# 同為大學

# 计算方法实验报告(三)



学院机械与能源工程学院专业机械设计制造及其自动化学号1852951姓名李腾指导教师李梦茹、陈茂林完成日期2020年11月27日

# 目录

→,	Jacob	i 迭代法和 Gauss-Seidel 迭代法	. 3
	1.1	实验代码	.3
	1.2	参数生成截图	.4
_,	非线性	上方程的解法(二分法)	.5
	2.1 实	<b>以验代码</b>	.5
	2.2参	数生成截图	.6
三、	非线性	上方程的解法(Newton 迭代法)	.6
	3.1 实	<b>以验代码</b>	.7
	3 2 参	<b>数</b> 生成截图	7

# 一、Jacobi 迭代法和 Gauss-Seidel 迭代法

对下列方程组,分别用 Jacobi 迭代法和 Gauss-Seidel 迭代法迭代求解,并观察是否收敛。

$$\begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = randi (20,3,1)$$

#### 1.1 实验代码

#### Jacobi 迭代函数

```
Gauss seidel.m × jacobi.m × test1.m × +
        % The jacobi iterative function
1
2
       % precision is the deviation between two successive iterations
3
       % A is the coefficient matrix
       % x0 is the initial value of x
       % b is the constant matrix
5
      [ function [y, iteration_count] = jacobi(A, b, x0, precision)
6
       D=diag(diag(A));
       U=-triu(A, 1);
9 —
       L=-tril(A,-1);
       B=D\(L+U);
10 —
       C=D\b;
11 -
12 -
       y=B*x0+C;
       iteration_count=1;
13 -
14 —
     while norm(y-x0)>precision
15 -
            x0=y;
16 -
            y=B*x0+C;
17 -
            iteration_count=iteration_count+1;
18 -
       - end
19 -
      L end
```

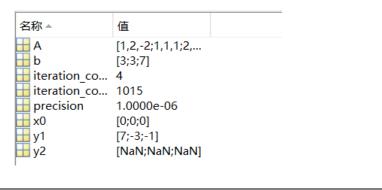
Gauss-Seidel 迭代函数

```
Gauss_seidel.m × jacobi.m × test1.m × +
        % The Gauss_seidel iterative function
1
2
        % precision is the deviation between two successive iterations
        % A is the coefficient matrix
3
        \% x0 is the initial value of x
4
5
        % b is the constant matrix
      ☐ function [y, iteration_count] = Gauss_seidel(A, b, x0, precision)
        D=diag(diag(A));
        U=-triu(A, 1);
9 —
        L=-tril(A,-1):
        G=(D-L)\setminus U;
10 -
        C=(D-L) \b;
11 -
12 -
        y=G*x0+C;
13 -
        iteration_count=1;
14 -
      while norm(y-x0)>precision
15 -
            x0=у;
16 -
            y=G*x0+C;
17 -
            iteration_count=iteration_count+1;
18 -
19 -
       ∟ end
```

#### 实验调用函数

```
Gauss seidel.m
                     jacobi.m × test1.m × +
1 —
       clear;
^2 ^-
       rand('seed', 1852951);
3
4 —
       A=[1,2,-2;1,1,1;2,2,1];
5 —
       b=randi (20, 3, 1);
       x0=[0;0;0];
6 —
       precision=le-6;
7 —
       [y1, iteration_count1] = jacobi(A, b, x0, precision)
8 —
       [y2, iteration_count2] Gauss_seidel (A, b, x0, precision)
9 —
```

#### 1.2 参数生成截图



```
y1 =
7
-3
-1

iteration_count1 =
4

y2 =

NaN
NaN
NaN
NaN
leration_count2 =

1015
```

# 二、非线性方程的解法(二分法)

用二分法求方程  $x^2 - x - a = 0$  的正根, 其中a = 0.5 + 1.5 rand(1),要求误差小于b = 0.03 + 0.03 rand(1)。

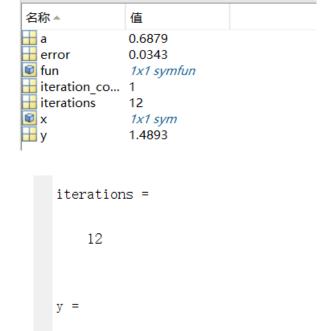
#### 2.1 实验代码

## 二分法迭代函数

```
dichotomy_solve.m × test2.m × +
       % The dichotomy iterative function
1
        \% x_upper is the upper limit of the interval
3
       % x down is the lower limit of the interval
        \% fun is the function corresponding to the equation
        % error is required error of solution
5
     function [iterations, y]=dichotomy_solve(x_upper, x_dowm, fun, error, iteration_count)
        x=(x_upper+x_dowm)/2;
 8 —
        f3=fun(x);
9 —
        f1=fun(x_upper);
10 —
        if(f1*f3<0)
11 -
            m=x-x\_upper;
12 —
            if(m>error)
13 —
                x_dowm=x;
14 —
                iteration_count=iteration_count+1;
15 —
                [iteration_count, y]=dichotomy_solve(x_upper, x_dowm, fun, error, iteration_count);
16 —
            else
17 —
18 —
            end
19 —
20 -
            m=x_dowm-x;
21 —
22 -
                x_upper=x;
23 —
                iteration_count=iteration_count+1;
24 -
                [iteration_count, y]=dichotomy_solve(x_upper, x_dowm, fun, error, iteration_count);
25 —
            e1se
26 —
27 —
            end
28 —
29 —
        iterations=iteration count:
30 —
```

#### 实验调用函数

## 2.2 参数生成截图



# 三、非线性方程的解法(Newton 迭代法)

对方程 $ax^b-e^x=0$ (其中,a=1+2rand(1),b=0.8+1.5rand(1) ),用Newton迭代法计算。

1.4893

#### 3.1 实验代码

#### Newton 迭代函数

```
newton.m × test3.m × +
        % The newton iterative function
 1
 2
        % f is the function corresponding to the equation
 3
        % f_diff is the first derivative of f
 4
        % max_iteration is the maximum number of iterations
 5
       % error is required error of solution
     ☐ function [iteration_count, y]=newton(x0, f, f_diff, max_iteration, error)
 7 —
       y=x0-f(x0)/f_diff(x0);
 8 —
        vpa(y, 5);
       iteration count=1;
10 - while abs(y-x0)>=error && iteration_count < max_iteration
11 —
           iteration_count=iteration_count+1;
12 -
            x0=у;
13 -
           y=vpa(x0-f(x0)/f_diff(x0), 6);
14 —
       end –
15 —
       l end
```

#### 实验调用函数

```
newton.m × test3.m × +
        clear;
1 -
        rand('seed', 1852951);
^{2} ^{-}
3
4 —
       syms x;
5 —
       a=1+2*rand(1);
       b=0.8+1.5*rand(1);
       f(x)=a*x^b-exp(x)+3;
7 —
       f_diff(x) = a*b*x^(b-1) - exp(x);
9 —
       error=0.001:
10 -
       x0=1;
11 -
       max_iteration=1000;
        [iteration_count, y] = newton(x0, f, f_diff, max_iteration, error)
12 -
```

## 3.2 参数生成截图

