实验报告三

李平治 PB19071501

2022年4月24日

I. 题目及其运行结果

1.1 题目

1.（Page186, Project10）用 Newton 迭代法求解非线性方程组

$$\begin{cases} f(x) = x^2 + y^2 - 1 = 0 \\ g(x) = x^3 - y = 0 \end{cases} \tag{1}$$

取 $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.6 \end{pmatrix}$，误差控制 $\max(|\Delta x_k|, |\Delta y_k|) \le 10^{-5}$.

输入：初始点 $(x_0, y_0) = (0.8, 0.6)$，精度控制 e，定义函数 $f(x), g(x)$.

输出：迭代次数 k，第 k 步的迭代解 (x_k, y_k).

2.（Page187, Project21(1)）用二阶 Rouge-Kutta 公式求解常微分方程组初值问题

$$\begin{cases} y'(x) = f(x, y) \\ y(a) = y_0 \end{cases}, a \le x \le b \tag{2}$$

(1) 求解初值问题

$$\begin{cases} y'(x) = y\sin\pi x \\ y(0) = 1 \end{cases} \tag{3}$$

输入：区间剖分点数 n，区间端点 a, b，定义函数 $y'(x) = f(x, y)$.

输出：y_k，$k = 1, 2, ..., n$.

3. （Page187, Project22）用改进的 Euler 公式求解常微分方程组初值问题计算公式：

$$\begin{pmatrix} \bar{y}_{n+1} \\ \bar{z}_{n+1} \end{pmatrix} = \begin{pmatrix} y_n \\ z_n \end{pmatrix} + h \begin{pmatrix} f(x_n, y_n, z_n) \\ g(x_n, y_n, z_n) \end{pmatrix} \tag{4}$$

$$\begin{pmatrix} y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} y_n \\ z_n \end{pmatrix} + \frac{h}{2} \left[\begin{pmatrix} f(x_n, y_n, z_n) \\ g(x_n, y_n, z_n) \end{pmatrix} + \begin{pmatrix} f(\bar{x}_{n+1}, \bar{y}_{n+1}, \bar{z}_{n+1}) \\ g(\bar{x}_{n+1}, \bar{y}_{n+1}, \bar{z}_{n+1}) \end{pmatrix} \right] \tag{5}$$

输入：区间剖分点数 N，区间端点 a, b，定义函数

$$y'(x) = f(x, y, z), z'(x) = g(x, y, z) \tag{6}$$

输出：(y_k, z_k)，$k = 1, 2, ..., N$

利用上述方法，求解课本 Page156 例题 7.7：

$$\begin{cases} \frac{\mathrm{d}u}{\mathrm{d}t} = 0.09u(1 - \frac{u}{20}) - 0.45uv \\ \frac{\mathrm{d}v}{\mathrm{d}t} = 0.06v(1 - \frac{v}{15}) - 0.001uv \\ u(0) = 1.6 \\ v(0) = 1.2 \end{cases} \tag{7}$$

1.2 结果

1.

取$x_0 = 0.8, y_0 = 0.6, epsilon = 1e - 5$

```
Step 1 : x = 0.8270491803278689 y = 0.5639344262295083
Step 2 : x = 0.8260323731676462 y = 0.5636236767037873
Step 3 : x = 0.8260313576552345 y = 0.5636241621608473
```

2.

取$n = 10, a = 0, b = 1$

```
x = 0.0 , y = 1.0
x = 0.1 , y = 1.01545
x = 0.2 , y = 1.06191
x = 0.3 , y = 1.13859
x = 0.4 , y = 1.24318
x = 0.5 , y = 1.37036
x = 0.6 , y = 1.51056
x = 0.7 , y = 1.64931
x = 0.8 , y = 1.76842
x = 0.9 , y = 1.84932
x = 1.0 , y = 1.87789
```

3.

取$N=3,a=1,b=4$

```
t = 1 , u = 1.6 , v = 1.2
t = 2 , u = 1.024566 , v = 1.266344
t = 3 , u = 0.640912 , v = 1.336601
t = 4 , u = 0.391211 , v = 1.410773
```

II. 使用算法

1. Newton迭代法求解线性方程组

对 $f(x,y),g(x,y)$ 在 (x_0,y_0) 作二元 Taylor 展开，并取其线性部分，得到方程组

$$\begin{cases} f(x,y)\approx f(x_0,y_0)+(x-x_0)\dfrac{\partial f(x_0,y_0)}{\partial x}+(y-y_0)\dfrac{\partial f(x_0,y_0)}{\partial y}=0 \\ g(x,y)\approx g(x_0,y_0)+(x-x_0)\dfrac{\partial g(x_0,y_0)}{\partial x}+(y-y_0)\dfrac{\partial g(x_0,y_0)}{\partial y}=0 \end{cases}$$

令 $x-x_0=\Delta x, y-y_0=\Delta y$, 则有

$$\begin{cases} \Delta x\dfrac{\partial f_1(x_0,y_0)}{\partial x}+\Delta y\dfrac{\partial f_1(x_0,y_0)}{\partial y}=-f_1(x_0,y_0) \\ \Delta x\dfrac{\partial f_2(x_0,y_0)}{\partial x}+\Delta y\dfrac{\partial f_2(x_0,y_0)}{\partial y}=-f_2(x_0,y_0) \end{cases} \tag{3.7}$$

如果

$$\det(J(x_0,y_0))=\left|\begin{matrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_1}{\partial y} \end{matrix}\right|_{(x_0,y_0)} \neq 0$$

解出 $\Delta x,\Delta y$

$$w_1=w_0+\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}=\begin{pmatrix} x_0+\Delta x \\ y_0+\Delta y \end{pmatrix}=\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

再列出方程组

$$\begin{cases} \dfrac{\partial f(x_1,y_1)}{\partial x}(x-x_1)+\dfrac{\partial f(x_1,y_1)}{\partial y}(y-y_1)=-f(x_1,y_1) \\ \dfrac{\partial g(x_1,y_1)}{\partial x}(x-x_1)+\dfrac{\partial g(x_1,y_1)}{\partial y}(y-y_1)=-g(x_1,y_1) \end{cases}$$

解出

$$\Delta x = x - x_1, \quad \Delta y = y - y_1$$

$$w_2 = \begin{pmatrix} x_1 + \Delta x \\ y_1 + \Delta x \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$$

继续做下去, 每一次迭代都是解一个类似式 (3.7) 的方程组

$$J(x_k, y_k) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} -f(x_k, y_k) \\ -g(x_k, y_k) \end{pmatrix}$$

$$\Delta x = x_{k+1} - x_k, \quad \Delta y = y_{k+1} - y_k$$

即

$$x_{k+1} = x_k + \Delta x, \quad y_{k+1} = y_k + \Delta y$$

直到 $\max(|\Delta x|, |\Delta y|) < \epsilon$ 为止.

基本上是Newton迭代求解非线性方程的拓展。

2. 二阶Runge-Kutta方法求解常微分方程组初值问题

Runge-Kutta 方法通常写成如下形式,

$$\begin{cases} y_{n+1} = y_n + h(c_1 k_1 + c_2 k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + ah, y_n + bhk_1) \end{cases} \tag{7.13}$$

若取 $c_1 = \frac{1}{2}, c_2 = \frac{1}{2}, a = 1, b = 1$, 得到式 (7.14) 的二阶 Runge-Kutta 公式:

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + h, y_n + hk_1) \end{cases} \tag{7.14}$$

比较式 (7.12) 得到 c_1, c_2, a, b 满足

$$\begin{cases} c_1 + c_2 = 1 \\ 2c_2 a = 1 \\ 2c_2 b = 1 \end{cases}$$

书中给出了两组系数，这里采用了第一种。

3.改进的Euler方法求解ODE初值问题

也可以用显式的 Euler 公式和隐式的梯形公式组成的预估-校正公式:

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(x_n, y_n), \\ y_{n+1} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})\right] \end{cases} \tag{7.8}$$

式 (7.8) 也称为改进的 Euler 公式, 它可合并写成

$$y_{n+1} = y_n + \frac{h}{2}(f(x_n, y_n) + f(x_{n+1}, y_n + hf(x_n, y_n)))$$

三元下的拓展为:

$$\begin{pmatrix} \bar{y}_{n+1} \\ \bar{z}_{n+1} \end{pmatrix} = \begin{pmatrix} y_n \\ z_n \end{pmatrix} + h \begin{pmatrix} f(x_n, y_n, z_n) \\ g(x_n, y_n, z_n) \end{pmatrix} \tag{4}$$

$$\begin{pmatrix} y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} y_n \\ z_n \end{pmatrix} + \frac{h}{2} \left[\begin{pmatrix} f(x_n, y_n, z_n) \\ g(x_n, y_n, z_n) \end{pmatrix} + \begin{pmatrix} f(\bar{x}_{n+1}, \bar{y}_{n+1}, \bar{z}_{n+1}) \\ g(\bar{x}_{n+1}, \bar{y}_{n+1}, \bar{z}_{n+1}) \end{pmatrix} \right] \tag{5}$$

改进即使用显示公式算出初始值，再用隐式公式进行一步迭代（a.k.a. 预估-校正）。

III. 结果分析

1.

迭代3次后满足精度，将最终结果带入计算，发现已非常准确

2.

用书中算法给出的两组系数相互对照，发现结果较为准确

3.

与书中例题解答对照，表面结果在误差范围内一致

附录

本次实验的Python代码如下

1.

```python
import numpy as np


def f_f(x, y):
    return x**2 + y**2 - 1


def f_g(x, y):
```

```python
    return x**3 - y


def Jaccobi(x, y):
    return np.array([[2*x, 2*y], [3*x**2, -1]])


def newton_iter(x0, y0, eps):
    x = x0
    y = y0
    k = 1
    while True:
        J = Jaccobi(x, y)
        J_inv = np.linalg.inv(J)
        f = np.array([f_f(x, y), f_g(x, y)])
        delta = -np.dot(J_inv, f)
        x += delta[0]
        y += delta[1]
        print('Step', k, ':', 'x =', x, 'y =', y)
        k += 1
        if np.linalg.norm(delta) < eps:
            break
    return x, y


x, y = newton_iter(0.8, 0.6, 1e-5)

```

2.

```python
import numpy as np


def y_drv(x, y):
    return np.sin(np.pi * x) * y


def rouge_kutta_2(n, a, b):
    h = (b - a) / n
    x = np.linspace(a, b, n + 1)
    y = np.zeros(n + 1)
    y[0] = 1
    for i in range(n):
        k1 = y_drv(x[i], y[i])
        k2 = y_drv(x[i] + h, y[i] + h * k1)
        y[i + 1] = y[i] + h * (k1 + k2) / 2
    return x, y

```

```python

x, y = rouge_kutta_2(10, 0, 1)
for i, j in zip(x, y):
    print("x =", round(i, 2), ", y =", round(j, 5))

```

3.

```python
import numpy as np

def y_drv(x, y, z):
    return 0.09 * y * (1 - y / 20) - 0.45 * y * z

def z_drv(x, y, z):
    return 0.06 * z * (1 - z / 15) - 0.001 * y * z

def euler_ode(N, a, b):
    h = (b - a) / N
    x = np.linspace(a, b, N + 1, dtype=np.int16)
    y = np.zeros(N + 1)
    z = np.zeros(N + 1)
    y[0] = 1.6
    z[0] = 1.2
    for i in range(N):
        _y = y[i] + h * y_drv(x[i], y[i], z[i])
        _z = z[i] + h * z_drv(x[i], y[i], z[i])
        y[i + 1] = y[i] + h / 2 * (y_drv(x[i], y[i], z[i]) + y_drv(x[i + 1], _y, _z))
        z[i + 1] = z[i] + h / 2 * (z_drv(x[i], y[i], z[i]) + z_drv(x[i + 1], _y, _z))
    return x, y, z


t, u, v = euler_ode(3, 1, 4)
for i in range(len(t)):
    print("t =", t[i], ", u =", round(u[i], 6), ", v =", round(v[i], 6))

```