0.888888888888889],
11                    [0.774596669241483,0.555555555555556])
12     elif n==4:
13         xw=np.array([-0.861136311594053,0.347854845137454],
14                     [-0.339981043584856,0.652145154862546],
15                     [0.339981043584856,0.652145154862546],
16                     [0.861136311594053,0.347854845137454])
17     elif n==5:
18         xw=np.array([-0.906179845938664,0.236926885056189],
19                     [-0.538469310105683,0.478628670499366],
20                     [0,0.568888888888889],
21                     [0.538469310105683,0.478628670499366],
22                     [0.906179845938664,0.236926885056189])
```

Ví dụ 2.12.

$$I = \int_1^3 \sin(\sqrt{x}) \times e^x dx$$

```
23 elif n==6:  
24     xw=np.array([[-0.932469514203152,0.171324492379170],  
25                 [-0.661209386466264,0.360761573048139],  
26                 [-0.238619186083197,0.467913934572691],  
27                 [0.238619186083197,0.467913934572691],  
28                 [0.661209386466264,0.360761573048139],  
29                 [0.932469514203152,0.171324492379170]])  
30 else: print('Chỉ xét n=1,2,3,4,5,6'); exit  
31 return xw  
32 def f(t,a,b):  
33     x=(b+a)/2+t*(b-a)/2 #Đổi biến x sang t  
34     fx=np.sin(x** (1/2))*np.exp(x) #Hàm cần tính tích phân  
35     y=fx*dx  
36     return y  
37 a,b = 1,3 #Cận dưới, cận trên  
38 for n in range(2,7): #n =2..6: Số điểm Gauss sử dụng  
39     Int=0; xw=gauss(n)  
40     for i in range(xw.shape[0]): #i duyệt qua hàng của xw  
41         Int += f(xw[i,0],a,b)*xw[i,1] #Công thức Gauss  
42     print('I(%d)=%10.8f'%(n,Int))
```

I(2)=17.00951821
I(3)=17.04644368
I(4)=17.04657018
I(5)=17.04656914
I(6)=17.04656909

2.3.5. Công thức Gauss cầu phương.

Trên cơ sở công thức Gauss cho hàm 1 biến \rightarrow Mở rộng cho hàm 2 biến và 3 biến.

Tọa độ và trọng số sử dụng lại của hàm 1 biến.

$$I_{1D} = \int_{-1}^1 f(x) dx \approx \sum_{i=1}^n w_i \times f(x_i)$$

$$I_{2D} = \int_{-1}^1 \int_{-1}^1 f(x, y) dx dy \approx \sum_{i=1}^n \sum_{j=1}^n w_i \times w_j \times f(x_i, y_j)$$

$$I_{3D} = \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 f(x, y, z) dx dy dz \approx \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n w_i \times w_j \times w_k \times f(x_i, y_j, z_k)$$

2.3.5. Công thức Gauss cầu phương.

Ví dụ 2.13. Sử dụng công thức Gauss tính tích phân:

Giải. $I = \int_{-1}^1 \int_{-1}^1 (0.2 + 25x - 200y^2 + 675x^3 - 900y^4 + 400x^5) dx dy$

```

1 def f(x, y):
2     fxy=0.2+25*x-200*y**2+675*x**3-900*y**4+400*x**5
3     return fxy
4 for n in range(2, 7):
5     xw=gauss(n); Int=0
6     for i in range(xw.shape[0]):
7         for j in range(xw.shape[0]):
8             Int += xw[i, 1]*xw[j, 1]*f(xw[i, 0], xw[j, 0])
9     print('I(%d)=%10.8f' % (n, Int))
I(2)=-665.86666667
I(3)=-985.86666667
I(4)=-985.86666667
I(5)=-985.86666667
I(6)=-985.86666667

```

$$I_{2D} \approx \sum_{i=1}^n \sum_{j=1}^n w_i \times w_j \times f(x_i, y_j)$$

2.3.5. Công thức Gauss cầu phương.

Ví dụ 2.14. Sử dụng công thức Gauss tính tích phân:

$$I = \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 (0.2 + 25x - 200y^2 + 675z^3 - 900y^4 + 400z^5) dx dy dz$$

Giải.

```

32| def f(x, y, z):
33|     fxyz=0.2+25*x-200*y**2+675*z**3-900*y**4+400*z**5
34|     return fxyz
35| for n in range(2, 7):
36|     xw=gauss(n); Int=0
37|     for i in range(xw.shape[0]):
38|         for j in range(xw.shape[0]):
39|             for k in range(xw.shape[0]):
40|                 Int += xw[i, 1]*xw[j, 1]*xw[k, 1]*f(xw[i, 0], xw[j, 0], xw[k, 0])
41|     print('I(%d)=%10.8f' % (n, Int))

```

I(2)=-1331.73333333

I(3)=-1971.73333333

I(4)=-1971.73333333

I(5)=-1971.73333333

I(6)=-1971.73333333

$$I_{3D} \approx \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n w_i \times w_j \times w_k \times f(x_i, y_j, z_k)$$

2.3.5. Công thức Gauss cầu phương.

Ví dụ 2.15. Sử dụng công thức Gauss tính tích phân:

$$I = \int_{-2}^5 \int_0^3 \int_{-3}^{-2} (5 + 3y + 2y^4 + 6x^2y^3 + y^2z^4 - 4xy^2z^7) dx dy dz$$

Giải.

```

1 def f(t,u,v,x1,x2,y1,y2,z1,z2):
2     x=(x2+x1)/2+t*(x2-x1)/2; dx=(x2-x1)/2 #Đổi biến x qua t
3     y=(y2+y1)/2+u*(y2-y1)/2; dy=(y2-y1)/2 #Đổi biến y qua u
4     z=(z2+z1)/2+v*(z2-z1)/2; dz=(z2-z1)/2 #Đổi biến z qua v
5     fxyz=5+3*y+2*y**4+6*x**2*y**3+y**2*z**4-4*x*y**2*z**7
6     fxyz=fxyz*dx*dy*dz
7     return fxyz
8 x1,x2 = -2,5 #Cận tích phân theo x
9 y1,y2 = 0,3 #Cận tích phân theo y
10 z1,z2 = -3,-2 #Cận tích phân theo z
11 for n in range(2,7): #Số điểm Gauss theo 1 phương: 2..6
12     xw=gauss(n); Int=0
13     for i in range(xw.shape[0]):
14         for j in range(xw.shape[0]):
15             for k in range(xw.shape[0]):
16                 Int += xw[i,1]*xw[j,1]*xw[k,1]*\
17                     f(xw[i,0],xw[j,0],xw[k,0],x1,x2,y1,y2,z1,z2)
18     print('I(%d)=%8.4f'%(n,Int))

```

$I(2) = 305657.6250$ $I(3) = 306833.8875$ $I(4) = 306836.2500$ $I(5) = 306836.2500$ $I(6) = 306836.2500$
--

Bài tập.

Bài tập 2.1. Tính xấp xỉ đạo hàm từ cấp 1 đến cấp 4 bằng sai phân hướng tâm với bước $h = 10^{-2}$ tại vị trí $x = 0.5$ của hàm số: $f(x) = -0.1x^6 - 0.15x^5 - 0.5x^2 + 1.2$, đánh giá sai số.

Bài tập 2.2. Cho hàm số $f(x) = \frac{\sqrt{x^3+4x^2+5}}{e^{2x+1}}$. Tính xấp xỉ của đạo hàm từ cấp 1 đến cấp 4 tại $x = 2$ với bước $h = 0.01$

Bài tập 2.3. Bằng thực nghiệm đo được độ võng theo phương y dọc theo phương x của dầm như bảng dưới. Biết: $E = 200GPa$; $I = 1 \times 10^{-9}m^4$

x[m]	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
y[cm]	0	-1.60	-3.03	-4.12	-4.74	-4.79	-4.26	-3.22	-1.88	-0.62	0

Dùng xấp xỉ của đạo hàm để tìm các giá trị moment uốn và lực cắt dọc theo dầm.

$$\text{Biết: } M = EI \frac{d^2y}{dx^2}; \quad Q = EI \frac{d^3y}{dx^3}.$$

Hàm độ võng chính xác của dầm dùng để kiểm chứng kết quả là:

$$y_{\text{chính xác}} = -4.17 \times 10^{-6}x^6 - 20.83 \times 10^{-6}x^5 + 23.44 \times 10^{-4}x^3 - 3.26 \times 10^{-2}x$$

Bài tập.

Bài tập 2.4. Dùng công thức hình thang với $n = 4$, tính: $I = \int_0^{\pi/2} (8x + 5 \cos x) dx$

Bài tập 2.5. Dùng công thức Simpson với $n = 4$, tính: $I = \int_0^{\pi/2} (8x + 5 \cos x) dx$

Bài tập 2.6. Dùng công thức hình thang với $n = 10$, tính: $I = \int_1^8 (x + e^{-3x}) dx$

Bài tập 2.7. Dùng công thức Simpson với $n = 10$, tính: $I = \int_1^8 (x + e^{-3x}) dx$

Bài tập 2.8. Dùng công thức hình thang với $n = 20$, tính: $I = \int_2^5 x^2 \sin(\sqrt{x}) \times e^x dx$

Bài tập 2.9. Dùng công thức Simpson với $n = 20$, tính: $I = \int_2^5 x^2 \sin(\sqrt{x}) \times e^x dx$

Bài tập.

Bài tập 2.10. Dùng công thức Gauss ba điểm, tính:

$$I = \int_{-1}^5 (4x^5 - 3x^4 + 8x^3 - 5x^2 + 2x - 5) dx$$

Bài tập 2.11. Dùng công thức Gauss hai điểm, tính:

$$I = \int_{-2}^5 \int_3^8 (5x^2y^3 - 3x^3y + 4xy) dx dy$$

Bài tập 2.12. Dùng công thức Gauss từ 2 đến 10 điểm, tính:

$$I = \int_{-2}^6 \int_{-4}^4 \int_{-1}^3 (5x^7y^8z^9 + 3x^6y^5 - 4y^2z^7 + x^{10}z^{11}) dx dy dz$$

Chương 3:

NỘI SUY – XẤP XỈ

Nội dung của chương.

3.1. Khái niệm.

3.2. Các phép nội suy.

 3.2.1. Nội suy tuyến tính.

 3.2.2. Nội suy Lagrange.

 3.2.3. Nội suy Newton.

3.3. Phương pháp bình phương cực tiểu để xấp xỉ hàm số.

3.1. Khái niệm.

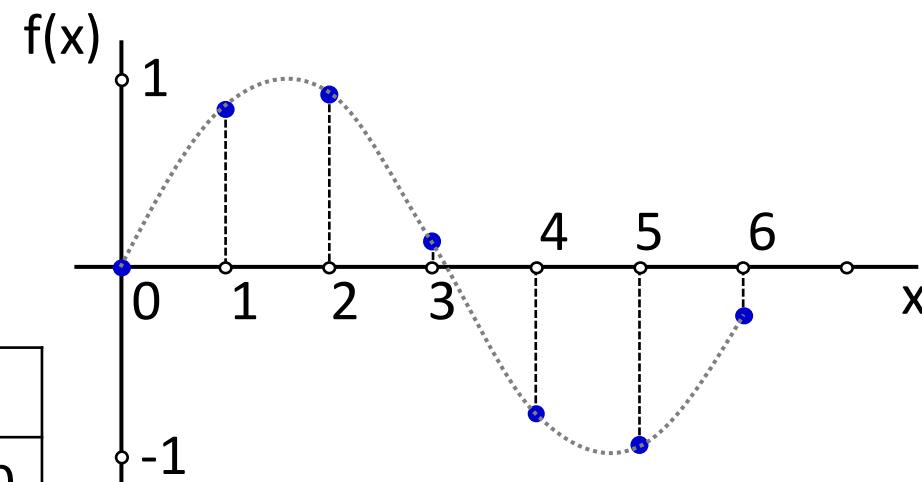
Nội suy là phương pháp ước tính giá trị của các điểm dữ liệu chưa biết trong phạm vi của một tập hợp rời rạc chứa một số điểm dữ liệu đã biết.

Trong kỹ thuật, khi có một số điểm dữ liệu đã biết giá trị bằng cách lấy mẫu thực nghiệm. Những điểm này là giá trị đại diện cho một hàm số của một biến số độc lập, gọi là hàm nội suy.

Thường chúng ta phải nội suy (ước tính) giá trị của hàm số này cho một giá trị trung gian của một biến độc lập.

Chẳng hạn, bằng thực nghiệm ta có bảng số liệu quan hệ giữa $x - f(x)$ như ở dưới. Khi biểu diễn lên đồ thị nhận được 7 điểm như hình vẽ. Ta cần tìm giá trị của hàm tại vị trí bất kỳ trong khoảng $0 \leq x \leq 6$

x	0	1	2	3	4	5	6
$f(x)$	0	0.8527	0.9112	0.1399	-0.7478	-0.9602	-0.2879



3.2. Các phép nội suy.

3.2.1. Nội suy tuyến tính.

Cho trước 2 điểm $A(x_1, y_1)$ và $B(x_2, y_2)$, nội suy tuyến tính là xác định phương trình đường thẳng đi qua 2 điểm này.

Phương trình của đường thẳng nội suy tuyến tính có dạng:

$$f(x) = \frac{x_2 - x}{x_2 - x_1} y_1 + \frac{x - x_1}{x_2 - x_1} y_2 = y_1 + \frac{y_2 - y_1}{x_2 - x_1} (x - x_1)$$

Ví dụ 3.1.

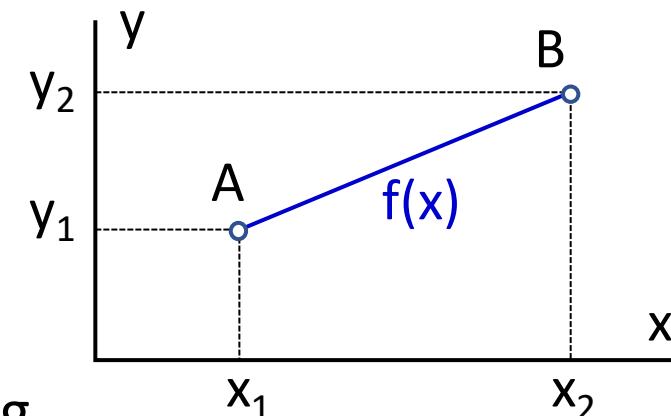
Bằng thực nghiệm, đo được giá trị của 2 điểm A, B như trên bảng.

1. Xác định hàm nội suy tuyến tính qua 2 điểm AB.
2. Xác định giá trị của hàm $f_{(x=97)}$.

Giải.

$$\begin{aligned} 1. \text{ Hàm nội suy: } f(x) &= 0.69 + \frac{0.6 - 0.69}{100 - 90} (x - 90) \\ &\Rightarrow f(x) = -0.009x + 1.5 \end{aligned}$$

$$2. f_{(x=97)} = -0.009 \times 97 + 1.5 = 0.627$$



	x_i	y_i
A	90	0.69
B	100	0.60

3.2.2. Nội suy Lagrange.

Gọi hàm nội suy $f(x) = a_0 + a_1 \textcolor{blue}{x} + a_2 \textcolor{blue}{x}^2 + \cdots + a_n \textcolor{blue}{x}^n$ (với các hệ số $a_0, a_1, a_2, \dots, a_n$ chưa biết) đi qua $n + 1$ điểm cho trước. Thay $n + 1$ điểm (cặp giá trị) đã biết vào phương trình nội suy, ta có hệ $n + 1$ phương trình để tìm các hệ số a_i :

$$\begin{cases} a_0 + \textcolor{red}{a}_1 x_0 + \textcolor{red}{a}_2 x_0^2 + \cdots + \textcolor{red}{a}_n x_0^n = y_0 \\ a_0 + \textcolor{red}{a}_1 x_1 + \textcolor{red}{a}_2 x_1^2 + \cdots + \textcolor{red}{a}_n x_1^n = y_1 \\ \cdots \\ a_0 + \textcolor{red}{a}_1 x_n + \textcolor{red}{a}_2 x_n^2 + \cdots + \textcolor{red}{a}_n x_n^n = y_n \end{cases}$$

Thay vì đi giải hệ phương trình, nội suy Lagrange đưa đến đa thức cần tìm tổng quát:

$$\begin{aligned} f(x) &= \frac{(\textcolor{blue}{x} - x_1) \times (\textcolor{blue}{x} - x_2) \times \cdots \times (\textcolor{blue}{x} - x_n)}{(x_0 - x_1) \times (x_0 - x_2) \times \cdots \times (x_0 - x_n)} \times y_0 \\ &\quad + \frac{(\textcolor{blue}{x} - x_0) \times (\textcolor{blue}{x} - x_2) \times \cdots \times (\textcolor{blue}{x} - x_n)}{(x_1 - x_0) \times (x_1 - x_2) \times \cdots \times (x_1 - x_n)} \times y_1 \\ &\quad \cdots \\ &\quad + \frac{(\textcolor{blue}{x} - x_0) \times (\textcolor{blue}{x} - x_1) \times \cdots \times (\textcolor{blue}{x} - x_{n-1})}{(x_n - x_0) \times (x_n - x_1) \times \cdots \times (x_n - x_{n-1})} \times y_n = \sum_{i=0}^n \prod_{j=0, j \neq i}^n \frac{\textcolor{blue}{x} - x_j}{x_i - x_j} \times y_i \end{aligned}$$

Ví dụ 3.2.

Bảng dưới là kết quả đo được khối lượng riêng của nước tại 4 nhiệt độ khác nhau.

1. Xác định hàm nội suy Lagrange của khối lượng riêng qua 4 điểm trên.
2. Xác định khối lượng riêng của nước ứng với nhiệt độ 10^0C và 60^0C .

Lần đo thứ i	0	1	2	3
Nhiệt độ $t[^0C]$	0	20	37	100
Khối lượng riêng $m[kg/m^3]$	999.8425	998.2071	993.3316	958.3665

Giải.

1. Hàm Lagrange bậc 3 (vì qua 4 điểm) được viết:

$$\sum_{i=0}^n \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j} \times y_i$$

$$m(t) = \frac{(t - 20)(t - 37)(t - 100)}{(0 - 20)(0 - 37)(0 - 100)} \times 999.8425 + \frac{(t - 0)(t - 37)(t - 100)}{(20 - 0)(20 - 37)(20 - 100)} \times 998.2071 \\ + \frac{(t - 0)(t - 20)(t - 100)}{(37 - 0)(37 - 20)(37 - 100)} \times 993.3316 + \frac{(t - 0)(t - 20)(t - 37)}{(100 - 0)(100 - 20)(100 - 37)} \times 958.3665$$

$$2. m_{(t=10^0C)} = 999.638 \frac{kg}{m^3}; m_{(t=60^0C)} = 982.8455 \frac{kg}{m^3}$$

Ví dụ 3.2.

Bảng dưới là kết quả đo được khối lượng riêng của nước tại 4 nhiệt độ khác nhau.

- Xác định hàm nội suy Lagrange của khối lượng riêng qua 4 điểm trên.
- Xác định khối lượng riêng của nước ứng với nhiệt độ $10^{\circ}C$ và $60^{\circ}C$.

Lần đo thứ i	0	1	2	3
Nhiệt độ $t[{}^{\circ}\text{C}]$	0	20	37	100
Khối lượng riêng $m[\text{kg/m}^3]$	999.8425	998.2071	993.3316	958.3665

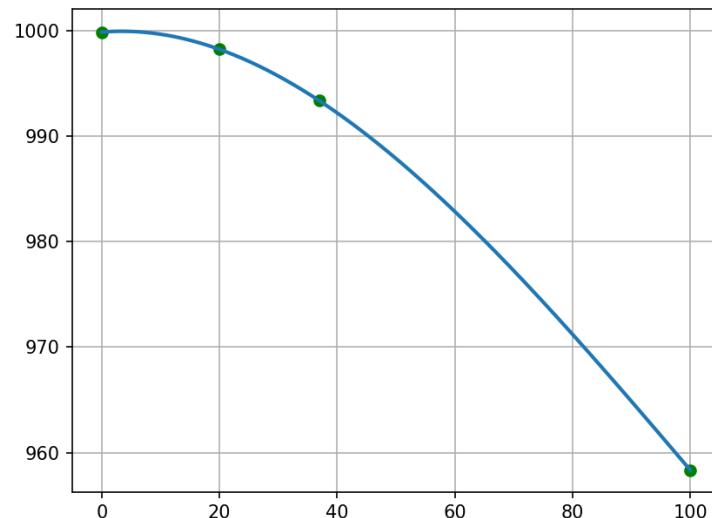
$$\sum_{i=0}^n \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j} \times y_i$$

Ví dụ 3.2.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 tT=np.array([0,20, 37, 100])
4 mT=np.array([999.8425, 998.2071, 993.3316, 958.3665])
5 def lagrange(t,tT,mT):
6     m=0
7     for i in range(len(tT)):
8         m1=mT[i]
9         for j in range(len(tT)):
10            if j != i:
11                m1 *= (t-tT[j]) / (tT[i]-tT[j])
12            m += m1
13    return m
14 print('Khối lượng riêng khi t=10oC: %5.2f kg/m^3'%lagrange(10,tT,mT))
15 print('Khối lượng riêng khi t=60oC: %5.2f kg/m^3'%lagrange(60,tT,mT))
16 tp=np.linspace(tT[0],tT[-1],100)
17 mp=lagrange(tp,tT,mT)
18 plt.plot(tT,mT, 'go'); plt.plot(tp,mp, lw=2)
19 plt.grid(); plt.show()
Khối lượng riêng khi t=10oC: 999.64 kg/m^3
Khối lượng riêng khi t=60oC: 982.85 kg/m^3

```



$$f(x) = \sum_{i=0}^n \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j} \times y_i$$

3.2.3. Nội suy Newton.

Nếu có $n + 1$ điểm đã biết tọa độ: $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ thì đa thức nội suy Newton sẽ có dạng:

$$f(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1})$$

Trong đó $a_0, a_1, a_2, \dots, a_n$ là các hệ số chưa biết và được tìm thông qua các đồng nhất sau:

Thay $x = x_0 \Rightarrow a_0 = y_0$

$$\text{Thay } x = x_1 \Rightarrow y_0 + a_1(x_1 - x_0) = y_1 \Rightarrow a_1 = \frac{y_1 - y_0}{x_1 - x_0}$$

$$\text{Thay } x = x_2 \Rightarrow y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = y_2$$

$$\Rightarrow a_2 = \frac{y_2 - y_0}{(x_2 - x_0)(x_2 - x_1)} - \frac{y_1 - y_0}{(x_2 - x_1)(x_1 - x_0)}$$

...

$$\text{Thay } x = x_n \Rightarrow \dots = y_n \Rightarrow a_n = \dots$$

Một cách tổng quát để xác định các hệ số $a_i, i = 0, 1, \dots, n$ thông qua khái niệm sai phân sau:

3.2.3. Nội suy Newton.

Gọi $f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$ là sai phân cấp 1 của hàm f trên $[x_i, x_{i+1}]$.

$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$ là sai phân cấp 2

$f[x_i, x_{i+1}, \dots, x_{i+p}] = \frac{f[x_{i+1}, x_{i+2}, \dots, x_{i+p}] - f[x_i, x_{i+1}, \dots, x_{i+p-1}]}{x_{i+p} - x_i}$ là sai phân cấp p

Ta có: $a_0 = y_0$.

$$a_1 = f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

$$a_2 = f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}.$$

...

$$a_n = f[x_0, x_1, \dots, x_n] = \frac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0}.$$

Thì đa thức nội suy Newton viết lại:

$$\begin{aligned} f(x) = y_0 + f[x_0, x_1] \times (\textcolor{blue}{x} - x_0) + f[x_0, x_1, x_2] \times (\textcolor{blue}{x} - x_0)(\textcolor{blue}{x} - x_1) + \\ \dots + f[x_0, x_1, \dots, x_n] \times (\textcolor{blue}{x} - x_0)(\textcolor{blue}{x} - x_1) \dots (\textcolor{blue}{x} - x_{n-1}) \end{aligned}$$

Ví dụ 3.3.

Làm lại Ví dụ 2.2 bằng theo đa thức nội suy Newton.

Giải.

1. Lập bảng tính sai phân:

t_i	$m(t_i)$	$m[x_i, x_{i+1}]$	$m[x_i, x_{i+1}, x_{i+2}]$	$m[x_i, x_{i+1}, x_{i+2}, x_{i+3}]$
0	999.8425			
		-0.08177		
20	998.2071		-0.00554	
		-0.28679		0.00002
37	993.3316		-0.00335	
		-0.55500		
100	958.3665			

Đa thức nội suy theo Newton:

$$m(t) = 999.8425 - 0.08177t - 0.00554t(t - 20) - 0.00002t(t - 20)(t - 37)$$

$$2. m_{(t=10^0C)} = 999.638 \frac{kg}{m^3}; m_{(t=60^0C)} = 982.8455 \frac{kg}{m^3}$$

Ví dụ 3.3.

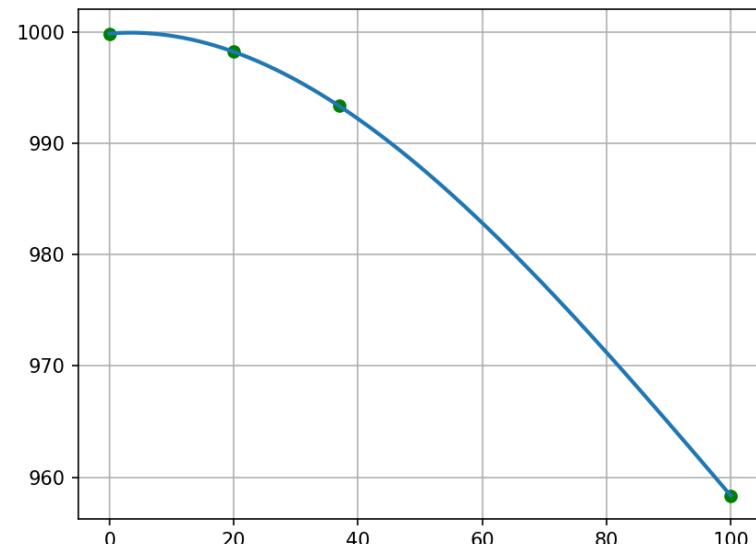
Code:

```

1 import numpy as np      #Nhập gói tính toán số vào chương trình
2 import matplotlib.pyplot as plt #Nhập gói vẽ vào chương trình
3 t0=0; t1=20; t2=37; t3=100  #Khai báo
4 m0=999.8425; m1=998.2071; m2=993.3316; m3=958.3665 #Khai báo
5 ms1_1=(m1-m0)/(t1-t0); ms1_2=(m2-m1)/(t2-t1); ms1_3=(m3-m2)/(t3-t2)
6 ms2_1=(ms1_2-ms1_1)/(t2-t0); ms2_2=(ms1_3-ms1_2)/(t3-t1)
7 ms3_1=(ms2_2-ms2_1)/(t3-t0)
8 def mt(t):
9     m=m0+ms1_1*(t-t0)+ms2_1*(t-t0)*(t-t1)+ms3_1*(t-t0)*(t-t1)*(t-t2)
10    return m
11 print('ms1_1=',ms1_1); print('ms1_2=',ms1_2); print('ms1_3=',ms1_3)
12 print('\n')
13 print('ms2_1=',ms2_1); print('ms2_2=',ms2_2)
14 print('\n')
15 print('ms3_1=',ms3_1)
16 print('\n')
17 t10=10; mt10=round(mt(t10),4); t60=60; mt60=round(mt(t60),4)
18 print('Khối lượng riêng khi t =',t10,'0C là: ',mt10,'[kg/m^3]')
19 print('Khối lượng riêng khi t =',t60,'0C là: ',mt60,'[kg/m^3]')
20
21 tTable=np.array([t0, t1, t2, t3]); mTable=np.array([m0, m1, m2, m3])
22 tp=np.linspace(0,100,500); mp=mt(tp)
23 plt.plot(tTable,mTable,'go'); plt.plot(tp,mp,lw=2)
24 plt.grid(); plt.show()

```

Out: | Khối lượng riêng khi t = 10 0C là: 999.638 [kg/m³]
| Khối lượng riêng khi t = 60 0C là: 982.8455 [kg/m³]



3.3. Phương pháp bình phương cực tiểu.

Trong thực tế, các giá trị y_i được xác định thông qua thực nghiệm đo đạc nên thường thiếu chính xác. Khi đó việc xây dựng một đa thức nội suy đi qua tất cả các điểm $M_i(x_i, y_i)$ cũng không còn chính xác.

Bài toán xấp xỉ thực nghiệm là tìm hàm $f(x)$ xấp xỉ của bảng $(x_i, y_i), i = 0, 1, 2, \dots, n$ theo phương pháp bình phương cực tiểu :

$$g(f) = \sum (f(x_i) - y_i)^2 \rightarrow \min$$

Hàm $f(x)$ tổng quát rất đa dạng. Để đơn giản, ta tìm hàm $f(x)$ theo dạng :

$$f(x) = a_1 f_1(x) + a_2 f_2(x) + \dots$$

Các hàm $f_1(x), f_2(x), \dots$ có thể là hàm lượng giác, lũy thừa, mũ, loga ...

3.3.1. Trường hợp $f(x) = a_1f_1(x) + a_2f_2(x)$.

Phương trình bình phương cực tiểu có dạng: $g(a_1, a_2) = \sum [a_1f_1(x_i) + a_2f_2(x_i) - y_i]^2$

Bài toán qui về tìm cực tiểu của hàm 2 biến $g(a_1, a_2)$.

Điểm dừng: (đạo hàm riêng theo biến a_1 và a_2)

$$\begin{cases} \frac{\partial g}{\partial a_1} = 2 \sum [a_1f_1(x_i) + a_2f_2(x_i) - y_i]f_1(x_i) = 0 \\ \frac{\partial g}{\partial a_2} = 2 \sum [a_1f_1(x_i) + a_2f_2(x_i) - y_i]f_2(x_i) = 0 \end{cases}$$

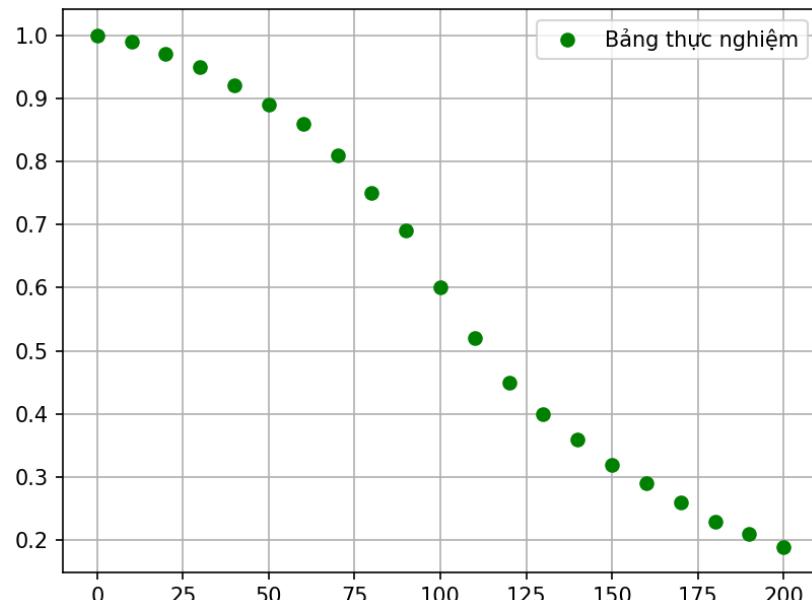
$$\Rightarrow \begin{cases} a_1 \sum f_1^2(x_i) + a_2 \sum f_1(x_i) \times f_2(x_i) = \sum y_i \times f_1(x_i) \\ a_1 \sum f_1(x_i) \times f_2(x_i) + a_2 \sum f_2^2(x_i) = \sum y_i \times f_2(x_i) \end{cases}$$

Ví dụ 3.4.

Bằng thực nghiệm, ta có bảng số liệu quan hệ giữa độ mảnh λ của thanh và hệ số giảm uốn dọc φ như bảng dưới. Dùng phương pháp bình phương cực tiểu dạng:

$f(\lambda) = a_1 + a_2\lambda$ để xác định hàm xấp xỉ của bảng.

λ	0	10	20	30	40	50	60	70	80	90	100					
φ	1	0.99	0.97	0.95	0.92	0.89	0.86	0.81	0.75	0.69	0.60					
						λ	110	120	130	140	150	160	170	180	190	200
						φ	0.52	0.45	0.40	0.36	0.32	0.29	0.26	0.23	0.21	0.19



Ví dụ 3.4.

Giải.

Viết lại điều kiện cực tiểu dạng $f(\lambda) = a_1 f_1(\lambda) + a_2 f_2(\lambda)$:

$$\begin{cases} a_1 \sum_{i=0}^{20} f_1^2(\lambda_i) + a_2 \sum_{i=0}^{20} f_1(\lambda_i) \times f_2(\lambda_i) = \sum_{i=0}^{20} \varphi_i \times f_1(\lambda_i) \\ a_1 \sum_{i=0}^{20} f_1(\lambda_i) \times f_2(\lambda_i) + a_2 \sum_{i=0}^{20} f_2^2(\lambda_i) = \sum_{i=0}^{20} \varphi_i \times f_2(\lambda_i) \end{cases}$$

Do $f(\lambda) = a_1 f_1(\lambda) + a_2 f_2(\lambda) \equiv a_1 + a_2 \lambda \Rightarrow f_1(\lambda) = 1; f_2(\lambda) = \lambda$

$$\Rightarrow \begin{cases} a_1 \sum_{i=0}^{20} 1 + a_2 \sum_{i=0}^{20} \lambda_i = \sum_{i=0}^{20} \varphi_i \\ a_1 \sum_{i=0}^{20} \lambda_i + a_2 \sum_{i=0}^{20} \lambda_i^2 = \sum_{i=0}^{20} \varphi_i \times \lambda_i \end{cases} \Rightarrow \begin{cases} 21a_1 + 2100a_2 = 12.66 \\ 2100a_1 + 287000a_2 = 901 \end{cases} \Rightarrow \begin{cases} a_1 = 1.0769 \\ a_2 = -0.0047 \end{cases}$$

Vậy, hàm xấp xỉ của bảng là: $\varphi(\lambda) = 1.0769 - 0.0047\lambda$

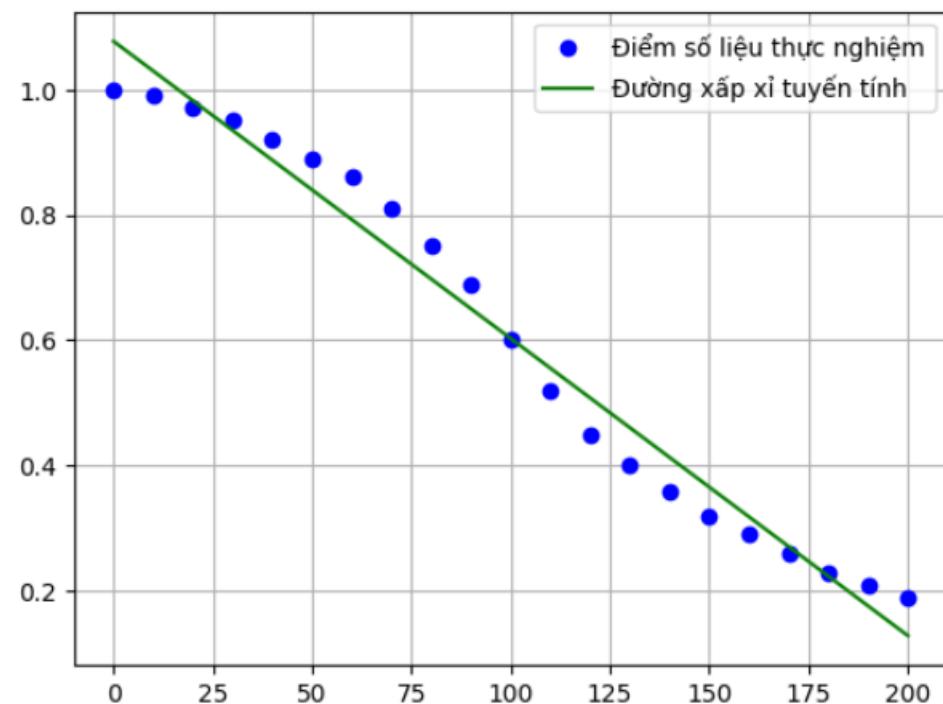
Ví dụ 3.4.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sympy as sp
4 lam = np.array([0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, \
5                 110, 120, 130, 140, 150, 160, 170, 180, 190, 200])
6 phi = np.array([1, 0.99, 0.97, 0.95, 0.92, 0.89, 0.86, 0.81, 0.75, 0.69, \
7                 0.60, 0.52, 0.45, 0.40, 0.36, 0.32, 0.29, 0.26, 0.23, 0.21, 0.19])
8 S1 = 0; SL = 0; SP = 0; SL2 = 0; SPL = 0
9 for i in range(len(lam)):
10     S1 += 1
11     SL += lam[i]
12     SP += phi[i]
13     SL2 += lam[i]**2
14     SPL += phi[i]*lam[i]
15 A = np.array([[S1, SL],
16               [SL, SL2]])
17 b = np.array([SP, SPL])
18 ai = np.linalg.solve(A,b)
19 lamb = sp.symbols('lamb')
20 phla = ai[0] + ai[1]*lamb
21 print('phi(lambda) = ', phla)
22 lamv = np.linspace(lam[0], lam[-1], 100)
23 phiv = ai[0] + ai[1]*lamv
24 plt.plot(lam, phi, 'bo', label = 'Điểm số liệu thực nghiệm')
25 plt.plot(lamv, phiv, '-g', label = 'Đường xấp xỉ tuyến tính')
26 plt.legend(); plt.grid(); plt.show()

```

$$\text{phi}(\lambda) = 1.07688311688312 - 0.00474025974025974\lambda$$



Ví dụ 3.5.

Bảng bên là các số liệu bằng thực nghiệm. Tìm hàm dạng $f(x) = a_1 \cos x + a_2 \sin x$ bằng phương pháp bình phương cực tiểu.

x	0	0.4	0.8	1.2	1.6	2.0
y	1.62	2.49	3.15	3.19	2.61	1.72

Giải.

Theo phương pháp bình phương cực tiểu, ta có $f_1(x) = \cos x$; $f_2(x) = \sin x$ suy ra:

$$\begin{cases} a_1 \sum_{i=0}^5 \cos^2 x_i + a_2 \sum_{i=0}^5 \cos x_i \times \sin x_i = \sum_{i=0}^5 y_i \times \cos x_i \\ a_1 \sum_{i=0}^5 \cos x_i \times \sin x_i + a_2 \sum_{i=0}^5 \sin^2 x_i = \sum_{i=0}^5 y_i \times \sin x_i \end{cases}$$

$$\Rightarrow \begin{cases} 2.6391a_1 + 0.7886a_2 = 6.4720 \\ 0.7886a_1 + 3.3609a_2 = 10.3754 \end{cases} \Rightarrow \begin{cases} a_1 = 1.6452 \\ a_2 = 2.7010 \end{cases}$$

Vậy hàm xấp xỉ là: $f(x) = 1.6452 \cos x + 2.7010 \sin x$

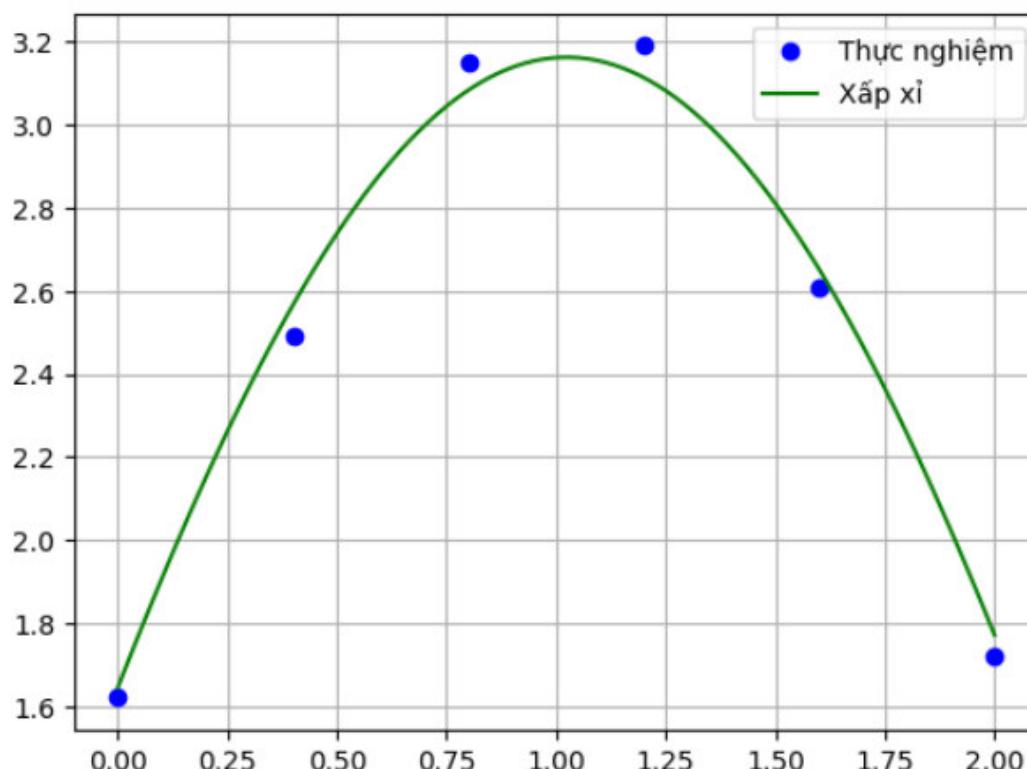
Ví dụ 3.5.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sympy as sp
4 xi=np.array([0, 0.4, 0.8, 1.2, 1.6, 2])
5 yi=np.array([1.62, 2.49, 3.15, 3.19, 2.61, 1.72])
6 c2=0; cs=0; yc=0; s2=0; ys=0
7 for i in range(len(xi)):
8     c2 += np.cos(xi[i])**2
9     cs += np.cos(xi[i])*np.sin(xi[i])
10    yc += yi[i]*np.cos(xi[i])
11    s2 += np.sin(xi[i])**2
12    ys += yi[i]*np.sin(xi[i])
13 A = np.array([[c2, cs],
14                 [cs, s2]])
15 b = np.array([yc, ys])
16 ai = np.linalg.solve(A, b)
17 x = sp.symbols('x')
18 fx = ai[0]*sp.cos(x)+ai[1]*sp.sin(x)
19 xv = np.linspace(xi[0], xi[-1], 100); fv = []
20 for i in range(len(xv)):
21     fv = np.append(fv, fx.subs(x, xv[i]))
22 print('f(x) =', fx)
23 plt.plot(xi, yi, 'bo', label='Thực nghiệm')
24 plt.plot(xv, fv, '-g', label='Xấp xỉ')
25 plt.grid(); plt.legend(); plt.show()

```

$$f(x) = 2.7010382951045 \sin(x) + 1.645244860746 \cos(x)$$



Ví dụ 3.6.

Bảng bên là các số liệu bằng thực nghiệm.

Tìm hàm dạng $f(x) = a_1x^2 + a_2\sin x$ bằng phương pháp bình phương cực tiểu.

Giải.

Theo phương pháp bình phương cực tiểu, ta có $f_1(x) = x^2$; $f_2(x) = \sin x$ suy ra:

$$\begin{cases} a_1 \sum_{i=0}^6 x_i^4 + a_2 \sum_{i=0}^6 x_i^2 \times \sin x_i = \sum_{i=0}^6 y_i \times x_i^2 \\ a_1 \sum_{i=0}^6 x_i^2 \times \sin x_i + a_2 \sum_{i=0}^6 \sin^2 x_i = \sum_{i=0}^6 y_i \times \sin x_i \end{cases}$$

$$\Rightarrow \begin{cases} 18832.1875a_1 + 6.6114a_2 = 12590.375 \\ 6.6114a_1 + 3.502a_2 = 65.9343 \end{cases}$$

$$\Rightarrow \begin{cases} a_1 = 0.6624 \\ a_2 = 17.5773 \end{cases}$$

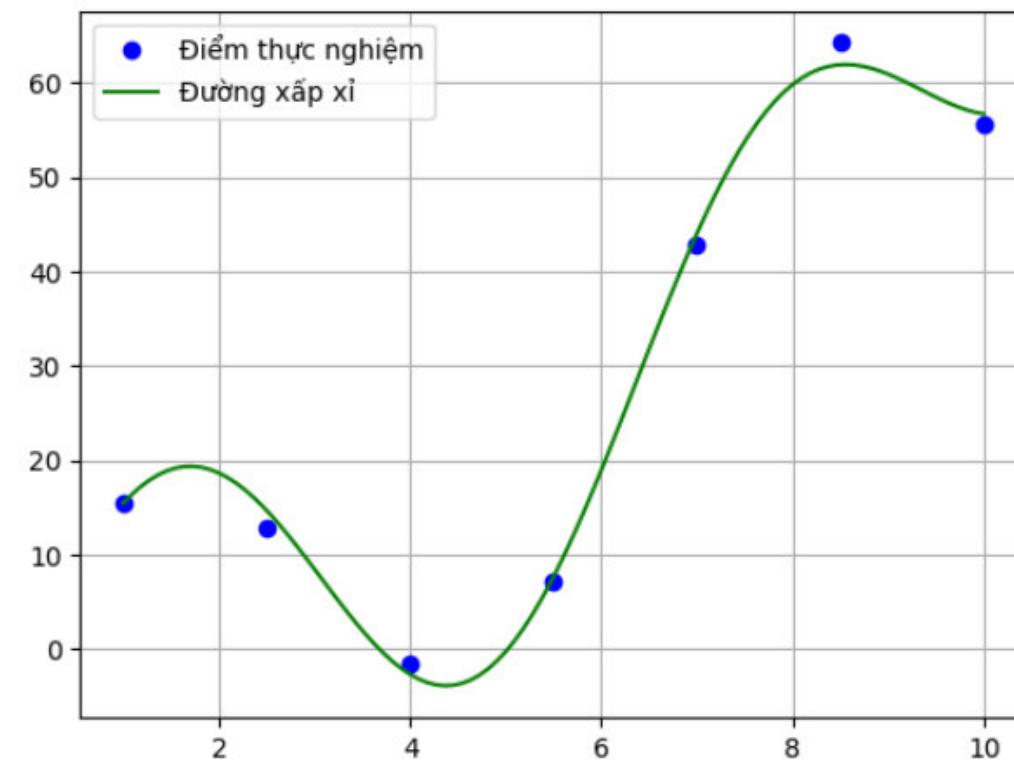
Vậy hàm xấp xỉ là: $f(x) = 0.6624x^2 + 17.5773 \sin x$

x	1	2.5	4	5.5	7	8.5	10
y	15.4	12.9	-1.5	7.2	42.9	64.2	55.6

Ví dụ 3.6.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sympy as sp
4 xi = np.array([1, 2.5, 4, 5.5, 7, 8.5, 10])
5 yi = np.array([15.4, 12.9, -1.5, 7.2, 42.9, 64.2, 55.6]
6 x4 = 0; x2s = 0; yx2 = 0; s2 = 0; ys = 0
7 for i in range(len(xi)):
8     x4 += xi[i]**4
9     x2s += xi[i]**2*np.sin(xi[i])
10    yx2 += yi[i]*xi[i]**2
11    s2 += np.sin(xi[i])**2
12    ys += yi[i]*np.sin(xi[i])
13 A = np.array([[x4, x2s],
14                 [x2s, s2]])
15 b = np.array([yx2, ys])
16 ai = np.linalg.solve(A, b)
17 x = sp.symbols('x')
18 fx = ai[0]*x**2 + ai[1]*sp.sin(x)
19 xv = np.linspace(xi[0], xi[-1], 100); fv = []
20 for i in range(len(xv)):
21     fv = np.append(fv, fx.subs(x, xv[i]))
22 print('f(x) =', fx)
23 plt.plot(xi, yi, 'bo', label='Điểm thực nghiệm')
24 plt.plot(xv, fv, '-g', label='Đường xấp xỉ')
25 plt.legend(); plt.grid(); plt.show()
```

$$f(x) = 0.662385289463896 \cdot x^2 + 17.577335922933 \cdot \sin(x)$$



3.3.2. Trường hợp $f(x) = a_1f_1(x) + a_2f_2(x) + a_3f_3(x)$.

Phương trình bình phương cực tiểu có dạng:

$$g(a_1, a_2, a_3) = \sum [a_1f_1(x_i) + a_2f_2(x_i) + a_3f_3(x_i) - y_i]^2$$

Bài toán qui về tìm cực tiểu của hàm 3 biến $g(a_1, a_2, a_3)$.

Điểm dừng: (đạo hàm riêng theo biến a_1, a_2 và a_3)

$$\begin{cases} \frac{\partial g}{\partial a_1} = 2 \sum [a_1f_1(x_i) + a_2f_2(x_i) + a_3f_3(x_i) - y_i]f_1(x_i) = 0 \\ \frac{\partial g}{\partial a_2} = 2 \sum [a_1f_1(x_i) + a_2f_2(x_i) + a_3f_3(x_i) - y_i]f_2(x_i) = 0 \\ \frac{\partial g}{\partial a_3} = 2 \sum [a_1f_1(x_i) + a_2f_2(x_i) + a_3f_3(x_i) - y_i]f_3(x_i) = 0 \end{cases}$$

3.3.2. Trường hợp $f(x) = a_1f_1(x) + a_2f_2(x) + a_3f_3(x)$.

$$\begin{cases} a_1 \sum f_1^2(x_i) + a_2 \sum f_1(x_i) \times f_2(x_i) + a_3 \sum f_1(x_i) \times f_3(x_i) = \sum y_i \times f_1(x_i) \\ a_1 \sum f_1(x_i) \times f_2(x_i) + a_2 \sum f_2^2(x_i) + a_3 \sum f_2(x_i) \times f_3(x_i) = \sum y_i \times f_2(x_i) \\ a_1 \sum f_1(x_i) \times f_3(x_i) + a_2 \sum f_2(x_i) \times f_3(x_i) + a_3 \sum f_3^2(x_i) = \sum y_i \times f_3(x_i) \end{cases}$$

Ví dụ 3.7.

Tìm hàm $f(x) = a_1 + a_2x + a_3x^2$ xấp xỉ của bảng dưới bằng phương pháp bình phương cực tiểu.

x	2	13.5	25	36.5	48	59.5	71	82.5	94
y	23.7	382.1	1055.4	2027.9	3105.8	5490.3	7059.2	9701.5	13448.3

Ví dụ 3.7.

Giải.

Theo phương pháp bình phương cực tiểu, ta có $f_1(x) = 1$; $f_2(x) = x$; $f_3(x) = x^2$ nên:

$$\left\{ \begin{array}{l} a_1 \sum_{i=0}^6 1 + a_2 \sum_{i=0}^6 x_i + a_3 \sum_{i=0}^6 x_i^2 = \sum_{i=0}^6 y_i \\ a_1 \sum_{i=0}^6 x_i + a_2 \sum_{i=0}^6 x_i^2 + a_3 \sum_{i=0}^6 x_i^3 = \sum_{i=0}^6 y_i \times x_i \\ a_1 \sum_{i=0}^6 x_i^2 + a_2 \sum_{i=0}^6 x_i^3 + a_3 \sum_{i=0}^6 x_i^4 = \sum_{i=0}^6 y_i \times x_i^2 \end{array} \right. \Rightarrow \left\{ \begin{array}{l} 9a_1 + 432a_2 + 28671a_3 = 42294.2 \\ 432a_1 + 28671a_2 + 2137968a_3 = 3147077.5 \\ 28671a_1 + 2137968a_2 + 169852148.25a_3 = 250469265.45 \end{array} \right.$$

$$\Rightarrow \begin{cases} a_1 = 188.6658 \\ a_2 = -10.8261 \\ a_3 = 1.5791 \end{cases}$$

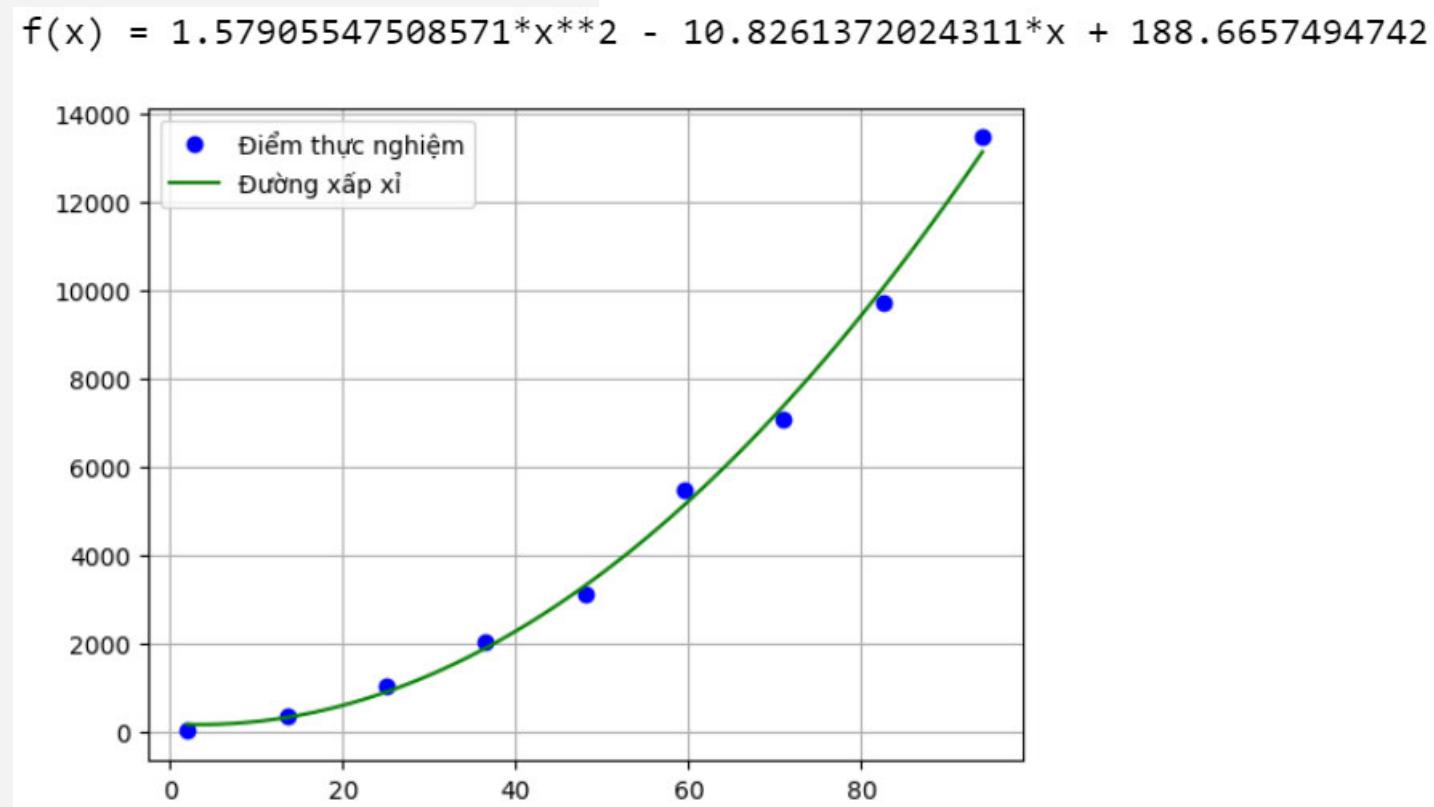
Vậy hàm xấp xỉ là: $f(x) = 188.6658 - 10.8261x + 1.5791x^2$

Ví dụ 3.7.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sympy as sp
4 xi = np.array([2, 13.5, 25, 36.5, 48, 59.5, 71, 82.5, 94])
5 yi = np.array([23.7, 382.1, 1055.4, 2027.9, 3105.8, 5490.3, 7059.2, 9701.5, 13448.3])
6 s1 = 0; sx = 0; sx2 = 0; sy = 0; sx3 = 0; syx = 0; sx4 = 0; syx2 = 0
7 for i in range(len(xi)):
8     s1 += 1
9     sx += xi[i]
10    sx2 += xi[i]**2
11    sy += yi[i]
12    sx3 += xi[i]**3
13    syx += yi[i]*xi[i]
14    sx4 += xi[i]**4
15    syx2 += yi[i]*xi[i]**2
16 A = np.array([[s1, sx, sx2],
17                 [sx, sx2, sx3],
18                 [sx2, sx3, sx4]])
19 b = np.array([sy, syx, syx2])
20 ai = np.linalg.solve(A, b)
21 x = sp.symbols('x')
22 fx = ai[0] + ai[1]*x + ai[2]*x**2
23 xv = np.linspace(xi[0], xi[-1], 100); fv = []
24 for i in range(len(xv)):
25     fv = np.append(fv, fx.subs(x, xv[i]))
26 print('f(x) =', fx)
27 plt.plot(xi, yi, 'bo', label='Điểm thực nghiệm')
28 plt.plot(xv, fv, '-g', label='Đường xấp xỉ')
29 plt.legend(); plt.grid(); plt.show()

```



3.3.3. Tổng quát, hàm xấp xỉ dạng đa thức bậc n

$$f(x) = a_0x^0 + a_1x^1 + a_2x^2 + \cdots + a_nx^n$$

Xác định các hệ số a_0, a_1, \dots, a_n bằng cách giải hệ phương trình bao gồm các hệ số là các tổng được tính từ bảng số liệu thực nghiệm.

$$\left[\begin{array}{cccc} \sum_{i=0}^n x_i^0 & \sum_{i=0}^n x_i^1 & \sum_{i=0}^n x_i^2 & \sum_{i=0}^n x_i^3 \\ \sum_{i=0}^n x_i^1 & \sum_{i=0}^n x_i^2 & \sum_{i=0}^n x_i^3 & \sum_{i=0}^n x_i^4 \\ \sum_{i=0}^n x_i^2 & \sum_{i=0}^n x_i^3 & \sum_{i=0}^n x_i^4 & \sum_{i=0}^n x_i^5 \\ \sum_{i=0}^n x_i^3 & \sum_{i=0}^n x_i^4 & \sum_{i=0}^n x_i^5 & \sum_{i=0}^n x_i^6 \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=0}^n x_i^k & \sum_{i=0}^n x_i^{k+1} & \sum_{i=0}^n x_i^{k+2} & \sum_{i=0}^n x_i^{k+3} \end{array} \right] \cdots \left[\begin{array}{c} \sum_{i=0}^n x_i^k \\ \sum_{i=0}^n x_i^{k+1} \\ \sum_{i=0}^n x_i^{k+2} \\ \sum_{i=0}^n x_i^{k+3} \\ \ddots \\ \sum_{i=0}^n x_i^{2k} \end{array} \right] = \left(\begin{array}{c} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_k \end{array} \right)$$

$sij \times ai = syxi \Rightarrow ai = sij^{-1} \times syxi$ sử dụng gói NumPy để giải: $ai = \text{numpy.linalg.solve}(sij, syxi)$

3.3.4. Hàm xấp xỉ dạng đặc biệt qui về bậc nhất.

1/ Hàm mũ: $y = ae^{bx}$

$$\Rightarrow \ln y = \ln a + bx$$

$$\Rightarrow Y = a_1 + bx$$

$$\begin{bmatrix} \sum 1 & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \times \begin{bmatrix} a_1 \\ b \end{bmatrix} = \begin{pmatrix} \sum \ln y_i \\ \sum \ln y_i \times x_i \end{pmatrix} \Rightarrow a_1, b \quad \Rightarrow a = e^{a_1}$$

2/ Hàm lũy thừa: $y = ax^b$

$$\Rightarrow \ln y = \ln a + b \ln x$$

$$\Rightarrow Y = a_1 + bX$$

$$\begin{bmatrix} \sum 1 & \sum \ln x_i \\ \sum \ln x_i & \sum (\ln x_i)^2 \end{bmatrix} \times \begin{bmatrix} a_1 \\ b \end{bmatrix} = \begin{pmatrix} \sum \ln y_i \\ \sum \ln y_i \times \ln x_i \end{pmatrix} \Rightarrow a_1, b; \quad a = e^{a_1}$$

3/ Hàm hữu tỉ: $y = \frac{ax}{b+x}$

$$\Rightarrow \frac{1}{y} = \frac{1}{a} + \frac{b}{a} \cdot \frac{1}{x}$$

$$\Rightarrow Y = a_1 + b_1 X$$

$$\begin{bmatrix} \sum 1 & \sum \frac{1}{x_i} \\ \sum \frac{1}{x_i} & \sum \frac{1}{x_i^2} \end{bmatrix} \times \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{pmatrix} \sum \frac{1}{y_i} \\ \sum \frac{1}{y_i} \cdot \frac{1}{x_i} \end{pmatrix} \Rightarrow a_1, b_1 \quad \Rightarrow \quad a = \frac{1}{a_1}, b = a \cdot b_1$$

Bài tập 3.1.

Bằng thực nghiệm, đo được giá trị của 2 điểm A, B như bảng bên.

1. Xác định hàm nội suy tuyến tính qua 2 điểm AB.
2. Xác định giá trị của hàm $f_{(x=74)}$.

	x_i	y_i
A	70	0.81
B	80	0.75

Bài tập 3.2.

Bảng dưới là kết quả đo được khối lượng riêng của nước tại 4 nhiệt độ khác nhau.

1. Xác định hàm nội suy Lagrange của khối lượng riêng qua 4 điểm trên.
2. Xác định khối lượng riêng của nước ứng với nhiệt độ $24^{\circ}C$ và $70^{\circ}C$.

Lần đo thứ i	0	1	2	3
Nhiệt độ $t[{}^{\circ}C]$	4	17	25	100
Khối lượng riêng $m[kg/m^3]$	999. 9750	998.7779	997. 0479	958.3665

Bài tập 3.3.

Giải lại Bài tập 2.2, sử dụng đa thức nội suy Newton.

Bài tập 3.4.

Bảng bên dưới là các số liệu bằng thực nghiệm. Tìm hàm $f(x) = a_1x + a_2 \sin x$ bằng phương pháp bình phương cực tiểu.

x	10	20	30	40	50	60
y	14.3888	11.7909	14.9884	26.1797	23.9814	30.3764

Bài tập 3.5.

Bảng bên dưới là các số liệu bằng thực nghiệm. Tìm hàm $f(x) = a_1 + a_2x + a_3x^2$ bằng phương pháp bình phương cực tiểu.

x	15	22	31	45	50	64	70	87
y	15.905	16.7507	13.4743	11.5074	17.6614	18.1206	31.8064	32.6331

Bài tập 3.6. Xác định hàm xấp xỉ bậc 3 bằng phương pháp bình phương cực tiểu.

Bậc 3: $f(x) = a + bx + cx^2 + dx^3$:

$$\begin{bmatrix} n & \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^4 \\ \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^4 & \sum_{i=1}^n x_i^5 \\ \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^4 & \sum_{i=1}^n x_i^5 & \sum_{i=1}^n x_i^6 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n y_i \cdot x_i \\ \sum_{i=1}^n y_i \cdot x_i^2 \\ \sum_{i=1}^n y_i \cdot x_i^3 \end{pmatrix}$$

$$x = [0. 0.63 1.25 1.88 2.5 3.13 3.75 4.38 5]$$

$$y = [17.2 14.5 15.4 19.5 21.3 29.1 33.7 40.1 49.9]$$

Bài tập 3.6. Xác định hàm xấp xỉ bậc 3 bằng phương pháp bình phương cực tiểu.

$$x = [0. \ 0.63 \ 1.25 \ 1.88 \ 2.5 \ 3.13 \ 3.75 \ 4.38 \ 5]$$
$$y = [17.2 \ 14.5 \ 15.4 \ 19.5 \ 21.3 \ 29.1 \ 33.7 \ 40.1 \ 49.9]$$

x	15	22	31	45	50	64	70	87
y	15.905	16.7507	13.4743	11.5074	17.6614	18.1206	31.8064	32.6331

Chương 4:

NGHIỆM CỦA PHƯƠNG TRÌNH PHI TUYẾN MỘT BIỀN

Nội dung của chương.

4.1. Khái niệm.

4.2. Phương pháp tìm nghiệm.

4.2.1. Phương pháp chia đôi khoảng.

4.2.2. Phương pháp Newton – Raphson.

4.2.3. Phương pháp Secant.

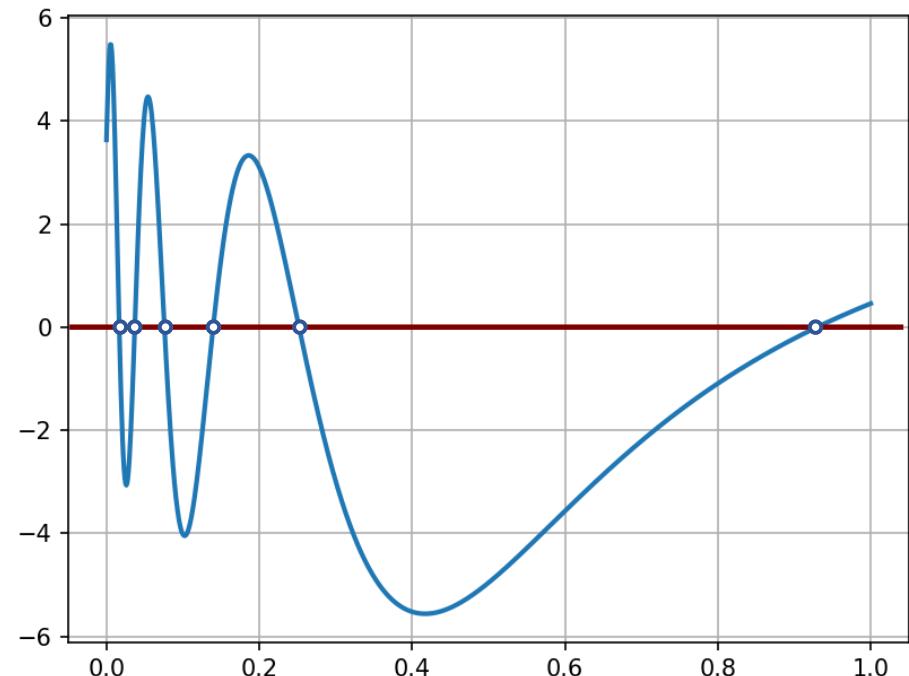
4.1. Khái niệm.

Để tìm nghiệm đúng của phương trình phi tuyến là một vấn đề khó khăn.

Chẳng hạn, cần tìm nghiệm của phương trình:

$$f(x) = 2 + 4 \cos\left(\frac{2-x}{0.1+x}\right) - (3-x)\sqrt{5x-x^2}$$

Rõ ràng, việc tìm được nghiệm đúng của phương trình này là không thể.



Thế nên, thay vì đi tìm nghiệm đúng của phương trình phi tuyến chúng ta sẽ chuyển qua tìm nghiệm gần đúng bằng các phương pháp giải lặp như:
Chia đôi khoảng, Newton – Raphson, Secant...

4.2.1 Phương pháp Bisection. (Chia đôi khoảng)

Cho trước sai số *error* và hai điểm $x = a, x = b$.

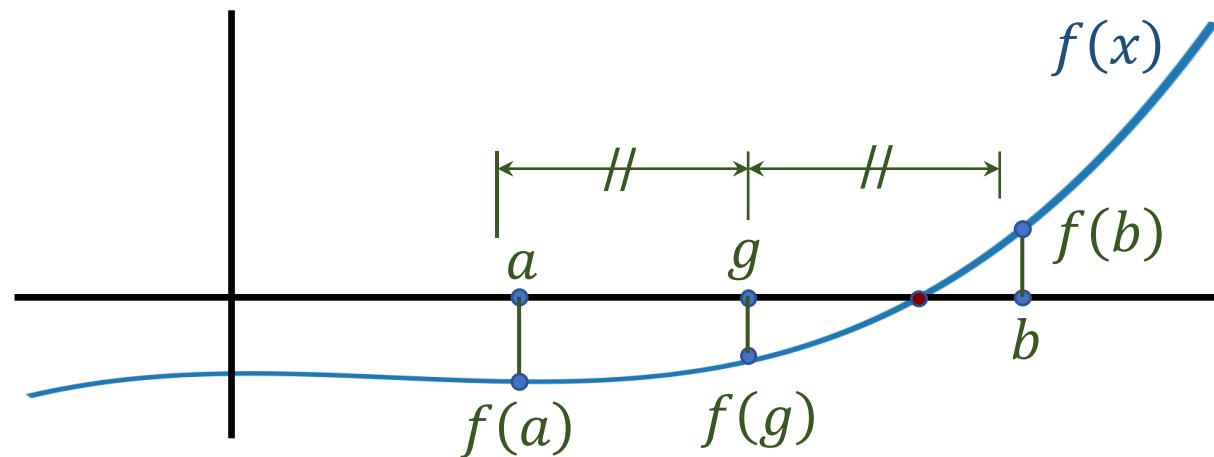
Tồn tại nghiệm trong khoảng (a, b) nếu $f(a)$ và $f(b)$ trái dấu: ($f(a) \times f(b) < 0$)

Xác định điểm g chính giữa a và b .

Nếu $f(a) \times f(g) < 0$ thì nghiệm sẽ nằm trong khoảng $(a, g) \rightarrow$ Gán $b = g$

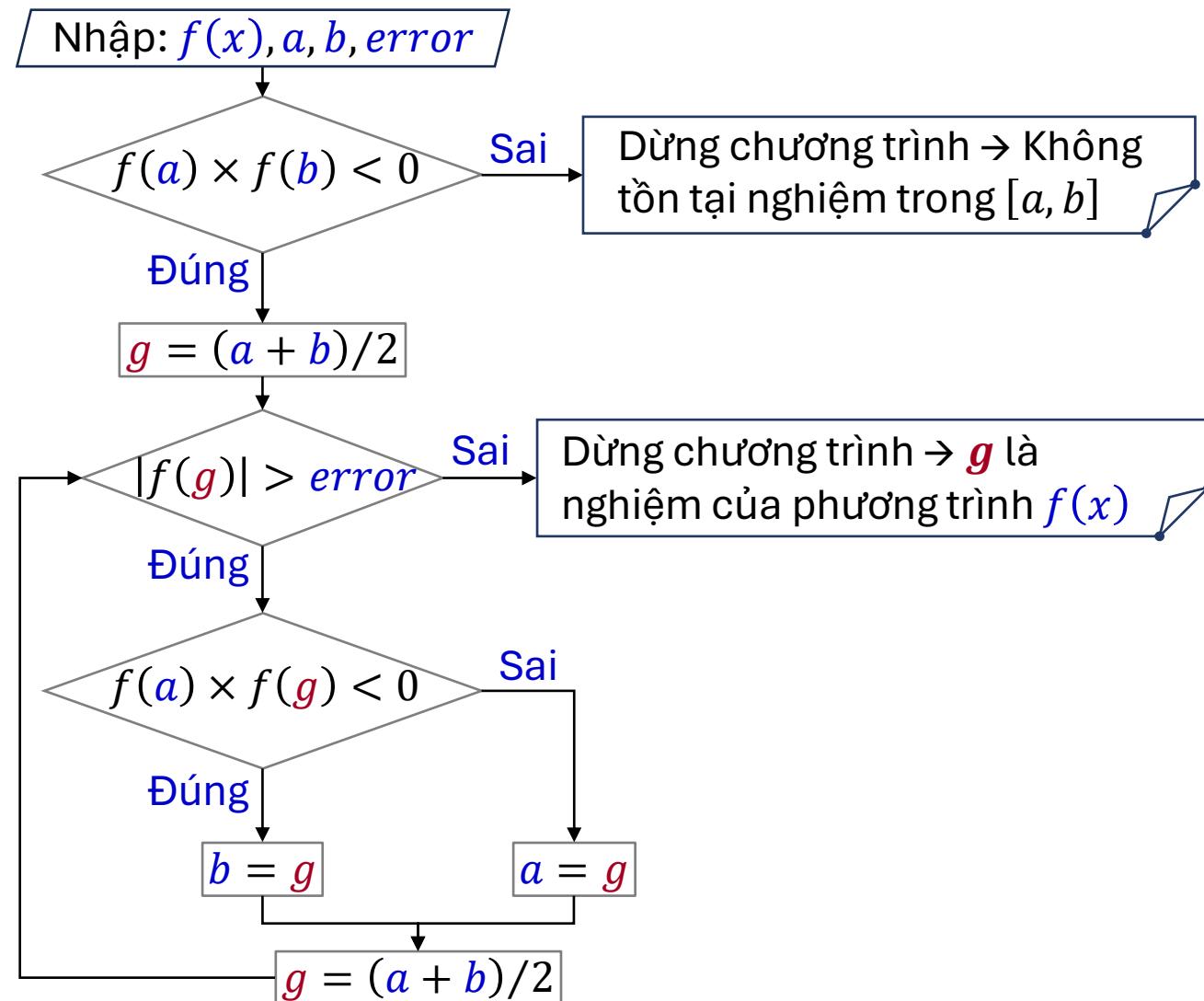
Nếu $f(a) \times f(g) > 0$ thì nghiệm sẽ nằm trong khoảng $(g, b) \rightarrow$ Gán $a = g$

Lặp lại bước tìm g cho đến khi $f(g) \leq \text{error}$ thì dừng $\rightarrow g$ là nghiệm.



Phương pháp này không tìm tất cả nghiệm mà chỉ tìm 1 nghiệm trong khoảng (a, b) . Đây là phương pháp đơn giản, nhưng hội tụ chậm.

4.2.1 Phương pháp Bisection. (Chia đôi khoảng)



4.2.1 Phương pháp Bisection. (Chia đôi khoảng)

Ví dụ 4.1: Tìm nghiệm trong khoảng $[0.6, 0.9]$ với sai số $err = 10^{-2}$ của phương trình:
 $f(x) = x^5 - 3x^3 + 1$ bằng phương pháp Bisection.

Giải.

$$f_{(a=0.6)} = 0.6^5 - 3 \times 0.6^3 + 1 = 0.43$$

$$f_{(b=0.9)} = 0.9^5 - 3 \times 0.9^3 + 1 = -0.60$$

$$g = (0.6 + 0.9)/2 = 0.75$$

$$f_{(g=0.75)} = 0.75^5 - 3 \times 0.75^3 + 1 = -0.03 \quad |-0.03| > 0.01$$

$$f_{(a)} \times f_{(g)} < 0 \Rightarrow g_1 = (0.6 + 0.75)/2 = 0.68$$

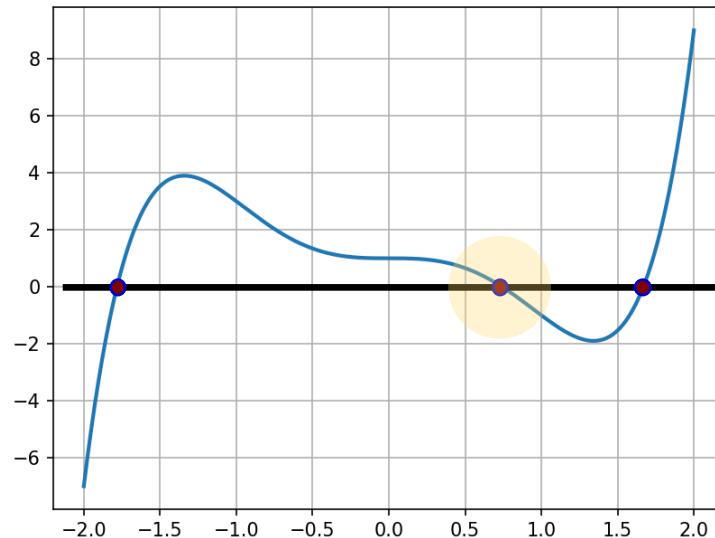
$$f_{(g_1=0.68)} = 0.68^5 - 3 \times 0.68^3 + 1 = 0.20 \quad > 0.01$$

$$f_{(a)} \times f_{(g_1)} > 0 \Rightarrow g_2 = (0.68 + 0.75)/2 = 0.72$$

$$f_{(g_2=0.72)} = 0.72^5 - 3 \times 0.72^3 + 1 = 0.07 \quad > 0.01$$

$$f_{(g_1)} \times f_{(g_2)} > 0 \Rightarrow g_3 = (0.72 + 0.75)/2 = 0.74$$

$$f_{(g_3=0.74)} = 0.74^5 - 3 \times 0.74^3 + 1 = 0.01 \quad = 0.01 \rightarrow \text{Dừng} \rightarrow \text{Nghiệm: } x = g_3 = 0.74$$



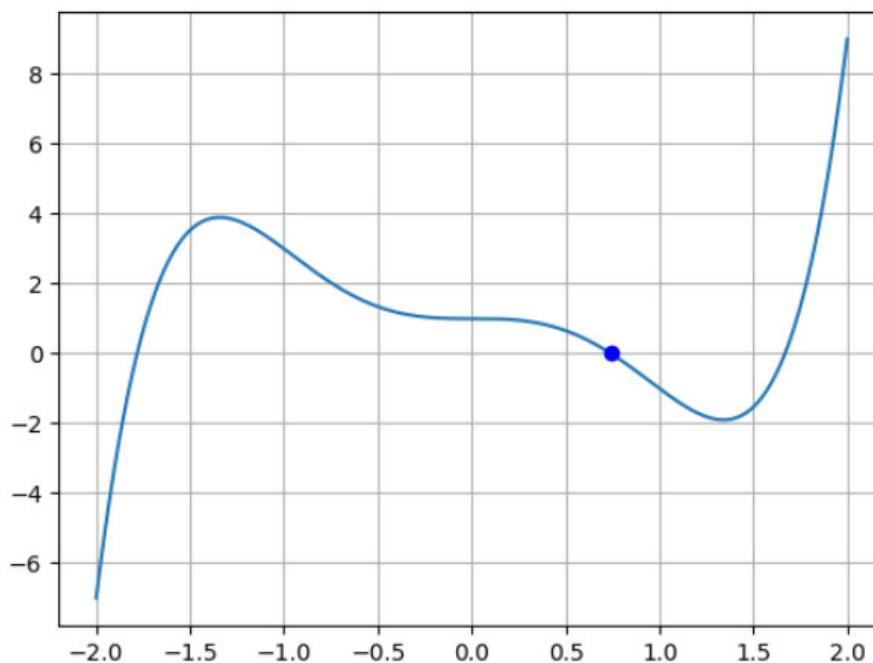
4.2.1 Phương pháp Bisection. (Chia đôi khoảng)

Ví dụ 4.1: Tìm nghiệm trong khoảng $[0.6, 0.9]$ với sai số $err = 10^{-2}$ của phương trình:
 $f(x) = x^5 - 3x^3 + 1$ bằng phương pháp Bisection.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 fx = lambda x: x**5 - 3*x**3 +1
4 er = 1e-2; a = 0.6; b = 0.9; i = 0
5 if fx(a)*fx(b)>0: print('Vô nghiệm trên [a, b]'); exit()
6 g = (a+b)/2
7 while abs(fx(g))>er:
8     if fx(a)*fx(g)<0: b=g
9     if fx(a)*fx(g)>0: a=g
10    g = (a+b)/2
11    i += 1
12    if i>1000: print('Lặp đến 1000 lần vẫn chưa tìm ra nghiệm'); break
13 print('Nghiệm x = ', round(g,4), 'sau', i, 'lần lặp.')
14 xp=np.linspace(-2,2,500); yp=fx(xp)
15 plt.plot(xp, yp, g, fx(g), 'bo'); plt.grid(); plt.show()
```

Nghiệm $x = 0.7406$ sau 4 lần lặp.



4.2.1 Phương pháp Bisection. (Chia đôi khoảng)

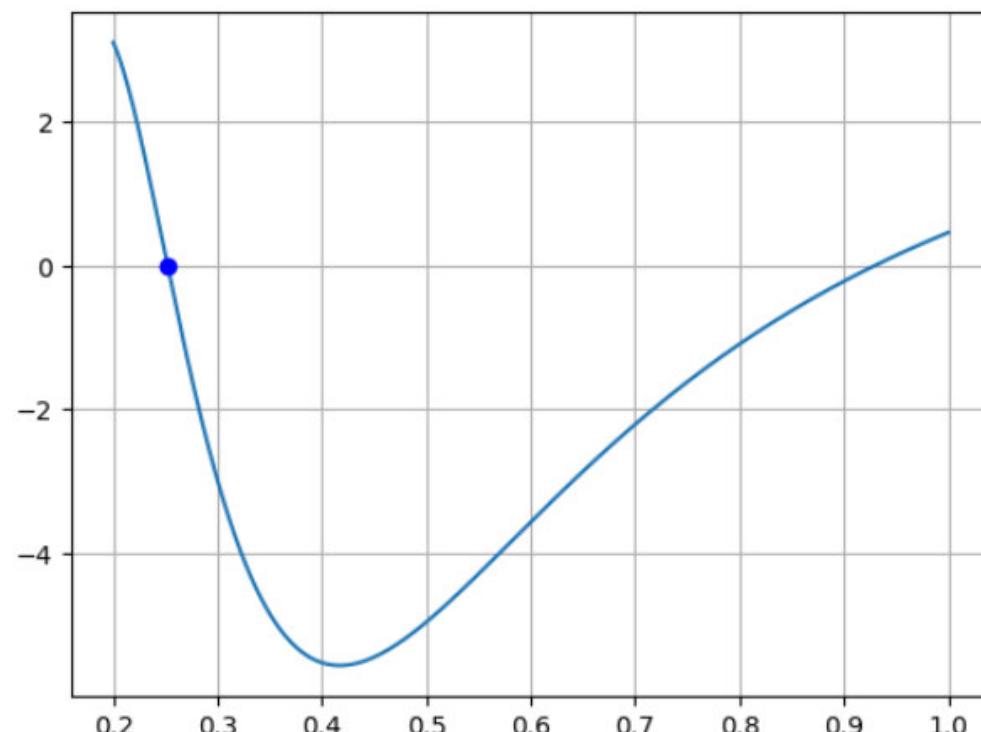
Ví dụ 4.2: Tìm nghiệm trong khoảng $[0.2, 0.4]$ với sai số $err = 10^{-5}$ của phương trình:

$$f(x) = 2 + 4 \cos\left(\frac{2-x}{0.1+x}\right) - (3-x)\sqrt{5x-x^2} \text{ bằng phương pháp Bisection.}$$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 fx = lambda x: 2+4*np.cos((2-x)/(0.1+x))-(3-x)*np.sqrt(5*x-x**2)
4 err=1e-5; a=0.2; b=0.4; i=0
5 if fx(a)*fx(b)>0: print('Vô nghiệm trên [a, b]'); exit()
6 g=(a+b)/2
7 while abs(fx(g))>err:
8     if fx(a)*fx(g)<0: b=g
9     if fx(a)*fx(g)>0: a=g
10    g=(a+b)/2
11    i += 1
12    if i>1000: print('Lặp đến 1000 lần vẫn chưa tìm ra nghiệm'); break
13 print('Nghiệm x =', round(g,4), 'sau', i, 'lần lặp.')
14 xp=np.linspace(0.2,1,500); yp=fx(xp)
15 plt.plot(xp, yp, g, fx(g), 'bo'); plt.grid(); plt.show()
```

Nghiệm $x = 0.252$ sau 19 lần lặp.



4.2.2. Phương pháp Newton – Raphson. (Tiếp tuyến)

Cho trước sai số *error* và điểm x_0 .

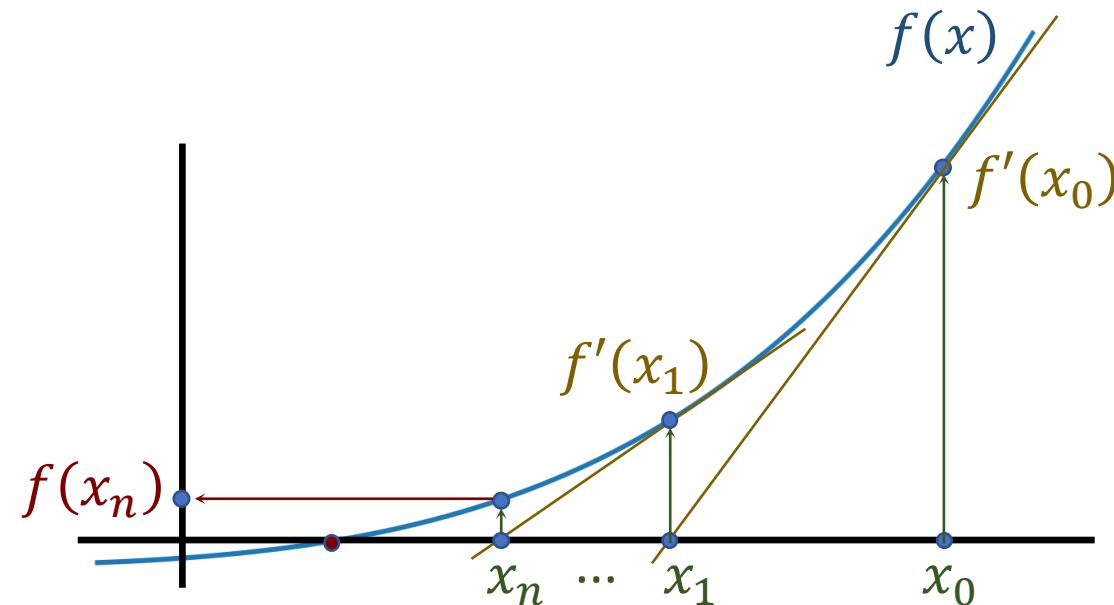
Tìm tiếp tuyến của $f(x)$ tại $x_0 \rightarrow$ cắt trục hoành tại x_1

Tìm tiếp tuyến của $f(x)$ tại $x_1 \rightarrow$ cắt trục hoành tại $x_2 \dots x_n$.

Tìm cho đến khi $f(x_n) \leq \text{error}$ thì dừng, và x_n là nghiệm.

Đường tiếp tuyến: $y - y_0 = y'(x - x_0) \Rightarrow 0 - f(x_0) = f'(x_0)(x_1 - x_0)$

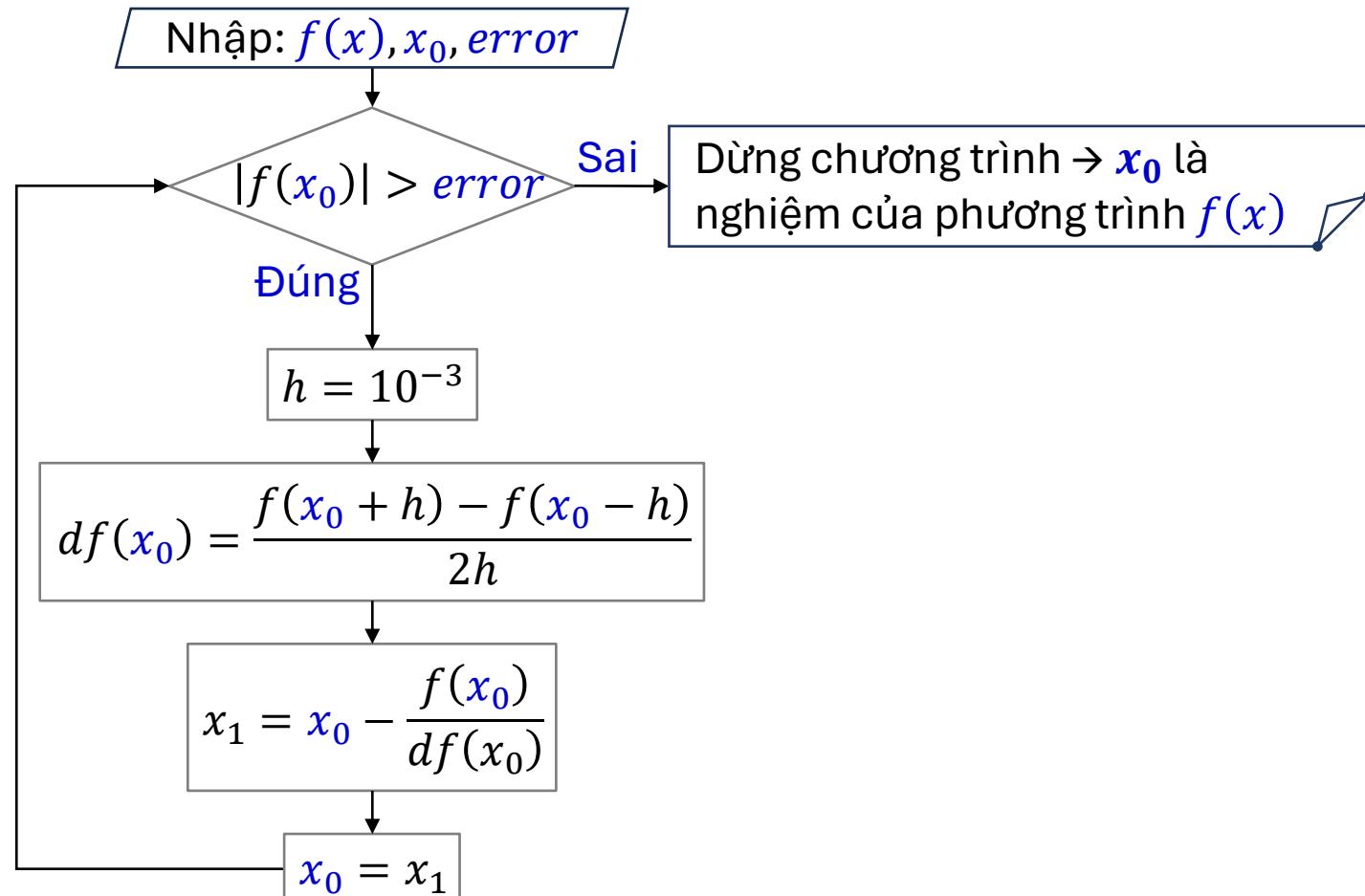
$$\Rightarrow x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



Ưu điểm: Hội tụ nhanh.

Nhược điểm: Phải tính đạo hàm cấp 1; Đồ thị phải đơn điệu (lồi hoặc lõm).

4.2.2. Phương pháp Newton – Raphson. (Tiếp tuyến)



4.2.2. Phương pháp Newton – Raphson. (Tiếp tuyến)

Ví dụ 4.3: Tìm nghiệm với điểm bắt đầu $x_0 = 0.65$ và $err = 10^{-2}$ của phương trình:
 $f(x) = x^5 - 3x^3 + 1$ bằng phương pháp Newton-Raphson.

Giải.

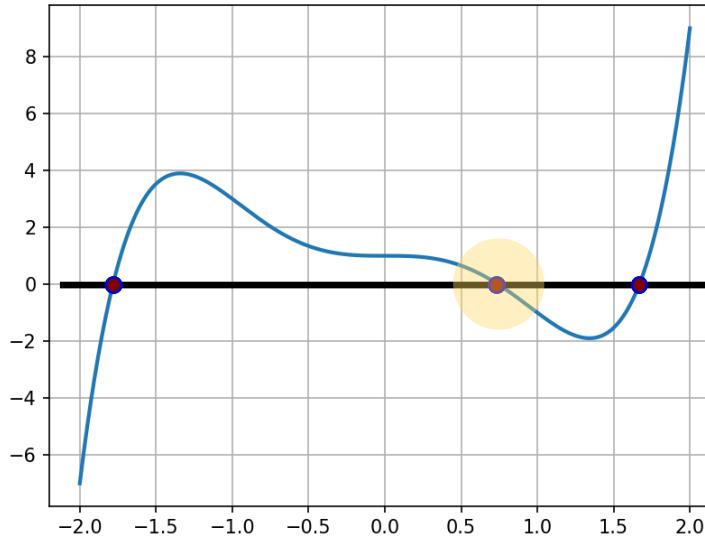
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$f'(x) = 5x^4 - 9x^2$$

$$x_1 = 0.65 - \frac{f(0.65)}{f'(0.65)} = 0.7504 \Rightarrow f(0.7504) = -0.0297 \Rightarrow |-0.0297| > 0.01$$

$$x_2 = 0.7504 - \frac{f(0.7504)}{f'(0.7504)} = 0.7419 \Rightarrow f(0.7419) = -0.0002 \Rightarrow |-0.0002| < 0.01 \rightarrow \text{Dừng}$$

→ Nghiệm: $x = x_2 = 0.7419$

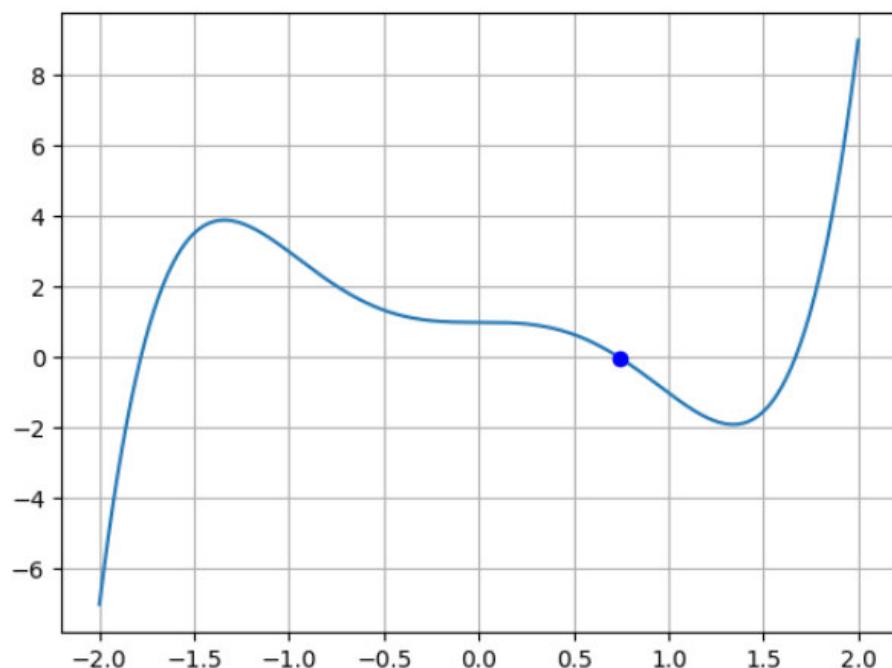


Ví dụ 4.3.:

$$x_0 = 0.6; \text{err} = 10^{-2}; f(x) = x^5 - 3x^3 + 1$$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 fx = lambda x: x**5 - 3*x**3 +1
4 df = lambda x: (fx(x+h) - fx(x-h))/2/h; h = 1e-3
5 x0 = 0.6; er = 1e-2; i = 0
6 while abs(fx(x0))>er:
7     x1 = x0 - fx(x0)/df(x0)
8     x0 = x1
9     i += 1
10    if i>1000: print('Lặp đến 1000 lần vẫn chưa tìm ra nghiệm'); break
11 print('Nghiệm x =', round(x0,4), 'sau', i, 'lần lặp.')
12 xp=np.linspace(-2,2,500); yp=fx(xp)
13 plt.plot(xp, yp, x0, fx(x0), 'bo'); plt.grid(); plt.show()
```

Nghiệm $x = 0.7422$ sau 2 lần lặp.



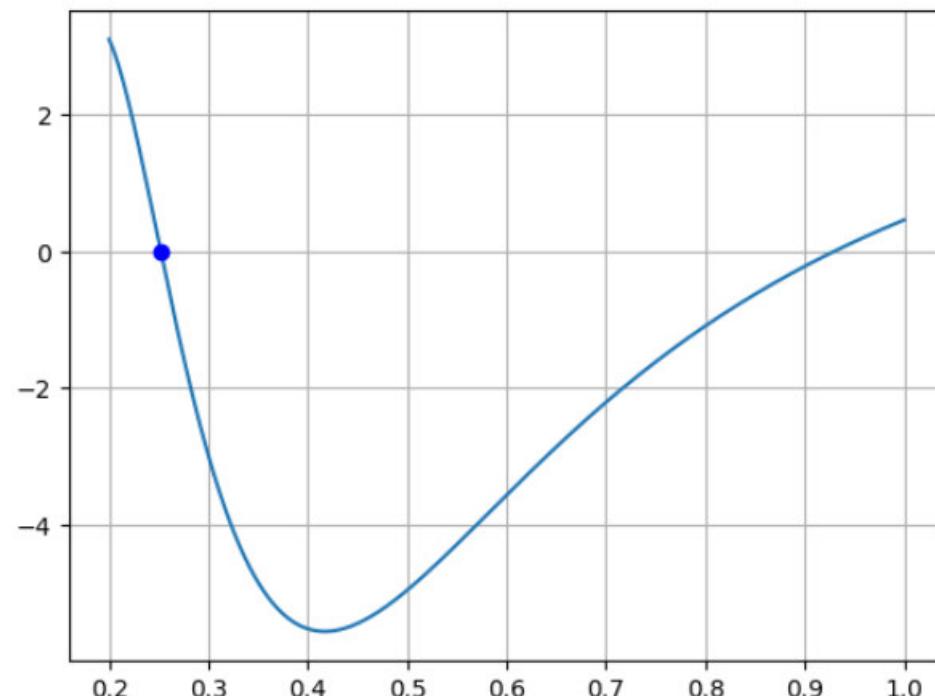
4.2.2. Phương pháp Newton – Raphson. (Tiếp tuyến)

Ví dụ 4.4: Tìm nghiệm với điểm bắt đầu $x_0 = 0.2$ và $err = 10^{-5}$ của phương trình:

$f(x) = 2 + 4 \cos\left(\frac{2-x}{0.1+x}\right) - (3-x)\sqrt{5x-x^2}$ bằng phương pháp Newton-Raphson.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 fx = lambda x: 2+4*np.cos((2-x)/(0.1+x))-(3-x)*np.sqrt(5*x-x**2)
4 dfx = lambda x: (fx(x+h)-fx(x-h))/2/h; h=1e-3
5 err = 1e-5; x0 = 0.2; i=0
6 while abs(fx(x0)) > err:
7     x1 = x0-fx(x0)/dfx(x0)
8     x0 = x1
9     i += 1
10    if i>1000: print('Lặp đến 1000 lần vẫn chưa tìm ra nghiệm'); break
11    print('Nghiệm x =', round(x0,4), 'sau', i, 'lần lặp.')
12 xp=np.linspace(0.2, 1, 500); yp=fx(xp)
13 plt.plot(xp, yp, x0, fx(x0), 'bo'); plt.grid(); plt.show()
```

Nghiệm $x = 0.252$ sau 4 lần lặp.



4.2.3. Phương pháp Secant. (Cát tuyến)

Cho trước sai số *error* và hai điểm x_1, x_2 bất kỳ

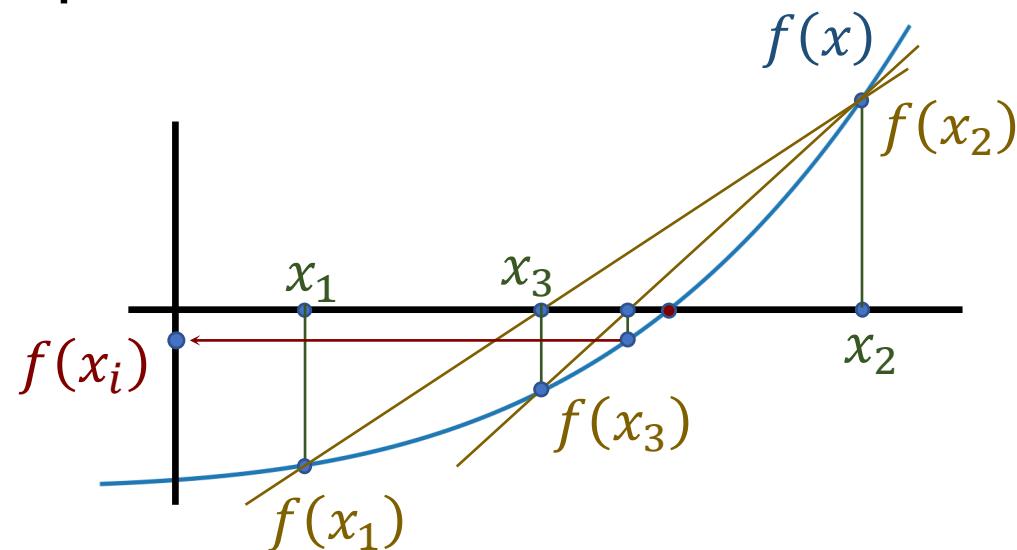
Tìm đường cát tuyến qua hai điểm $x_1, x_2 \rightarrow$ Cắt trực hoành tại x_3 .

... Tìm đường cát tuyến qua hai điểm $x_{i-2}, x_{i-1} \rightarrow$ Cắt trực hoành tại x_i

Tìm cho đến khi $f(x_i) \leq \text{error}$ thì dừng, và x_i là nghiệm.

$$\text{Đường cát tuyến: } \frac{x_{i-1} - x_{i-2}}{x - x_{i-2}} = \frac{f(x_{i-1}) - f(x_{i-2})}{f(x) - f(x_{i-2})}$$

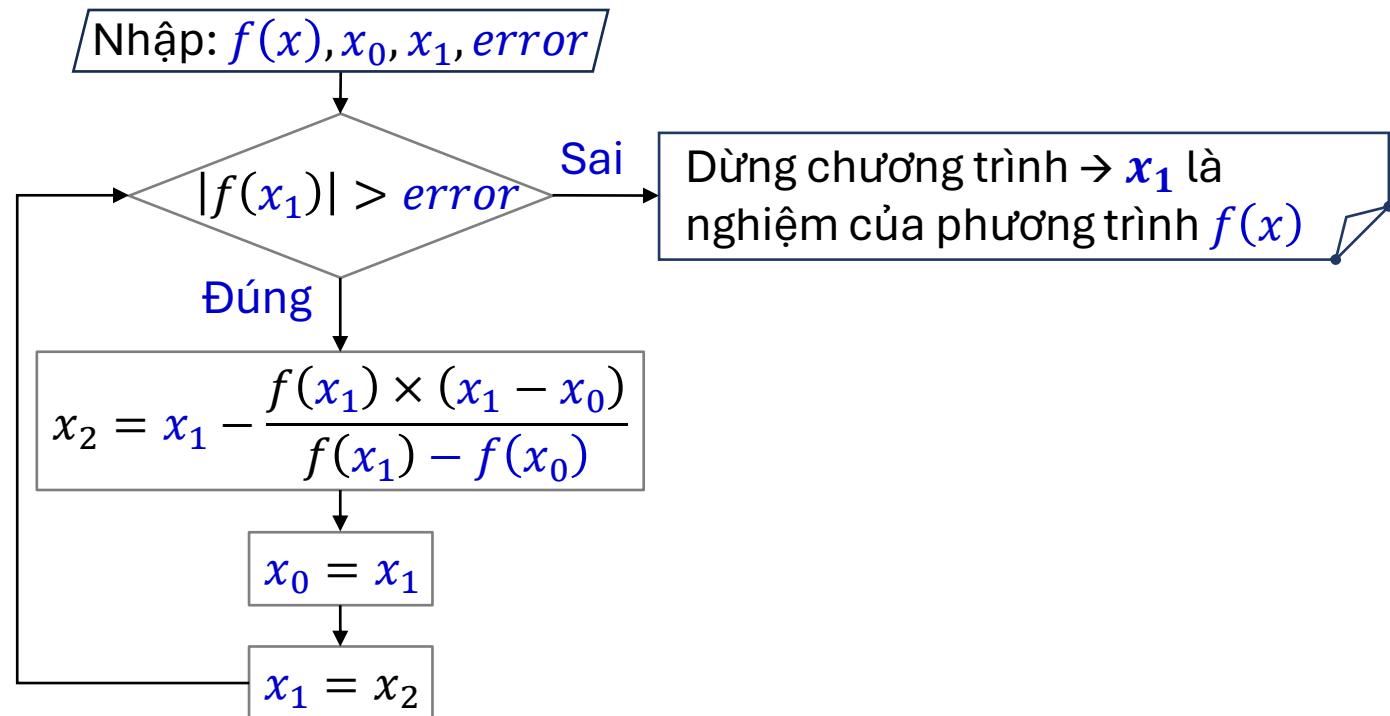
$$\Rightarrow x_i = x_{i-2} - \frac{f(x_{i-2}) \times (x_{i-1} - x_{i-2})}{f(x_{i-1}) - f(x_{i-2})}$$



Ưu điểm: Khắc phục 2 nhược điểm của phương pháp tiếp tuyến.

Nhược điểm: Hội tụ nhanh hơn phương pháp Bisection nhưng vẫn chậm.

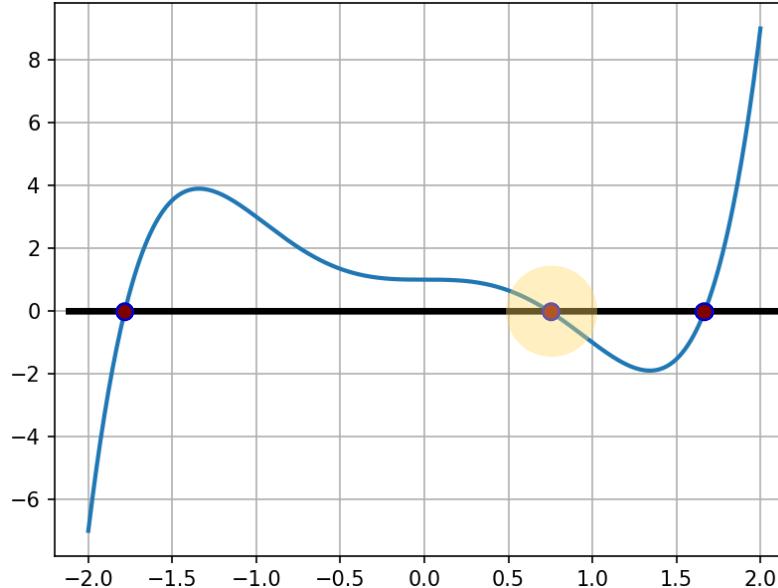
4.2.3. Phương pháp Secant. (Cát tuyến)



4.2.3. Phương pháp Secant. (Cát tuyến)

Ví dụ 4.5: Tìm nghiệm với hai điểm bắt đầu $x_0 = 0.6$ và $x_1 = 0.9$ với sai số $err = 10^{-2}$ của phương trình: $f(x) = x^5 - 3x^3 + 1$ bằng phương pháp Secant.

$$x_{i+1} = x_i - \frac{f(x_i) \times (x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$



Giải.

$$x_2 = 0.9 - \frac{f(0.9) \times (0.9 - 0.6)}{f(0.9) - f(0.6)} = 0.7256 \Rightarrow f_{(0.7256)} = 0.0550 > 0.01$$

$$x_3 = 0.7256 - \frac{f_{(0.7256)} \times (0.7256 - 0.9)}{f_{(0.7256)} - f_{(0.9)}} = 0.7403 \Rightarrow f_{(0.7403)} = 0.0051 < 0.01 \rightarrow \text{Dừng}$$

→ Nghiệm: $x = x_3 = 0.7403$

3.2.3. Phương pháp Secant. (Cát tuyến)

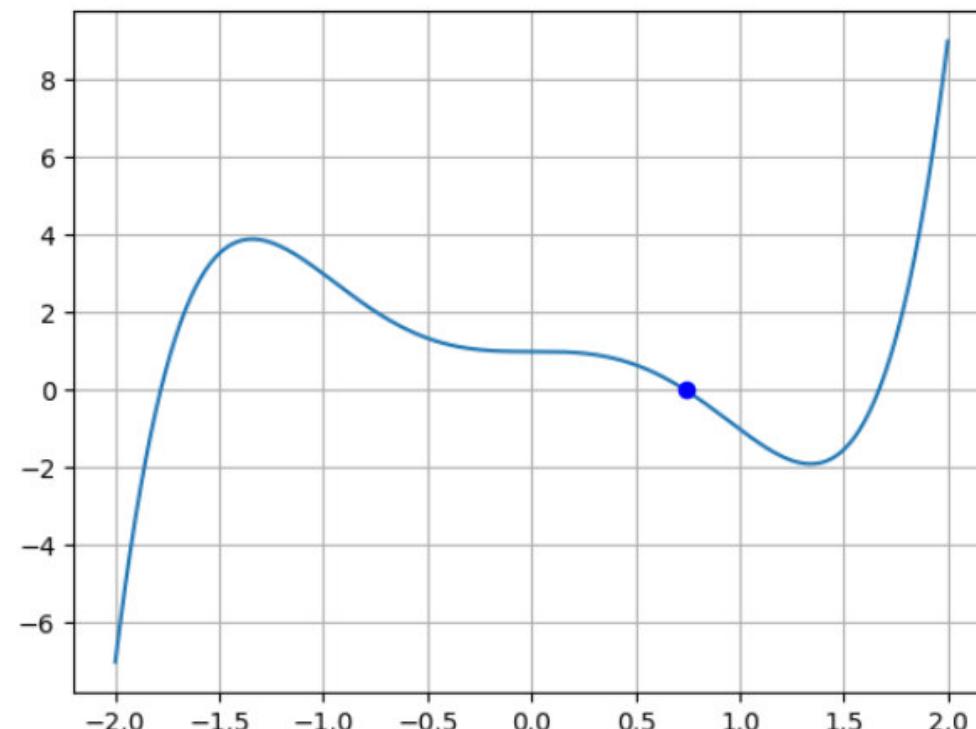
Ví dụ 4.5: Tìm nghiệm với hai điểm bắt đầu $x_0 = 0.6$ và $x_1 = 0.9$ với sai số $err = 10^{-2}$ của phương trình: $f(x) = x^5 - 3x^3 + 1$ bằng phương pháp Secant.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 fx = lambda x: x**5 - 3*x**3 + 1
4 x0 = 0.6; x1 = 0.9; er = 1e-2; i = 0
5 while abs(fx(x1))>er:
6     x2 = x1 - fx(x1)*(x1 - x0)/(fx(x1) - fx(x0))
7     x0 = x1
8     x1 = x2
9     i += 1
10    if i>1000: print('Lặp đến 1000 lần vẫn chưa tìm ra nghiệm'); break
11 print('Nghiệm x =', round(x1, 4), 'sau', i, 'lần lặp.')
12 xp=np.linspace(-2, 2, 500); yp=fx(xp)
13 plt.plot(xp, yp, x1, fx(x1), 'bo'); plt.grid(); plt.show()

```

Nghiệm $x = 0.7403$ sau 2 lần lặp.



4.2.3. Phương pháp Secant. (Cát tuyến)

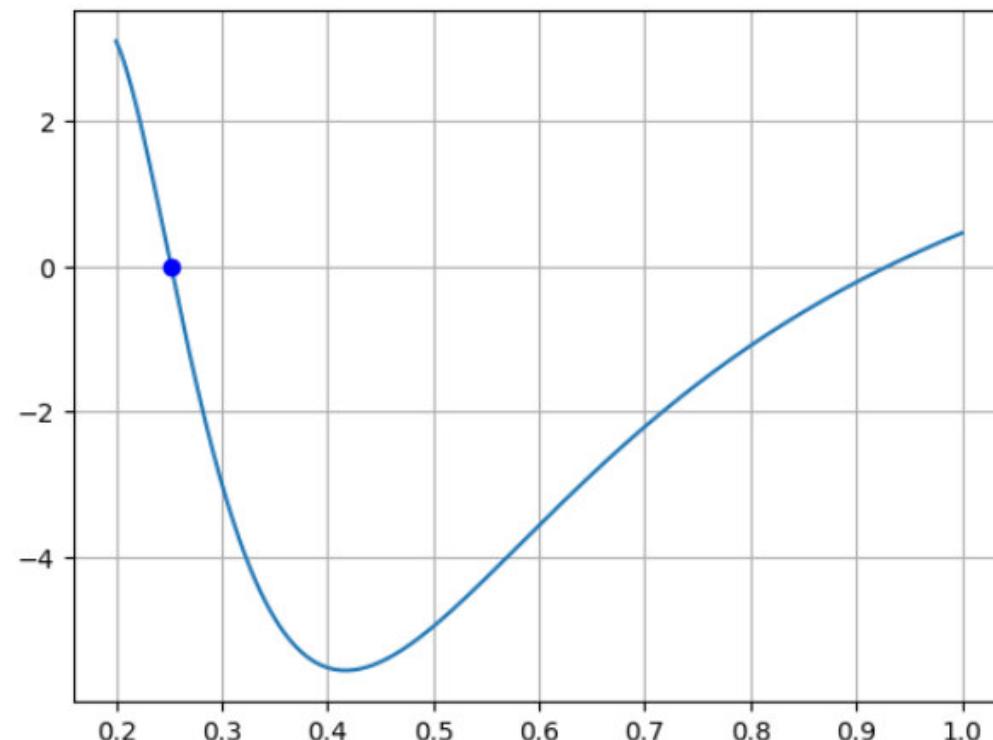
Ví dụ 4.6: Tìm nghiệm với hai điểm bắt đầu $x_0 = 0.2, x_1 = 0.4$ và sai số $err = 10^{-5}$ của phương trình: $f(x) = 2 + 4 \cos\left(\frac{2-x}{0.1+x}\right) - (3-x)\sqrt{5x-x^2}$ bằng phương pháp Secant.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 fx = lambda x: 2+4*np.cos((2-x)/(0.1+x))-(3-x)*np.sqrt(5*x-x**2)
4 err=1e-5; x0=0.2; x1=0.4; i=0
5 while abs(fx(x1)) > err:
6     x2=x1-fx(x1)*(x1-x0)/(fx(x1)-fx(x0))
7     x0 = x1
8     x1 = x2
9     i += 1
10    if i>1000: print('Lặp đến 1000 lần vẫn chưa tìm ra nghiệm'); break
11 print('Nghiệm x =', round(x1, 4), 'sau', i, 'lần lặp.')
12 xp=np.linspace(0.2, 1, 500); yp=fx(xp)
13 plt.plot(xp, yp, x1, fx(x1), 'bo'); plt.grid(); plt.show()

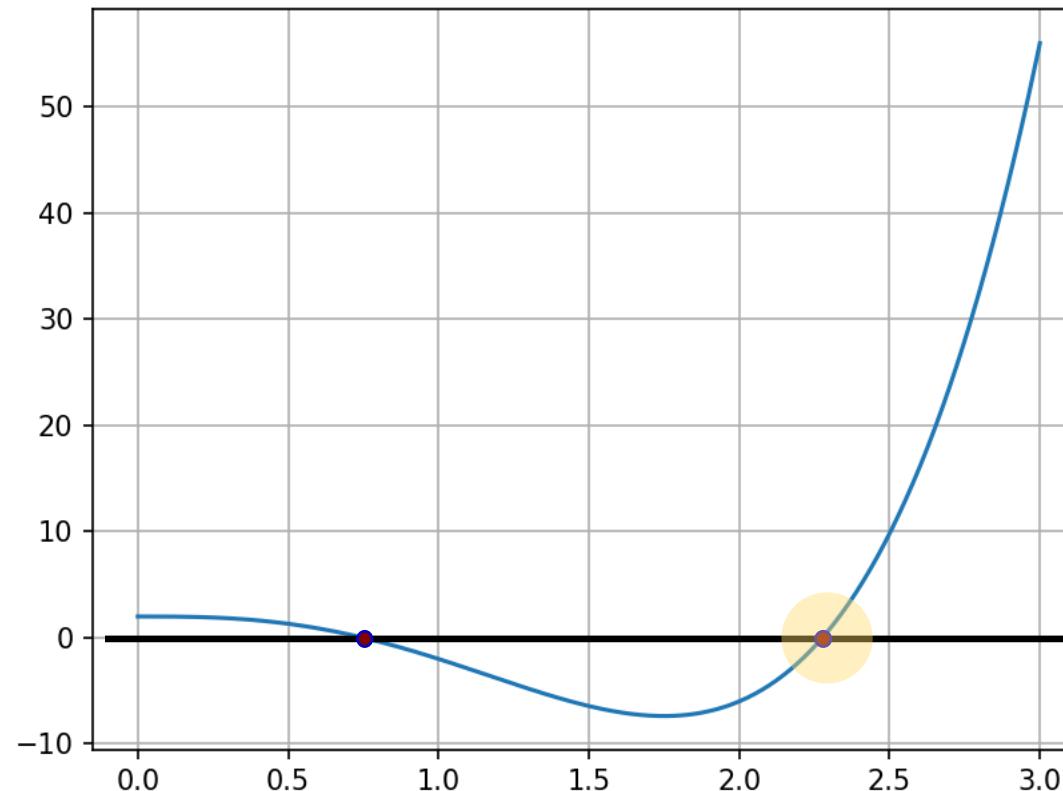
```

Nghiệm $x = 0.252$ sau 5 lần lặp.



Bài tập.

Bài tập 4.1: Cho hàm số: $f(x) = 3x^4 - 7x^3 + 2$



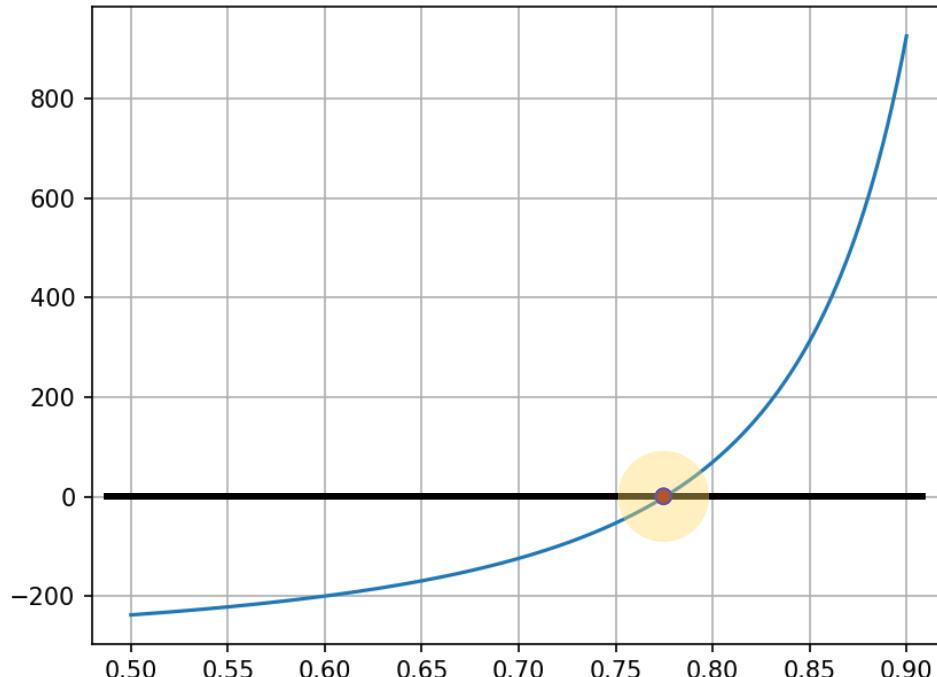
Yêu cầu: Giải trực tiếp, dùng máy tính bỏ túi Casio.

- Tìm nghiệm trong đoạn $[2, 2.5]$ với sai số $err = 10^{-2}$ bằng phương pháp Bisection.
- Tìm nghiệm với điểm bắt đầu $x_0 = 2$ và $err = 10^{-2}$ theo Newton - Raphson.
- Tìm nghiệm với hai điểm ban đầu $x_0 = 2, x_1 = 2.5$ và sai số $err = 10^{-2}$ theo Secant.

Bài tập.

Bài tập 4.2: Cho hàm số:

$$f(x) = 100 \times \left(\frac{x}{0.6}\right)^{0.625} \times \left(\frac{1-x}{0.4}\right)^{-1.625} - 300$$



- a. Tìm nghiệm trong đoạn $[0.75, 0.8]$ với sai số $err = 10^{-4}$ bằng phương pháp Bisection.
- b. Tìm nghiệm với điểm bắt đầu $x_0 = 0.75$ và $err = 10^{-4}$ theo Newton - Raphson.
- c. Tìm nghiệm với hai điểm ban đầu $x_0 = 0.75, x_1 = 0.8$ và sai số $err = 10^{-4}$ theo Secant.

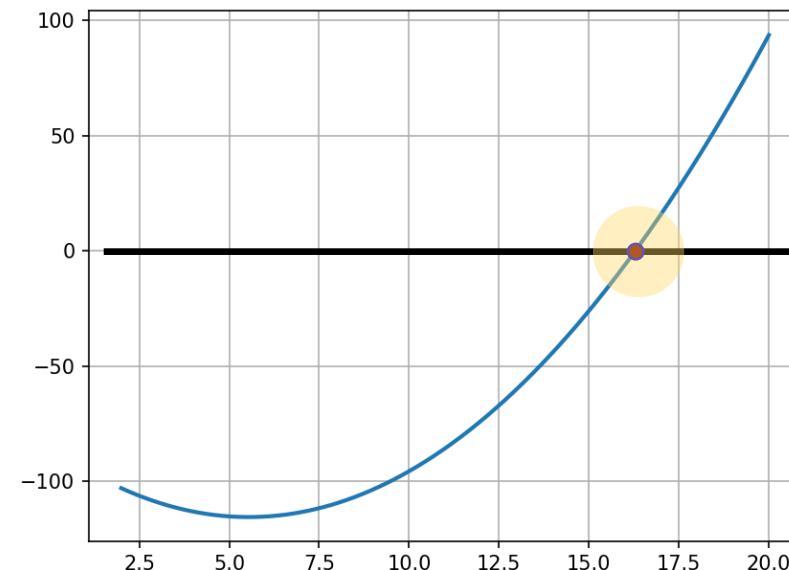
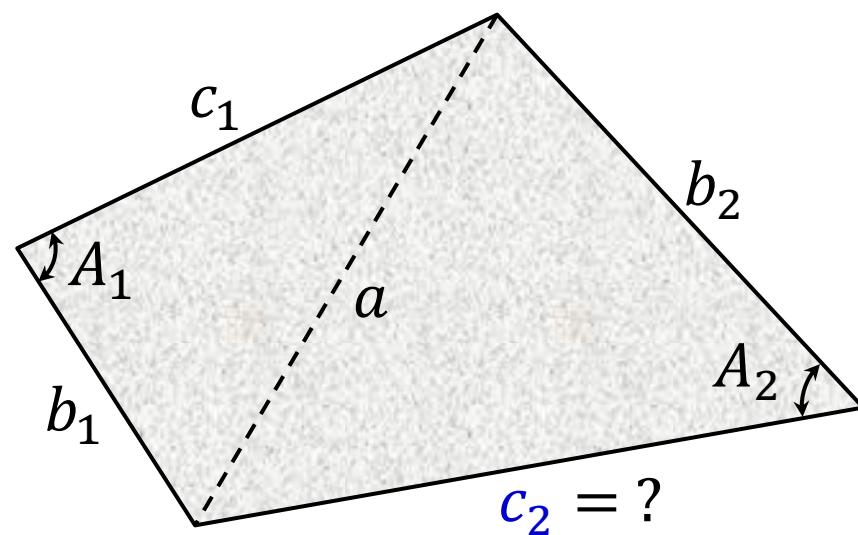
Bài tập.

Bài tập 4.3:

Cho tứ giác như hình vẽ. Hệ thức lượng trong tam giác: $a^2 = b_1^2 + c_1^2 - 2b_1c_1 \cos A_1$.

Kết hợp 2 tam giác: $\Rightarrow b_2^2 + c_2^2 - 2b_2c_2 \cos A_2 - (b_1^2 + c_1^2 - 2b_1c_1 \cos A_1) = 0$

Biết: $b_1 = 11.5m$; $c_1 = 15.7m$; $A_1 = 85^\circ$; $b_2 = 16.2m$; $A_2 = 70^\circ$.



Yêu cầu: Giải phương trình để tìm chiều dài của cạnh c_2 . Lưu ý: Radian = Độ/180.

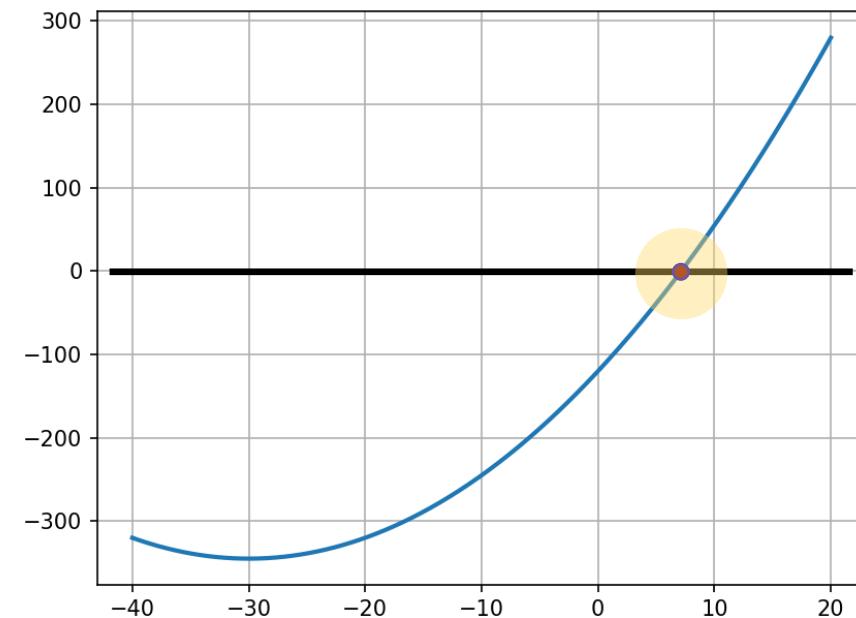
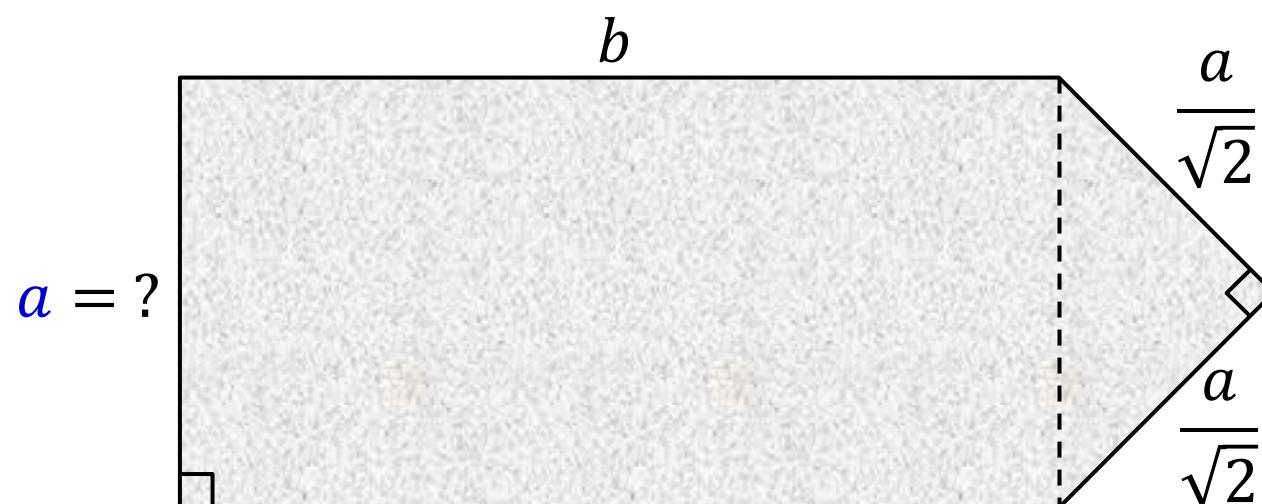
- Theo phương pháp Bisection, khoảng tìm nghiệm là $[15, 17.5]$, sai số $err = 10^{-2}$.
- Theo Newton – Raphson, với điểm bắt đầu $x_0 = 15$ và $err = 10^{-2}$.
- Theo Secant, với hai điểm ban đầu $x_0 = 15, x_1 = 17.5$ và sai số $err = 10^{-2}$.

Bài tập.

Bài tập 4.4: Cho đa giác như hình vẽ. Diện tích của đa giác được tính:

$$S = \textcolor{blue}{a} \times b + \frac{1}{2} \times \frac{\textcolor{blue}{a}}{\sqrt{2}} \times \frac{\textcolor{blue}{a}}{\sqrt{2}} = \textcolor{blue}{ab} + 0.25\textcolor{blue}{a}^2$$

Biết: $S = 120m^2$; $b = 15m$. Xác định chiều dài cạnh $\textcolor{blue}{a}$.



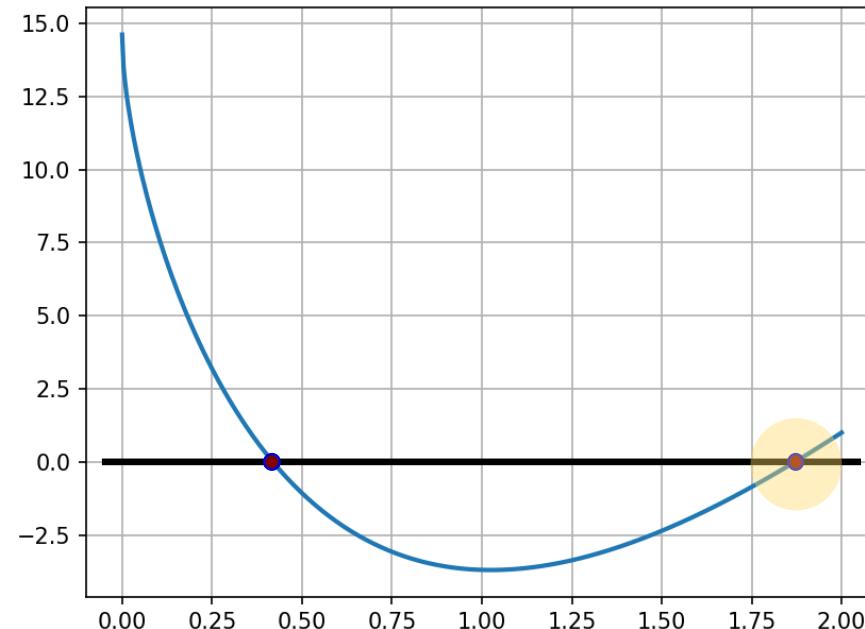
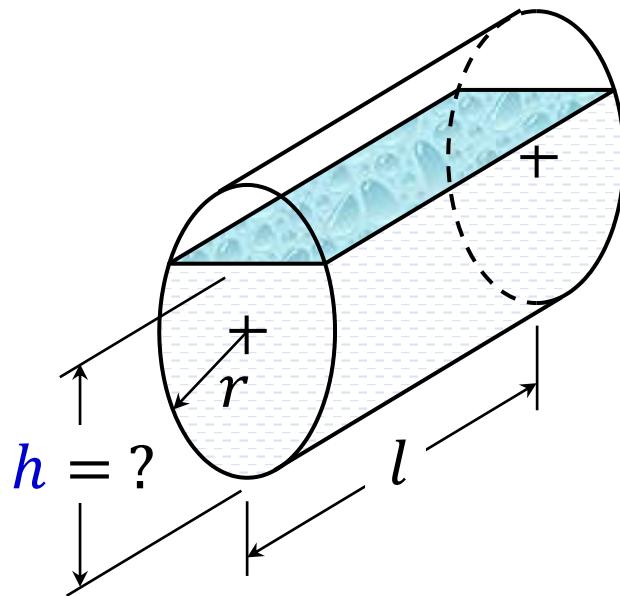
- a. Theo phương pháp Bisection, khoảng tìm nghiệm là $[5, 10]$, sai số $err = 10^{-2}$.
- b. Theo Newton – Raphson, với điểm bắt đầu $x_0 = 5$ và $err = 10^{-2}$.
- c. Theo Secant, với hai điểm ban đầu $x_0 = 5, x_1 = 10$ và sai số $err = 10^{-2}$.

Bài tập.

Bài tập 4.5: Công thức tính thể tích một phần của hình trụ:

$$V = \left[r^2 \cos^{-1} \left(\frac{r-h}{r} \right) - (r-h)\sqrt{2rh - h^2} \right] l$$

Biết: $r = 2m$; $l = 4m$; $V = 15m^3$. Xác định chiều cao h .



- a. Theo phương pháp Bisection, khoảng tìm nghiệm là $[1.75, 2]$, sai số $err = 10^{-2}$.
- b. Theo Newton – Raphson, với điểm bắt đầu $x_0 = 1.75$ và sai số $err = 10^{-2}$.
- c. Theo Secant, với hai điểm ban đầu $x_0 = 1.75, x_1 = 2$ và sai số $err = 10^{-2}$.

Bài tập.

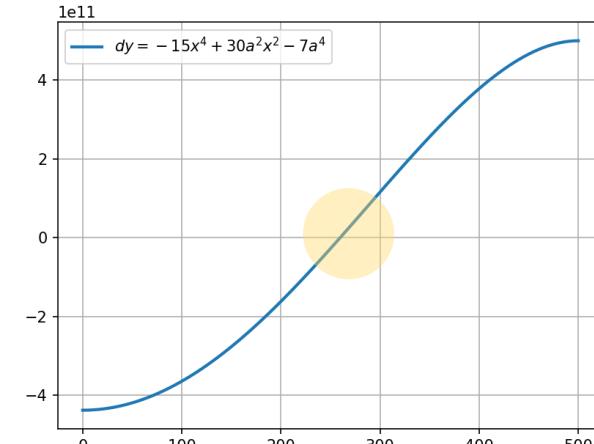
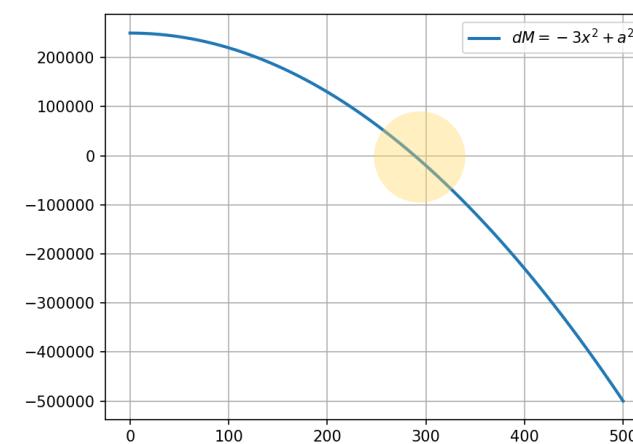
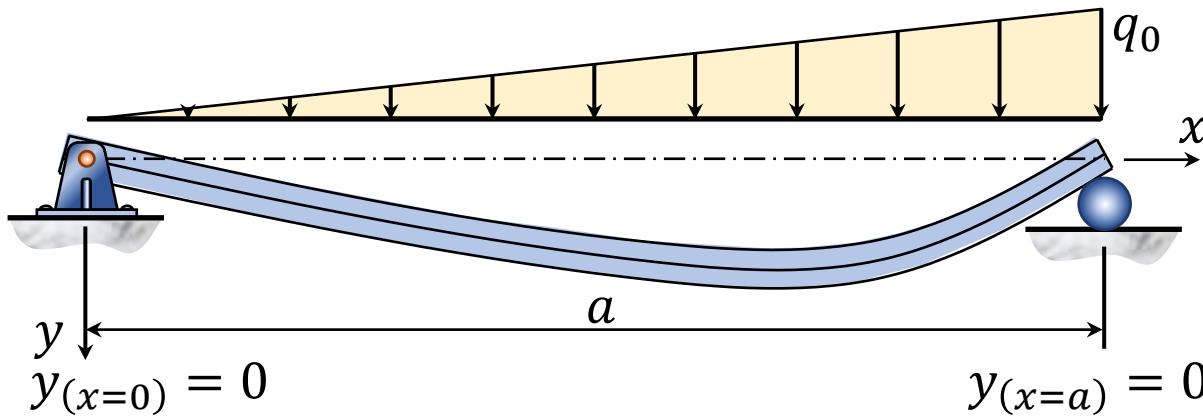
Bài tập 4.6: Dầm liên kết và chịu lực như hình vẽ. Các phương trình của dầm:

Moment uốn: $M = \frac{q_0 x}{6a} (a^2 - x^2)$.

Độ võng: $y = \frac{q_0 x}{360 E.I.a} (-3x^4 + 10a^2x^2 - 7a^4)$.

Biết: $a = 500\text{cm}$; $E = 1.2 \times 10^4 \frac{\text{kN}}{\text{cm}^2}$; $I = 4.6 \times 10^6 \text{cm}^4$; $q_0 = 2.8 \frac{\text{kN}}{\text{cm}}$.

- Xác định vị trí có moment uốn lớn nhất (giải phương trình: $dM/dx = 0$).
- Xác định vị trí có độ võng lớn nhất (giải phương trình: $dy/dx = 0$).



Bài tập.

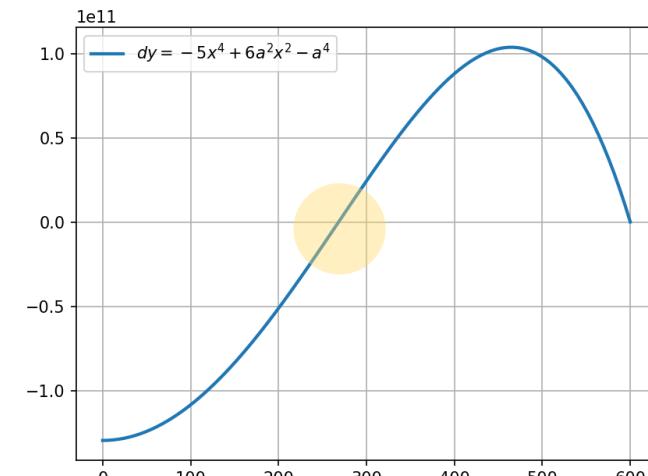
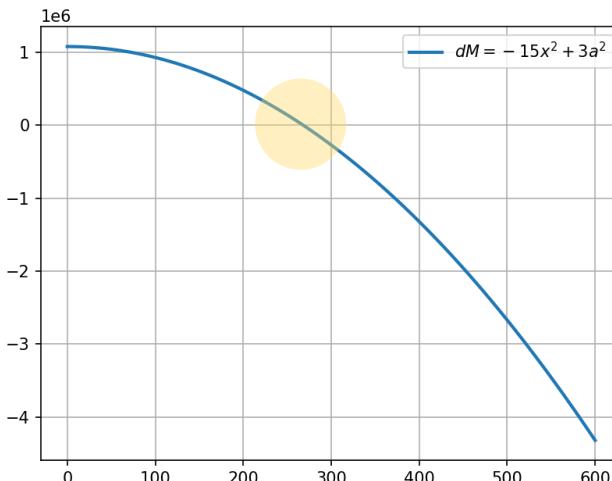
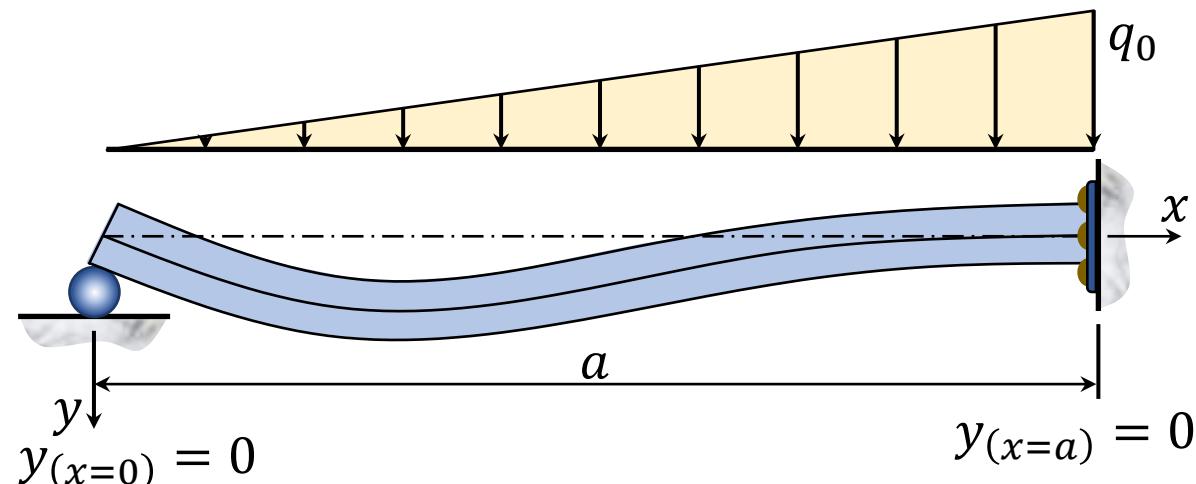
Bài tập 4.7: Dầm liên kết và chịu lực như hình vẽ. Các phương trình của dầm:

$$\text{Moment uốn: } M = \frac{q_0 x}{30a} (3a^2 - 5x^2).$$

$$\text{Độ võng: } y = \frac{q_0}{120E.I.a} (-x^5 + 2a^2x^3 - a^4x).$$

$$\text{Biết: } a = 600\text{cm}; E = 2.10^4 \frac{\text{kN}}{\text{cm}^2}; I = 5.10^4 \text{cm}^4; q_0 = 2.5 \frac{\text{kN}}{\text{cm}}.$$

- Xác định vị trí có moment uốn lớn nhất (giải phương trình: $dM/dx = 0$).
- Xác định vị trí có độ võng lớn nhất (giải phương trình: $dy/dx = 0$).



Bài tập.

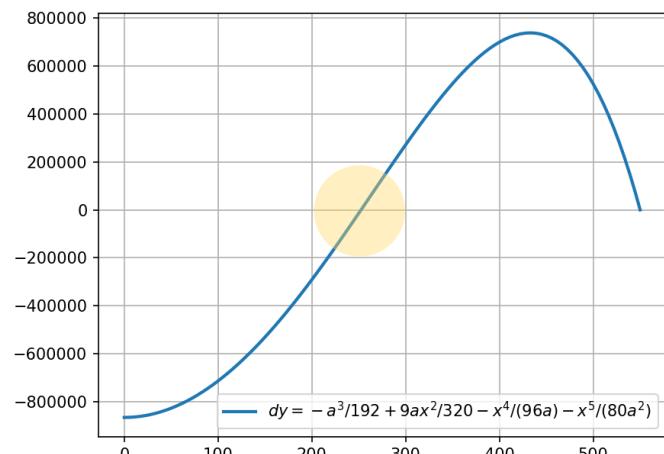
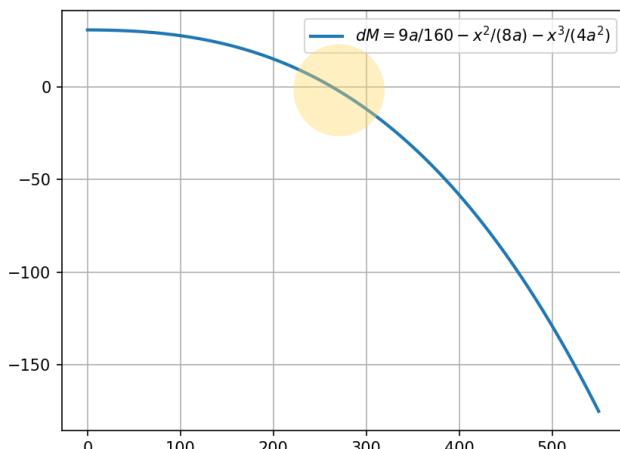
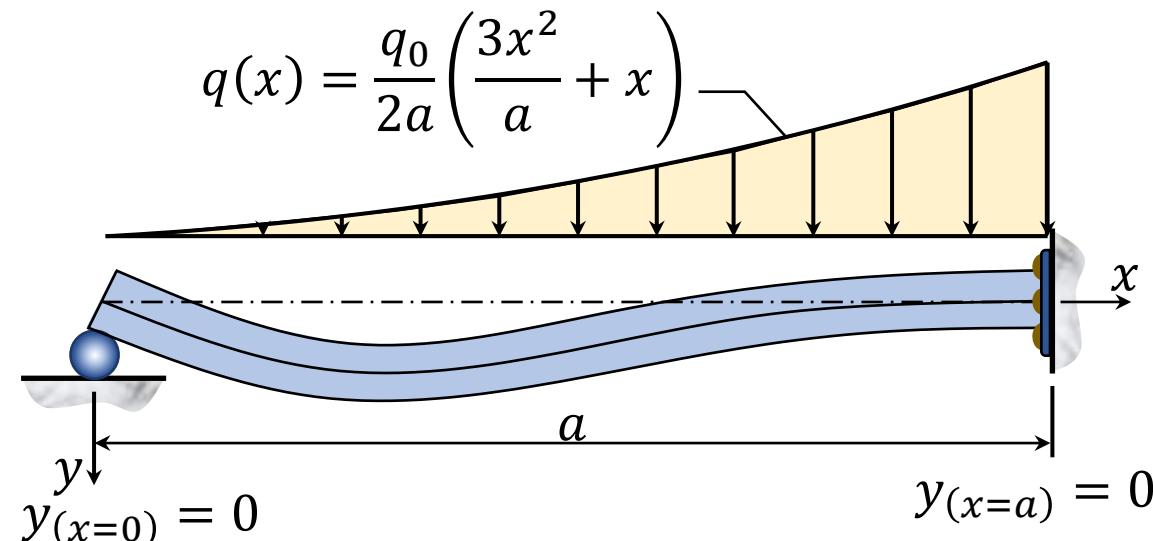
Bài tập 4.8: Dầm liên kết và chịu lực như hình vẽ. Các phương trình của dầm:

$$\text{Moment uốn: } M = \frac{q_0}{480a^2} (27a^3x - 20ax^3 - 30x^4).$$

$$\text{Độ võng: } y = \frac{q_0}{480E.I.a^2} (-2.5a^5x + 4.5a^3x^3 - ax^5 - x^6).$$

$$\text{Biết: } a = 550\text{cm}; E = 1.5 \times 10^4 \frac{\text{kN}}{\text{cm}^2}; I = 3 \times 10^4 \text{cm}^4; q_0 = 1 \frac{\text{kN}}{\text{cm}}.$$

- a. Xác định vị trí có moment uốn lớn nhất (giải phương trình: $dM/dx = 0$).
- b. Xác định vị trí có độ võng lớn nhất (giải phương trình: $dy/dx = 0$).



Bài tập.

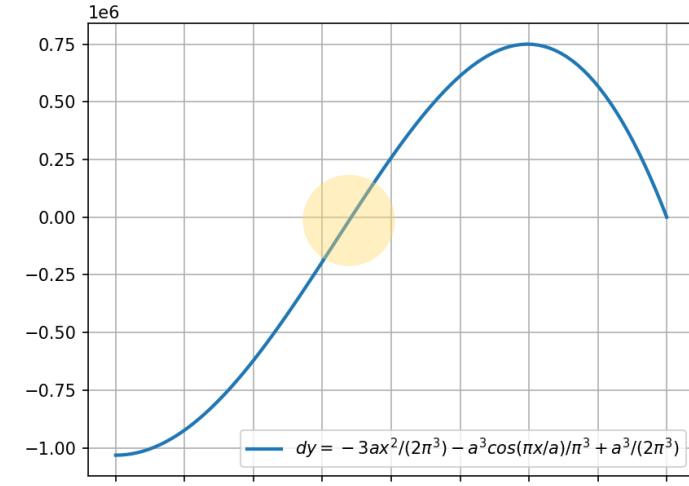
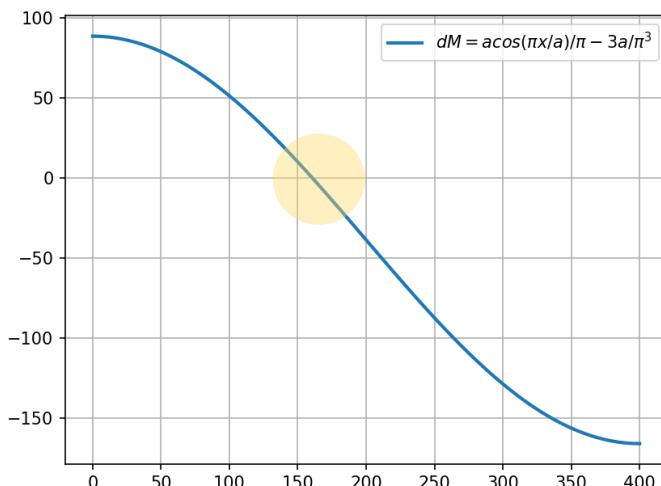
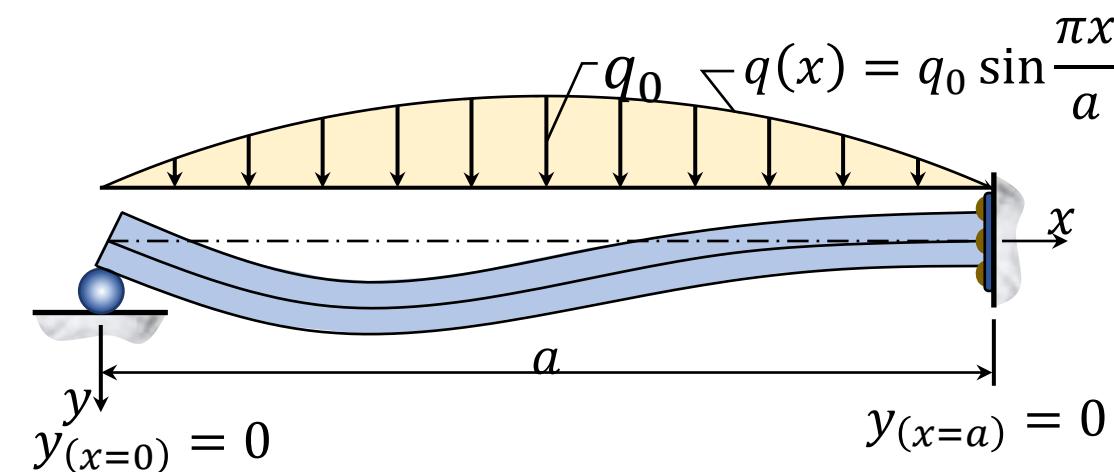
Bài tập 4.9: Dầm liên kết và chịu lực như hình vẽ. Các phương trình của dầm:

$$\text{Moment uốn: } M = \frac{q_0 a}{\pi^2} \left(\frac{3}{\pi} x - a \sin \frac{\pi x}{a} \right).$$

$$\text{Độ võng: } y = \frac{q_0 a}{2\pi^3 EI} \left(-a^2 x + \frac{2a^3}{\pi} \sin \frac{\pi x}{a} + x^3 \right).$$

$$\text{Biết: } a = 400\text{cm}; E = 9.10^3 \frac{kN}{cm^2}; I = 4.10^5 cm^4; q_0 = 1.5 \frac{kN}{cm}.$$

- a. Xác định vị trí có moment uốn lớn nhất (giải phương trình: $dM/dx = 0$).
- b. Xác định vị trí có độ võng lớn nhất (giải phương trình: $dy/dx = 0$).



Chương 5:

MA TRẬN

Nội dung của chương.

5.1. Ma trận (Matrix) và định thức.

5.1.1. Ma trận.

5.1.2. Định thức của ma trận.

5.1.3. Tính định thức.

5.1.4. Ma trận nghịch đảo.

5.2. Chuẩn của ma trận và chuẩn của véc tơ.

5.2.1. Chuẩn của ma trận.

5.2.2. Chuẩn của véc tơ.

5.1. Ma trận và định thức.

5.1.1. Ma trận.

Một bảng hình chữ nhật được tạo bởi $m \times n$ phần tử gồm m hàng và n cột được gọi là ma trận.

Ma trận A ký hiệu: $A = (a_{i,j})_{m \times n}$.

Phần tử $a_{i,j}$ ($i = 0 \dots m$; $j = 0 \dots n$) nằm ở vị trí hàng thứ i và cột thứ j .

$$A = \begin{pmatrix} 1 & -5 & 7 \\ -4 & 2 & -3 \end{pmatrix} \begin{matrix} \leftarrow \text{Hàng 0} \\ \leftarrow \text{Hàng 1} \end{matrix}$$

↑ ↑ ↑
 Cột 0 Cột 1 Cột 2

Ma trận có kích thước 2×3

Các ma trận đặc biệt.

Ma trận vuông hoặc ma trận chỉ có 1 (hàng hoặc cột) tồn tại các dạng đặc biệt sau:

$$\text{zeros} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Ma trận không.

$$\text{eye} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ma trận đơn vị.

$$B = (3 \quad -1 \quad 2 \quad 5)$$

Ma trận hàng.

$$C = \begin{pmatrix} 2 \\ 5 \\ -3 \\ 7 \end{pmatrix}$$

Ma trận cột.

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -5 \end{pmatrix}$$

Ma trận đường chéo chính.

$$\text{triu} = \begin{pmatrix} 3 & -4 & 0 & 1 \\ 0 & 1 & -2 & 3 \\ 0 & 0 & 2 & 5 \\ 0 & 0 & 0 & -6 \end{pmatrix}$$

Ma trận tam giác trên.

$$\text{tril} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 \\ 7 & 6 & -1 & 0 \\ 9 & 1 & 2 & 4 \end{pmatrix}$$

Ma trận tam giác dưới.

Các ma trận đặc biệt.

Chuyển trí (chuyển vị) của ma trận $A = (a_{i,j})_{m \times n}$ là ma trận: $A^T = (a_{j,i})_{n \times m}$

```

1| import numpy as np
2| A=np.array([[1,3,-4,5],
3|             [7,-2,6,1],
4|             [-3,8,-9,2]])
5| print('A=\n',A)
6| At=A.T; print('\nAt=\n',At)

```

$$\begin{aligned} A = \\ [[1 & 3 & -4 & 5] \\ [7 & -2 & 6 & 1] \\ [-3 & 8 & -9 & 2]] \end{aligned}$$

$$\begin{aligned} At = \\ [[1 & 7 & -3] \\ [3 & -2 & 8] \\ [-4 & 6 & -9] \\ [5 & 1 & 2]] \end{aligned}$$

Ma trận đối của ma trận $A = (a_{i,j})_{m \times n}$ là ma trận: $-A = (-a_{i,j})_{m \times n}$

```

1| import numpy as np
2| A=np.array([[1,3,-4,5],
3|             [7,-2,6,1],
4|             [-3,8,-9,2]])
5| print('A=\n',A)
6| An=-A; print('\nAn=\n',An)

```

$$\begin{aligned} A = \\ [[1 & 3 & -4 & 5] \\ [7 & -2 & 6 & 1] \\ [-3 & 8 & -9 & 2]] \end{aligned}$$

$$\begin{aligned} An = \\ [[-1 & -3 & 4 & -5] \\ [-7 & 2 & -6 & -1] \\ [3 & -8 & 9 & -2]] \end{aligned}$$

Cách khai báo để tạo ma trận trong Python.

```
1 import numpy as np
2 A=np.array([[1,2,3],[4,5,6],[7,8,9]])
3 print('A=\n',A)
4 b=np.array([1,2,3,4,5]); print('b=',b)
5 C=np.ones((3,4)); print('C=\n',C)
6 D=np.zeros((2,4)); print('D=\n',D)
7 E=np.eye(3); print('E=\n',E)
8 F=np.tri(3,3); print('F=\n',F)
9 G=np.tril([[1,2,3],
10                 [4,5,6],
11                 [7,8,9]]); print('G=\n',G)
12 K=np.triu([[1,2,3],
13                 [4,5,6],
14                 [7,8,9]]); print('K=\n',K)
15 H=A.T; print('H=\n',H)
```

A=

[1 2 3]
[4 5 6]
[7 8 9]

b= [1 2 3 4 5]

C=

[1. 1. 1. 1.]
[1. 1. 1. 1.]
[1. 1. 1. 1.]

D=

[0. 0. 0. 0.]
[0. 0. 0. 0.]

E=

[1. 0. 0.]
[0. 1. 0.]
[0. 0. 1.]

F=

[1. 0. 0.]
[1. 1. 0.]
[1. 1. 1.]

G=

[1 0 0]
[4 5 0]
[7 8 9]

K=

[1 2 3]
[0 5 6]
[0 0 9]

H=

[1 4 7]
[2 5 8]
[3 6 9]

Các tính chất của ma trận.

Với ma trận A, B, C có cùng kích thước.

$$1. A + B = B + A$$

$$2. A + (B + C) = (A + B) + C$$

$$3. \alpha(A + B) = \alpha A + \alpha B$$

$$4. (\alpha + \beta)A = \alpha A + \beta A$$

$$5. A + 0 = 0 + A = A$$

Các phép toán trên ma trận.

1. Phép cộng: $(a_{i,j})_{m \times n} + (b_{i,j})_{m \times n} = (a_{i,j} + b_{i,j})_{m \times n}$

$$\begin{pmatrix} -1 & 4 & 3 \\ 2 & 7 & -5 \end{pmatrix} + \begin{pmatrix} 2 & 8 & 4 \\ -6 & 5 & 1 \end{pmatrix} = \begin{pmatrix} -1 + 2 & 4 + 8 & 3 + 4 \\ 2 - 6 & 7 + 5 & -5 + 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 12 & 7 \\ -4 & 12 & -4 \end{pmatrix}$$

2. Phép trừ: $(a_{i,j})_{m \times n} - (b_{i,j})_{m \times n} = (a_{i,j} - b_{i,j})_{m \times n}$

$$\begin{pmatrix} -1 & 4 & 3 \\ 2 & 7 & -5 \end{pmatrix} - \begin{pmatrix} 2 & 8 & 4 \\ -6 & 5 & 1 \end{pmatrix} = \begin{pmatrix} -1 - 2 & 4 - 8 & 3 - 4 \\ 2 + 6 & 7 - 5 & -5 - 1 \end{pmatrix}$$

$$= \begin{pmatrix} -3 & -4 & -1 \\ 8 & 2 & -6 \end{pmatrix}$$

Các phép toán trên ma trận.

3. Phép nhân mảng: $(a_{i,j})_{m \times n} * (b_{i,j})_{m \times n} = (a_{i,j} * b_{i,j})_{m \times n}$

$$\begin{pmatrix} -1 & 4 & 3 \\ 2 & 7 & -5 \end{pmatrix} * \begin{pmatrix} 2 & 8 & 4 \\ -6 & 5 & 1 \end{pmatrix} = \begin{pmatrix} -1 * 2 & 4 * 8 & 3 * 4 \\ 2 * (-6) & 7 * 5 & -5 * 1 \end{pmatrix}$$

$$= \begin{pmatrix} -2 & 32 & 12 \\ -12 & 35 & -5 \end{pmatrix}$$

4. Phép chia mảng: $(a_{i,j})_{m \times n} / (b_{i,j})_{m \times n} = (a_{i,j} / b_{i,j})_{m \times n}$

$$\begin{pmatrix} -1 & 4 & 3 \\ 2 & 7 & -5 \end{pmatrix} / \begin{pmatrix} 2 & 8 & 4 \\ -6 & 5 & 1 \end{pmatrix} = \begin{pmatrix} -1/2 & 4/8 & 3/4 \\ 2/(-6) & 7/5 & (-5)/1 \end{pmatrix}$$

$$= \begin{pmatrix} -0.5 & 0.5 & 0.75 \\ -0.33 & 1.4 & -5 \end{pmatrix}$$

Các phép toán trên ma trận.

5. Phép nhân ma trận: $(a_{i,j})_{m \times \textcolor{red}{n}} \times (b_{i,j})_{\textcolor{red}{n} \times p} = \left(\sum_{k=1}^n c_{i,k} = a_{i,k} \times b_{k,j} \right)_{m \times p}$

$$\begin{pmatrix} -1 & 4 & 3 \\ 2 & 7 & -5 \end{pmatrix} \times \begin{pmatrix} 2 & -6 \\ 8 & 5 \\ 4 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} -1 * 2 + 4 * 8 + 3 * 4 & -1 * (-6) + 4 * 5 + 3 * 1 \\ 2 * 2 + 7 * 8 - 5 * 4 & 2 * (-6) + 7 * 5 - 5 * 1 \end{pmatrix} = \begin{pmatrix} 42 & 29 \\ 40 & 18 \end{pmatrix}$$

```

1| import numpy as np
2| A=np.array([[-1,4,3],
3|             [2,7,-5]])
4| print('A=\n',A)
5| B=np.array([[2,-6],
6|             [8,5],
7|             [4,1]])
8| print('B=\n',B)
9| C=A@B; print('C=\n',C)

```

```

A=
[[ -1   4   3]
 [  2   7  -5]]
B=
[[  2  -6]
 [  8   5]
 [  4   1]]
C=
[[42  29]
 [40  18]]

```

5.1. Ma

5.1.2. Định

Định thức

Định thức

Phần bù đ

Ngoài ra định thức của ma trận \mathbf{A} có thể được khai triển theo các *ma trận phần bù* \mathbf{A}_{ij} của ma trận \mathbf{A} . Trong đó ma trận phần bù \mathbf{A}_{ij} là ma trận được tạo thành bằng cách xoá đi dòng thứ i và cột thứ j của ma trận \mathbf{A} .

$$A = \begin{pmatrix} 2 \\ -3 \\ 4 \\ -6 \end{pmatrix}$$

$$\det(\mathbf{A}) = \sum_{j=1}^n (-1)^{i+j} a_{ij} \det(\mathbf{A}_{ij})$$

$$\Rightarrow M_{2,3} = (-1)^{2+3} \times \begin{pmatrix} -3 & 6 & 2 & 8 \\ 4 & -5 & 7 & 3 \\ -6 & 9 & 0 & -4 \end{pmatrix} = - \begin{pmatrix} 2 & -4 & 3 \\ 4 & -5 & 3 \\ -6 & 9 & -4 \end{pmatrix}$$

Các tính chất của định thức.

1. $|A^T| = |A|$
2. $|A * B| = |A| * |B|$
3. $|\alpha * A_{n \times n}| = \alpha^n |A|$
4. $|A^p| = |A|^p$
5. Nếu A có 1 hàng hoặc cột bằng 0 thì $|A| = 0$
6. Nếu A có 2 hàng hoặc 2 cột có quan hệ tỷ lệ nhau thì $|A| = 0$
7. Khi đổi 2 hàng (hoặc 2 cột) cho nhau thì định thức đổi dấu.
8. Định thức của ma trận tam giác bằng tích của các phần tử trên đường chéo chính.
9. $|A + B| \neq |A| + |B|$

5.1. Ma trận và định thức.

5.1.3. Tính định thức. Python: `numpy.linalg.det()`

Khai triển theo hàng 1: (bằng qui nạp)

$$A = (a_{11})_{1 \times 1} \Rightarrow |A| = a_{11}$$

$$M_{i,j} = (-1)^{i+j} \times \boxed{\begin{array}{l} \text{Định thức suy ra từ ma trận } A \\ \text{sau khi bỏ đi hàng } i, \text{cột } j \end{array}}_{n-1}$$

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}_{2 \times 2} \Rightarrow |A| = a_{11} \times M_{11} + a_{12} \times M_{12} = a_{11} \times a_{22} - a_{12} \times a_{21}$$

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}_{n \times n} \Rightarrow |A| = a_{11} \times M_{11} + a_{12} \times M_{12} + \dots + a_{1n} \times M_{1n}$$

Khai triển theo cột thứ i :

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}_{n \times n} \Rightarrow |A| = a_{1i} \times M_{1i} + a_{2i} \times M_{2i} + \dots + a_{ni} \times M_{ni}$$

Lưu ý: Chọn các hàng (hoặc cột) có nhiều phần tử bằng 0 để khai triển thì sẽ nhanh.

5.1. Ma trận và định thức.

5.1.3. Tính định thức.

Ví dụ 5.1. Tính định thức của ma trận:

$$A = \begin{pmatrix} 1 & 4 & 3 \\ 2 & 7 & -5 \\ 6 & -1 & 8 \end{pmatrix}$$

Giải: Khai triển theo hàng 1.

$$|A| = 1 \times (-1)^{1+1} \times \begin{vmatrix} 7 & -5 \\ -1 & 8 \end{vmatrix} + 4 \times (-1)^{1+2} \times \begin{vmatrix} 2 & -5 \\ 6 & 8 \end{vmatrix} + 3 \times (-1)^{1+3} \times \begin{vmatrix} 2 & 7 \\ 6 & -1 \end{vmatrix}$$

Tiếp tục khai triển theo hàng 1.

$$= 7 \times (-1)^{1+1} \times 8 - 5 \times (-1)^{1+2} \times (-1)$$

$$-4 \times [2 \times (-1)^{1+1} \times 8 - 5 \times (-1)^{1+2} \times 6]$$

$$+ 3 \times [2 \times (-1)^{1+1} \times (-1) + 7 \times (-1)^{1+2} \times 6] = -265$$

```

1 import numpy as np
2 import numpy.linalg as la
3 A=np.array([[1,4,3],
4             [2,7,-5],
5             [6,-1,8]])
6 print('det(A) =',la.det(A))

```

5.1. Ma trận và định thức.

5.1.3. Tính định thức.

Ví dụ 5.2. Tính định thức của ma trận: $A = \begin{pmatrix} 1 & 4 & 0 & -2 \\ 3 & 5 & 2 & 8 \\ 2 & 7 & 0 & -6 \\ 6 & -1 & 0 & 9 \end{pmatrix}$

Giải:

Khai triển theo cột 3: $|A| = 2 \times (-1)^{2+3} \times \begin{vmatrix} 1 & 4 & -2 \\ 2 & 7 & -6 \\ 6 & -1 & 9 \end{vmatrix}$

Khai triển theo hàng 1:

$$= -2 \times 1 \times (-1)^{1+1} \begin{vmatrix} 7 & -6 \\ -1 & 9 \end{vmatrix} - 2 \times 4 \times (-1)^{1+2} \begin{vmatrix} 2 & -6 \\ 6 & 9 \end{vmatrix} - 2 \times (-2) \times (-1)^{1+3} \begin{vmatrix} 2 & 7 \\ 6 & -1 \end{vmatrix}$$

Tiếp tục khai triển theo hàng 1:

$$\begin{aligned} &= -2 \times 7 \times (-1)^{1+1} \times 9 - 2 \times (-6) \times (-1)^{1+2} \times (-1) \\ &\quad + 8 \times 2 \times (-1)^{1+1} \times 9 + 8 \times (-6) \times (-1)^{1+2} \times 6 \\ &\quad + 4 \times 2 \times (-1)^{1+1} \times (-1) + 4 \times 7 \times (-1)^{1+2} \times 6 = 142 \end{aligned}$$

5.1. Ma trận và định thức.

5.1.4. Ma trận nghịch đảo. Python: `numpy.linalg.inv()`

Cho ma trận vuông $A_{n \times n}$, nếu tồn tại ma trận B sao cho: $A \times B = B \times A = I_{n \times n}$ thì B được gọi là ma trận nghịch đảo của ma trận A , Ký hiệu A^{-1} .

Nếu ma trận $A_{n \times n}$ có $\det(A) \neq 0$ thì sẽ tồn tại ma trận nghịch đảo của A (khả nghịch).

Gọi phần bù đại số của phần tử a_{ij} của ma trận $A_{n \times n}$ là M_{ij} :

$$M_{ij} = (-1)^{i+j} \times \left| \begin{array}{c} \text{Định thức suy ra từ ma trận } A \\ \text{sau khi bỏ đi hàng } i, \text{ cột } j \end{array} \right|_{n-1}$$

Ta có công thức tìm ma trận nghịch đảo của ma trận $A_{n \times n}$ là:

$$A^{-1} = \frac{1}{\det(A)} \times \begin{pmatrix} M_{11} & M_{12} & \dots & M_{1n} \\ M_{21} & M_{22} & \dots & M_{2n} \\ \dots & \dots & \dots & \dots \\ M_{n1} & M_{n2} & \dots & M_{nn} \end{pmatrix}^T$$

5.1.4. Ma trận nghịch đảo.

Ví dụ 5.3. Tìm ma trận nghịch đảo của ma trận: $A = \begin{pmatrix} 1 & 4 & 3 \\ 2 & 7 & -5 \\ 6 & -1 & 8 \end{pmatrix}$

```
1| import numpy as np  
2| A=np.array([[1,4,3],[2,7,-5],[6,-1,8]])  
3| print('inv(A) =',np.linalg.inv(A))
```

Giải: Ở ví dụ 4.1 có $\det(A) = -265$

$$M_{11} = (-1)^{1+1} \begin{vmatrix} 7 & -5 \\ -1 & 8 \end{vmatrix} = 51; M_{12} = (-1)^{1+2} \begin{vmatrix} 2 & -5 \\ 6 & 8 \end{vmatrix} = -46; M_{13} = (-1)^{1+3} \begin{vmatrix} 2 & 7 \\ 6 & -1 \end{vmatrix} = -44$$

$$M_{21} = (-1)^{2+1} \begin{vmatrix} 4 & 3 \\ -1 & 8 \end{vmatrix} = -35; M_{22} = (-1)^{2+2} \begin{vmatrix} 1 & 3 \\ 6 & 8 \end{vmatrix} = -10; M_{23} = (-1)^{2+3} \begin{vmatrix} 1 & 4 \\ 6 & -1 \end{vmatrix} = 25$$

$$M_{31} = (-1)^{3+1} \begin{vmatrix} 4 & 3 \\ 7 & -5 \end{vmatrix} = -41; M_{32} = (-1)^{3+2} \begin{vmatrix} 1 & 3 \\ 2 & -5 \end{vmatrix} = 11; M_{33} = (-1)^{3+3} \begin{vmatrix} 1 & 4 \\ 2 & 7 \end{vmatrix} = -1$$

$$\Rightarrow A^{-1} = \frac{1}{-265} \times \begin{pmatrix} 51 & -35 & -41 \\ -46 & -10 & 11 \\ -44 & 25 & -1 \end{pmatrix} = \begin{pmatrix} -0.1925 & 0.1321 & 0.1547 \\ 0.1736 & 0.0377 & -0.0415 \\ 0.1660 & -0.0943 & 0.0038 \end{pmatrix}$$

5.1.4. Ma trận nghịch đảo.

Ví dụ 5.4. Dùng phép biến đổi sơ cấp trên hàng tìm ma trận nghịch đảo của ma trận A .

Giải: $AI = (A|I)$ $\xrightarrow{\text{Sử dụng các phép biến đổi sơ cấp trên hàng}} (I|A^{-1})$

$$AI = \left(\begin{array}{ccc|cc} 1 & 4 & 3 & 1 & 0 & 0 \\ 2 & 7 & -5 & 0 & 1 & 0 \\ 6 & -1 & 8 & 0 & 0 & 1 \end{array} \right) \xrightarrow{\begin{array}{l} h_2 = h_2 - h_1 \times 2 \\ h_3 = h_3 - h_1 \times 6 \end{array}} \left(\begin{array}{ccc|cc} 1 & 4 & 3 & 1 & 0 & 0 \\ 0 & -1 & -11 & -2 & 1 & 0 \\ 0 & -25 & -10 & -6 & 0 & 1 \end{array} \right)$$

$$\xrightarrow{h_2 = h_2 \times (-1)} \left(\begin{array}{ccc|cc} 1 & 4 & 3 & 1 & 0 & 0 \\ 0 & 1 & 11 & 2 & -1 & 0 \\ 0 & -25 & -10 & -6 & 0 & 1 \end{array} \right) \xrightarrow{\begin{array}{l} h_1 = h_1 - h_2 \times 4 \\ h_3 = h_3 + h_2 \times 25 \end{array}} \left(\begin{array}{ccc|cc} 1 & 0 & -41 & -7 & 4 & 0 \\ 0 & 1 & 11 & 2 & -1 & 0 \\ 0 & 0 & 265 & 44 & -25 & 1 \end{array} \right)$$

$$\xrightarrow{h_3 = h_3 / 265} \left(\begin{array}{ccc|ccc} 1 & 0 & -41 & -7 & 4 & 0 \\ 0 & 1 & 11 & 2 & -1 & 0 \\ 0 & 0 & 1 & 44/265 & -25/265 & 1/265 \end{array} \right)$$

Ví dụ 5.4.

$$A = \begin{pmatrix} 1 & 4 & 3 \\ 2 & 7 & -5 \\ 6 & -1 & 8 \end{pmatrix}$$

$$AI = (A|I) \xrightarrow{\text{Sử dụng các phép biến đổi sơ cấp trên hàng}} (I|A^{-1})$$

$$\left(\begin{array}{ccc|ccc} 1 & 0 & -41 & -7 & 4 & 0 \\ 0 & 1 & 11 & 2 & -1 & 0 \\ 0 & 0 & 1 & 44/265 & -25/265 & 1/265 \end{array} \right)$$

$$\xrightarrow{\begin{array}{l} h_1=h_1+h_3\times 41 \\ h_2=h_2-h_3\times 11 \end{array}} \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -51/265 & 7/53 & 41/265 \\ 0 & 1 & 0 & 46/265 & 2/53 & -11/265 \\ 0 & 0 & 1 & 44/265 & -25/265 & 1/265 \end{array} \right)$$

$$\Rightarrow A^{-1} = \begin{pmatrix} -51/265 & 7/53 & 41/265 \\ 46/265 & 2/53 & -11/265 \\ 44/265 & -25/265 & 1/265 \end{pmatrix} = \begin{pmatrix} -0.1925 & 0.1321 & 0.1547 \\ 0.1736 & 0.0377 & -0.0415 \\ 0.1660 & -0.0943 & 0.0038 \end{pmatrix}$$

5.1.4. Ma trận nghịch đảo.

Tạo hàm để: $(A|I) \xrightarrow{\text{Phép chia, trừ trên hàng}} (I|A^{-1})$

```
1 def inv(A):
2     n=A.shape[0]; AI=np.c_[A,np.eye(n)]
3     for i in range(n):
4         for j in range(i,n):
5             if abs(AI[i,i])<abs(AI[j,j]):
6                 temp=AI[i].copy()
7                 AI[i]=AI[j].copy()
8                 AI[j]=temp.copy()
9
10    for i in range(n):
11        ss=AI[i,i].copy()
12        for j in range(2*n):
13            AI[i,j] /= ss
14
15    for j in range(n):
16        if i!=j:
17            temp=AI[j,i].copy()
18            for k in range(2*n):
19                AI[j,k] -= temp*AI[i,k]
20
21    return AI[:,n:2*n]
```

Ghép 2 ma trận A và I

Sắp xếp phần tử trụ $AI[i, i]$ từ lớn đến nhỏ → Giảm sai số làm tròn

Chia các phần tử trên hàng i cho $AI[i, i] \rightarrow AI[i, i] = 1$

Khử các phần tử trên cột j khi $j \neq i \rightarrow AI[i, j] = 0$

5.2. Chuẩn của ma trận và chuẩn của véc tơ.

5.2.1. Chuẩn của ma trận.

Chuẩn của ma trận $A = (a_{ij})$ là một số thực được ký hiệu $\|A\|$ thỏa các điều kiện:

- ✓ $\|A\| \geq 0$
- ✓ $\|\alpha \times A\| = |\alpha| \times \|A\|$
- ✓ $\|A + B\| = \|A\| + \|B\|$
- ✓ $\|A \times B\| = \|A\| \times \|B\|$

Thường phổ biến 3 chuẩn của ma trận sau: $\|A\|_1 = \max_j \sum_i |a_{ij}|$ (Chuẩn cột)

$$\|A\|_2 = \sqrt{\sum_{i,j} |a_{ij}|^2}$$
 Chuẩn Euclide)

$$\|A\|_\infty = \max_i \sum_j |a_{ij}|$$
 (Chuẩn hàng)

5.2. Chuẩn của ma trận và chuẩn của véc tơ.

5.2.1. Chuẩn của véc tơ.

Chuẩn của véc tơ (ma trận 1 hàng hoặc 1 cột) $X = (x_1 \quad x_2 \quad \dots \quad x_n)$ là:

$$\|X\|_1 = |x_1| + |x_2| + \dots + |x_n| = \sum_{i=1}^n |x_i|$$

$$\|X\|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \dots + |x_n|^2} = \sqrt{\sum_{i=1}^n |x_i|^2}$$

$$\|X\|_\infty = \max_i(|x_i|)$$

5.2. Chuẩn của ma trận và chuẩn của véc tơ.

Ví dụ 5.5. Tính 3 chuẩn phổ biến của ma trận:

Giải.

$$\|A\|_1 = \max_j \sum_i |a_{ij}| = \max(2 + 1 + 5, \quad |-4| + 7 + |-5|, \quad 0 + |-6| + 3) = 16$$

$$\|A\|_2 = \sqrt{\sum_{i,j} |a_{ij}|^2} = \sqrt{\lambda_{max}(A^T \times A)}$$

$$\|A\|_\infty = \max_i \sum_j |a_{ij}| = \max(2 + |-4| + 0, \quad 1 + 7 + |-6|, \quad 5 + |-5| + 3) = 14$$

$$A = \begin{pmatrix} 2 & -4 & 0 \\ 1 & 7 & -6 \\ 5 & -5 & 3 \end{pmatrix}$$

Với λ_{max} là giá trị riêng lớn nhất của ma trận $(A^T \times A)$

```
1 import numpy as np
2 import numpy.linalg as LA
3 A=np.array([[2,-4,0],[1,7,-6],[5,-5,3]])
4 print('norm(A,1) =',LA.norm(A,1))
5 print('norm(A,2) =',LA.norm(A,2))
6 print('norm(A,inf) =',LA.norm(A,np.inf))
```

```
norm(A,1) = 16.0
norm(A,2) = 11.640109146773208
norm(A,inf) = 14.0
```

5.2. Chuẩn của ma trận và chuẩn của véc tơ.

Ví dụ 5.6. Tính 3 chuẩn phổ biến của véc tơ: $X = (1 \quad -5 \quad 7 \quad 0 \quad -8)$

Giải.

$$\|X\|_1 = \sum_{i=1}^n |x_i| = 1 + |-5| + 7 + 0 + |-8| = 21$$

$$\|X\|_2 = \sqrt{1^2 + |-5|^2 + 7^2 + 0^2 + |-8|^2} = 11.789826122551595$$

$$\|X\|_\infty = \max(1, |-5|, 7, 0, |-8|) = 8$$

```
1 import numpy as np
2 import numpy.linalg as LA
3 X=np.array([1,-5,7,0,-8])
4 print('norm(X,1) =',LA.norm(X,1))
5 print('norm(X,2) =',LA.norm(X,2))
6 print('norm(X,inf) =',LA.norm(X,np.inf))
```

```
norm(X,1) = 21.0
norm(X,2) = 11.789826122551595
norm(X,inf) = 8.0
```

Bài tập.

Bài tập 5.1. Tính trực tiếp phép nhân 2 ma trận sau:

$$A = \begin{pmatrix} -3 & 2 & 7 & 6 \\ 1 & 0 & -2 & 8 \\ -5 & 4 & 9 & -1 \end{pmatrix} \times \begin{pmatrix} 5 & 3 \\ 4 & -2 \\ 0 & 1 \\ -7 & 6 \end{pmatrix} = ?$$

Bài tập 5.2. Tính trực tiếp định thức của ma trận: $A = \begin{pmatrix} 1 & 4 & 0 & 8 \\ 3 & 2 & 2 & 7 \\ 0 & 0 & 0 & -4 \\ 0 & 7 & -6 & 5 \end{pmatrix}$

Bài tập 5.3. Tìm trực tiếp ma trận nghịch đảo của ma trận:

$$A = \begin{pmatrix} 7 & 1 & 5 & -2 \\ 8 & 0 & 2 & 7 \\ -3 & 0 & 0 & -4 \\ 4 & 0 & -1 & 3 \end{pmatrix}$$

Bài tập.

Bài tập 5.4. Dùng phép biến đổi sơ cấp trên hàng để biến ma trận $A|I$ thành $I|A^{-1}$, từ đó xác định được ma trận nghịch đảo của A .

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

Bài tập 5.5. Tính trực tiếp chuẩn $\|A\|_1$ của ma trận: $A = \begin{pmatrix} -9 & -4 & 7 \\ 1 & 7 & -8 \\ 5 & -6 & 6 \end{pmatrix}$

Bài tập 5.6. Tính trực tiếp chuẩn $\|X\|_2$ của véc tơ: $X = \begin{pmatrix} -9 \\ -7 \\ 12 \\ 4 \end{pmatrix}$

Kiểm tra.

Kiểm tra. Dùng phép biến đổi sơ cấp trên hàng để biến ma trận $A|I$ thành $I|A^{-1}$, từ đó xác định được ma trận nghịch đảo của A . **s** là số cuối của mssv

$$A = \begin{pmatrix} 3+s & -9+s & 4+s \\ 1+s & -2+s & 7+s \\ -5+s & 8+s & 2+s \end{pmatrix}$$

Chương 6:

GIẢI HỆ PHƯƠNG TRÌNH TUYẾN TÍNH VÀ PHI TUYẾN

Nội dung của chương.

6.1. Hệ phương trình đại số tuyến tính.

6.1.1. Phương pháp thông qua ma trận nghịch đảo.

6.1.2. Phương pháp Cramer.

6.1.3. Phương pháp Gauss.

6.1.4. Phương pháp lặp đơn.

6.2. Hệ phương trình phi tuyến.

6.1. Hệ phương trình đại số tuyến tính.

Hệ phương trình đại số tuyến tính có dạng:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{12}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

Viết dưới dạng ma trận:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}. \text{ Hoặc: } Ax = b$$

Cần tìm nghiệm: $x = (x_1, x_2, \dots, x_n)$. Nếu $\det(A) \neq 0$ thì hệ có nghiệm duy nhất.

6.1.1. Phương pháp thông qua ma trận nghịch đảo.

Tìm nghiệm thông qua 2 phép toán nghịch đảo và nhân ma trận:

$$x = A^{-1} @ b$$

```
import numpy as np
x = np.linalg.inv(A) @ b
```

6.1. Hệ phương trình đại số tuyến tính.

6.1.2. Phương pháp Cramer.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}. \text{ Hay: } A\mathbf{x} = \mathbf{b}$$

Nghiệm theo phương pháp Cramer:

$$x_i = \frac{\det(A_i)}{\det(A)}$$

Trong đó: A_i là ma trận có được từ ma trận A bằng cách thay cột thứ i bằng cột b .

$$A_i = \begin{pmatrix} a_{11} & a_{12} & \dots & \boxed{b_1} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & \boxed{b_2} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & \boxed{b_n} & \dots & a_{nn} \end{pmatrix}$$

Cột thứ: 1 2 ... ***i*** ... n

6.1.2. Phương pháp Cramer.

Ví dụ 6.1. Dùng phương pháp Cramer để giải hệ:

$$\begin{cases} 2x_1 + x_2 - x_3 = 3 \\ x_1 - x_2 + 2x_3 = 2 \\ 3x_1 - 2x_2 + x_3 = -1 \end{cases}$$

Giải.

$$|A| = \begin{vmatrix} 2 & 1 & -1 \\ 1 & -1 & 2 \\ 3 & -2 & 1 \end{vmatrix} = 2 \begin{vmatrix} -1 & 2 \\ -2 & 1 \end{vmatrix} - 1 \begin{vmatrix} 1 & -1 \\ -2 & 1 \end{vmatrix} + 3 \begin{vmatrix} 1 & -1 \\ -1 & 2 \end{vmatrix} = 2(-1 + 4) - (1 - 2) + 3(2 - 1) = 10$$

$$|A_1| = \begin{vmatrix} 3 & 1 & -1 \\ 2 & -1 & 2 \\ -1 & -2 & 1 \end{vmatrix} = 3 \begin{vmatrix} -1 & 2 \\ -2 & 1 \end{vmatrix} - 2 \begin{vmatrix} 1 & -1 \\ -2 & 1 \end{vmatrix} - 1 \begin{vmatrix} 1 & -1 \\ -1 & 2 \end{vmatrix} = 3(-1 + 4) - 2(1 - 2) - (2 - 1) = 10$$

$$|A_2| = \begin{vmatrix} 2 & 3 & -1 \\ 1 & 2 & 2 \\ 3 & -1 & 1 \end{vmatrix} = 2 \begin{vmatrix} 2 & 2 \\ -1 & 1 \end{vmatrix} - 1 \begin{vmatrix} 3 & -1 \\ -1 & 1 \end{vmatrix} + 3 \begin{vmatrix} 3 & -1 \\ 2 & 2 \end{vmatrix} = 2(2 + 2) - (3 - 1) + 3(6 + 2) = 30$$

$$|A_3| = \begin{vmatrix} 2 & 1 & 3 \\ 1 & -1 & 2 \\ 3 & -2 & -1 \end{vmatrix} = 2 \begin{vmatrix} -1 & 2 \\ -2 & -1 \end{vmatrix} - 1 \begin{vmatrix} 1 & 3 \\ -2 & -1 \end{vmatrix} + 3 \begin{vmatrix} 1 & 3 \\ -1 & 2 \end{vmatrix} = 2(1 + 4) - (-1 + 6) + 3(2 + 3) = 20$$

$$\Rightarrow x_1 = 10/10 = 1; \quad x_2 = 30/10 = 3; \quad x_3 = 20/10 = 2$$

```
1 import numpy as np
2 A=np.array([[2,1,-1],[1,-1,2],[3,-2,1]])
3 b=np.array([3,2,-1])
4 x1,x2,x3=np.linalg.solve(A,b)
5 x1,x2,x3=round(x1,4),round(x2,4),round(x3,4)
6 print('x1 =',x1,'; x2 =',x2,'; x3 =',x3)
```

6.1. Hệ phương trình đại số tuyến tính.

6.1.3. Phương pháp Gauss. (Gồm hai quá trình thuận và ngược)

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}.$$

Quá trình thuận: Biến đổi ma trận A về thành ma trận tam giác trên.

$$A \rightarrow A^{(n)} = \begin{pmatrix} 1 & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & 1 & \dots & a_{2n}^{(2)} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}; b^{(n)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \dots \\ b_n^{(n)} \end{pmatrix}.$$

$a_{ij}^{(k)}$ là phần tử hàng i cột j sau bước biến đổi thứ (k) của ma trận A

$b_i^{(k)}$ là phần tử hàng i sau bước biến đổi thứ (k) của ma trận b

6.1.3. Phương pháp Gauss.

Kết quả dẫn đến hệ phương trình có dạng:

$$\left\{ \begin{array}{l} x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)} \\ x_2 + \cdots + a_{1n}^{(2)}x_n = b_2^{(2)} \\ \cdots \quad \cdots \quad \cdots \\ x_n = b_b^{(n)} \end{array} \right.$$

Quá trình ngược: Lần lượt tính nghiệm $x_n; x_{n-1}; \dots; x_1$.

$$x_n = b_n^{(n)}$$

$$x_{n-1} = b_{n-1}^{(n-1)} - a_{n-1}^{(n-1)} \times x_n$$

...

$$x_i = b_i^{(i)} - \sum_{k=i+1}^n a_{ik}^{(i)} \times x_k$$

...

$$x_1 = b_1^{(1)} - \sum_{k=2}^n a_{1k}^{(1)} \times x_k$$

6.1.3. Phương pháp Gauss.

Ví dụ 6.2. Dùng phương pháp Gauss để giải hệ:

$$\begin{cases} 2x_1 + x_2 - x_3 = 3 \\ x_1 - x_2 + 2x_3 = 2 \\ 3x_1 - 2x_2 + x_3 = -1 \end{cases}$$

Giải: Quá trình thuận.

Viết dưới dạng ma trận:

$$\left(\begin{array}{ccc|c} a_{11} = 2 & a_{12} = 1 & a_{13} = -1 & b_1 = 3 \\ a_{21} = 1 & a_{22} = -1 & a_{23} = 2 & b_2 = 2 \\ a_{31} = 3 & a_{32} = -2 & a_{33} = 1 & b_3 = -1 \end{array} \right)$$

Do $a_{11} \neq 0$ (Nếu $a_{11} = 0$ thì đổi hàng 1 cho hàng 2), chia hàng 1 cho $a_{11} = 2$:

$$\Rightarrow \left(\begin{array}{ccc|c} 1 & a_{12}^{(1)} = 1/2 & a_{13}^{(1)} = -1/2 & b_1^{(1)} = 3/2 \\ a_{21} = 1 & a_{22} = -1 & a_{23} = 2 & b_2 = 2 \\ a_{31} = 3 & a_{32} = -2 & a_{33} = 1 & b_3 = -1 \end{array} \right)$$

Khử a_{21}, a_{31} : Hàng 2 = Hàng 2 – Hàng 1 $\times a_{21}$ và Hàng 3 = Hàng 3 – Hàng 1 $\times a_{31}$:

$$\Rightarrow \left(\begin{array}{ccc|c} 1 & a_{12}^{(1)} = 1/2 & a_{13}^{(1)} = -1/2 & b_1^{(1)} = 3/2 \\ 0 & a_{22}^{(1)} = -1 - 1/2 \times 1 & a_{23}^{(1)} = 2 + 1/2 \times 1 & b_2^{(1)} = 2 - 3/2 \times 1 \\ 0 & a_{32}^{(1)} = -2 - 1/2 \times 3 & a_{33}^{(1)} = 1 + 1/2 \times 3 & b_3^{(1)} = -1 - 3/2 \times 3 \end{array} \right)$$

6.1.3. Phương pháp Gauss.

$$\left(\begin{array}{ccc|c} 1 & a_{12}^{(1)} = 1/2 & a_{13}^{(1)} = -1/2 & b_1^{(1)} = 3/2 \\ 0 & a_{22}^{(1)} = -3/2 & a_{23}^{(1)} = 5/2 & b_2^{(1)} = 1/2 \\ 0 & a_{32}^{(1)} = -7/2 & a_{33}^{(1)} = 5/2 & b_3^{(1)} = -11/2 \end{array} \right)$$

Chia hàng 2 cho $a_{22} = -3/2$:

$$\Rightarrow \left(\begin{array}{ccc|c} 1 & a_{12}^{(1)} = 1/2 & a_{13}^{(1)} = -1/2 & b_1^{(1)} = 3/2 \\ 0 & 1 & a_{23}^{(2)} = -5/3 & b_2^{(2)} = -1/3 \\ 0 & a_{32}^{(1)} = -7/2 & a_{33}^{(1)} = 5/2 & b_3^{(1)} = -11/2 \end{array} \right)$$

Khử $a_{32}^{(1)}$: Hàng 3 = Hàng 3 - Hàng 2 $\times a_{32}^{(1)}$:

$$\Rightarrow \left(\begin{array}{ccc|c} 1 & a_{12}^{(1)} = 1/2 & a_{13}^{(1)} = -1/2 & b_1^{(1)} = 3/2 \\ 0 & 1 & a_{23}^{(2)} = -5/3 & b_2^{(2)} = -1/3 \\ 0 & 0 & a_{33}^{(2)} = 5/2 + 5/3 \times (-7/2) & b_3^{(2)} = -11/2 + 1/3 \times (-7/2) \end{array} \right)$$

6.1.3. Phương pháp Gauss.

$$\left(\begin{array}{ccc|c} 1 & a_{12}^{(1)} = 1/2 & a_{13}^{(1)} = -1/2 \\ 0 & 1 & a_{23}^{(2)} = -5/3 \\ 0 & 0 & a_{33}^{(2)} = -10/3 \end{array} \right) | \left(\begin{array}{c} b_1^{(1)} = 3/2 \\ b_2^{(1)} = -1/3 \\ b_3^{(2)} = -20/3 \end{array} \right)$$

Chia hàng 3 cho $a_{33} = -10/3$:

$$\Rightarrow \left(\begin{array}{ccc|c} 1 & a_{12}^{(1)} = 1/2 & a_{13}^{(1)} = -1/2 \\ 0 & 1 & a_{23}^{(2)} = -5/3 \\ 0 & 0 & 1 \end{array} \right) | \left(\begin{array}{c} b_1^{(1)} = 3/2 \\ b_2^{(1)} = -1/3 \\ b_3^{(2)} = 2 \end{array} \right) \Rightarrow \left\{ \begin{array}{l} x_1 + \frac{1}{2}x_2 - \frac{1}{2}x_3 = \frac{3}{2} \\ x_2 - \frac{5}{3}x_3 = -\frac{1}{3} \\ x_3 = 2 \end{array} \right.$$

Quá trình ngược:

$$x_3 = 2 \Rightarrow x_2 = -\frac{1}{3} + \frac{5}{3} \times 2 = 3 \Rightarrow x_1 = \frac{3}{2} + \frac{1}{2} \times 2 - \frac{1}{2} \times 3 = 1$$

6.1.3. Phương pháp Gauss.

Ví dụ 6.3. Dùng biến đổi sơ cấp trên hàng: Đổi hàng; Nhân hoặc chia hàng cho 1 số; Cộng hoặc trừ 2 hàng cho nhau để tạo thành hàng mới, áp dụng giải hệ cho bên cạnh:

$$\begin{cases} 2x_1 + x_2 - 2x_3 = 1 \\ x_1 - 2x_2 + 2x_3 = 0 \\ 2x_1 - 2x_2 + x_3 = -1 \end{cases}$$

Giải:

$$\begin{array}{l}
 Ab = \left(\begin{array}{ccc|c} 2 & 1 & -2 & 1 \\ 1 & -2 & 2 & 0 \\ 2 & -2 & 1 & -1 \end{array} \right) \xrightarrow{h_1 \leftrightarrow h_2} \left(\begin{array}{ccc|c} 1 & -2 & 2 & 0 \\ 2 & 1 & -2 & 1 \\ 2 & -2 & 1 & -1 \end{array} \right) \xrightarrow{\begin{array}{l} h_2 = h_2 - h_1 \times 2 \\ h_3 = h_3 - h_1 \times 2 \end{array}} \left(\begin{array}{ccc|c} 1 & -2 & 2 & 0 \\ 0 & 5 & -6 & 1 \\ 0 & 2 & -3 & -1 \end{array} \right) \\
 \xrightarrow{h_2 = h_2 / 5} \left(\begin{array}{ccc|c} 1 & -2 & 2 & 0 \\ 0 & 1 & -6/5 & 1/5 \\ 0 & 2 & -3 & -1 \end{array} \right) \xrightarrow{\begin{array}{l} h_1 = h_1 + h_2 \times 2 \\ h_3 = h_3 - h_2 \times 2 \end{array}} \left(\begin{array}{ccc|c} 1 & 0 & -2/5 & 2/5 \\ 0 & 1 & -6/5 & 1/5 \\ 0 & 0 & -3/5 & -7/5 \end{array} \right) \\
 \xrightarrow{h_3 = h_3 \times (-5/3)} \left(\begin{array}{ccc|c} 1 & 0 & -2/5 & 2/5 \\ 0 & 1 & -6/5 & 1/5 \\ 0 & 0 & 1 & 7/3 \end{array} \right) \xrightarrow{\begin{array}{l} h_1 = h_1 + h_3 \times 2/5 \\ h_2 = h_2 + h_3 \times 6/5 \end{array}} \left(\begin{array}{ccc|c} 1 & 0 & 0 & 4/3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 7/3 \end{array} \right)
 \end{array}$$

Vậy, nghiệm của hệ là: $x_1 = 4/3$; $x_2 = 3$; $x_3 = 7/3$

6.1.3. Phương pháp Gauss.

Tạo hàm để: $(A|b) \xrightarrow{\text{Phép chia, trừ trên hàng}} (I|X)$

```
1 def solve(A, b):
2     n=A.shape[0]; Ab=np.c_[A, b] • Ghép 2 ma trận A và b
3     for i in range(n):
4         for j in range(i, n):
5             if abs(Ab[i, i])<abs(Ab[j, j]):
6                 temp=Ab[i].copy()
7                 Ab[i]=Ab[j].copy()
8                 Ab[j]=temp.copy()
9
10    for i in range(n):
11        ss=Ab[i, i].copy()
12        for j in range(n+1):
13            Ab[i, j] /= ss • Sắp xếp phần tử trụ  $Ab[i, i]$ 
14            từ lớn đến nhỏ → Giảm sai
15            số làm tròn
16            for j in range(n):
17                if i!=j:
18                    temp=Ab[j, i].copy()
19                    for k in range(n+1):
20                        Ab[j, k] -= temp*Ab[i, k] • Chia các phần tử trên hàng
21                        i cho  $Ab[i, i] \rightarrow Ab[i, i] = 1$ 
22
23    return Ab[:, :-1] • Khử các phần tử trên cột j
24                                khi  $j \neq i \rightarrow Ab[i, j] = 0$ 
```

6.1. Hệ phương trình đại số tuyến tính.

6.1.4. Phương pháp lặp đơn.

Hệ phương trình có dạng: $A \times x = b$ hay:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

Biến đổi tương đương hệ đã cho về dạng: $x = C \times x + d$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}; \quad C \times x = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}; \quad d = \begin{pmatrix} d_1 \\ d_2 \\ \dots \\ d_n \end{pmatrix}.$$

Cho trước bộ nghiệm ban đầu: $x^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \dots \\ x_n^{(0)} \end{pmatrix}$.

Nếu đây: $x^{(n+1)} = C \times x^{(n)} + d$ hội tụ về $x^{(*)}$ khi $n \rightarrow \infty$ thì $x^{(*)}$ là nghiệm của hệ.

6.1.4. Phương pháp lặp đơn.

✓ Điều kiện hội tụ:

Nếu $\|C\|_p < 1$ thì dãy lặp sẽ hội tụ về nghiệm. $\|C\|_p$ là chuẩn tùy chọn của C .

✓ Sai số:

Gọi x^* là nghiệm đúng của hệ, ta có sai số sau:

$$\|x^k - x^*\|_p \leq \frac{\|C\|_p}{1 - \|C\|_p} \times \|x^k - x^{k-1}\|_p$$

Hoặc: $\|x^k - x^*\|_p \leq \frac{(\|C\|_p)^k}{1 - \|C\|_p} \times \|x^1 - x^0\|_p$

6.1.4. Phương pháp lặp đơn.

Ví dụ 6.4. Dùng phương pháp giải lặp, tìm nghiệm gần đúng của hệ phương trình:

$$\begin{cases} 3x_1 + 0.45x_2 - 0.09x_3 = 6 \\ 0.08x_1 + 4x_2 - 0.6x_3 = 10 \\ 0.2x_1 - 0.15x_2 + 5x_3 = 15 \end{cases}$$

Giải:

$$\Leftrightarrow \begin{cases} x_1 = -0.15x_2 + 0.03x_3 + 2 \\ x_2 = -0.02x_1 + 0.15x_3 + 2.5 \\ x_3 = -0.04x_1 + 0.03x_2 + 3 \end{cases} \Leftrightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 & -0.15 & 0.03 \\ -0.02 & 0 & 0.15 \\ -0.04 & 0.03 & 0 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 2 \\ 2.5 \\ 3 \end{pmatrix}$$

$$\Leftrightarrow x = C \times x + d$$

$\|C\|_1 = 0.18 < 1$ do đó dãy lặp $x^{(n+1)} = C \times x^{(n)} + d$ sẽ hội tụ về nghiệm bài toán.

$$\text{Cho } x^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{pmatrix} = \begin{pmatrix} 0 & -0.15 & 0.03 \\ -0.02 & 0 & 0.15 \\ -0.04 & 0.03 & 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 2 \\ 2.5 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2.5 \\ 3 \end{pmatrix}$$

Ví dụ 6.4.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 & -0.15 & 0.03 \\ -0.02 & 0 & 0.15 \\ -0.04 & 0.03 & 0 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 2 \\ 2.5 \\ 3 \end{pmatrix}. \quad \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{pmatrix} = \begin{pmatrix} 2 \\ 2.5 \\ 3 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{pmatrix} = \begin{pmatrix} 0 & -0.15 & 0.03 \\ -0.02 & 0 & 0.15 \\ -0.04 & 0.03 & 0 \end{pmatrix} \times \begin{pmatrix} 2 \\ 2.5 \\ 3 \end{pmatrix} + \begin{pmatrix} 2 \\ 2.5 \\ 3 \end{pmatrix} = \begin{pmatrix} 1.72 \\ 2.91 \\ 3 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} x_1^{(3)} \\ x_2^{(3)} \\ x_3^{(3)} \end{pmatrix} = \begin{pmatrix} 0 & -0.15 & 0.03 \\ -0.02 & 0 & 0.15 \\ -0.04 & 0.03 & 0 \end{pmatrix} \times \begin{pmatrix} 1.72 \\ 2.91 \\ 3 \end{pmatrix} + \begin{pmatrix} 2 \\ 2.5 \\ 3 \end{pmatrix} = \begin{pmatrix} 1.65 \\ 2.91 \\ 3.02 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} x_1^{(4)} \\ x_2^{(4)} \\ x_3^{(4)} \end{pmatrix} = \begin{pmatrix} 0 & -0.15 & 0.03 \\ -0.02 & 0 & 0.15 \\ -0.04 & 0.03 & 0 \end{pmatrix} \times \begin{pmatrix} 1.65 \\ 2.91 \\ 3.02 \end{pmatrix} + \begin{pmatrix} 2 \\ 2.5 \\ 3 \end{pmatrix} = \begin{pmatrix} 1.65 \\ 2.92 \\ 3.02 \end{pmatrix}$$

Ví dụ 6.4.

$$C = \begin{pmatrix} 0 & -0.15 & 0.03 \\ -0.02 & 0 & 0.15 \\ -0.04 & 0.03 & 0 \end{pmatrix}; \begin{pmatrix} x_1^{(3)} \\ x_2^{(3)} \\ x_3^{(3)} \end{pmatrix} = \begin{pmatrix} 1.65 \\ 2.91 \\ 3.02 \end{pmatrix}; \begin{pmatrix} x_1^{(4)} \\ x_2^{(4)} \\ x_3^{(4)} \end{pmatrix} = \begin{pmatrix} 1.65 \\ 2.92 \\ 3.02 \end{pmatrix}$$

Khi không có nghiệm chính xác:

$$\|C\|_1 = 18; \|x^{(4)} - x^{(3)}\|_1 = \left\| \begin{pmatrix} 1.65 \\ 2.92 \\ 3.02 \end{pmatrix} - \begin{pmatrix} 1.65 \\ 2.91 \\ 3.02 \end{pmatrix} \right\|_1 = 0.0074$$

$$error_{max} = \frac{\|C\|_1}{1 - \|C\|_1} \times \|x^{(4)} - x^{(3)}\|_1 = \frac{0.18}{1 - 0.18} \times 0.01 = 0.0016$$

Khi có nghiệm chính xác: $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1.65261911 \\ 2.92017268 \\ 3.02150042 \end{pmatrix}$ (Nếu có sai số là do làm tròn)

$$error = \|x^{(4)} - x\|_1 = \left\| \begin{pmatrix} 1.65 \\ 2.92 \\ 3.02 \end{pmatrix} - \begin{pmatrix} 1.65261911 \\ 2.92017268 \\ 3.02150042 \end{pmatrix} \right\|_1 = 0.0013$$

Ví dụ 6.4.

```

1| import numpy as np
2| C=np.array([[0,-0.15,0.03],
3|             [-0.02,0,0.15],
4|             [-0.04,0.03,0]])
5| d=np.array([2,2.5,3])
6| x0=np.array([0,0,0]); print('x0 =',x0)
7| x1=C@x0+d; print('x1 =',x1)
8| x2=C@x1+d; print('x2 =',x2)
9| x3=C@x2+d; print('x3 =',x3)
10| x4=C@x3+d; print('x4 =',x4)
11| norm4_3=sum(abs(x4-x3))
12| normC=np.linalg.norm(C,1)
13| Norm_max=normC*norm4_3/(1-normC)
14| print('norm_max =',Norm_max)
15| A=np.array([[3,0.45,-0.09],
16|             [0.08,4,-0.6],
17|             [0.2,-0.15,5]])
18| b=np.array([6,10,15])
19| x=np.linalg.solve(A,b); print('x =',x)
20| print('norm(x4-x) =',sum(abs(x4-x)))

```

x0 = [0 0 0]
x1 = [2. 2.5 3.]
x2 = [1.715 2.91 2.995]
x3 = [1.65335 2.91495 3.0187]
x4 = [1.6533185 2.919738 3.0213145]
norm_max = 0.0016318536585366208
x = [1.65261911 2.92017268 3.02150042]
norm(x4-x) = 0.0013199857254804215

6.2. Hệ phương trình phi tuyến.

Hệ phương trình:

$$\begin{pmatrix} a_{11}(x_i) & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22}(x_k) & \dots & a_{2n}(x_1) \\ \dots & \dots & \dots & \dots \\ a_{n1}(x_2) & a_{n2} & \dots & a_{nn} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n(x_i) \end{pmatrix}.$$

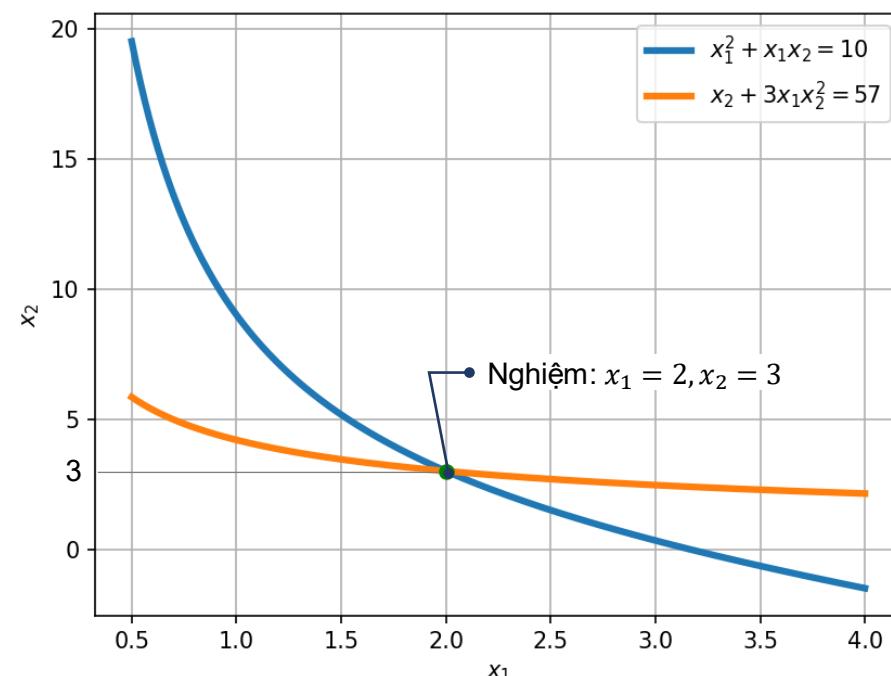
Hoặc: $A(x_i) \times x = b(x_i)$.

Khi có một số phần tử trong A hay b còn phụ thuộc vào x_i thì hệ là phi tuyến.

Ví dụ 6.5. Giải lặp hệ phương trình:

$$\begin{cases} x_1^2 + x_1 x_2 = 10 \\ x_2 + 3x_1 x_2^2 = 57 \end{cases}$$

Nhìn vào đồ thị, ta thấy: $x_1 = 2, x_2 = 3$ là nghiệm.



Ví dụ 6.5.

$$\begin{cases} x_1^2 + x_1 x_2 = 10 \\ x_2 + 3x_1 x_2^2 = 57 \end{cases}$$

Cho trước nghiệm ban đầu: $\begin{cases} x_1^{(0)} = 1.5 \\ x_2^{(0)} = 3.5 \end{cases}$

✓ Biến đổi hệ đã cho về dạng:

$$\Rightarrow \begin{cases} x_1 = \frac{10 - x_1^2}{x_2} \\ x_2 = 57 - 3x_1 x_2^2 \end{cases} \quad \Rightarrow \begin{cases} x_1 = \frac{10 - (1.5)^2}{3.5} = 2.3485 \\ x_2 = 57 - 3(1.5)(3.5)^2 = 1.8750 \end{cases}$$

$$\Rightarrow \begin{cases} x_1 = \frac{10 - (2.3485)^2}{1.8750} = 2.3918 \\ x_2 = 57 - 3(2.3485)(1.8750)^2 = 32.2307 \end{cases} \quad \Rightarrow \begin{cases} x_1 = \frac{10 - (2.3918)^2}{32.2307} = 0.1328 \\ x_2 = 57 - 3(2.3918)(32.2307)^2 = -7396.9348 \end{cases}$$

Rõ ràng, cách đưa về dạng như trên làm cho việc giải lặp phân kỳ nên không tìm được nghiệm.

Ví dụ 6.5.

$$\begin{cases} x_1^2 + x_1 x_2 = 10 \\ x_2 + 3x_1 x_2^2 = 57 \end{cases}$$

Cho trước nghiệm ban đầu: $\begin{cases} x_1^{(0)} = 1.5 \\ x_2^{(0)} = 3.5 \end{cases}$

✓ Biến đổi hệ đã cho về dạng:

$$\Rightarrow \begin{cases} x_1 = \sqrt{10 - x_1 x_2} \\ x_2 = \sqrt{\frac{57 - x_2}{3x_1}} \end{cases} \Rightarrow \begin{cases} x_1 = \sqrt{10 - (1.5)(3.5)} = 2.1794 \\ x_2 = \sqrt{\frac{57 - (3.5)}{3(1.5)}} = 3.4480 \end{cases} \Rightarrow \begin{cases} x_1 = \sqrt{10 - (2.1794)(3.4480)} = 1.5765 \\ x_2 = \sqrt{\frac{57 - (3.4480)}{3(2.1794)}} = 2.8619 \end{cases}$$

$$\Rightarrow \begin{cases} x_1 = \sqrt{10 - (1.5765)(2.8619)} = 2.3427 \\ x_2 = \sqrt{\frac{57 - (2.8619)}{3(1.5765)}} = 3.3833 \end{cases} \Rightarrow \begin{cases} x_1 = \sqrt{10 - (2.3427)(3.3833)} = 1.44012 \\ x_2 = \sqrt{\frac{57 - (3.3833)}{3(2.3427)}} = 2.7621 \end{cases}$$

Trường hợp này, nghiệm thu được qua các dây lặp không ổn định nên cũng không tìm được nghiệm.

6.2. Hệ phương trình phi tuyến.

Phương pháp giải lặp Newton – Raphson:

- ✓ Xét hàm 1 biến, khai triển Taylor:

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

Xét khai triển đến cấp 1. Khi x_{i+1} là nghiệm của phương trình thì $f(x_{i+1}) = 0$:

$$\Rightarrow x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Đây là kết quả giải phương trình phi tuyến bằng phương pháp Newton – Raphson trong chương 3.

6.2. Hệ phương trình phi tuyến.

✓ Xét tiếp hệ 2 phương trình, 2 biến:

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases} \Rightarrow \begin{cases} f_1^{(i+1)} = f_1^{(i)} + \frac{\partial f_1^{(i)}}{\partial x_1} \cdot (x_1^{(i+1)} - x_1^{(i)}) + \frac{\partial f_1^{(i)}}{\partial x_2} \cdot (x_2^{(i+1)} - x_2^{(i)}) \\ f_2^{(i+1)} = f_2^{(i)} + \frac{\partial f_2^{(i)}}{\partial x_1} \cdot (x_1^{(i+1)} - x_1^{(i)}) + \frac{\partial f_2^{(i)}}{\partial x_2} \cdot (x_2^{(i+1)} - x_2^{(i)}) \end{cases}$$

Nếu $\begin{pmatrix} x_1^{(i+1)} \\ x_2^{(i+1)} \end{pmatrix}$ là nghiệm thì $\begin{pmatrix} f_1^{(i+1)} \\ f_2^{(i+1)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

$$\Rightarrow \begin{cases} \frac{\partial f_1^{(i)}}{\partial x_1} \cdot x_1^{(i+1)} + \frac{\partial f_1^{(i)}}{\partial x_2} \cdot x_2^{(i+1)} = -f_1^{(i)} + \frac{\partial f_1^{(i)}}{\partial x_1} \cdot x_1^{(i)} + \frac{\partial f_1^{(i)}}{\partial x_2} \cdot x_2^{(i)} \\ \frac{\partial f_2^{(i)}}{\partial x_1} \cdot x_1^{(i+1)} + \frac{\partial f_2^{(i)}}{\partial x_2} \cdot x_2^{(i+1)} = -f_2^{(i)} + \frac{\partial f_2^{(i)}}{\partial x_1} \cdot x_1^{(i)} + \frac{\partial f_2^{(i)}}{\partial x_2} \cdot x_2^{(i)} \end{cases}$$

Viết hệ trên dưới dạng ma trận:

6.2. Hệ phương trình phi tuyến.

✓ Xét tiếp hệ 2 phương trình, 2 biến:

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix} \times \begin{pmatrix} x_1^{(i+1)} \\ x_2^{(i+1)} \end{pmatrix} = \begin{pmatrix} -f_1^{(i)} \\ -f_2^{(i)} \end{pmatrix} + \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix} \times \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} x_1^{(i+1)} \\ x_2^{(i+1)} \end{pmatrix} = \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \end{pmatrix} + \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix}^{-1} \times \begin{pmatrix} -f_1^{(i)} \\ -f_2^{(i)} \end{pmatrix}$$

$$\text{Đặt: } J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix}$$

Gọi là ma trận Jacobian. Hệ viết lại: $x^{(i+1)} = x^{(i)} - J^{(i),-1} \cdot f^{(i)}$

6.2. Hệ phương trình phi tuyến.

$$\begin{pmatrix} x_1^{(i+1)} \\ x_2^{(i+1)} \end{pmatrix} = \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \end{pmatrix} - \begin{pmatrix} \frac{\partial f_1^{(i)}}{\partial x_1} & \frac{\partial f_1^{(i)}}{\partial x_2} \\ \frac{\partial f_2^{(i)}}{\partial x_1} & \frac{\partial f_2^{(i)}}{\partial x_2} \end{pmatrix}^{-1} \times \begin{pmatrix} f_1^{(i)} \\ f_2^{(i)} \end{pmatrix}$$

Viết dưới dạng phương trình:

$$\Rightarrow \begin{cases} x_1^{(i+1)} = x_1^{(i)} - \frac{f_1^{(i)} \times \frac{\partial f_2^{(i)}}{\partial x_2} - f_2^{(i)} \times \frac{\partial f_1^{(i)}}{\partial x_2}}{\frac{\partial f_1^{(i)}}{\partial x_1} \times \frac{\partial f_2^{(i)}}{\partial x_2} - \frac{\partial f_2^{(i)}}{\partial x_1} \times \frac{\partial f_1^{(i)}}{\partial x_1}} \\ x_2^{(i+1)} = x_2^{(i)} - \frac{f_2^{(i)} \times \frac{\partial f_1^{(i)}}{\partial x_1} - f_1^{(i)} \times \frac{\partial f_2^{(i)}}{\partial x_1}}{\frac{\partial f_1^{(i)}}{\partial x_1} \times \frac{\partial f_2^{(i)}}{\partial x_2} - \frac{\partial f_2^{(i)}}{\partial x_1} \times \frac{\partial f_1^{(i)}}{\partial x_1}} \end{cases}$$

Để ý, mẫu số là định thức của ma trận Jacobian:

$$\det(J) = \det \begin{pmatrix} \frac{\partial f_1^{(i)}}{\partial x_1} & \frac{\partial f_1^{(i)}}{\partial x_2} \\ \frac{\partial f_2^{(i)}}{\partial x_1} & \frac{\partial f_2^{(i)}}{\partial x_2} \end{pmatrix}$$

6.2. Hệ phương trình phi tuyến.

✓ Tổng quát cho hệ có n phương trình và có n biến:

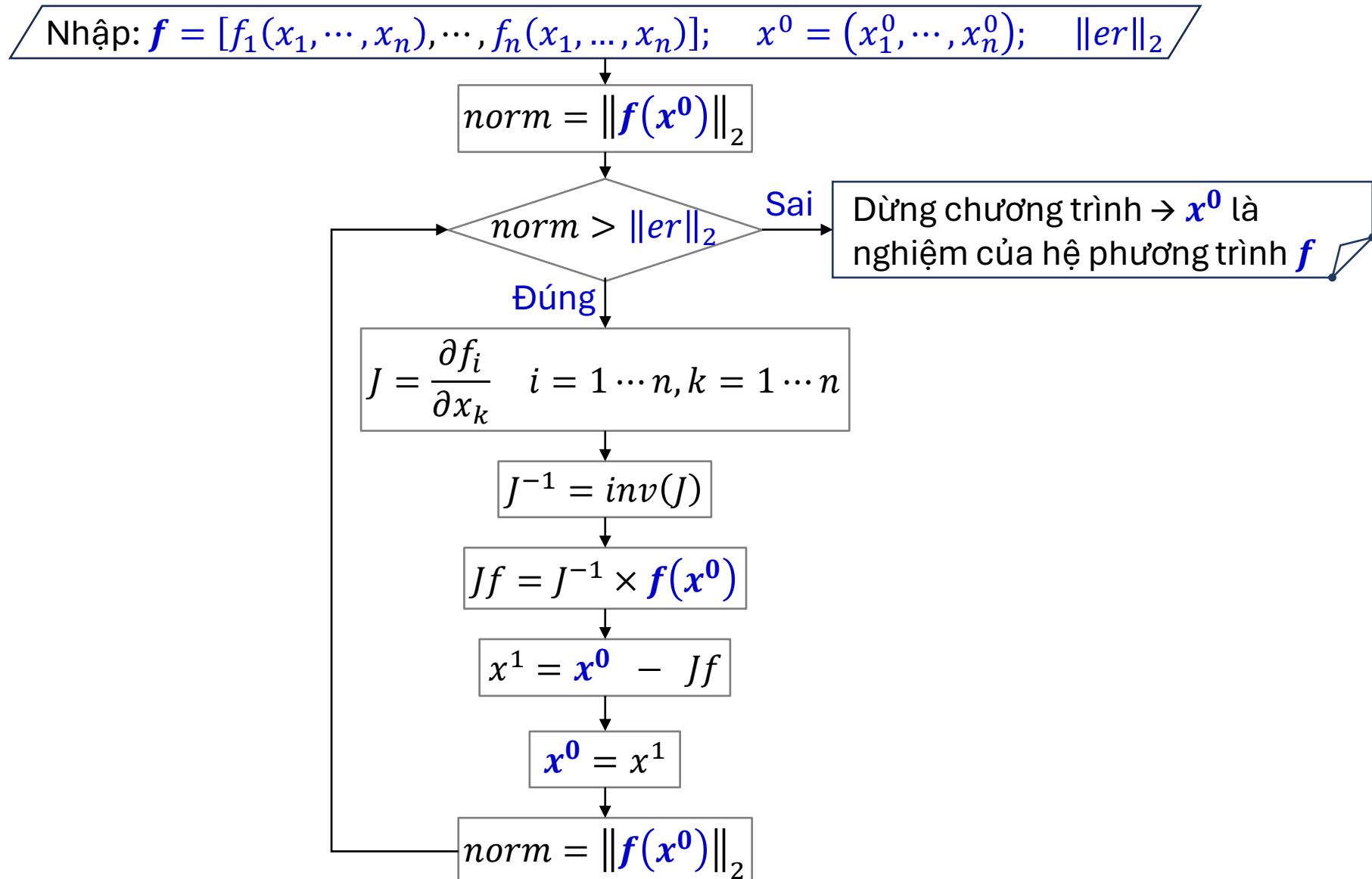
$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \dots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

$$\Rightarrow \begin{pmatrix} x_1^{(i+1)} \\ x_2^{(i+1)} \\ \dots \\ x_n^{(i+1)} \end{pmatrix} = \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \dots \\ x_n^{(i)} \end{pmatrix} - \begin{pmatrix} \frac{\partial f_1^{(i)}}{\partial x_1} & \frac{\partial f_1^{(i)}}{\partial x_2} & \dots & \frac{\partial f_1^{(i)}}{\partial x_n} \\ \frac{\partial f_2^{(i)}}{\partial x_1} & \frac{\partial f_2^{(i)}}{\partial x_2} & \dots & \frac{\partial f_2^{(i)}}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_n^{(i)}}{\partial x_1} & \frac{\partial f_n^{(i)}}{\partial x_2} & \dots & \frac{\partial f_n^{(i)}}{\partial x_n} \end{pmatrix}^{-1} \times \begin{pmatrix} f_1^{(i)} \\ f_2^{(i)} \\ \dots \\ f_n^{(i)} \end{pmatrix}$$

$$x^{(i+1)} = x^{(i)} - J^{(i),-1} \times f^{(i)}$$

6.2. Hệ phương trình phi tuyến.

Lưu đồ phương pháp Newton – Raphson:



6.2. Hệ phương trình phi tuyến.

Ví dụ 6.6. Cho bộ nghiệm ban đầu $x_1 = 1.5$; $x_2 = 3.5$ và sai số theo chuẩn 2 $\|er\|_2 = 0.5$. Sử dụng phương pháp Newton – Raphson để tìm nghiệm của hệ.

$$\begin{cases} x_1^2 + x_1 x_2 = 10 \\ x_2 + 3x_1 x_2^2 = 57 \end{cases}$$

$$x^{(i+1)} = x^{(i)} - J^{(i),-1} \times f^{(i)}$$

Giải.

$$x^{(0)} = \begin{pmatrix} 1.5 \\ 3.5 \end{pmatrix}; \begin{cases} f_1(x_1, x_2) = x_1^2 + x_1 x_2 - 10 \\ f_2(x_1, x_2) = x_2 + 3x_1 x_2^2 - 57 \end{cases} \Rightarrow J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 & x_1 \\ 3x_2^2 & 1 + 6x_1 x_2 \end{pmatrix}$$

$$\Rightarrow x^{(1)} = \begin{pmatrix} 1.5 \\ 3.5 \end{pmatrix} - \begin{pmatrix} 2(1.5) + 3.5 & 1.5 \\ 3(3.5)^2 & 1 + 6(1.5)(3.5) \end{pmatrix}^{-1} \times \begin{pmatrix} (1.5)^2 + (1.5)(3.5) - 10 \\ 3.5 + 3(1.5)(3.5)^2 - 57 \end{pmatrix} = \begin{pmatrix} 2.04 \\ 2.84 \end{pmatrix}$$

$$\Rightarrow \left\| \begin{pmatrix} (2.04)^2 + (2.04)(2.84) - 10 \\ (2.84) + 3(2.04)(2.84)^2 - 57 \end{pmatrix} \right\|_2 = 4.7987 > \|er\|_2 = 0.5$$

Ví dụ 6.6.

$$x^{(i+1)} = x^{(i)} - J^{(i),-1} \times f^{(i)}$$

$$x^{(0)} = \begin{pmatrix} 1.5 \\ 3.5 \end{pmatrix}; \quad \begin{cases} f_1(x_1, x_2) = x_1^2 + x_1 x_2 - 10 \\ f_2(x_1, x_2) = x_2 + 3x_1 x_2^2 - 57 \end{cases}; \quad J = \begin{pmatrix} 2x_1 + x_2 & x_1 \\ 3x_2^2 & 1 + 6x_1 x_2 \end{pmatrix}; \quad x^{(1)} = \begin{pmatrix} 2.04 \\ 2.84 \end{pmatrix}$$

$$\Rightarrow x^{(2)} = \begin{pmatrix} 2.04 \\ 2.84 \end{pmatrix} - \begin{pmatrix} 2(2.04) + 2.84 & 2.04 \\ 3(2.84)^2 & 1 + 6(2.04)(2.84) \end{pmatrix}^{-1} \times \begin{pmatrix} (2.04)^2 + (2.04)(2.84) - 10 \\ 2.84 + 3(2.04)(2.84)^2 - 57 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

$$\Rightarrow \left\| \begin{pmatrix} (2)^2 + (2)(3) - 10 \\ (3) + 3(2)(3)^2 - 57 \end{pmatrix} \right\|_2 = 0.0 < \|er\|_2 = 0.5 \quad \text{Dừng.}$$

Vậy $x_1 = 2$; $x_2 = 3$ là nghiệm của hệ phương trình đã cho.

6.2. Hệ phương trình phi tuyến.

Ví dụ 6.7. Cho bộ nghiệm ban đầu $x_1 = 1; x_2 = 1; x_3 = 1$ và sai số theo chuẩn 2 $\|er\|_2 = 0.1$, sử dụng phương pháp Newton – Raphson để tìm nghiệm của hệ.

$$\begin{cases} x_1^2 + x_1 x_2^2 = 4 \\ x_1 x_2 - 4x_2 x_3 = 7 \\ x_1 x_3^3 - x_2^2 x_3 = 9 \end{cases}$$

$$x^{(i+1)} = x^{(i)} - J^{(i),-1} \times f^{(i)}$$

Giải.

$$x^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}; \quad f = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} x_1^2 + x_1 x_2^2 - 4 \\ x_1 x_2 - 4x_2 x_3 - 7 \\ x_1 x_3^3 - x_2^2 x_3 - 9 \end{pmatrix}; \quad J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{pmatrix}$$

$$\Rightarrow J = \begin{pmatrix} 2x_1 + x_2^2 & 2x_1 x_2 & 0 \\ x_2 & x_1 - 4x_3 & -4x_2 \\ x_3^3 & -2x_2 x_3 & 3x_1 x_3^2 - x_2^2 \end{pmatrix}$$

Số lượng phép tính quá lớn \rightarrow Viết chương trình tính: $f, J, J^{-1}, x - J^{-1} \times f$

Ví dụ 6.7.

```

1 import numpy as np
2 import numpy.linalg as LA
3 def Newton(x1,x2,x3):
4     X=np.array([x1,x2,x3])
5     F=np.array([x1**2+x1*x2**2-4,\n
6                 x1*x2-4*x2*x3-7,\n
7                 x1*x3**3-x2**2*x3-9])
8     J=np.array([[2*x1+x2**2, 2*x1*x2, 0],\n
9                 [x2, x1-4*x3, -4*x2],\n
10                [x3**3, -2*x2*x3, 3*x1*x3**2-x2**2]])
11    Jinv=LA.inv(J)
12    Xnew=X-Jinv@F
13    Fnew=np.array([Xnew[0]**2+Xnew[0]*Xnew[1]**2-4,\n
14                  Xnew[0]*Xnew[1]-4*Xnew[1]*Xnew[2]-7,\n
15                  Xnew[0]*Xnew[2]**3-Xnew[1]**2*Xnew[2]-9])
16    Norm=LA.norm(Fnew)
17    return [Xnew, Norm]
18 x1=1; x2=1; x3=1; er2=0.1; i=0
19 X, Norm=Newton(x1,x2,x3)
20 while Norm>er2:
21     i += 1
22     print('\nX(i=%d)=[%4.2f, %6.2f, %5.2f]'%(i,X[0],X[1],X[2]))
23     print('Norm(i=%d)=%4.2f'%(i,Norm))
24     X, Norm=Newton(X[0],X[1],X[2])
25     print('\nX(i=%d)=[%4.2f, %6.2f, %5.2f]'%(i+1,X[0],X[1],X[2]))
26     print('Norm(i=%d)=%4.2f'%(i+1,Norm))

```

$$x^{(i+1)} = x^{(i)} - J^{(i),-1} \times f^{(i)}$$

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$F = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$$

$$J = \begin{pmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ j_{31} & j_{32} & j_{33} \end{pmatrix}$$

$$X(i=1) = [3.59, -1.89, 1.31]$$

$$\text{Norm}(i=1) = 22.74$$

$$X(i=2) = [2.37, -1.26, 1.66]$$

$$\text{Norm}(i=2) = 5.66$$

$$X(i=3) = [1.61, -1.16, 1.87]$$

$$\text{Norm}(i=3) = 1.22$$

$$X(i=4) = [1.50, -1.10, 1.96]$$

$$\text{Norm}(i=4) = 0.06$$

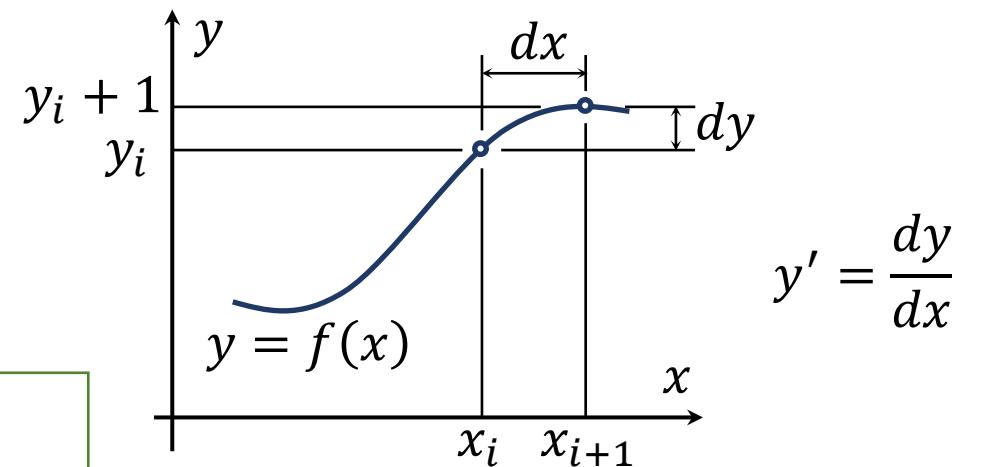
Ví dụ 6.7.

Trường hợp không tính đạo hàm của f để tìm $J \rightarrow$ Thay bằng tính sai phân.

```

1 import numpy as np
2 import numpy.linalg as LA
3 def Newton(x1,x2,x3):
4     X=np.array([x1,x2,x3])
5     F=np.array([x1**2+x1*x2**2-4,\n
6                 x1*x2-4*x2*x3-7,\n
7                 x1*x3**3-x2**2*x3-9])
8
9     h=1e-2
10    dFx1i=np.array([(x1+h)**2+(x1+h)*x2**2-4,\n
11                      (x1+h)*x2-4*x2*x3-7,\n
12                      (x1+h)*x3**3-x2**2*x3-9])
13    dFx1=(dFx1i-F)/h #d(F)/dx1
14    dFx2i=np.array([x1**2+x1*(x2+h)**2-4,\n
15                      x1*(x2+h)-4*(x2+h)*x3-7,\n
16                      x1*x3**3-(x2+h)**2*x3-9])
17    dFx2=(dFx2i-F)/h #d(F)/dx2
18    dFx3i=np.array([x1**2+x1*x2**2-4,\n
19                      x1*x2-4*x2*(x3+h)-7,\n
20                      x1*(x3+h)**3-x2**2*(x3+h)-9])
21    dFx3=(dFx3i-F)/h #d(F)/dx3
J=np.c_[dFx1,dFx2,dFx3]

```



$$h = dx \Rightarrow f'(x_i) = \frac{f(x_i + h) - f(x_i)}{h}$$

X(i=1)=[3.59, -1.88, 1.31]

Norm(i=1)=22.61

X(i=2)=[2.36, -1.25, 1.66]

Norm(i=2)=5.61

X(i=3)=[1.61, -1.16, 1.87]

Norm(i=3)=1.23

X(i=4)=[1.50, -1.10, 1.96]

Norm(i=4)=0.06

6.2. Hệ phương trình phi tuyến.

Ví dụ 6.8. Cho $x_1^{(0)} = 1; x_2^{(0)} = 1; x_3^{(0)} = 1$ và sai số theo chuẩn 2 $\|er\|_2 = 0.1$, sử dụng phương pháp Newton – Raphson để tìm nghiệm của hệ.

$$\begin{cases} 5x_1 - 2 \cos(x_2 x_3^2) - 1 = 0 \\ 8x_2 + \sqrt{x_1^2 + \sin(x_3)^3 + 0.5} + 1 = 0 \\ 4e^{-x_1 x_2} + 50x_3 + 7\pi - 2 = 0 \end{cases}$$

Giải.

$$x^{(i+1)} = x^{(i)} - J^{(i),-1} \times f^{(i)}$$

$$x^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}; \quad f = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} 5x_1 - 2 \cos(x_2 x_3^2) - 1 \\ 8x_2 + \sqrt{x_1^2 + \sin(x_3)^3 + 0.5} + 1 \\ 4e^{-x_1 x_2} + 50x_3 + 7\pi - 2 \end{pmatrix}; \quad J = \begin{pmatrix} \frac{\Delta f_1}{\Delta x_1} & \frac{\Delta f_1}{\Delta x_2} & \frac{\Delta f_1}{\Delta x_3} \\ \frac{\Delta f_2}{\Delta x_1} & \frac{\Delta f_2}{\Delta x_2} & \frac{\Delta f_2}{\Delta x_3} \\ \frac{\Delta f_3}{\Delta x_1} & \frac{\Delta f_3}{\Delta x_2} & \frac{\Delta f_3}{\Delta x_3} \end{pmatrix}$$

Ví dụ 6.8.

```

1 import numpy as np
2 import numpy.linalg as LA
3 def f(x1,x2,x3): #Hàm cần tính tích phân
4     g=np.array([5*x1-2*np.cos(x2*x3**2)-1,\n
5                 8*x2+(x1**2+np.sin(x3)**3+0.5)**(1/2)+1,\n
6                 4*np.exp(-x1*x2)+50*x3+7*np.pi-2])
7     return g
8 def df(x1,x2,x3):
9     x=np.array([x1,x2,x3]) #Nghiệm ban đầu
10    h=1e-3 #Bước tính sai phân
11    dgx1=(f(x1+h,x2,x3)-f(x1-h,x2,x3))/2/h #Sai phân theo x1
12    dgx2=(f(x1,x2+h,x3)-f(x1,x2-h,x3))/2/h #Sai phân theo x2
13    dgx3=(f(x1,x2,x3+h)-f(x1,x2,x3-h))/2/h #Sai phân theo x3
14    J=np.c_[dgx1,dgx2,dgx3] #Nối 3 cột của ma trận Jacobian
15    Jinv=LA.inv(J) #Tìm nghịch đảo của ma trận Jacobian
16    g=f(x1,x2,x3)
17    xN=x-Jinv@g # Tìm nghiệm bằng phương pháp Newton - Raphson
18    gN=f(xN[0],xN[1],xN[2]) #Tính giá trị của hàm theo nghiệm mới
19    Norm=LA.norm(gN) #Tính chuẩn 2 của véc tơ
20    return xN, Norm

```

$$x^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$f = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} 5 * x_1 - 2 \cos(x_2 x_3^2) - 1 \\ 8x_2 + \sqrt{x_1^2 + \sin(x_3)^3 + 0.5} + 1 \\ 4e^{-x_1 x_2} + 50x_3 + 7\pi - 2 \end{pmatrix}$$

$$J = \begin{pmatrix} \frac{\Delta f_1}{\Delta x_1} & \frac{\Delta f_1}{\Delta x_2} & \frac{\Delta f_1}{\Delta x_3} \\ \frac{\Delta f_2}{\Delta x_1} & \frac{\Delta f_2}{\Delta x_2} & \frac{\Delta f_2}{\Delta x_3} \\ \frac{\Delta f_3}{\Delta x_1} & \frac{\Delta f_3}{\Delta x_2} & \frac{\Delta f_3}{\Delta x_3} \end{pmatrix}$$

Ví dụ 6.8.

```
21| x1=1; x2=1; x3=1; er2=0.1; i=0
22| x, Norm=df(x1,x2,x3)
23| while Norm>er2:
24|     i += 1
25|     print ('\nX(i=%d)=[%7.4f,%7.4f,%7.4f]'%(i,x[0],x[1],x[2]))
26|     print ('Norm(i=%d)=%4.2f'%(i,Norm))
27|     x, Norm=df(x[0],x[1],x[2])
28| print ('\nX(i=%d)=[%7.4f,%7.4f,%7.4f]'%(i+1,x[0],x[1],x[2]))
29| print ('Norm(i=%d)=%4.2f'%(i+1,Norm))
```

X(i=1)=[1.8273, -0.3059, -0.4433]

Norm(i=1)=7.82

X(i=2)=[0.5995, -0.2224, -0.4658]

Norm(i=2)=1.27

X(i=3)=[0.5994, -0.2336, -0.4918]

Norm(i=3)=0.00

Bài tập.

Bài tập 6.1. Dùng phương pháp Cramer để giải hệ:

$$\begin{cases} -5x_1 + 4x_2 + 6x_3 = 4 \\ 2x_1 + 7x_2 - 2x_3 = 1 \\ 3x_1 - 2x_2 + x_3 = -5 \end{cases}$$

Bài tập 6.2. Dùng phương pháp Gauss để giải hệ:

$$\begin{cases} 3x_1 - 2x_2 - 5x_3 = 3 \\ 2x_1 - 4x_2 + 2x_3 = 4 \\ -x_1 - 2x_2 + 3x_3 = -5 \end{cases}$$

Bài tập 6.3. Dùng phép biến đổi sơ cấp trên hàng để biến ma trận: $A|b$ thành $I|X$, từ đó tìm được nghiệm X của hệ:

$$\begin{cases} 3x_1 - 2x_2 - 5x_3 = -2 \\ 2x_1 - 4x_2 + 2x_3 = 7 \\ -x_1 - 2x_2 + 3x_3 = 9 \end{cases}$$

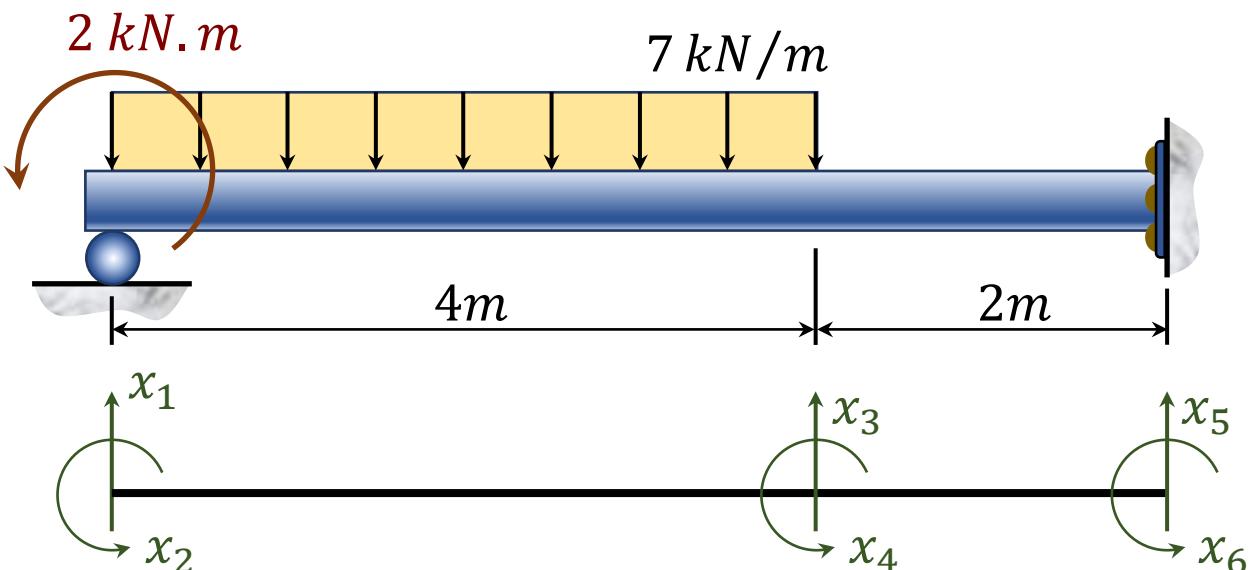
Bài tập 6.4. Dùng phương pháp lặp đơn để tìm nghiệm gần đúng của hệ:

$$\begin{cases} 0.25x_1 + 5x_2 - 0.5x_3 = 0.6 \\ x_1 - 0.1x_2 + 5x_3 = 6 \\ -2x_1 + 0.1x_2 + 4x_3 = 1 \end{cases}$$

Bài tập.

Bài tập 6.5. Dầm cho như hình vẽ. Bằng phương pháp phần tử hữu hạn với hệ 2 phần tử 6 bậc tự do, có được hệ phương trình đại số tuyến tính sau:

$$\begin{pmatrix} 1 & 3 & 0 & 0 & 0 & 0 \\ 3 & 4 & 2 & 1 & 3 & 0 \\ 0 & 2 & 9 & 4 & 1 & 2 \\ 0 & 1 & 4 & 7 & 5 & 0 \\ 0 & 3 & 1 & 5 & 6 & 2 \\ 0 & 0 & 2 & 0 & 2 & 3 \end{pmatrix} \times \begin{pmatrix} -1 \\ x_2 \\ x_3 \\ x_4 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} f_1 \\ -1 \\ -4 \\ 2 \\ f_5 \\ f_6 \end{pmatrix}$$



Giải hệ tìm: x_2 , x_3 , x_4 và f_1 , f_5 , f_6

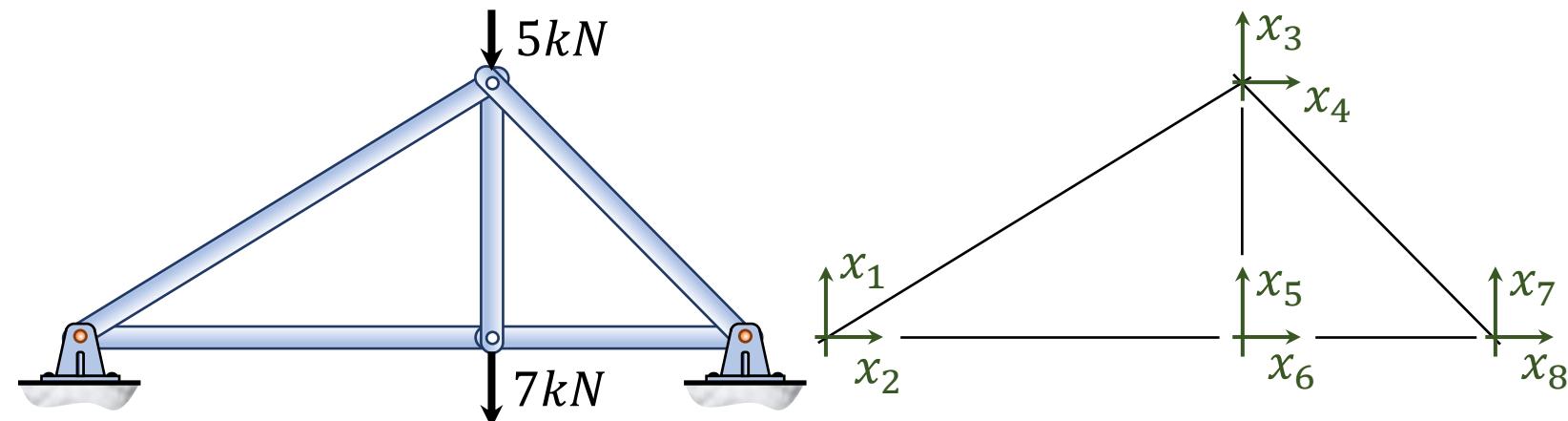
Bài tập.

Bài tập 6.6. Hệ giàn cho như hình vẽ. Bằng phương pháp phần tử hữu hạn với hệ 5 phần tử 8 bậc tự do, có được hệ phương trình đại số tuyến tính sau:

Giải hệ tìm:

x_2, x_3, x_4, x_5, x_6
và f_1, f_2, f_7, f_8

$$\begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 7 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 9 & 3 & 4 & -1 & 0 & 0 \\ 0 & 0 & 3 & 8 & 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 1 & 9 & 4 & -3 & 0 \\ 0 & 0 & -1 & 0 & 4 & 6 & 2 & 0 \\ 0 & 0 & 0 & 0 & -3 & 2 & 8 & -4 \\ 0 & 0 & 0 & 0 & 0 & 0 & -4 & 3 \end{pmatrix} \times \begin{pmatrix} 2 \\ 1 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ -5 \\ 0 \\ -7 \\ 0 \\ f_7 \\ f_8 \end{pmatrix}$$



Bài tập.

Bài tập 6.7. Cho $x_1^{(0)} = 1; x_2^{(0)} = 1; x_3^{(0)} = 1$ và sai số theo chuẩn 2 $\|er\|_2 = 0.1$, sử dụng phương pháp Newton – Raphson để tìm nghiệm của hệ sau.

$$\begin{cases} 0.5x_1x_2x_3 + 6x_2 - x_3^2 - 5 = 0 \\ x_1x_2^2 + x_3 - 0.4x_3^3 + 8 = 0 \\ x_1 + 0.2x_1^2x_2^2 - 2x_2 - 1 = 0 \end{cases}$$

X (i=136) = [-28.7865, -0.4179, 4.2486]
Norm (i=136) = 0.00

Bài tập 6.8. Cho $x_1^{(0)} = 0; x_2^{(0)} = 0; x_3^{(0)} = 0$ và sai số theo chuẩn 2 $\|er\|_2 = 0.1$, sử dụng phương pháp Newton – Raphson để tìm nghiệm của hệ sau.

$$\begin{cases} x_1 + \cos(x_1x_2x_3) - 1 = 0 \\ \sqrt[5]{1 - x_1} + 0.1x_2^2 + 0.05x_3^2 - 2x_3 = 0 \\ x_1 - 0.5x_2^2 - 2x_2 + x_3 - 2 = 0 \end{cases}$$

X (i=4) = [0.0000, -0.9487, 0.5526]
Norm (i=4) = 0.00

Chương 7:

PHƯƠNG TRÌNH VI PHÂN THƯỜNG

Nội dung của chương.

7.1. Bài toán trị đầu.

 7.1.1. Khái niệm.

 7.1.2. Phương pháp Euler.

 7.1.3. Phương pháp Heun.

 7.1.4. Phương pháp Runge Kutta bậc 2.

 7.1.5. Phương pháp Runge Kutta bậc 4.

7.2. Bài toán trị biên.

 7.2.1. Phương pháp bắn

 7.2.2. Phương pháp sai phân hữu hạn

7.1. Bài toán trị đầu.

7.1.1. Khái niệm.

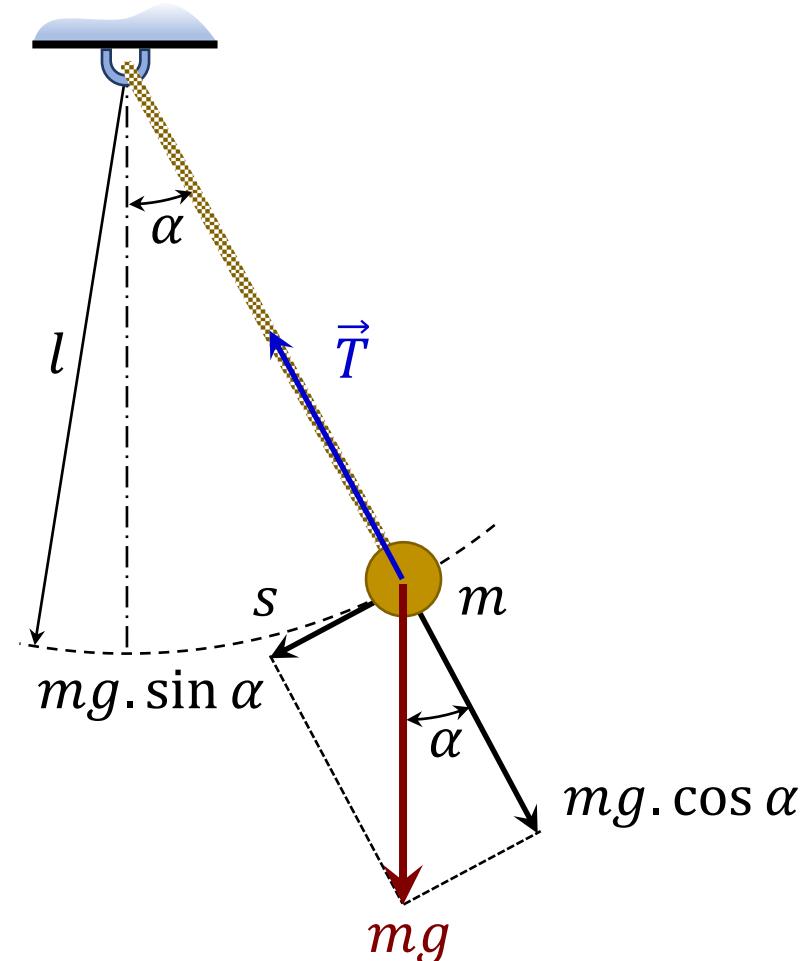
Khi α đủ nhỏ ($\alpha < 10^0$) $\Rightarrow \alpha \approx \sin \alpha = \frac{s}{l}$

Theo định luật 2 Newton: $-mg \cdot \sin \alpha = ma = ms''$

$$\Rightarrow s'' + \frac{g}{l} \times s = 0 \quad \text{Hay: } \frac{d^2s}{dt^2} + \frac{g}{l} \times s = 0$$

Khi α là lớn \rightarrow Việc tìm nghiệm rất khó

\Rightarrow Phải sử dụng phương pháp số.



7.1. Bài toán trị đầu.

7.1.2. Phương pháp Euler.

Xét phương trình vi phân: $\begin{cases} \frac{dy}{dt} = f(t, y(t)) \text{ Hay: } y'(t) = f(t, y(t)) \\ t_0 \leq t \leq t_n \\ y|_{t=t_0} = y_0 \end{cases}$

Chia $[t_0, t_n]$ thành n khoảng \rightarrow Thời gian của mỗi khoảng: $h = \frac{t_n - t_0}{n}$

Các mốc thời gian khảo sát: $t = (t_0, \quad t_0 + h, \quad t_0 + 2h, \dots, \quad t_0 + nh)$

Nếu $y(t)$ là nghiệm duy nhất, khai triển Taylor trong khoảng $[t_{i-1}, t_i]$:

$$y(t_i) = y(t_{i-1}) + y'(t_{i-1}) \times (t_i - t_{i-1}) + y''(t_{i-1}) \frac{(t_i - t_{i-1})^2}{2} + \dots$$

Chỉ xét đến vi phân cấp 1.

Các xấp xỉ: $y_i \approx y(t_i)$; $y_{i-1} \approx y(t_{i-1})$; $y'_{i-1} \approx y'(t_{i-1}) = f(t_{i-1}, y_{i-1})$

và $h = t_i - t_{i-1}$ suy ra: $y_i = y_{i-1} + h \times f(t_{i-1}, y_{i-1}); \quad i = 1, \dots, n$

7.1.2. Phương pháp Euler.

$$y_i = y_{i-1} + h \times f(t_{i-1}, y_{i-1}); \quad i = 1, \dots, n$$

Trình tự các bước thực hiện:

1/ Chia khoảng thời gian $[t_0, t_n]$ thành n khoảng.

2/ Tại (t_0, y_0) kẻ đường thẳng có hệ số góc:

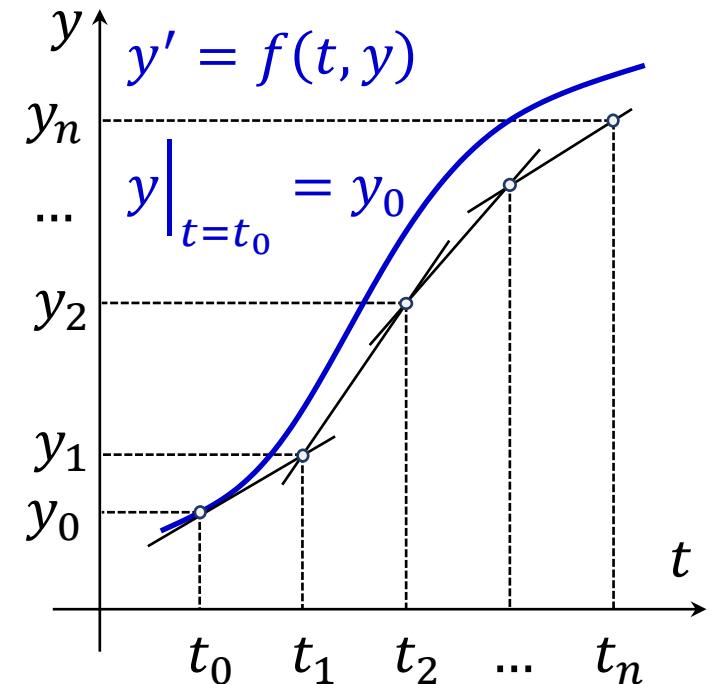
$y'(t_0) = f(t_0, y_0)$ cắt t_1 tại y_1 là xấp xỉ của $y(t_1)$

3/ Tại (t_1, y_1) kẻ đường thẳng có hệ số góc:

$y'(t_1) = f(t_1, y_1)$ cắt t_2 tại y_2 là xấp xỉ của $y(t_2)$

Thực hiện cho đến khi tìm được y_n là xấp xỉ của hàm $y(t_n)$

Sai số: $\Delta_a = \max \left(\sum_{i=1}^n |y(t_i) - y_i| \right)$



7.1.2. Phương pháp Euler.

$$y_i = y_{i-1} + h \times f(t_{i-1}, y_{i-1}); \quad i = 1, \dots, n$$

Ví dụ 7.1. Xấp xỉ nghiệm của bài toán Cauchy bằng phương pháp Euler. Biết nghiệm chính xác của bài toán là: $y(t) = (t + 1)^2 - 0.5e^t$

$$\begin{cases} y'(t) = y - t^2 + 1; & 0 \leq t \leq 2 \\ y|_{t=0} = 0.5 \end{cases}$$

Giải.

$$t_0 = 0; \quad y_0 = 0.5$$

$$h = \frac{t_n - t_0}{n} = \frac{2 - 0}{n}$$

Công thức xấp xỉ nghiệm của bài toán:

$$y_i = y_{i-1} + h(y_{i-1} - t_{i-1}^2 + 1)$$

Ví dụ 7.1.

$$h = \frac{t_n - t_0}{n} = \frac{2 - 0}{n}; \quad t_0 = 0; \quad y_0 = 0.5;$$

$$y_i = y_{i-1} + h(y_{i-1} - t_{i-1}^2 + 1)$$

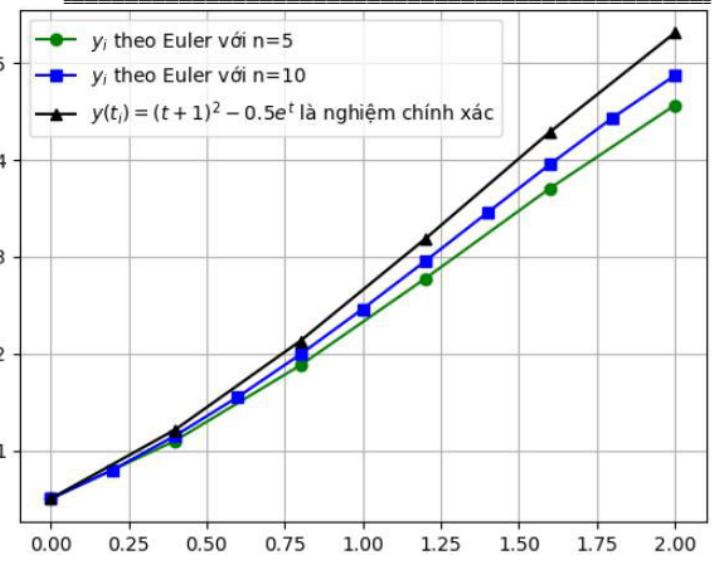
$$y(t) = (t+1)^2 - 0.5e^t$$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from tabulate import tabulate
4 def ODEe(t0, tn, y0, n):
5     h = (tn - t0)/n
6     t = np.linspace(t0, tn, n+1)
7     yi = np.zeros(len(t), dtype=float)
8     yi[0] = y0
9     for i in range(len(t)):
10         if i > 0:
11             yi[i] = yi[i-1]+h*(yi[i-1]-t[i-1]**2+1)
12     y = (t+1)**2-0.5*np.exp(t)
13     return t, yi, y
14 t0, tn, y0 = 0, 2, 0.5
15 n5, n10 = 5, 10
16 Title = np.array(['i', 'ti', 'yi', 'y(ti)', '|y(ti)-yi|'])
17
18 t5, yi5, y5 = ODEe(t0, tn, y0, n5)
19 Table5 = np.c_[range(len(t5)), t5, y5, yi5, abs(y5-yi5)]
20 Table5 = np.vstack((Title, Table5))
21 print(tabulate(Table5, headers='firstrow', tablefmt='fancy_grid'))
22
23 t10, yi10, y10 = ODEe(t0, tn, y0, n10)
24 Table10 = np.c_[range(len(t10)), t10, y10, yi10, abs(y10-yi10)]
25 Table10 = np.vstack((Title, Table10))
26 print(tabulate(Table10, headers='firstrow', tablefmt='fancy_grid'))
27
28 plt.plot(t5, yi5, '-go', label='$y_i$ theo Euler với n=%d'%n5)
29 plt.plot(t10, yi10, '-bs', label='$y_i$ theo Euler với n=%d'%n10)
30 plt.plot(t5, y5, '-k^', label='$y(t_i)=(t+1)^2-0.5e^t$ là nghiệm chính xác')
31 plt.grid(); plt.legend(); plt.show()

```

i	ti	yi	y(ti)	y(ti)-yi
0	0	0.5	0.5	0
1	0.4	1.21409	1.1	0.114088
2	0.8	2.12723	1.876	0.25123
3	1.2	3.17994	2.7704	0.409542
4	1.6	4.28348	3.70256	0.580924
5	2	5.30547	4.55958	0.745888



i	ti	yi	y(ti)	y(ti)-yi
0	0	0.5	0.5	0
1	0.2	0.829299	0.8	0.0292986
2	0.4	1.21409	1.152	0.0620877
3	0.6	1.64894	1.5504	0.0985406
4	0.8	2.12723	1.98848	0.13875
5	1	2.64086	2.45818	0.182683
6	1.2	3.17994	2.94981	0.23013
7	1.4	3.7324	3.45177	0.280627
8	1.6	4.28348	3.95013	0.333356
9	1.8	4.81518	4.42815	0.387023
10	2	5.30547	4.86578	0.439687

7.1. Bài toán trị đầu.

7.1.3. Phương pháp Heun (Euler cải tiến).

Để giảm sai số xấp xỉ, công thức Euler thay $f(t_{i-1}, y_{i-1})$ bởi $\frac{f(t_{i-1}, y_{i-1}) + f(t_i, y_i)}{2}$

$$\Rightarrow y(t_i) \approx \textcolor{red}{y}_i = y_{i-1} + h \frac{f(t_{i-1}, y_{i-1}) + f(t_i, \textcolor{blue}{y}_i)}{2}$$

Vì giá trị cần tìm $\textcolor{blue}{y}_i$ có mặt cả 2 vế của phương trình nên giải, tìm là phức tạp.

Để đơn giản, thay $\textcolor{blue}{y}_i$ bên vế phải bằng: $y_{i-1} + hf(t_{i-1}, y_{i-1})$

Công thức trở thành:

$$y(t_i) \approx \textcolor{red}{y}_i = y_{i-1} + h \frac{f(t_{i-1}, y_{i-1}) + f(t_i, y_{i-1} + hf(t_{i-1}, y_{i-1}))}{2}; \quad i = 1, \dots, n$$

7.1.3. Phương pháp Heun (Euler cải tiến).

$$y(t_i) \approx y_i = y_{i-1} + h \frac{f(t_{i-1}, y_{i-1}) + f(t_i, y_{i-1} + hf(t_{i-1}, y_{i-1}))}{2}; \quad i = 1, \dots, n$$

Ví dụ 7.2. Giải phương trình vi phân sau bằng phương pháp Heun. Biết nghiệm chính xác của bài toán là: $y(t) = (t + 1)^2 - 0.5e^t$

$$\begin{cases} y'(t) = y - t^2 + 1; & 0 \leq t \leq 2 \\ y|_{t=0} = 0.5 \end{cases}$$

Giải.

$$t_0 = 0; \quad y_0 = 0.5; \quad h = \frac{t_n - t_0}{n} = \frac{2 - 0}{n}; \quad f(t, y) = y - t^2 + 1$$

Công thức xấp xỉ nghiệm của bài toán:

$$y_i = y_{i-1} + h \times \frac{f(t_{i-1}, y_{i-1}) + f(t_i, y_{i-1} + h \times f(t_{i-1}, y_{i-1})))}{2}; \quad i = 1, \dots, n$$

Ví dụ 7.2.

$$t_0 = 0; \quad y_0 = 0.5; \quad t_n = 2; \quad h = \frac{t_n - t_0}{n}; \quad f(t, y) = y - t^2 + 1; \quad y(t) = (t + 1)^2 - 0.5e^t$$

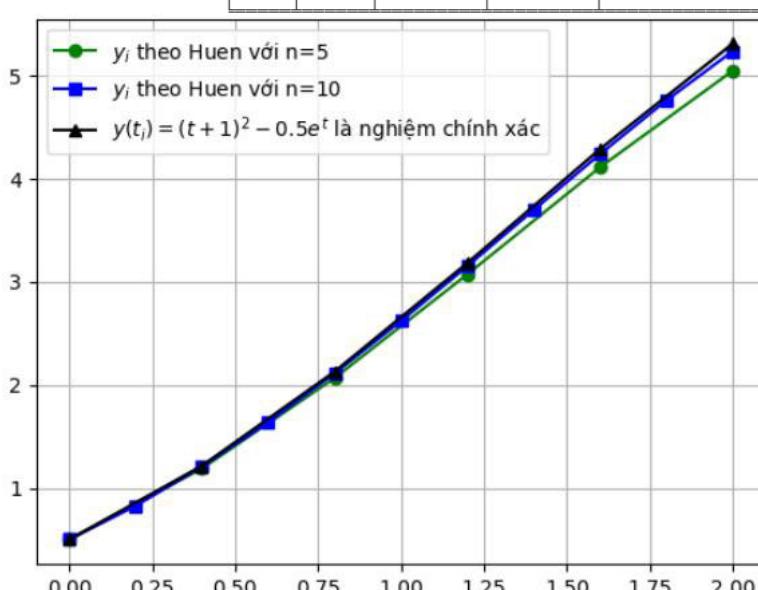
$$y_i = y_{i-1} + h \times \frac{f(t_{i-1}, y_{i-1}) + f(t_i, y_{i-1} + h \times f(t_{i-1}, y_{i-1})))}{2}; \quad i = 1, \dots, n$$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from tabulate import tabulate
4 f = lambda t, y: y-t**2+1
5 def ODEh(to, tn, y0, n):
6     h = (tn-t0)/n
7     t = np.linspace(t0, tn, n+1)
8     yi = np.zeros(len(t), dtype=float)
9     yi[0] = y0
10    for i in range(len(t)):
11        if i > 0:
12            yi[i] = yi[i-1] + h*(f(t[i-1], yi[i-1]) + f(t[i], \
13                yi[i-1]+h*f(t[i-1], yi[i-1]))/2
14    y = (t+1)**2-0.5*np.exp(t)
15    return t, yi, y
16 t0, tn, y0 = 0, 2, 0.5
17 n5, n10 = 5, 10
18 Title = np.array(['i', 'ti', 'yi', 'y(ti)', '|y(ti)-yi|'])
19
20 t5, yi5, y5 = ODEh(t0, tn, y0, n5)
21 Table5 = np.c_[range(len(t5)), t5, y5, yi5, abs(y5-yi5)]
22 Table5 = np.vstack((Title, Table5))
23 print(tabulate(Table5, headers='firstrow', tablefmt='fancy_grid'))
24
25 t10, yi10, y10 = ODEh(t0, tn, y0, n10)
26 Table10 = np.c_[range(len(t10)), t10, y10, yi10, abs(y10-yi10)]
27 Table10 = np.vstack((Title, Table10))
28 print(tabulate(Table10, headers='firstrow', tablefmt='fancy_grid'))
29
30 plt.plot(t5, yi5, '-go', label='$y_i$ theo Huen với n=%d'%n5)
31 plt.plot(t10, yi10, '-bs', label='$y_i$ theo Huen với n=%d'%n10)
32 plt.plot(t5, y5, '-k^', label='$y(t_i)=(t+1)^2-0.5e^t$ là nghiệm chính xác')
33 plt.grid(); plt.legend(); plt.show()

```

i	ti	yi	y(ti)	y(ti)-yi	i	ti	yi	y(ti)	y(ti)-yi
0	0	0.5	0.5	0	0	0	0.5	0.5	0
1	0.4	1.21409	1.188	0.0260877	1	0.2	0.829299	0.826	0.00329862
2	0.8	2.12723	2.06544	0.0617895	2	0.4	1.21409	1.20692	0.00716765
3	1.2	3.17994	3.06965	0.11029	3	0.6	1.64894	1.63724	0.0116982
4	1.6	4.28348	4.10788	0.1756	4	0.8	2.12723	2.11024	0.0169938
5	2	5.30547	5.04287	0.262604	5	1	2.64086	2.61769	0.0231715



7.1. Bài toán trị đầu.

7.1.4. Phương pháp Runge – Kutta.

Công thức Runge – Kutta bậc 2.

Mục đích của phương pháp là để giảm sai số của nghiệm.

Ý tưởng của phương pháp là làm tăng độ chính xác y_{t_i} tại điểm $t_{i-1} + h$ bằng cách đưa vào trong khoảng $[t_{i-1}, t_{i-1} + h]$ điểm trung gian $t_{i-1} + h/2$, công thức:

$$\begin{cases} y(t_i) \approx y(t_{i-1}) + \frac{h}{2} \times (k_1 + k_2) \\ k_1 = f(t_{i-1}, y_{i-1}) \\ k_2 = f(t_{i-1} + h, y_{i-1} + h \times k_1) \end{cases}$$

$$y(t_i) \approx y(t_{i-1}) + h \times f\left(t_{i-1} + \frac{h}{2}, y(t_{i-1}) + \frac{h}{2} \times f(t_{i-1}, y(t_{i-1}))\right)$$

7.1. Bài toán trị đầu.

7.1.4. Phương pháp Runge – Kutta.

Công thức Runge – Kutta bậc 4.

$$\left\{ \begin{array}{l} y_i = y_{i-1} + \frac{h}{6} (k_1^{i-1} + 2k_2^{i-1} + 2k_3^{i-1} + k_4^{i-1}) \\ k_1^{i-1} = f(t_{i-1}, y_{i-1}) \\ k_2^{i-1} = f\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_1^{i-1}\right) \\ k_3^{i-1} = f\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_2^{i-1}\right) \\ k_4^{i-1} = f(t_{i-1} + h, y_{i-1} + h \times k_3^{i-1}) \\ i = 1, 2, \dots, n \end{array} \right.$$

Đây là phương pháp có độ chính xác cao và được sử dụng phổ biến trong kỹ thuật.

7.1.4. Phương pháp Runge – Kutta.

Ví dụ 7.3. Giải phương trình vi phân sau bằng phương pháp Runge – Kutta bậc 4. Biết nghiệm chính xác của bài toán là: $y(t) = (t + 1)^2 - 0.5e^t$

$$\begin{cases} y'(t) = y - t^2 + 1; & 0 \leq t \leq 2 \\ y|_{t=0} = 0.5 \end{cases}$$

Giải. $t_0 = 0; y_0 = 0.5; t_n = 2; h = \frac{t_n - t_0}{n}; f(\textcolor{red}{t}, \textcolor{green}{y}) = \textcolor{green}{y} - \textcolor{red}{t}^2 + 1$

Công thức xấp xỉ nghiệm của bài toán:

$$k_1^{i-1} = \textcolor{blue}{f}(t_{i-1}, y_{i-1}); \quad k_2^{i-1} = \textcolor{blue}{f}\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_1^{i-1}\right)$$

$$k_3^{i-1} = \textcolor{blue}{f}\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_2^{i-1}\right); \quad k_4^{i-1} = \textcolor{blue}{f}(t_{i-1} + h, y_{i-1} + h \times k_3^{i-1})$$

$$y_i = y_{i-1} + \frac{h}{6}(k_1^{i-1} + 2k_2^{i-1} + 2k_3^{i-1} + k_4^{i-1}); \quad i = 1, 2, \dots, n$$

Ví dụ 7.3. $t_0 = 0; y_0 = 0.5; t_n = 2; h = \frac{t_n - t_0}{n}$ $f(t, y) = y - t^2 + 1$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from tabulate import tabulate
4 f = lambda t, y: y-t**2+1
5 def ODErk4(t0, tn, y0, n):
6     h = (tn-t0)/n
7     t = np.linspace(t0, tn, n+1)
8     yi = np.zeros(len(t), dtype=float)
9     yi[0] = y0
10    for i in range(len(t)):
11        if i > 0:
12            k1 = f(t[i-1], yi[i-1])
13            k2 = f(t[i-1]+h/2, yi[i-1]+h*k1/2)
14            k3 = f(t[i-1]+h/2, yi[i-1]+h*k2/2)
15            k4 = f(t[i-1]+h, yi[i-1]+h*k3)
16            yi[i] = yi[i-1] + h*(k1+2*k2+2*k3+k4)/6
17    y = (t+1)**2-0.5*np.exp(t)
18    return t, yi, y
19 t0, tn, y0 = 0, 2, 0.5
20 n5, n10 = 5, 10
21 Title = np.array(['i', 'ti', 'yi', 'y(ti)', '|y(ti)-yi|'])
22 t5, yi5, y5 = ODErk4(t0, tn, y0, n5)
23 Table5 = np.c_[range(len(t5)), t5, y5, yi5, abs(y5-yi5)]
24 Table5 = np.vstack((Title, Table5))
25 print(tabulate(Table5, headers='firstrow', tablefmt='fancy_grid'))
26 t10, yi10, y10 = ODErk4(t0, tn, y0, n10)
27 Table10 = np.c_[range(len(t10)), t10, y10, yi10, abs(y10-yi10)]
28 Table10 = np.vstack((Title, Table10))
29 print(tabulate(Table10, headers='firstrow', tablefmt='fancy_grid'))
30 plt.plot(t5, yi5, '-go', label='$y_i$ theo Runge-Kutta với n=%d'%n5)
31 plt.plot(t10, yi10, '-bs', label='$y_i$ theo Runge-Kutta với n=%d'%n10)
32 plt.plot(t5, y5, '-k^', label='$y(t_i)=(t+1)^2-0.5e^t$ là nghiệm chính xác')
33 plt.grid(); plt.legend(); plt.show()

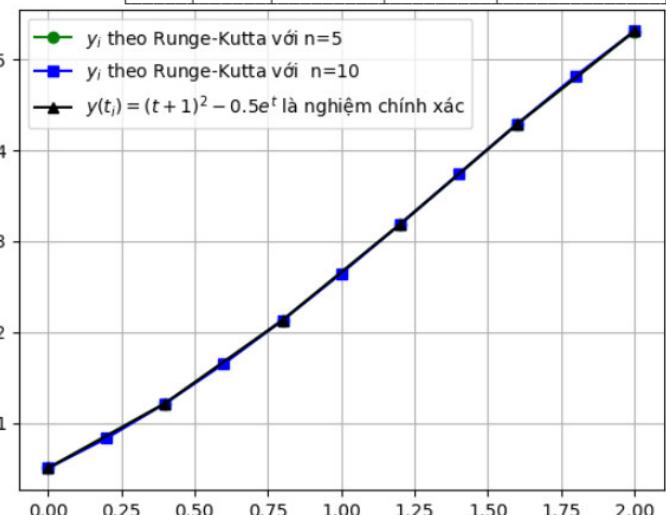
```

$$k_1^{i-1} = f(t_{i-1}, y_{i-1}); \quad k_2^{i-1} = f\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_1^{i-1}\right)$$

$$k_3^{i-1} = f\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_2^{i-1}\right); \quad k_4^{i-1} = f\left(t_{i-1} + h, y_{i-1} + h \times k_3^{i-1}\right)$$

$$y_i = y_{i-1} + \frac{h}{6}(k_1^{i-1} + 2k_2^{i-1} + 2k_3^{i-1} + k_4^{i-1}); \quad i = 1, 2, \dots, n$$

i	ti	yi	y(ti)	y(ti)-yi	i	ti	yi	y(ti)	y(ti)-yi
0	0	0.5	0.5	0	0	0	0.5	0.5	0
1	0.4	1.21409	1.21392	0.000167651	1	0.2	0.829299	0.829293	5.28759e-06
2	0.8	2.12723	2.12683	0.000395274	2	0.4	1.21409	1.21408	1.14405e-05
3	1.2	3.17994	3.17924	0.00070131	3	0.6	1.64894	1.64892	1.85828e-05
4	1.6	4.28348	4.28238	0.00110783	4	0.8	2.12723	2.1272	2.68508e-05
5	2	5.30547	5.30383	0.00163966	5	1	2.64086	2.64082	3.6393e-05



i	ti	yi	y(ti)	y(ti)-yi
0	0	0.5	0.5	0
1	0.2	0.829299	0.829293	5.28759e-06
2	0.4	1.21409	1.21408	1.14405e-05
3	0.6	1.64894	1.64892	1.85828e-05
4	0.8	2.12723	2.1272	2.68508e-05
5	1	2.64086	2.64082	3.6393e-05
6	1.2	3.17994	3.17989	4.73684e-05
7	1.4	3.7324	3.73234	5.99437e-05
8	1.6	4.28348	4.28341	7.42895e-05
9	1.8	4.81518	4.81509	9.05732e-05
10	2	5.30547	5.30536	0.00010895

7.1. Bài toán trị đầu.

Ví dụ 7.4. Giải phương trình vi phân bằng 3 phương pháp Euler, Heun, Runge - Kutta. Biết nghiệm chính xác của bài toán là: $y(t) = \frac{4}{1.3}(e^{0.8t} - e^{-0.5t}) + 2e^{-0.5t}$

$$\begin{cases} y'(t) = 4e^{0.8t} - 0.5y; & 0 \leq t \leq 4 \\ y|_{t=0} = 2 \end{cases}$$

Giải. $t_0 = 0; y_0 = 2; t_n = 4; h = \frac{t_n - t_0}{n}; f(t, y) = 4e^{0.8t} - 0.5y$

Công thức nghiệm:

Euler: $y_i = y_{i-1} + h \times f(t_{i-1}, y_{i-1}); i = 1, 2, \dots, n$

Heun: $y_i = y_{i-1} + \frac{h}{2} \times [f(t_{i-1}, y_{i-1}) + f(t_i, y_{i-1} + h \times f(t_{i-1}, y_{i-1}))]; i = 1, 2, \dots, n$

$$k_1^{i-1} = f(t_{i-1}, y_{i-1}); \quad k_2^{i-1} = f\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_1^{i-1}\right)$$

Runge-Kutta: $k_3^{i-1} = f\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_2^{i-1}\right); k_4^{i-1} = f(t_{i-1} + h, y_{i-1} + h \times k_3^{i-1})$

$$y_i = y_{i-1} + \frac{h}{6} (k_1^{i-1} + 2k_2^{i-1} + 2k_3^{i-1} + k_4^{i-1}); i = 1, 2, \dots, n$$

Ví dụ 7.4.

$$\mathbf{f}(t, y) = 4e^{0.8t} - 0.5y$$

$$\mathbf{yE}_i = y_{i-1} + h \times \mathbf{f}(t_{i-1}, y_{i-1})$$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from tabulate import tabulate
4 f = lambda t, y: 4*np.exp(0.8*t)-0.5*y
5 def ODE(t0, tn, y0, n):
6     h = (tn - t0)/n
7     t = np.linspace(t0, tn, n+1)
8     yE = np.zeros(len(t), dtype=float); yE[0] = y0
9     yH = np.zeros(len(t), dtype=float); yH[0] = y0
10    yR = np.zeros(len(t), dtype=float); yR[0] = y0
11    for i in range(len(t)):
12        if i > 0:
13            yE[i]=yE[i-1]+h*f(t[i-1],yE[i-1]) #1. Euler
14            yH[i]=yH[i-1]+h*(f(t[i-1],yH[i-1])+f(t[i],yH[i-1]+h*f(t[i-1],yH[i-1])))/2 #2. Heun
15            k1=f(t[i-1],yR[i-1]) #3. Runge-Kutta:
16            k2=f(t[i-1]+h/2,yR[i-1]+h*k1/2)
17            k3=f(t[i-1]+h/2,yR[i-1]+h*k2/2)
18            k4=f(t[i-1]+h,yR[i-1]+h*k3)
19            yR[i]=yR[i-1]+h*(k1+2*k2+2*k3+k4)/6
20    y = 4*(np.exp(0.8*t)-np.exp(-0.5*t))/1.3+2*np.exp(-0.5*t)
21    return t, yE, yH, yR, y
22 t0, tn, y0, n = 0, 4, 2, 5
23 Title = np.array(['i', 'ti', 'yEi', 'yHi', 'yRi', 'y(ti)'])
24 t, yE, yH, yR, y = ODE(t0, tn, y0, n)
25 Table = np.c_[range(len(t)), t, yE, yH, yR, y]
26 Table = np.vstack((Title, Table))
27 print(tabulate(Table, headers='firstrow', tablefmt='fancy_grid'))
28 plt.plot(t, yE, '-mo', label='$y_i$ theo Euler')
29 plt.plot(t, yH, '-g^', label='$y_i$ theo Heun')
30 plt.plot(t, yR, '-bs', label='$y_i$ theo Runge-Kute 4')
31 plt.plot(t, y, '-k^', label='$y(t)=4(e^{0.8t}-e^{-0.5t})/1.3+2e^{-0.5t}$')
32 plt.grid(); plt.legend(); plt.show()

```

$$\mathbf{yH}_i = y_{i-1} + \frac{h}{2} \times [\mathbf{f}(t_{i-1}, y_{i-1}) + \mathbf{f}(t_i, y_{i-1} + h \times \mathbf{f}(t_{i-1}, y_{i-1}))]$$

$$k_1^{i-1} = \mathbf{f}(t_{i-1}, y_{i-1})$$

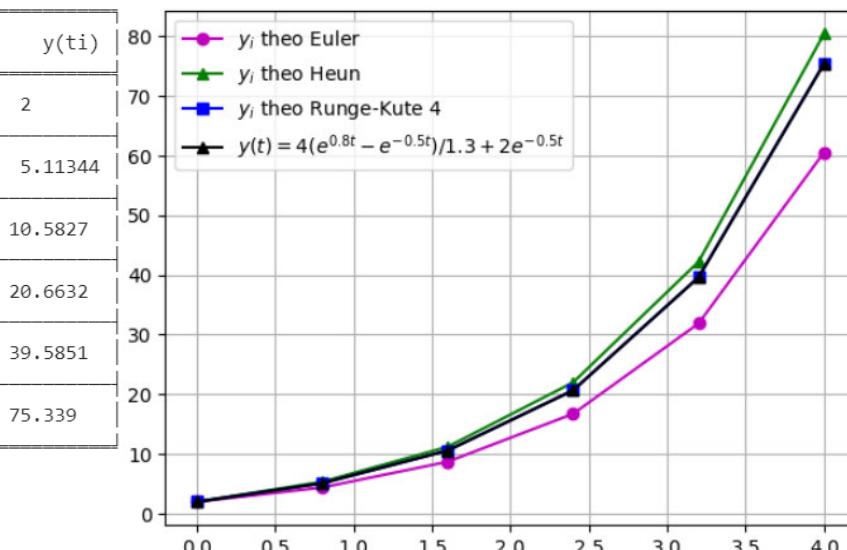
$$k_2^{i-1} = \mathbf{f}\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_1^{i-1}\right)$$

$$k_3^{i-1} = \mathbf{f}\left(t_{i-1} + \frac{h}{2}, y_{i-1} + \frac{h}{2} \times k_2^{i-1}\right)$$

$$k_4^{i-1} = \mathbf{f}(t_{i-1} + h, y_{i-1} + h \times k_3^{i-1})$$

$$\mathbf{yR}_i = y_{i-1} + \frac{h}{6} (k_1^{i-1} + 2k_2^{i-1} + 2k_3^{i-1} + k_4^{i-1})$$

i	ti	yEi	yHi	yRi	y(ti)
0	0	2	2	2	2
1	0.8	4.4	5.35437	5.11545	5.11344
2	1.6	8.70874	11.2162	10.5879	10.5827
3	2.4	16.7345	21.9933	20.6742	20.6632
4	3.2	31.8678	42.2009	39.6067	39.5851
5	4	60.5153	80.367	75.3804	75.339



Bài tập 7.1. Sử dụng phương pháp Euler để giải phương trình vi phân sau:

$$\begin{cases} y'(t) = t - 2y; & 0 \leq t \leq 3 \\ y|_{t=0} = 1 \end{cases}$$

Biết nghiệm chính xác của bài toán là: $y(t) = \frac{1}{4}(2t - 1 - 5e^{-2t})$

Bài tập 7.2. Sử dụng phương pháp Euler cải tiến để giải phương trình vi phân của bài