Mục lục

Chương 2

Giải phương trình một biến

2.1 Phương pháp chia đôi

Xét phương trình

$$f(x) = 0, \quad x \in [a, b]. \tag{$*$}$$

Giả sử

1) f liên tục trên đoạn $[a, b]$; và 2) $f(a)$ và $f(b)$ trái dấu.

Khi đó

a) $(*)$ có nghiệm.

b) Xét dãy đoạn chứa nghiệm $[a_n, b_n]$ xác định bởi

   i) $[a_0, b_0] = [a, b]$.

   ii) Đặt

$$c_n = \frac{a_n + b_n}{2}. \tag{2.1}$$

Trường hợp 1: Nếu $f(c_n) = 0$ thì $x^* = c_n$, và dừng thuật toán.

Trường hợp 2: Nếu $f(a_n)f(c_n) < 0$, thì $[a_{n+1}, b_{n+1}] = [a_n, c_n]$; ngược lại, tức là $f(c_n)f(b_n) < 0$, thì $[a_{n+1}, b_{n+1}] = [c_n, b_n]$.

thì

i) $\lim\limits_{n\to\infty} a_n = \lim\limits_{n\to\infty} b_n = x^*$, là một nghiệm của $(*)$.

ii) Chọn nghiệm gần đúng bất kỳ $x_n \in [a_n, b_n]$, chẳng hạn đơn giản $x_n = a_n$, ta có công thức đánh giá sai số

$$|x_n - x^*| \le \underbrace{b_n - a_n}_{\varepsilon_n}.$$

Ở đây $[a_0, b_0] \supset [a_1, b_1] \supset \ldots$ là dãy đoạn lồng nhau, và $b_{n+1} - a_{n+1} = \dfrac{b_n - a_n}{2}$, nên

$$\varepsilon_n = b_n - a_n = \frac{b-a}{2^n}.$$

Ví dụ 2.1. Xét phương trình $x^3 + 2x - 1 = 0$, $x \in [0, 2]$.

a) Kiểm tra điều kiện thực hiện phương pháp

b) Xây dựng dãy đoạn chứa nghiệm sau 5 bước

c) Đánh giá sai số của nghiệm gần đúng chọn được sau 5 bước.

d) Cho một nghiệm gần đúng với sai số 10^{-2}.

e) Cần thực hiện bao nhiêu bước lặp để thu được nghiệm có sai số 10^{-6}.

Giải. a) $f(x) = x^3 + 2x - 1$. Dễ thấy f liên tục trên $[0, 2]$; và $f(0) = -1 < 0$ và $f(2) = 11 > 0$ trái dấu.

```
f = lambda x: x**3 + 2*x - 1
f(0), f(2)
```

b, c) Bảng tính khoảng gần đúng và đánh giá sai số

n	a_n	b_n	c_n	$f(a_n)$	$f(b_n)$	$f(c_n)$	ε_n
0	0	2	1	−	+	+	2
1	0	1	0.5	−	+	+	1
2	0	0.5	0.25	−	+	−	0.5
3	0.25	0.5	0.375	−	+	−	0.25
4	0.375	0.5	0.4375	−	+	−	0.125
5	0.4375	$\boxed{0.5}$	0.46875	−	+	+	0.0625

Khoảng chứa nghiệm sau 5 bước là [0.4375, 0.5].

```python
1  a, b = 0, 2
2  for _ in range(5):
3      c = (a + b) / 2
4      if f(c) == 0:
5          print(f'Nghiệm đúng: {c}')
6          break
7      elif f(a) * f(c) < 0:
8          b = c
9      else:
10         a = c
11     print(a, b)  # a_n, b_n với n = 1, 2, ..., 5
12                  # Để đánh giá sai số ε_n = b_n − a_n, ta in thêm b - a
```

Để đánh giá sai số $\varepsilon_n = b_n - a_n$, tại dòng 11, ta in thêm `b - a`.

d) Ta lập tiếp bảng tính ở ý (c) đến khi sai số (ô đóng khung) nhỏ hơn 10^{-2}.

n	a_n	b_n	c_n	$f(a_n)$	$f(b_n)$	$f(c_n)$	ε_n
6	0.4375	0.46875	0.453125	$-$	$+$	$-$	0.03125
7	0.453125	0.46875	0.460938	$-$	$+$	$+$	0.015625
8	0.453125	0.460938	0.457031	$-$	$+$	$+$	$\boxed{0.0078125}$

Nghiệm gần đúng với sai số 10^{-2} là $x_8 = a_8 = 0.453125$.

```python
while b - a > 10 ** -2:  # chỉ cần thay dòng 2 bởi dòng này
```

e) $\dfrac{b-a}{2^n} < 10^{-6} \Rightarrow n > \log_2 \dfrac{b-a}{10^{-6}} = 20.9316$. Chọn số bước lặp $n = 21$.

```python
from sympy import *
log((2 - 0) / 10**-6, 2.)
```

□

2.2 Phương pháp Newton và mở rộng

Xét phương trình

$$f(x) = 0, \quad x \in [a, b]. \tag{$*$}$$

Giả sử

1) f', f'' không đổi dấu trên $[a, b]$; và 2) $f(a)$ và $f(b)$ trái dấu.

Khi đó

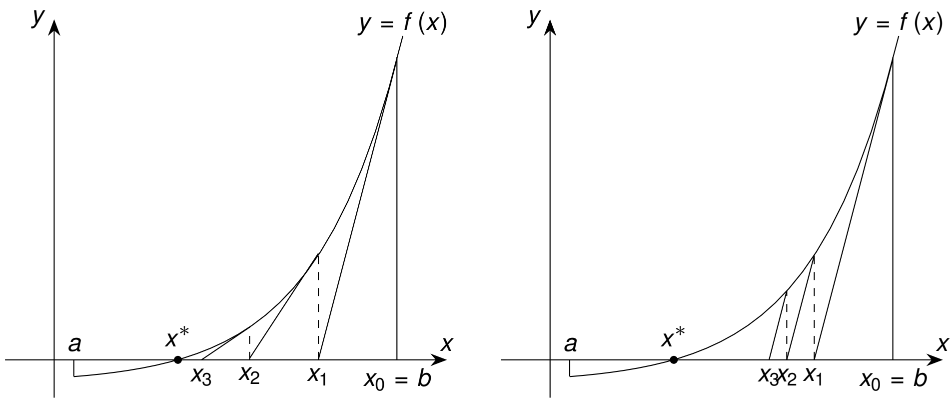
a) $(*)$ có nghiệm duy nhất $x^* \in (a, b)$.

b) Xét dãy nghiệm gần đúng $\{x_n\}$:

   i) $x_0 \in [a, b]$ sao cho $f(x_0)$ cùng dấu với f'' (thường chọn x_0 là a hoặc b).

   ii) Công thức lặp

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, \ldots \tag{2.2}$$



thì

   i) $\{x_n\}$ đơn điệu và $\lim\limits_{n\to\infty} x_n = x^*$.

   ii) Công thức sai số

$$\left|x_n - x^*\right| \le \frac{M}{2m}\left|x_n - x_{n-1}\right|^2, \quad n = 1, 2, \ldots \tag{2.3}$$

   trong đó $M \ge \left|f''(x)\right| \ \forall x \in [a, b]$, và $m = \min\{\left|f'(a)\right|, \ \left|f'(b)\right|\}$.

Để xác định M, có thể khảo sát hàm số $\left|f''(x)\right|$ trên $[a, b]$, có vài phương pháp sau. Các phương pháp cho các giá trị M khác nhau.

- Khảo sát hàm số $f''(x)$ trên $[a, b]$, từ đó khảo sát được $\left|f''(x)\right|$.
- Đánh giá sơ bộ miền giá trị của hàm sơ cấp và áp dụng tính chất của hàm trị tuyệt đối.
- Đánh giá "thô" (không chính xác cho lắm!) bằng cách chia lưới và quan sát đồ thị. Vì đánh giá M không phải là mục đích chính của bài toán, nên trong thực hành, ta chấp nhận cả phương pháp này.

Công thức Newton cải biên

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)} \tag{2.4}$$

có số phép tính mỗi bước ít hơn nhưng tốc độ hội tụ chậm hơn công thức Newton.

Ví dụ 2.2. Giải phương trình $x^3 - x^2 - 3 = 0 \quad (*)$ trên đoạn $[1, 4]$.

a) Kiểm tra điều kiện thực hiện phương pháp: nhận biết bằng đồ thị, hoặc đánh giá chính xác bằng giải tích.

b) Tìm nghiệm gần đúng sau 3 bước (tới x_3).

c) Tìm sai số của các nghiệm gần đúng ở ý trên.

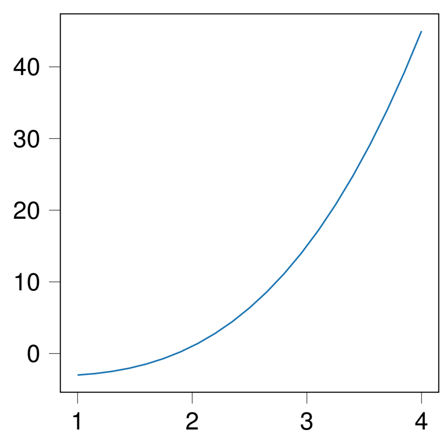
d) Tìm nghiệm gần đúng với sai số 10^{-6}, cho biết số bước lặp đã thực hiện.

e) Tìm nghiệm gần đúng sau 3 bước theo công thức Newton cải biên.

Giải. Đặt $f(x) = x^3 - x^2 - 3$.

```
f = lambda x: x**3 - x**2 - 3
```

a) Đánh giá bằng đồ thị:



Nhận xét	Quan sát
$f'(x) > 0 \ \forall x \in [1, 4]$	đồ thị đi lên theo hướng từ trái sang phải
$f''(x) > 0 \ \forall x \in [1, 4]$	đồ thị võng xuống
$f(1) < 0$	đầu bên trái của đồ thị thấp hơn mức 0 ở trục đứng
$f(4) > 0$	đầu bên phải của đồ thị cao hơn mức 0 ở trục đứng

```
import numpy as np
X = np.linspace(1, 4, 101)  # chia đoạn [1,4] thành 101
```

```
    đoạn bằng nhau
Y = [ f(x) for x in X ]

import matplotlib.pyplot as plt
plt.plot(X, Y);
```

Đánh giá bằng giải tích: Ta có $f''(x) = 6x - 2 \geq 4 > 0,\ \forall x \in [1,4]$, suy ra f' đơn điệu trên đoạn $[1,4]$, nên $f'(x) \geq f'(1) = 1 > 0,\ \forall x \in [1,4]$. Sau đó, tính $f(1) = -3 < 0$ và $f(4) = 45 > 0$.

```
from sympy import *
x = symbols('x')

f(x).diff(x, 2)           # f''(x)

f(x).diff().subs(x, 1)    # f'(1)
f(1), f(4)
```

b) $f(4)$ cùng dấu với $f'' \Rightarrow x_0 = 4$. Công thức lặp

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^3 - x_n^2 - 3}{3x_n^2 - 2x_n}$$

cho ta các nghiệm gần đúng sau 3 bước lặp

n	x_n
0	4
1	2.875
2	2.21883
3	1.92841

```
t = symbols('t')
df = lambda x: f(t).diff().subs(t, x)   # định nghĩa đạo
    hàm f'(x)

x = 4.                   # chú ý dấu .
for _ in range(3):       # chỉ số chạy _ không dùng đến trong
    chu trình
    x = x - f(x) / df(x)
    print(x)
```

c) Công thức sai số

$$|x_n - x^*| \leq \underbrace{\frac{M}{2m}|x_n - x_{n-1}|^2}_{\varepsilon_n}$$

trong đó

$$\left|f''(x)\right| = f''(x) = 6x - 2 \leq 22,\ \forall x \in [1,4] \Rightarrow M = 22$$
$$m = \min\left\{\left|f'(1)\right|, \left|f'(4)\right|\right\} = \min\{1, 40\} = 1.$$

Ta có sai số tương ứng của các nghiệm gần đúng sau 3 bước lặp.

n	x_n	ε_n
0	4	
1	2.875	13.9219
2	2.21883	4.73619
3	1.92841	0.927734

```
M = 22
m = min( abs(df(1)), abs(df(4)) )

x0 = 4.
for _ in range(3):
    x = x0 - f(x0) / df(x0)
    ss = M / 2 / m * (x - x0)**2  # biến x và x0 đại diện cho x_n và x_{n-1}
    x0 = x
    print(x, ss)
```

Trong thực hành, để đánh giá M, có thể dùng lệnh sau

```
d2f = lambda x: f(t).diff(t, 2).subs(t, x)  # định nghĩa f''(x)
Y = [ abs(d2f(x)) for x in X ]
M = max(Y)
```

d) Để tìm nghiệm với sai số 10^{-6}, ta thực hiện các bước lặp ở ý (c) đến khi sai số $\leq 10^{-6}$.

n	x_n	ε_n
4	1.86641	0.0422842
5	1.86371	0.0000803536
6	1.86371	$\boxed{2.76859 \cdot 10^{-10}}$

Nghiệm gần đúng có sai số 10^{-6} là $x_6 = 1.86371$.

```python
x0 = 4.
n = 0
while True:
    x = x0 - f(x0) / df(x0)
    ss = M / 2 / m * (x -x0)**2
    x0 = x
    n += 1
    print(n, x, ss)
    if ss < 10**-6:
        break
```

d) Công thức Newton cải biên

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)} = x_n - \frac{x_n^3 - x_n^2 - 3}{40}.$$

Ta được nghiệm gần đúng sau 3 bước lặp

n	x_n
0	4
1	2.875
2	2.56255
3	2.38103

```python
x = 4.
for _ in range(3):
  x = x - f(x) / df(4)
  print(x)
```

□

Một ứng dụng đơn giản của công thức Newton, là tìm nghịch đảo $x = \frac{1}{a}$ của số thực a. Xét phương trình

$$f(x) = \frac{1}{x} - a = 0.$$

Ta có $f'(x) = -\frac{1}{x^2}$, $f''(x) = \frac{2}{x^3}$. Theo công thức Newton

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{\frac{1}{x_n} - a}{-\frac{1}{x_n^2}} = x_n(2 - ax_n) \tag{2.5}$$

trong đó x_0 được chọn sao cho $|1 - ax_0| < 1$. Khi đó

$$\begin{aligned}
&1 - ax_{n+1} = 1 - a\left(2x_n - ax_n^2\right) = (1 - ax_n)^2 \\
&\Rightarrow 1 - ax_n = (1 - ax_0)^{2^n} \Rightarrow \frac{1}{a} - x_n = \frac{1}{a}(1 - ax_0)^{2^n} \xrightarrow[n\to\infty]{} 0 \\
&\Rightarrow x_n \xrightarrow[n\to\infty]{} \frac{1}{a}.
\end{aligned}$$

Ví dụ 2.3. Tính gần đúng $\frac{1}{9.347}$ bằng dãy lặp trên tới khi có hai phần tử của dãy trùng nhau tới 5 chữ số sau dấu phẩy.

Giải. Với $a = 9.347$, có thể chọn $x_0 = 0.1$, ta có $|1 - ax_0| = 0.0653 < 1$. Công thức lặp (2.5) cho ta bảng giá trị

n	x_n
0	0.1
1	0.10653
2	$\underline{0.10698}4$
3	$\underline{0.10698}6$

□

Một ứng dụng nữa, khai căn $x = \sqrt{a}$. Xét phương trình $f(x) = x^2 - a = 0$. Ta có $f'(x) = 2x$, $f''(x) = 2$, và công thức lặp

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right), \tag{2.6}$$

trong đó ta thường chọn $x_0 = a$.

Ví dụ 2.4. Tính gần đúng $\sqrt{2}$.

Giải. Chọn $x_0 = 2$. Công thức (2.6) cho dãy nghiệm gần đúng tới khi hai phần tử liên tiếp trùng nhau tới 5 chữ số sau dấu phẩy:

n	x_n
0	2
1	1.5
2	1.41667
3	1.41422
4	$\underline{1.41421}$
5	$\underline{1.41421}$

□

2.3 Lặp điểm bất động

Xét phương trình

$$x = g(x), \quad x \in [a, b]. \tag{$*$}$$

Giả sử g là ánh xạ co trên $[a, b]$, tức là

1) $g(x) \in [a,b]\ \forall x \in [a,b]$; và

2) $\exists q < 1,\ \forall x, y \in [a,b]$:

$$|g(x) - g(y)| \leq q\,|x - y|\,. \tag{$**$}$$

Khi đó

a) ($*$) có nghiệm duy nhất $x^* \in [a,b]$.

b) Xét dãy nghiệm gần đúng $\{x_n\}$:

i) $x_0 \in [a,b]$ bất kì (thường chọn x_0 là a, b, hoặc $\dfrac{a+b}{2}$).

ii) Công thức lặp

$$x_{n+1} = g(x_n)\,,\quad n = 0, 1, \ldots \tag{2.7}$$

thì

i) $\lim\limits_{n\to\infty} x_n = x^*$.

ii) Công thức đánh giá sai số (tiên nghiệm, hậu nghiệm):

$$\begin{aligned} |x_n - x^*| &\leq \frac{q^n}{1-q}\,|x_1 - x_0| \\ |x_n - x^*| &\leq \frac{q}{1-q}\,|x_n - x_{n-1}| \end{aligned} \tag{2.8}$$

Giả sử $|g'(x)| \leq q < 1,\ \forall x \in [a,b]$. Khi đó $\forall x, y \in [a,b]$:

$$|g(x) - g(y)| = |g'(c)| \cdot |x - y| \leq q\,|x - y| \tag{2.9}$$

tức là điều kiện ($**$) thỏa mãn.

Công thức hậu nghiệm đánh giá sai số của nghiệm gần đúng tốt hơn công thức tiên nghiệm, do sai số nhỏ hơn. Tuy nhiên, công thức tiên nghiệm giúp ta đánh giá sai số tại lặp nào đó mà không cần phải tìm nghiệm gần đúng nào.

Ví dụ 2.5. Biến đổi phương trình trong Ví dụ 2.2

$$x^3 - x^2 - 3 = 0 \Leftrightarrow x^3 = x^2 + 3 \Leftrightarrow x = \sqrt[3]{x^2 + 3}$$

với $x \in [1,4]$.

a) Kiểm tra điều kiện thực hiện phương pháp: khảo sát hàm số hoặc phương pháp đồ thị.

b) Cho trước $x_0 = 2.5$, tính nghiệm gần đúng tới x_3.

c) Tìm sai số của các nghiệm gần đúng ở trên.

d) Tìm nghiệm với sai số 10^{-4} và xác định số bươc lặp đã thực hiện.

e) Không cần tìm dãy nghiệm gần đúng, hãy xác định cần bao nhiêu bước lặp để thu được nghiệm có sai số 10^{-10}.

Giải. a) Đặt $g(x) = \sqrt[3]{x^2+3}$.

```
from sympy import *
g = lambda x: (x**2 + 3) ** Rational(1, 3)
```

Khảo sát hàm số: Ta có $g'(x) = \dfrac{2x}{3\left(x^2+3\right)^{2/3}} > 0,\ \forall x \in [1,4]$, suy ra g đồng biến trên $[1,4]$, nên

$$g(1) \le g(x) \le g(4) \Rightarrow 1.5874 \le g(x) \le 2.6684 \Rightarrow 1 \le g(x) \le 4.$$

Mặt khác, $g''(x) = \dfrac{2\left(9-x^2\right)}{9\left(x^2+3\right)^{5/3}}$. Điểm dừng* trên đoạn $[1,4]$ của $g'(x)$ thỏa mãn $g''(x) = 0 \Leftrightarrow x = 3$. Do đó

$$\begin{aligned}\left|g'(x)\right| &\le \max\left\{\left|g'(1)\right|, \left|g'(4)\right|, \left|g'(3)\right|\right\} = \\ &= \max\left\{0.264567, 0.374513, 0.381571\right\} = 0.381571 = q < 1.\end{aligned}$$

```
g(x).diff()
N(g(1), 6), N(g(4), 6)

g(x).diff(x, 2).simplify()

t = symbols('t')
dg = lambda x: g(t).diff().subs(t, x)

arr = [ N(abs(dg(x)), 6) for x in [1, 4, 3] ]
q = max(arr)
q
```

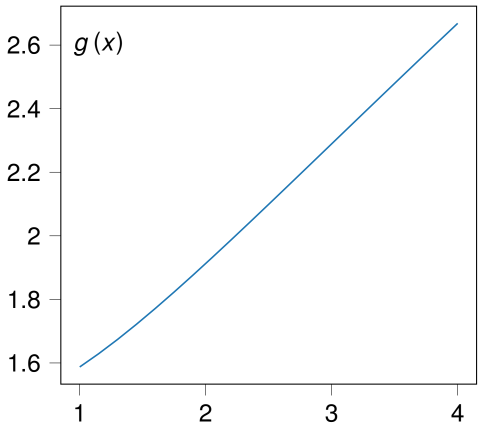
Phương pháp đồ thị: Ta có $1.58740 \le g(x) \le 2.66840,\ \forall x \in [1,4]$, suy ra

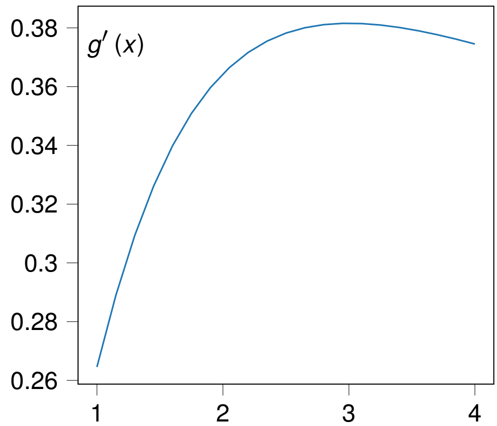
$$1 \le g(x) \le 4,\ \forall x \in [1,4].$$

Mặt khác

$$\left|g'(x)\right| \le 0.381570 = q < 1,\ \forall x \in [1,4].$$

*Việc tìm điểm dừng của $g'(x)$ có khi dẫn dến một phương trình còn khó hơn phương trình ban đầu.






```python
import numpy as np
X = np.linspace(1, 4, 101)
Y = [ g(x) for x in X ]

import matplotlib.pyplot as plt
plt.plot(X, Y);

min(Y), max(Y)

Y = [ abs(dg(x)) for x in X ]
plt.plot(X, Y);

q = max(Y)
q
```

b) Với $x_0 = 2.5$ và công thức lặp

$$x_{n+1} = g(x_n) = \sqrt[3]{x_n^2 + 3}$$

ta có nghiệm gần đúng sau 3 bước

n	x_n
0	2.5
1	2.09917
2	1.94927
3	1.8945

```python
x = 2.5
for _ in range(3):
    x = g(x)
    print(x)
```

c) Đánh giá sai số của nghiệm theo công thức hậu nghiệm

$$|x_n - x^*| \leq \frac{q}{1-q}\left|x_n - x_{n-1}\right| = \varepsilon_n$$

ta được bảng sau

n	x_n	ε_n
0	2.5	
1	2.09917	0.247314
2	1.94927	0.0924898
3	1.8945	0.033789

```python
x0 = 2.5
for _ in range(3):
    x = g(x0)
    ss = q / (1-q) * abs(x - x0)
    x0 = x
    print(x, ss)
```

d) Thực hiện tiếp các bước lặp ở ý (c) đến khi sai số $\leq 10^{-4}$

n	x_n	ε_n
4	1.87475	0.0121887
5	1.86766	0.0043737
...		
9	1.86377	$\boxed{0.000071735}$

```python
x0 = 2.5
n = 0
while True:
    x = g(x0)
    ss = q/(1-q) * abs(x - x0)
    x0 = x
    n += 1
    print(n, x, ss)
    if ss < 10**-4:
        break
```

Nghiệm gần đúng có sai số 10^{-4} là $x_9 = 1.86377$.

e) Xét sai số theo công thức tiên nghiệm

$$\frac{q^n}{1-q}|x_1 - x_0| < 10^{-10} \Rightarrow q^n < \frac{10^{-10}(1-q)}{|x_1 - x_0|}$$
$$\Rightarrow n > \log_q \frac{10^{-10}(1-q)}{|x_1 - x_0|} = 23.4491 \Rightarrow \text{ chọn } n = 24.$$

```python
x0 = 2.5
x1 = g(x0)
log(10**-10 * (1-q) / abs(x1 - x0), q)
```

□

Trong Ví dụ 2.2 và 2.5, phương pháp đánh giá giá trị của hàm số, cũng như kỹ thuật lập trình khá giống nhau.

2.4 Phân tích sai số của các phương pháp lặp

2.5 Tăng tốc độ hội tụ

2.6 Nghiệm của đa thức và phương pháp Müller

Định lý 2.1 (Định lý chặn nghiệm Cauchy). *Cho* $P(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n$, $a_0 \neq 0$. *Đặt* $M = \max\limits_{1\le i\le n} \dfrac{|a_i|}{|a_0|}$. *Khi đó mọi nghiệm thực của P đều nằm trong khoảng* $[-(M+1), M+1]$.

Định nghĩa 2.1. *Đa thức* $P(x)$ *gọi là khử bội nếu không có đa thức* $Q(x)$ *bậc dương sao cho* $Q^2 \mid P$.

Định lý 2.2 (Sturm). *Với mọi đa thức* $P(x)$, $\dfrac{P}{\gcd(P, P')}$ *là khử bội.*

Ngoài ra P và $\dfrac{P}{\gcd(P, P')}$ có cùng tập nghiệm.

Định lý 2.3. *Dãy Sturm của đa thức một biến* $P(x)$ *xác định bởi*

$$\begin{aligned} P_0 &= P \\ P_1 &= P' \\ P_{i+1} &= -\left(P_{i-1} \bmod P_i\right) \end{aligned}$$

đến khi $P_{k+1} = 0$.

Ký hiệu $V(a)$ *là số lần đổi dấu của dãy* $P_0(a), P_1(a), \ldots, P_k(a)$, *trong đó mỗi số 0 (nếu có) trong dãy chỉ tính là một lần đổi dấu. Khi đó, số nghiệm của P trong khoảng* $(a, b]$ *là* $V(a) - V(b)$.

Tóm tắt Python

- Từ khóa
  - break
  - Dấu _

Tài liệu tham khảo

[1] Phạm Kỳ Anh. *Giải tích số*. Đại học Quốc gia Hà Nội, 2002. 284 trang.

[2] Richard L. Burden, Douglas J. Faires **and** Annette M. Burden. *Numerical Analysis*. phiên bản 10. Cengage Learning, 2016. 918 trang.

[3] NumPy community. *NumPy User Guide*. phiên bản 1.22.0. 531 trang. URL: https://numpy.org/doc/stable.

[4] SciPy community. *SciPy Reference Guide*. phiên bản 1.8.1. 3584 trang. URL: https://docs.scipy.org/doc.

[5] Phan Văn Hạp **and** Lê Đình Thịnh. *Phương pháp tính và các thuật toán*. Nhà xuất bản Giáo dục, 2000. 400 trang.

[6] Doãn Tam Hòe. *Toán học tính toán*. Đại học Quốc gia Hà Nội, 2009. 240 trang.

[7] Matplotlib development team. *Matplotlib documentation*. phiên bản 3.5.1. URL: https://matplotlib.org/3.5.1/tutorials/index.html.

[8] SymPy Development Team. *SymPy Documentation*. phiên bản 1.8. 2750 trang. URL: https://github.com/sympy/sympy/releases.