Toán học tính toán: Python buổi 5/10

4 Bài toán giá trị ban đầu của phương trình vi phân thường

4.1 Phương pháp Picard

```
1 f = lambda x, y: y - x

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

4 x0, y0 = 0, 2

5 # Cách 1: đệ quy
6 def y(n, x):
7   if n == 0:
8     return y0
9   return y0 + f(t, y(n-1, t)).integrate((t, x0, x))

10 y(1, x)

11 # Cách 2: lặp
12 y = y0 + 0*x   # biểu thức phải xuất hiện biến mới dùng được cú pháp thay thế (xem dòng 14)
13 for _ in range(3):
14   y = y.subs(x, t)
15   y = y0 + f(t, y).integrate((t, x0, x))
16   display(y)
```

Mã 26: Phương pháp Picard: Ví dụ 1

```
f = lambda x, y: [x * y[0] - y[1], y[0] + y[1] - 1]
f(0, [1, 2])

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

import numpy as np

x0 = 1
y0 = np.array([-1, 2]) + 0*x

# Cách 1: đệ quy
def y(n, x):
  if n == 0:
    return y0
  return y0 + [fi.integrate((t, x0, x)) for fi in f(t, y(n-1, t))]

y(1, x)

# Cách 2: lặp
y = y0.copy()   # không được gán y = y0, vì khi đó thay đổi y sẽ làm thay đổi y0
for i in range(2):
  y = [yi.subs(x, t) for yi in y]
  y = y0 + [fi.integrate((t, x0, x)) for fi in f(t, y)]
```

```
    display(y)
```

Mã 27: Phương pháp Picard: Ví dụ 2

```
f = lambda x, y: [y[1], y[2], x * y[2] - y[0]]
f(0, [1, 2, 3])

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

import numpy as np

x0 = -1
y0 = np.array([1, 0, -2]) + 0*x

# Cách 1: đệ quy
def y(n, x):
    if n == 0:
        return y0
    return y0 + [fi.integrate((t, x0, x)) for fi in f(t, y(n-1, t))]

y(1, x)

# Cách 2: lặp
y = y0.copy()
for i in range(2):
    y = [yi.subs(x, t) for yi in y]
    y = y0 + [fi.integrate((t, x0, x)) for fi in f(t, y)]
    display(y)
```

Mã 28: Phương pháp Picard: Ví dụ 3

4.2 Phương pháp Taylor

```
from sympy import *
x = symbols('x')
y = symbols('y', cls=Function)

P = 2
for k in range(1, 4):
    d = y(x).diff(x, k)
    for _ in range(k):
        d = d.subs(y(x).diff(), y(x) - x).simplify()   # mỗi bước lặp hạ được 1 cấp đạo hàm
    d = d.subs({x: 0, y(x): 2})
    P += d / factorial(k) * (x - 0)**k                # y_k(x)

P
```

Mã 29: Phương pháp Taylor: Ví dụ 1

```
from sympy import *
x = symbols('x')
y, z = symbols('y z', cls=Function)

P = 2
```

```
for k in range(1, 4):
  d = y(x).diff(x, k)   # d = z(x).diff(x, k)
  for _ in range(k):
    d = d.subs({y(x).diff(): x * y(x) - z(x), z(x).diff(): y(x) + z(x) - 1}).simplify()
  d = d.subs({x: 0, y(x): 2})
  P += d / factorial(k) * (x - 0)**k

P
```

Mã 30: Phương pháp Taylor: Ví dụ 2

```
from sympy import *
x = symbols('x')
y = symbols('y', cls=Function)

P = 1 + 0*(x+1) + (-2) / factorial(2) * (x+1)**2
for k in range(3, 6):
  d = y(x).diff(x, k)
  for _ in range(k - 2): # 2 = cấp của phương trình vi phân - 1
    d = d.subs({y(x).diff(x, 3): x * y(x).diff(x, 2) - y(x)}).simplify()
  d = d.subs({x: -1, y(x): 1, y(x).diff(): 0, y(x).diff(x, 2): -2})
  P += d / factorial(k) * (x + 1)**k

P
```

Mã 31: Phương pháp Taylor: Ví dụ 3

4.3 Phương pháp Euler

```
                             # 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 32: Phương pháp Euler

4.4 Phương pháp Runge-Kutta RK4

```
                             # 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]
```

```python
                        # 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 33: Phương pháp RK4