科学计算引论作业(四)

谢悦晋　U202210333

Oct 15th, 2023

3.10 (实验题) 给定线性方程组

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ -2 & 3 & 4 & 5 & 6 \\ -3 & -4 & 5 & 6 & 7 \\ -4 & -5 & -6 & 7 & 8 \\ -5 & -6 & -7 & -8 & 9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 55 \\ 66 \\ 63 \\ 36 \\ -25 \end{pmatrix}.$$

试以 $\|X^{(k+1)} - X^{(k)}\|_\infty < 10^{-5}$ 作为终止准则,分别利用 Jacobi 迭代法、Gauss-Seidel 迭代法及具有最佳松弛因子的 JOR 迭代方法和 SOR 迭代法求解该方程组,并比较这些方法的计算时间.

解：Jacobi 迭代法、Gauss-Seidel 迭代法是无法收敛的，而JOR 迭代方法和 SOR 迭代法可以收敛，结果如下：

```python
from tqdm import tqdm
import numpy as np
x = np.ones(5)
x_new = np.zeros(5)
b = np.array([55, 66, 63, 36, -25])
A = np.array([[1, 2, 3, 4, 5 ], [-2, 3, 4, 5, 6], [-3, -4, 5, 6, 7], [-4, -5, -6, 7, 8], [-5, -6, -7, -8, 9]])
num_iter = 100
epsilon = 10e-5
diff = np.zeros(5)

for i in tqdm(range(num_iter)):
    for j in range(5):
        x_new[j] = (b[j] - np.dot(A[j,:], x) + A[j,j]*x[j])/A[j,j]
        diff = np.abs(x_new - x)
    if np.max(diff) < epsilon:
        print(f"Converged after {i+1} iterations: {x}")
        break
    x = x_new.copy()
    print(x) # 输出变量x的值作为方程Ax=b的解
```

在 2023.10.15 20:28:06 于 17ms内执行

```
100%|██████████| 100/100 [00:00<00:00, 16718.37it/s]
[-1.82074255e+58 -7.91298497e+57 -3.94239603e+57 -6.29439127e+56
  2.12169825e+57]
[ 1.95624233e+58 -1.00760869e+58 -1.94698939e+58 -2.18603699e+58
 -1.90163692e+58]
```

```python
import numpy as np
x = np.array([1.5, 2.1, 3, 4, 5])
x_new = np.zeros(5)
b = np.array([55, 66, 63, 36, -25])
A = np.array([[1, 2, 3, 4, 5 ], [-2, 3, 4, 5, 6], [-3, -4, 5, 6, 7], [-4, -5, -6, 7, 8], [-5, -6, -7, -8, 9]])
num_iter = 100
max_diff = 1e-5
diff = np.zeros(5)

for i in tqdm(range(num_iter)):
    for j in range(5):
        x_old = x.copy()
        x[j] = (b[j] - np.dot(A[j,:j], x[:j]) - np.dot(A[j,(j+1):], x[(j+1):]))/A[j,j]
        diff[j] = np.abs(x[j] - x_old[j])
    if np.max(diff) < max_diff:
        break
    print(x)
```

在 2023.10.15 20:29:42 于 11ms内执行

```
100%|██████████| 100/100 [00:00<00:00, 16720.37it/s]
[-5.40337027e+160 -5.71176547e+160 -9.10899456e+160 -1.56894142e+161
 -2.78406355e+161]
[2.40711349e+162 2.54449853e+162 4.05790879e+162 6.98937860e+162
 1.24025499e+163]
```

```python
from tqdm import tqdm
import numpy as np
x = np.ones(5)
x_new = np.zeros(5)
b = np.array([55, 66, 63, 36, -25])
A = np.array([[1, 2, 3, 4, 5 ], [-2, 3, 4, 5, 6], [-3, -4, 5, 6, 7], [-4, -5, -6, 7, 8], [-5, -6, -7, -8, 9]])
num_iter = 1000
epsilon = 10e-5
diff = np.zeros(5)
omega = 0.05

for i in tqdm(range(num_iter)):
    for j in range(5):
        x_new[j] = x[j] + omega * (b[j] - np.dot(A[j,:], x))/A[j,j]
        diff = np.abs(x_new - x)
    if np.max(diff) < epsilon:
        print(f"Converged after {i+1} iterations: {x}")
        break
    x = x_new.copy()
    print(x)
```

在 2023.10.15 20:31:50 于 18ms内执行

```
28%|██▊       | 277/1000 [00:00<00:00, 19159.02it/s]
[1.00223327 2.0010367  3.00056119 4.00015122 4.99979577]
[1.00195458 2.00102971 3.00064682 4.00028021 4.99993112]
[1.00161803 2.0009838  3.00070231 4.00039048 5.00006079]
```

```python
import numpy as np
x = np.zeros(5)
x_new = np.zeros(5)
b = np.array([55, 66, 63, 36, -25])
A = np.array([[1, 2, 3, 4, 5 ], [-2, 3, 4, 5, 6], [-3, -4, 5, 6, 7], [-4, -5, -6, 7, 8], [-5, -6, -7, -8, 9]])
num_iter = 10000
max_diff = 1e-5
diff = np.zeros(5)
omega = 0.333

for i in tqdm(range(num_iter)):
    for j in range(5):
        x_old = x.copy()
        x[j] = x[j] + omega * (b[j] - np.dot(A[j,:j], x[:j]) - np.dot(A[j,j:], x[j:]))/A[j,j]
        diff[j] = np.abs(x[j] - x_old[j])
    if np.max(diff) < max_diff:
        print(f"Converged after {i+1} iterations: {x}")
        break
    print(x)
```

在 2023.10.15 20:34:02 于 8ms内执行

```
  0%|          | 32/10000 [00:00<00:00, 16037.49it/s]
[0.9999817  1.99997333 2.99998192 4.00001575 4.99998607]
[1.00002582 1.99999651 2.99999237 4.00001771 4.99999798]
[1.00000695 1.99999412 2.9999886  4.00000925 4.99999842]
Converged after 33 iterations: [1.00001026 1.99999934 2.99999131 4.00000609 5.00000024]
```

4.1 给定函数 $f(x)=x\exp(x)(1+\exp(x))$ 及插值节点 $x_0=1.00, x_1=1.02, x_2=1.04, x_3=1.06$,试构造 3 次 Lagrange 插值多项式计算 $f(1.03)$ 的逼近值,并导出其误差估计.

解：

Lagrange插值多项式：

$$L_3(x)=\sum_{i=0}^{3} f(x_i)l_i(x), \quad l_i(x)=\prod_{j=0,j\neq i}^{3}\frac{(x-x_j)}{x_i-x_j}$$

结果如下：

```python
def Lagrange(arr_x, arr_y, _x):
    l = [0 for j in range(len(arr_x))]
    result = 0
    for i in range(0, len(arr_x)):
        denominator = 1
        molecular = 1
        for j in range(0, len(arr_x)):
            if i != j:
                denominator = denominator * (arr_x[i] - arr_x[j])
                molecular = molecular * (_x - arr_x[j])
        l[i] = molecular / denominator
        result = result + l[i] * arr_y[i]
    return result
def function(x):
    return x * np.exp(x) * (1 + np.exp(x))
x_arr = np.array([1.00, 1.02, 1.04, 1.06])
y_arr=[]
for i in range(len(x_arr)):
    y_arr.append(function(x_arr[i]))
y = Lagrange(x_arr, y_arr, 1.03)
y, function(1.03), y - function(1.03)
```

在 2023.10.15 21:28:17 于 4ms内执行

```
(10.966445234880203, 10.966446714368054, -1.4794878513413323e-06)
```