

TÀI LIỆU ĐƯỢC SỬ DỤNG CHO MÔN THI TOÁN TOÁN HỌC TÍNH TOÁNHọ tên : **Giang Nguyễn Hưởng**MSSV : **0014568**

```
1 f = lambda x: x**3 + 2*x - 1
2 f(0), f(2)
3 a, b = 0, 2
4 for _ in range(5):
5     c = (a + b) / 2
6     if f(c) == 0:
7         print(f' NGHIỆM ĐÚNG: {c} ')
8         break
9     elif f(a) * f(c) < 0:
10        b = c
11    else:
12        a = c
13    ss = b - a      #  $\varepsilon_n$ 
14    print(a, b)    #  $a_n, b_n$ 
```

Mã 1:

```
1 import numpy as np
2 B = np.array( [[-0.21, -0.28,  0.05],
3               [ 0.19,  0.01, -0.26],
4               [ 0.39, -0.12, -0.06]] )
5 g = [-0.9, 3.8, -2.9]
6 q = np.linalg.norm(B, np.inf)
7 q
8 x0 = [0, 2, -1]      #  $x^{(k-1)}$ 
9 for _ in range(3):
10    x = B.dot(x0) + g   #  $x^{(k)}$ 
11    ss = q / (1-q) * np.linalg.norm(x - x0, np.inf)  #  $\varepsilon_k$ 
12    x0 = x
13    print(x, ss)
14 x = [0, 2, -1]
15 for _ in range(4):
16     for i in range(3):
17         x[i] = B[i].dot(x) + g[i]  #  $x_i^{(k)}$ 
```

```
18 print(x)
```

Mã 2:

```
1 X = ...
2 Y = ...

3 (-3*Y[0] + 4*Y[1] - Y[2]) / 2 / h
4 [ (Y[i+1] - Y[i-1]) / 2 / h for i in range(1, 10) ]
5 (3*Y[10] - 4*Y[9] + Y[8]) / 2 / h

6 (Y[2] - 2*Y[1] + Y[0]) / h**2
7 [ (Y[i+1] - 2*Y[i] + Y[i-1]) / h**2 for i in range(1, 10) ]
8 (Y[10] - 2*Y[9] + Y[8]) / h**2

9 sum( [(X[i] - X[i-1]) * (Y[i] + Y[i-1]) / 2 for i in range(1, 11)]
      )

10 sum( [(X[2*i] - X[2*i-2]) * (Y[2*i] + 4*Y[2*i-1] + Y[2*i-2]) / 6
        for i in range(1, 6)] )
```

Mã 3:

```
1 # VD2, 3: import numpy as np
2 f = lambda x, y: y - x # VD2: f = lambda x, y: np.array([x * y[0] - y[1],
3 # VD3: f = lambda x, y: np.array([y[1], y[2], x *
4 # VD2: X = [1, 1.1, 1.3, 1.5]
5 # VD3: X = [-1, -0.8, -0.6, -0.5]
6 y = 2 # VD2: y = [-1, 2]
7 # VD3: y = [1, 0, -2]
8 for n in range(3):
9     h = X[n+1] - X[n]
10    y = y + h * f(X[n], y)
11    print(y)
```

Mã 4:

```
1 d = lambda k, i: Y[i] if k == 0 else d(k-1, i+1) - d(k-1, i)
2 [ [d(k, i) for i in range(4-k)] for k in range(4) ]
```

```

3 from sympy import *
4 x, t = symbols('x t')

5 P = 0 # Y[3]
6 for k in range(4):
7     prod = d(k, 0) / factorial(k) # d(k, 3-k)
8     for i in range(k):
9         prod *= t - i # t + i
10    P += prod

11 P

12 P.subs(t, (x - X[0]) / 1).expand() # X[3]

```

Mã 5:

```

1 # VD2, 3: import numpy as np
2 f = lambda x, y: y - x # VD2: f = lambda x, y: np.array([x * y[0] - y[1],
3     y[0] + y[1] - 1])
4 # VD3: f = lambda x, y: np.array([y[1], y[2], x *
5     y[2] - y[0]])
6 X = [0, 0.2, 0.3, 0.5] # VD2: X = [1, 1.1, 1.3, 1.5]
7 # VD3: X = [-1, -0.8, -0.6, -0.5]
8 y = 2 # VD2: y = [-1, 2]
9 # VD3: y = [1, 0, -2]
10 for n in range(len(X) - 1):
11     h = X[n+1] - X[n]
12     k1 = h * f(X[n], y)
13     k2 = h * f(X[n] + h/2, y + k1/2)
14     k3 = h * f(X[n] + h/2, y + k2/2)
15     k4 = h * f(X[n] + h, y + k3)
16     y = y + (k1 + 2*k2 + 2*k3 + k4)/6
17 print(y)

```

Mã 6:

```

1 X = [1, 1.3, 1.7, 2.]
2 Y = [3.5, 4., 4.6, 5.2]

3 from sympy import *
4 x = symbols('x')
5 cs = [1 + 0*x, x, log(x)]

```

```

6 V = [ [cs_i.subs(x, X_k) for X_k in X] for cs_i in cs ]
7 import numpy as np
8 A = [ [np.dot(V_i, V_j) for V_j in V] for V_i in V ]
9 b = [np.dot(Y, V_i) for V_i in V]
10 hs = np.linalg.solve(
11     np.array(A).astype(float),
12     np.array(b).astype(float) )
13 P = hs.dot(cs)
14 [P.subs(x, X_k) for X_k in X]
15 errors = [Y_k - P.subs(x, X_k) for (X_k, Y_k) in zip(X, Y)]
16 np.linalg.norm(np.array(errors).astype(float))

```

Mã 7:

```

1 from sympy import *
2 f = lambda x: sin(x)
3 a, b = 0, pi
4 import numpy as np
5 X = np.linspace(np.float32(0), np.float32(b), 11)
6 Y = [ f(x) for x in X ]
7 sum( [(X[i] - X[i-1]) * (Y[i] + Y[i-1]) / 2 for i in range(1, 11)]
8       )
9 plot(f(x).diff(x, 2), (x, a, b))
10 M2 = 1
11 N(M2 * (b-a)**3 / 12 / 10**2, 6)
12 sum( [(X[2*i] - X[2*i-2]) * (Y[2*i] + 4*Y[2*i-1] + Y[2*i-2]) / 6
13       for i in range(1, 6)] )
14 plot(f(x).diff(x, 4), (x, a, b))
15 M4 = 1

```

14 $N(M4 * (b-a)**5 / 180 / 10**4, 6)$

Mã 8:

```

1 X = [-0.7, 1.7, -4.9, 3.1, -1.3]
2 Y = [-2.9, -1.1, -2.9, 1.5, 0.8]
3 Z = [7.1, 5.8, -3.1, -1, -8.7]

4 from sympy import *
5 x, y = symbols('x y')
6 cs = [1 + 0*x, x, y]

7 V = [ [cs_i.subs({x: X_k, y: Y_k}) for (X_k, Y_k) in zip(X, Y)] for
        cs_i in cs ]

8 import numpy as np
9 A = [ [np.dot(V_i, V_j) for V_j in V] for V_i in V ]
10 b = [np.dot(Z, V_i) for V_i in V]

11 hs = np.linalg.solve(
12     np.array(A, dtype=float),
13     np.array(b, dtype=float) )

14 P = hs.dot(cs)

15 [P.subs({x: X_k, y: Y_k}) for (X_k, Y_k) in zip(X, Y)]
16 errors = [Z_k - P.subs({x: X_k, y: Y_k}) for (X_k, Y_k, Z_k) in zip
            (X, Y, Z)]
17 np.linalg.norm( np.array(errors, dtype=float) )

```

Mã 9:

```

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

2 from sympy import *
3 x = symbols('x')

4 plot(f(x), (x, 1, 4))

5 plot(abs(f(x).diff(x, 2)), (x, 1, 4))
6 M = 22.5

```

```

7 t = symbols('t')
8 df = lambda x: f(t).diff().subs(t, x)
9 m = min( abs(df(1)), abs(df(4)) )

10 x0 = 4. #  $x_{n-1}$ 
11 for _ in range(3):
12     x = N(x0 - f(x0) / df(x0), 6) #  $x_n$ 
13     ss = N(M / 2 / m * (x - x0)**2, 6) #  $\varepsilon_n$ 
14     x0 = x
15     print(x, ss)

```

Mã 10:

```

1 A = [[-15.4, 1, 6.3], [-4.2, 10.8, 3.3], [-2.4, 5.3, 15.9]]
2 b = [30, 25, -10]

3 m = lambda i, j: -A[i][j] / A[i][i] if i != j else 0
4 import numpy as np
5 B = np.array([ [ m(i, j) for j in range(3)] for i in range(3) ])
6 g = [ b[i] / A[i][i] for i in range(3) ]

```

Mã 11:

```

1 X = [-1, 0, 1, 2]
2 Y = [4, 3, 2, 7]

3 def L(i, x):
4     prod = 1
5     for j in range(4):
6         if j != i:
7             prod *= (x - X[j]) / (X[i] - X[j])
8     return prod

9 from sympy import *
10 x = symbols('x')
11 L(0, x)
12 L(0, x).expand()

13 P = 0
14 for i in range(4):
15     P += Y[i] * L(i, x)

```

16 P.expand()

Mã 12:

```
1 from sympy import *
2 g = lambda x: root(x**2 + 3, 3)
3 x = symbols('x')
4 plot(g(x), (x, 1, 4))
5 plot(abs(g(x).diff()), (x, 1, 4))
6 q = 0.38
7 x0 = 2.5 #  $x_{n-1}$ 
8 for _ in range(3):
9     x = N(g(x0), 6) #  $x_n$ 
10    ss = N(q / (1-q) * abs(x - x0), 6) #  $\varepsilon_n$ 
11    x0 = x
12    print(x, ss)
```

Mã 13: