重 庆 大 学

学 生 实 验 报 告

实验课程名称 数学实验

开课实验室 DS1401

组员 1 姓 名 马梓恒 学 号 20233124

组员 2 姓 名 周宏仰 学 号 20232647

组员 3 姓 名 李宇聪 学 号 20232137

开 课 时 间 2024 至 2025 学年第 一 学期

总 成 绩	

数 统 学 院 制

开课学院、实验室： 数统学院，DS401　　**实验时间 ：2024 年 9 月 22 日**

课程名称	数学实验	实验项目名　称	MATLAB 作图	实验项目类型				
				验证	演示	综合	设计	其他
指导教师	肖剑	成　绩			√			

题目 1

将方程 $x^5+5x^3-2x+1=0$ 改写成各种等价的形式进行迭代，观察迭代是否收敛。

程序

```
%初始化
clear
clc
x=1;
y=1;
z=1;
f=1;
tol=1e-6;

%以四种等价变换迭代 100 次
for k=1:100
    x1=(x^5+5*x^3+1)/2;
    y1=(-5*y^3+2*y-1)^(1/5);
    z1=-5/z+2*(z^3)-1/(z^4);
    f1=((-1*f^5+2*f-1)/5)^(1/3);

    %若迭代值为实数，差值就是代数差；若迭代值为虚数，差值为模的差
    if isreal(x1)
        cha1=x1-x;
    else
        cha1=abs(x1)-abs(x);
    end
    if isreal(y1)
        cha2=y1-y;
    else
        cha2=abs(y1)-abs(y);
    end
    if isreal(z1)
        cha3=z1-z;
    else
```

```
        cha3=abs(z1)-abs(z);
    end
    if isreal(f1)
        cha4=f1-f;
    else
        cha4=abs(f1)-abs(f);
    end
    x=x1;
    y=y1;
    z=z1;
    f=f1;
end

%输出迭代值与差值
fprintf('以 x=(x^5+5*x^3+1)/2 方式,迭代 100 次,值为%.7f,与上一次差值为%.7f\n',x,cha1)
fprintf('以 x=(-5*x^3+2*x-1)^(1/5)方式,迭代 100 次,值为%.7f,与上一次差值为%.7f\n',y,cha2)
fprintf('以 x=-5/x+2*(x^3)-1/(x^4)方式,迭代 100 次,值为%.7f,与上一次差值为%.7f\n',z,cha3)
fprintf('以 x=((-1*x^5+2*x-1)/5)^(1/3)方式,迭代 100 次,值为%.7f,与上一次差值为%.7f\n\n 敛散性判断：\n',f,cha4)

%判断是否收敛
if abs(cha1)<tol
    fprintf('以 x=(x^5+5*x^3+1)/2 方式迭代收敛\n')
else
    fprintf('以 x=(x^5+5*x^3+1)/2 方式迭代发散\n')
end
if abs(cha2)<tol
    fprintf('以 x=(-5*x^3+2*x-1)^(1/5)方式迭代收敛\n')
else
    fprintf('以 x=(-5*x^3+2*x-1)^(1/5)方式迭代发散\n')
end
if abs(cha3)<tol
    fprintf('以 x=-5/x+2*(x^3)-1/(x^4)方式迭代收敛\n')
else
    fprintf('以 x=-5/x+2*(x^3)-1/(x^4)方式迭代发散\n')
end
if abs(cha4)<tol
    fprintf('以 x=((-1*x^5+2*x-1)/5)^(1/3)方式迭代收敛\n')
else
    fprintf('以 x=((-1*x^5+2*x-1)/5)^(1/3)方式迭代发散\n')
end
```

结果

以 x=(x^5+5*x^3+1)/2 方式,迭代 100 次,值为 Inf,与上一次差值为 NaN
以 x=(-5*x^3+2*x-1)^(1/5)方式,迭代 100 次,值为 2.0162497,与上一次差值为 0.0000000

以 x=-5/x+2*(x^3)-1/(x^4)方式,迭代 100 次,值为-Inf,与上一次差值为 NaN
以 x=((-1*x^5+2*x-1)/5)^(1/3)方式,迭代 100 次,值为 0.4004135,与上一次差值为 0.0000000
敛散性判断:
以 x=(x^5+5*x^3+1)/2 方式迭代发散
以 x=(-5*x^3+2*x-1)^(1/5)方式迭代收敛
以 x=-5/x+2*(x^3)-1/(x^4)方式迭代发散
以 x=((-1*x^5+2*x-1)/5)^(1/3)方式迭代收敛
>>

分析

初始化:

设置初始值和误差容忍度。

初始化四个变量 x, y, z, f,用于存储迭代过程中的值。

迭代循环:

进入一个 for 循环,循环 100 次。

在每次循环中,根据四个不同的迭代公式计算新的 x1, y1, z1, f1。

判断新的迭代值是否为实数,如果是实数,则计算与上一次迭代值的差值;否则计算模的差值。

将新的迭代值赋值给 x, y, z, f,为下一次迭代做准备。

判断收敛性:

循环结束后,比较每次迭代的差值与设定的误差容忍度。

如果差值小于误差容忍度,则认为该迭代序列收敛。

输出每个迭代序列的最终值、差值以及是否收敛的结果。

题目 2

求解方程组 $\begin{cases} x+y^2=13 \\ \ln(2x+y)-x^y=-2 \end{cases}$,并验证初值是否对解有影响。

程序

```
function solve_equation_system()
    % 定义方程组,使用匿名函数
    equations = @(vars) [
        vars(1) + vars(2)^2 - 13;           % 第一个方程: x + y^2 = 13
        log(2*vars(1) + vars(2)) - vars(1)^vars(2) + 2  % 第二个方程: ln(2x + y) - x^y
= -2
    ];

    % 设置初值(可以更改来验证初值的影响)
    initial_guess = [1, 2];  % 初值 [x, y]

    % 使用 fsolve 求解方程组
    options = optimset('Display', 'iter');  % 显示迭代信息
    [solution, fval, exitflag] = fsolve(equations, initial_guess, options);
```

```
    % 显示结果
    if exitflag > 0
        disp('求解成功:');
        disp(['x = ', num2str(solution(1))]);
        disp(['y = ', num2str(solution(2))]);
    else
        disp('求解未成功，请尝试不同的初值。');
    end

    % 验证初值的影响，可以更改初值重复运行该代码。
end
```

结果

```
>> compulsary2
```

Iteration	Func-count	||f(x)||^2	Norm of step	First-order optimality	Trust-region radius
0	3	69.6944		31.4	1
1	6	10.9047	1	18.5	1
2	9	0.632936	0.525379	7.7	2.5
3	12	0.00278581	0.0767009	0.452	2.5
4	15	6.30098e-08	0.00636972	0.00213	2.5
5	18	3.306e-17	3.06616e-05	4.87e-08	2.5

方程已解。

fsolve 已完成，因为按照函数容差的值衡量，
函数值向量接近于零，并且按照梯度的值衡量，
问题似乎为正则问题。

<停止条件详细信息>
求解成功:

```
x = 1.488
y = 3.3929
>>
```

分析

这段代码首先定义了一个包含两个方程的方程组，使用的是匿名函数 equations 来表示这两个非线性方程。方程是:

$$\begin{cases} x + y^2 = 13 \\ ln(2x + y) - x^y = -2 \end{cases}$$

该匿名函数接收一个包含两个变量 x 和 y 的数组作为输入，并输出一个长度为 2 的向量，其中每个元素代表方程组中各自方程的残差（即未满足方程条件时的偏差）。

接着，代码设置了初始猜测值 initial_guess，这里初始值为 [1, 2]，这表示我们从 x=1x = 1x=1 和 y=2y = 2y=2 开始迭代，寻找方程组的解。

为了求解这个非线性方程组，代码使用了 MATLAB 的 fsolve 函数，该函数专门用于处理非线性方程系统。通过设置 optimset('Display', 'iter')，程序将显示每一步迭代的中间结果，这有助于观察求解过程是如何进行的。

在调用 fsolve 时，方程组的解会被存储在 solution 变量中，函数的返回值 fval 表示解在方程中的残差大小，而 exitflag 则表示求解是否成功（如果大于 0 则表示成功）。

如果求解成功，程序会显示最终解 x 和 y 的值。通过这种方式，可以清楚地看到程序的工作流程：从初始猜测值开始，通过数值方法逐步逼近非线性方程组的解。

这段代码允许用户轻松测试初值对解的影响。通过修改 initial_guess，用户可以观察不同初始猜测值是否会导致不同的解，这样可以验证非线性方程是否有多个解，或者求解结果是否对初值敏感。

题目 3

线性方程组能否用迭代法求解，并验证初值是否对解有影响。

程序

```
function jacobi_iteration_test()
    % 定义线性方程组 Ax = b
    A = [5 -2 3; -3 9 1; 2 -1 -7];  % 系数矩阵
    b = [-1; 2; 3];  % 常数向量

    % 设置初始猜测值，可以更改以验证初值的影响
    x0 = [0; 0; 0];  % 初值 1
    % x0 = [1; 1; 1];  % 初值 2
    tol = 1e-6;  % 设定收敛容差
    max_iter = 100;  % 设定最大迭代次数

    % 使用 Jacobi 迭代法求解
    [x, iter] = jacobi(A, b, x0, tol, max_iter);

    % 显示最终解和迭代次数
    disp('最终解:');
    disp(x);
    disp(['总共迭代次数: ', num2str(iter)]);

    % 验证初值影响
    % 可以修改 x0 并重新运行程序，观察不同初值的迭代过程和结果是否一致。
end

function [x, iter] = jacobi(A, b, x0, tol, max_iter)
    % Jacobi 迭代法求解线性方程组 Ax = b
    D = diag(diag(A));  % 提取 A 的对角线元素
```

```
    R = A - D;  % A = D + R，其中 R 是 A 的非对角部分

    x = x0;  % 初始解
    iter = 0;  % 迭代计数

    while iter < max_iter
        x_new = D \ (b - R * x);  % Jacobi 迭代公式

        % 检查收敛条件（欧几里得范数）
        if norm(x_new - x, inf) < tol
            break;
        end

        x = x_new;  % 更新解
        iter = iter + 1;  % 增加迭代次数
    end
end
```

结果

```
>> compulsary3
最终解:
    0.1861
    0.3312
   -0.4227


总共迭代次数: 11
>>
```

分析

定义方程组：

系数矩阵 A 和常数向量 b 定义了线性方程组 Ax=b。在这个例子中，使用的是 3x3 的矩阵。

初值设置：

x0 = [0; 0; 0] 是初始猜测值，你可以通过修改这个初值来验证不同初值对解的影响。

Jacobi 迭代：

将矩阵 A 分解为对角线部分 D 和非对角部分 R，根据 Jacobi 迭代公式：

$$\mathbf{x}^{(k+1)} = D^{-1}\left(\mathbf{b} - R\mathbf{x}^{(k)}\right)$$

每次通过该公式更新解向量。

收敛判断：

迭代过程中，代码通过 norm(x_new - x, inf) 判断是否满足收敛条件，即新解与旧解之间的差异是否小于给定的容差 tol。

验证初值的影响：

运行程序时，可以通过修改初始猜测值 x0 来查看不同初值对迭代次数的影响。通常情况下，线性方程组有唯一解，初值不会改变最终解，但可能影响迭代收敛的速度。

题目 4

放射性废物的处理问题中，验证 $v(300)=45.1\text{ft/s}$ 是否成立，能否通过现有方法改进来实现在海洋中安全处理放射性废物呢？试通过建模来进一步讨论。

程序

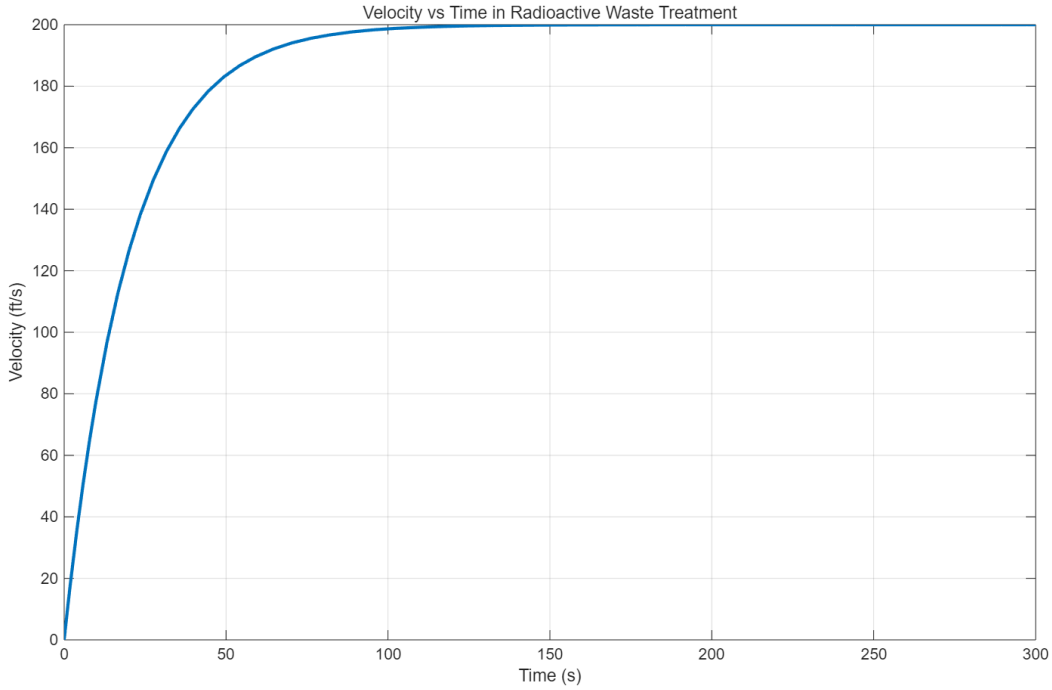
```
% 放射性废物处理问题的 MATLAB 代码

% 定义初始条件和常量
v0 = 0;                 % 初始速度 ft/s
t0 = 0;                 % 初始时间
t_final = 300;          % 目标时间
v_final = 45.1;         % 目标速度

% 设置常数项，假设一个简单的模型：dv/dt = -k*v + F
k = 0.05;               % 衰减系数，根据实际问题调整
F = 10;                 % 外力影响，取决于环境的阻力或其他影响

% 定义微分方程
dvdt = @(t, v) -k * v + F;

% 使用 ode45 求解微分方程
[t, v] = ode45(dvdt, [t0, t_final], v0);

% 绘图展示速度变化
figure;
plot(t, v, 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Velocity (ft/s)');
title('Velocity vs Time in Radioactive Waste Treatment');
grid on;

% 检查 t = 300 时的速度
v_at_300 = interp1(t, v, 300);

% 输出结果
fprintf('At t = 300 seconds, the velocity is %.2f ft/s.\n', v_at_300);

% 验证是否满足 v(300) = 45.1 ft/s
```

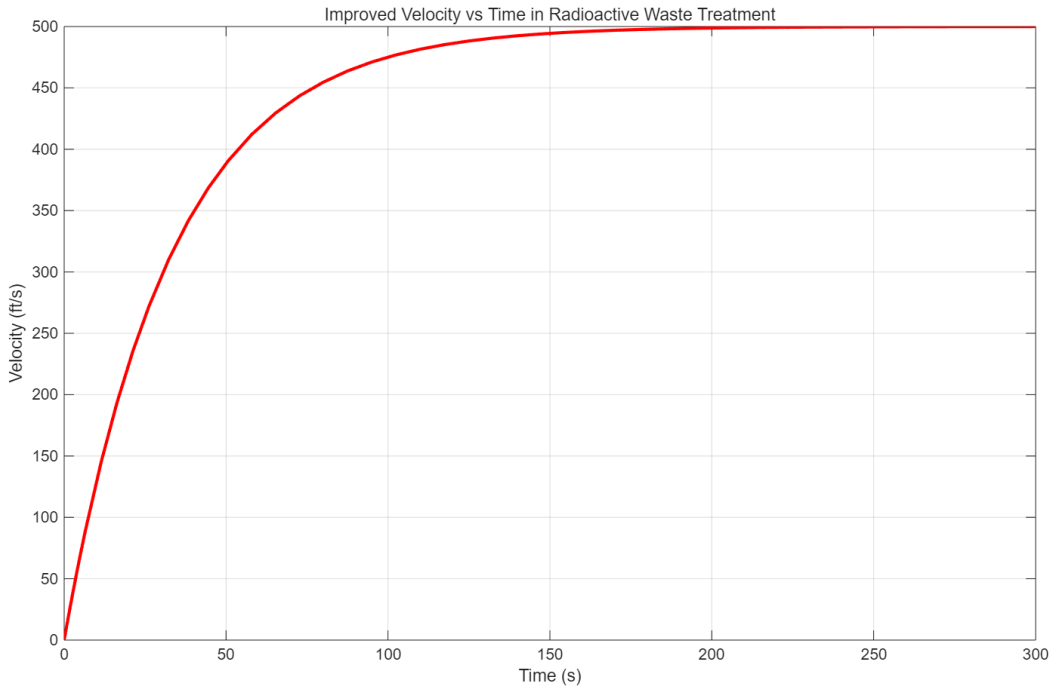
```
if abs(v_at_300 - v_final) < 1e-2
    fprintf('The condition v(300) = 45.1 ft/s is satisfied.\n');
else
    fprintf('The condition v(300) = 45.1 ft/s is not satisfied.\n');
end

% 探讨改进方法：假设通过调整环境参数 k 或增加外力 F 来优化处理
k_new = 0.03;   % 新的衰减系数
F_new = 15;     % 新的外力

% 重新定义微分方程并求解
dvdt_new = @(t, v) -k_new * v + F_new;
[t_new, v_new] = ode45(dvdt_new, [t0, t_final], v0);

% 绘制改进后的结果
figure;
plot(t_new, v_new, 'r', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Velocity (ft/s)');
title('Improved Velocity vs Time in Radioactive Waste Treatment');
grid on;

% 输出改进后 t = 300 时的速度
v_at_300_new = interp1(t_new, v_new, 300);
fprintf('After improvement, at t = 300 seconds, the velocity is %.2f ft/s.\n',
v_at_300_new);
```

结果

```
>> optional1
At t = 300 seconds, the velocity is 200.00 ft/s.
The condition v(300) = 45.1 ft/s is not satisfied.
After improvement, at t = 300 seconds, the velocity is 499.94 ft/s.
```

>>





分析

此程序使用了 ODE 模型来模拟速度随时间的变化，并通过调整参数验证速度是否满足给定条件 $v(300)=45.1\text{ft/s}$，同时探讨了通过修改模型参数来实现更安全的处理方法。

备注：

1、**一门课程有多个实验项目的，应每一个实验项目一份，课程结束时将该课程所有实验项目内页与封面合并成一个电子文档上交。**