hw1

20989977 Zhang Mingtao

2024/2/9

0.

```
library(reticulate)
```

1.

```
import numpy as np
# 1.(2)

# 定义函数及其导数
def f(x):
    return 4*x*np.sin(x) - 4*np.sin(x)**2 - x**2

def f_prime(x):
    return 4*np.sin(x) + 4*x*np.cos(x) - 8*np.sin(x)*np.cos(x) - 2*x

# (i)
# 牛顿迭代法求解函数零点
def newton_method(x0, tol=1e-5, max_iter=1000):
    x = x0
    iter_count = 0

    while iter_count < max_iter:
        iter_count += 1
        delta_x = f(x) / f_prime(x)
        x -= delta_x
        # print(x)

        if np.abs(delta_x) < tol:
            break

    if iter_count == max_iter:
        print("达到最大迭代次数但未收敛到解")

    return x

# 初始值
x0 = 1.5

# 求解函数的零点
zero_point1 = newton_method(x0)

print("Newton's method: 函数的零点解为:", zero_point1)

# (ii)
# 割线法求解函数零点
```

```
## Newton’s method: 函数的零点解为: 1.8954885493408413
```

```
def secant_method(x0, x1, tol=1e-5, max_iter=100):
    x = x1
    x_prev = x0
    iter_count = 0

    while iter_count < max_iter:
        iter_count += 1
        delta_x = f(x) * (x - x_prev) / (f(x) - f(x_prev))
        x_prev = x
        x -= delta_x
        # print(x)

        if np.abs(delta_x) < tol:
            break

    if iter_count == max_iter:
        print("达到最大迭代次数但未收敛到解")

    return x

# 初始值
x0 = 1.5
x1 = 2.0

# 求解函数的零点
zero_point2 = secant_method(x0, x1)

print("the secant method: 函数的零点解为:", zero_point2)
```

```
## the secant method: 函数的零点解为: 1.895509296308531
```

2.

```
# 2.(2)

# 定义函数
def F(x):
    x1, x2 = x
    f1 = 1 + x1**2 - 4*x2**2 + np.exp(x1)*np.cos(2*x2)
    f2 = 4*x1*x2 + np.exp(x1)*np.sin(2*x2)
    return np.array([f1, f2])

# 定义雅可比矩阵
def J(x):
    x1, x2 = x
    df1_dx1 = 2*x1 + np.exp(x1)*np.cos(2*x2)
    df1_dx2 = -8*x2 - 2*np.exp(x1)*np.sin(2*x2)
    df2_dx1 = 4*x2 + np.exp(x1)*np.sin(2*x2)
    df2_dx2 = 4*x1 + 2*np.exp(x1)*np.cos(2*x2)
    return np.array([[df1_dx1, df1_dx2], [df2_dx1, df2_dx2]])

# 牛顿迭代法求解非线性方程组
def newton_method(x0, max_iter=5):
    x = x0

    for i in range(max_iter):
        delta_x = np.linalg.solve(J(x), -F(x))
        x += delta_x

    return x

# 初始值
x0 = np.array([-1, 2], dtype=np.float64)

# 求解非线性方程组
solution = newton_method(x0, max_iter=5)

print("Newton's method: 迭代结果:", solution)
```

```
## Newton's method: 迭代结果: [-0.29316269  0.58632991]
```

3.

```
# 3
def P(x):
    return (-5*x**2+3*x+26)/6

P(1)
```

```
## 4.0
```

```
P(2)
```

```
## 2.0
```

```
P(-1)
```

```
## 3.0
```