TÀI LIỆU ĐƯỢC SỬ DỤNG CHO MÔN THI TOÁN TOÁN HỌC TÍNH TOÁN

Họ tên : **Nguyễn Hồng An**

MSSV : **0000268**

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

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

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


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


t = symbols('t')
df = lambda x: f(t).diff().subs(t, x)

m = min( abs(df(1)), abs(df(4)) )


x0 = 4.                              # x_{n-1}
for _ in range(3):
    x = N(x0 - f(x0) / df(x0), 6)    # x_n
    ss = N(M / 2 / m * (x - x0)**2, 6) # ε_n
    x0 = x
    print(x, ss)
```

Mã 1:

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

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

g = [ b[i] / A[i][i] for i in range(3) ]
```

Mã 2:

```
import numpy as np
B = np.array( [[-0.21, -0.28,  0.05],
```

```
                [ 0.19,  0.01, -0.26],
                [ 0.39, -0.12, -0.06]] )
g = [-0.9, 3.8, -2.9]

q = np.linalg.norm(B, np.inf)
q

x0 = [0, 2, -1]         # x^(k-1)
for _ in range(3):
    x = B.dot(x0) + g   # x^(k)
    ss = q / (1-q) * np.linalg.norm(x - x0, np.inf)  # ε_k
    x0 = x
    print(x, ss)

x = [0, 2, -1]
for _ in range(4):
    for i in range(3):
        x[i] = B[i].dot(x) + g[i]  # x_i^(k)
    print(x)
```

Mã 3:

```
from sympy import *

g = lambda x: root(x**2 + 3, 3)

x = symbols('x')

plot(g(x), (x, 1, 4))

plot(abs(g(x).diff()), (x, 1, 4))
q = 0.38

x0 = 2.5                               # x_{n-1}
for _ in range(3):
    x = N(g(x0), 6)                    # x_n
    ss = N(q / (1-q) * abs(x - x0), 6)  # ε_n
    x0 = x
    print(x, ss)
```

Mã 4:

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

```python
f(0), f(2)

a, b = 0, 2
for _ in range(5):
    c = (a + b) / 2
    if f(c) == 0:
        print(f'Nghiệm đúng: {c}')
        break
    elif f(a) * f(c) < 0:
        b = c
    else:
        a = c
    ss = b - a     # ε_n
    print(a, b)   # a_n, b_n
```

Mã 5:

```python
from sympy import *

f = lambda x: sin(x)

a, b = 0, pi

import numpy as np

X = np.linspace(np.float32(0), np.float32(b), 11)
Y = [ f(x) for x in X ]

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

plot(f(x).diff(x, 2), (x, a, b))
M2 = 1
N(M2 * (b-a)**3 / 12 / 10**2, 6)


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)] )

plot(f(x).diff(x, 4), (x, a, b))
M4 = 1
N(M4 * (b-a)**5 / 180 / 10**4, 6)
```

Mã 6:

```python
                              # VD2, 3: import numpy as np
f = lambda x, y: y - x        # VD2: f = lambda x, y: np.array([x * y[0] - y[1], y[0] + y[1] - 1])
                              # VD3: f = lambda x, y: np.array([y[1], y[2], x * y[2] - y[0]])
X = [0, 0.2, 0.3, 0.5]        # VD2: X = [1, 1.1, 1.3, 1.5]
                              # VD3: X = [-1, -0.8, -0.6, -0.5]

y = 2                         # VD2: y = [-1, 2]
                              # VD3: y = [1, 0, -2]
for n in range(3):
  h = X[n+1] - X[n]
  y = y + h * f(X[n], y)
  print(y)
```

Mã 7:

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

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

V = [ [cs_i.subs(x, X_k) for X_k in X] for cs_i in cs ]

import numpy as np
A = [ [np.dot(V_i, V_j) for V_j in V] for V_i in V ]
b = [np.dot(Y, V_i) for V_i in V]

hs = np.linalg.solve(
    np.array(A).astype(float),
    np.array(b).astype(float) )

P = hs.dot(cs)

[P.subs(x, X_k) for X_k in X]
errors = [Y_k - P.subs(x, X_k) for (X_k, Y_k) in zip(X, Y)]
np.linalg.norm(np.array(errors).astype(float))
```

Mã 8:

```python
X = [-0.7, 1.7, -4.9, 3.1, -1.3]
Y = [-2.9, -1.1, -2.9, 1.5, 0.8]
```

```
Z = [7.1, 5.8, -3.1, -1, -8.7]

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

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

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

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

P = hs.dot(cs)

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

Mã 9:

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

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

from sympy import *
x = symbols('x')
L(0, x)
L(0, x).expand()

P = 0
for i in range(4):
```

```
    P += Y[i] * L(i, x)

P.expand()
```

Mã 10:

```
d = lambda k, i: Y[i] if k == 0 else d(k-1, i+1) - d(k-1, i)

[ [d(k, i) for i in range(4-k)] for k in range(4) ]

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

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

P

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

Mã 11:

```
                     # VD2, 3: import numpy as np
f = lambda x, y: y - x   # VD2: f = lambda x, y: np.array([x * y[0] - y[1],
   y[0] + y[1] - 1])
                     # VD3: f = lambda x, y: np.array([y[1], y[2], x *
   y[2] - y[0]])
X = [0, 0.2, 0.3, 0.5]   # VD2: X = [1, 1.1, 1.3, 1.5]
                     # VD3: X = [-1, -0.8, -0.6, -0.5]

y = 2                    # VD2: y = [-1, 2]
                     # VD3: y = [1, 0, -2]
for n in range(len(X) - 1):
    h = X[n+1] - X[n]
    k1 = h * f(X[n]      , y)
    k2 = h * f(X[n] + h/2, y + k1/2)
    k3 = h * f(X[n] + h/2, y + k2/2)
    k4 = h * f(X[n] + h  , y + k3)
    y = y + (k1 + 2*k2 + 2*k3 + k4)/6
```

```
    print(y)
```

Mã 12:

```
X = ...
Y = ...

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


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


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

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 13: