学号 WA2214014 专业 人工智能 姓名 杨跃浙
实验日期 **24.12.06** 教师签字 成绩

实验报告

【实验名称】 **实验一：有效数字和误差限**

【实验目的】

1. 掌握有效数字和误差限的计算方法，理解有效数字在数值计算中的意义。

2. 熟悉 MATLAB 环境下的基本编程操作，包括函数的定义和调用。

3. 理解并实现三种常用数值迭代方法（牛顿法、不动点迭代法、割线法），掌握其原理及适用场景。

4. 能够通过数值方法求解方程的零点，并结合图形直观展示解的过程和结果。

5. 培养解决实际问题的能力，提高程序设计和调试的熟练度。

【实验内容】

实验要求：

1. 创建一个文件夹存放.m 文件，文件夹命名中需要包含个人姓名缩写，如：code_xy.
2. 理解算法及程序代码，尽量自己编写、调试；
3. 要求在函数名称后追加个人姓名缩写作为标识符，如 bisect_xy;

4. 实验关键过程和结果截图填入实验报告中，截图中应包含个人标识信息.

实验任务:

2 为必做题；3、4、5 至少选做 1 题。
1.MATLAB 入门基本练习（参看 MATLAB 简介、参考书及 PPT 等）

2. 已知圆周率π的真值为π=3.1415926….

(1) 若π的近似值取为$\pi^* = 3.1415$，问π^*有几位有效数字，并给出误差限；

(2) 若π的近似值取为$\pi^* = 3.141593$,问π^*有几位有效数字，并给出误差限。

3. 利用牛顿法求解函数 $f(x) = e^x - x - 5$ 在 3.8 附近的零点并画图展示。

4. 利用不动点迭代法求解函数 $f(x) = e^{-}x - x$ 在 0.1 附近的零点并画图展示。

5. 利用割线法求解函数 $f(x) = e^x - x - 5$ 的零点并画图展示，初始迭代点取$x_0 = 3.8, x_1 = 3.6$。

calculate_yyz.m

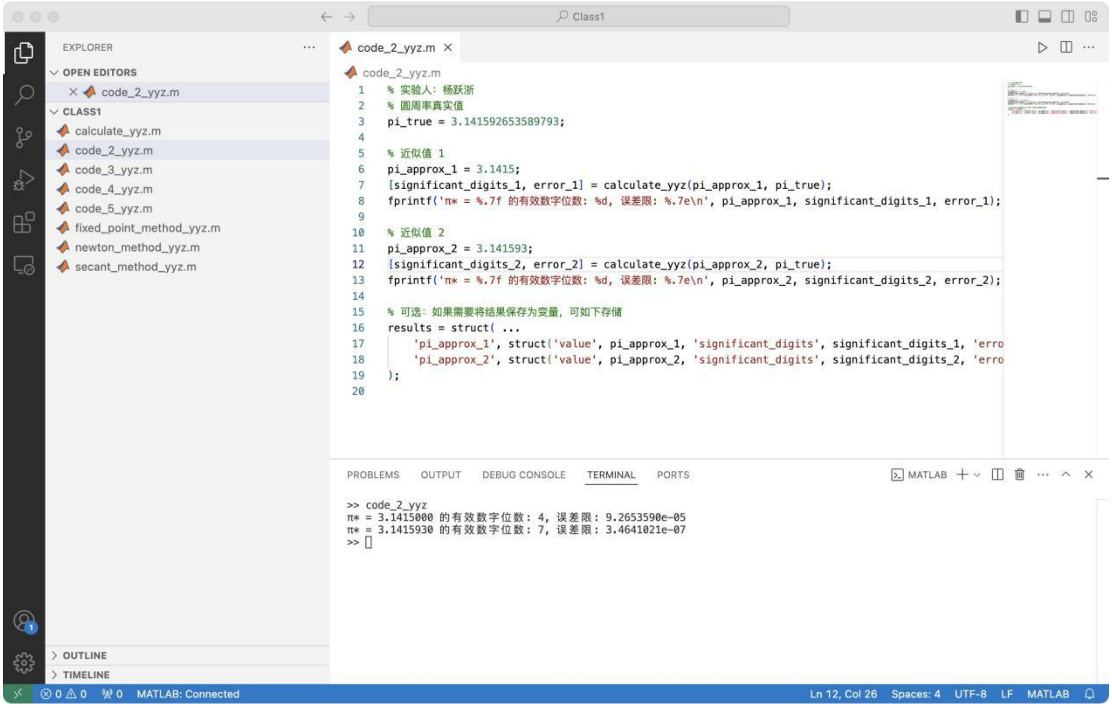
```
function [significant_digits, error] =
calculate_yyz(approx_value, true_value)
% 计算误差
error = abs(approx_value - true_value);
% 计算整数位数
integer_digits = count_integer_digits_yyz(approx_value);
% 计算有效数字
n = 0;
while error < 0.5 * 10^(-n-1)
n = n + 1;
end
% 总有效数字为整数位数 + 小数部分的有效数字
significant_digits = integer_digits + n;
end

function integer_digits = count_integer_digits_yyz(value)
% 计算整数部分的位数
integer_digits = length(num2str(floor(value)));
end
```

code_2_yyz.m

```
% 实验人: 杨跃浙
% 圆周率真实值
```

```
pi_true = 3.141592653589793;

% 近似值 1
pi_approx_1 = 3.1415;
[significant_digits_1, error_1] = calculate_yyz(pi_approx_1, pi_true);
fprintf('π* = %.7f 的有效数字位数: %d, 误差限: %.7e\n', pi_approx_1, significant_digits_1, error_1);

% 近似值 2
pi_approx_2 = 3.141593;
[significant_digits_2, error_2] = calculate_yyz(pi_approx_2, pi_true);
fprintf('π* = %.7f 的有效数字位数: %d, 误差限: %.7e\n', pi_approx_2, significant_digits_2, error_2);

% 可选：如果需要将结果保存为变量，可如下存储
results = struct( ...
'pi_approx_1', struct('value', pi_approx_1, 'significant_digits', significant_digits_1, 'error_limit', error_1), ...
'pi_approx_2', struct('value', pi_approx_2, 'significant_digits', significant_digits_2, 'error_limit', error_2) ...
);
```

结果:



newton_method_yyz.m

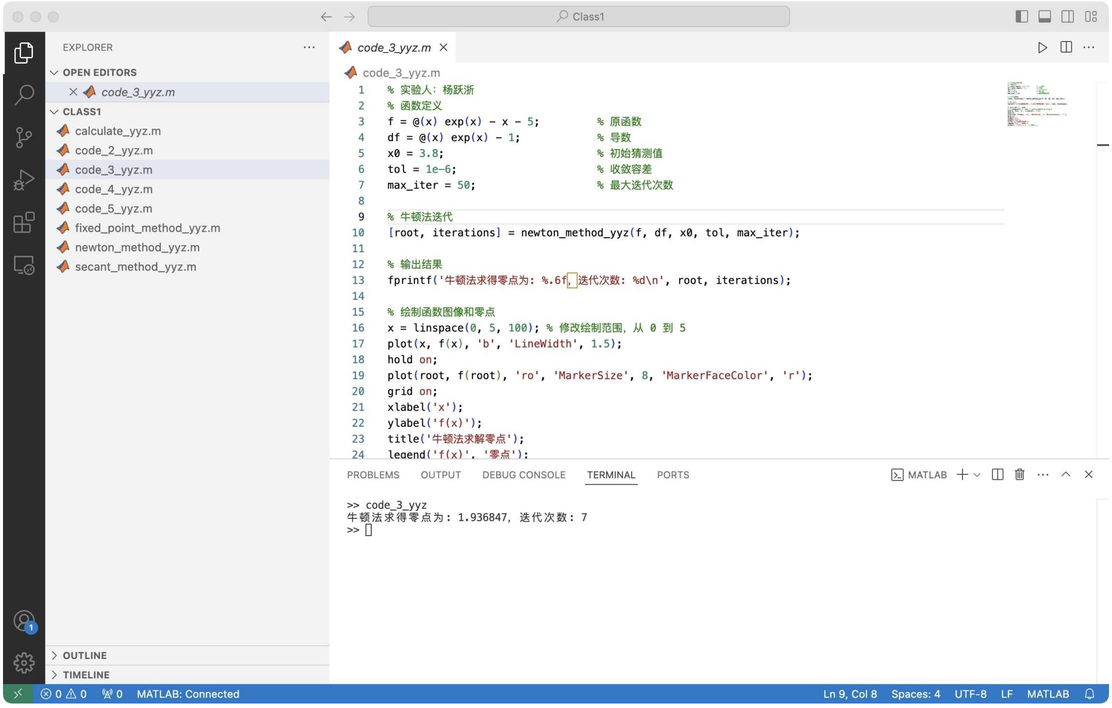
```
function [root, iterations] = newton_method_yyz(f, df, x0, tol, max_iter)
x = x0;
for k = 1:max_iter
x_new = x - f(x) / df(x);
if abs(x_new - x) < tol
root = x_new;
iterations = k;
return;
end
x = x_new;
end
error('牛顿法未能在指定迭代次数内收敛');
end
```

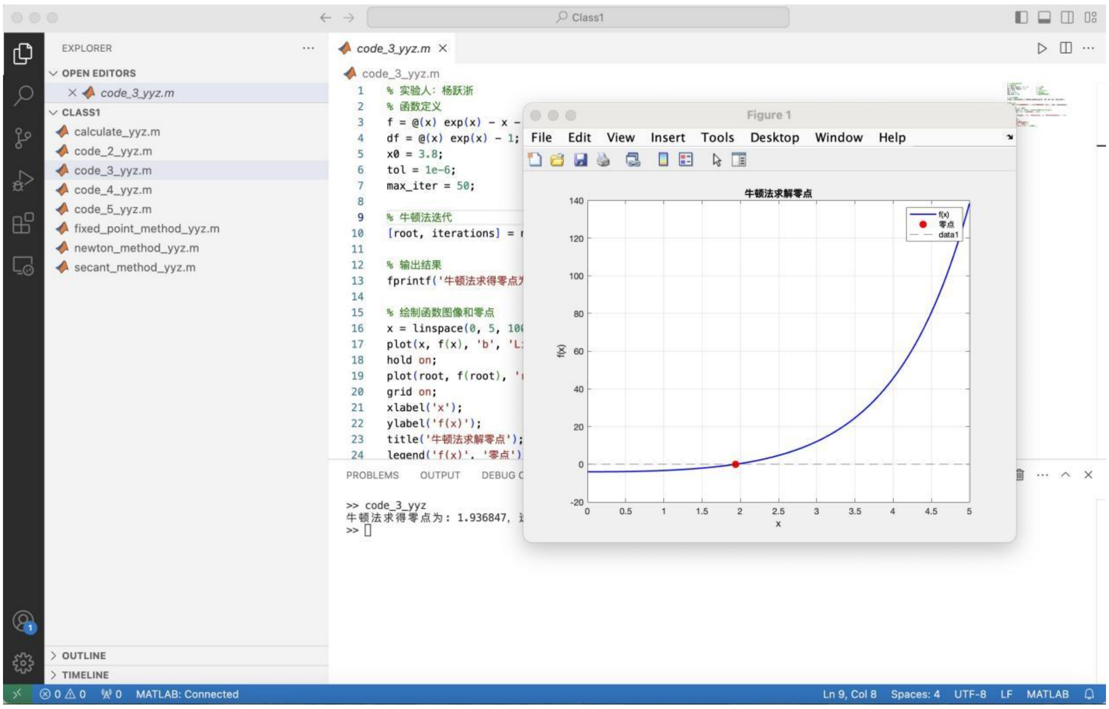
code_3_yyz.m

```
% 实验人：杨跃浙
% 函数定义
f = @(x) exp(x) - x - 5; % 原函数
df = @(x) exp(x) - 1; % 导数
x0 = 3.8; % 初始猜测值
tol = 1e-6; % 收敛容差
```

```
max_iter = 50; % 最大迭代次数

% 牛顿法迭代
[root, iterations] = newton_method_yyz(f, df, x0, tol,
max_iter);

% 输出结果
fprintf('牛顿法求得零点为: %.6f, 迭代次数: %d\n', root,
iterations);

% 绘制函数图像和零点
x = linspace(0, 5, 100); % 修改绘制范围, 从 0 到 5
plot(x, f(x), 'b', 'LineWidth', 1.5);
hold on;
plot(root, f(root), 'ro', 'MarkerSize', 8, 'MarkerFaceColor',
'r');
grid on;
xlabel('x');
ylabel('f(x)');
title('牛顿法求解零点');
legend('f(x)', '零点');
yline(0, '--k'); % 添加 x 轴辅助线
```

结果:





fixed_point_method_yyz.m

```
function [root, iterations] = fixed_point_method_yyz(g, x0,
tol, max_iter)
x = x0;
for k = 1:max_iter
x_new = g(x);
if abs(x_new - x) < tol
root = x_new;
```

```
iterations = k;
return;
end
x = x_new;
end
error('不动点迭代法未能在指定迭代次数内收敛');
end
```

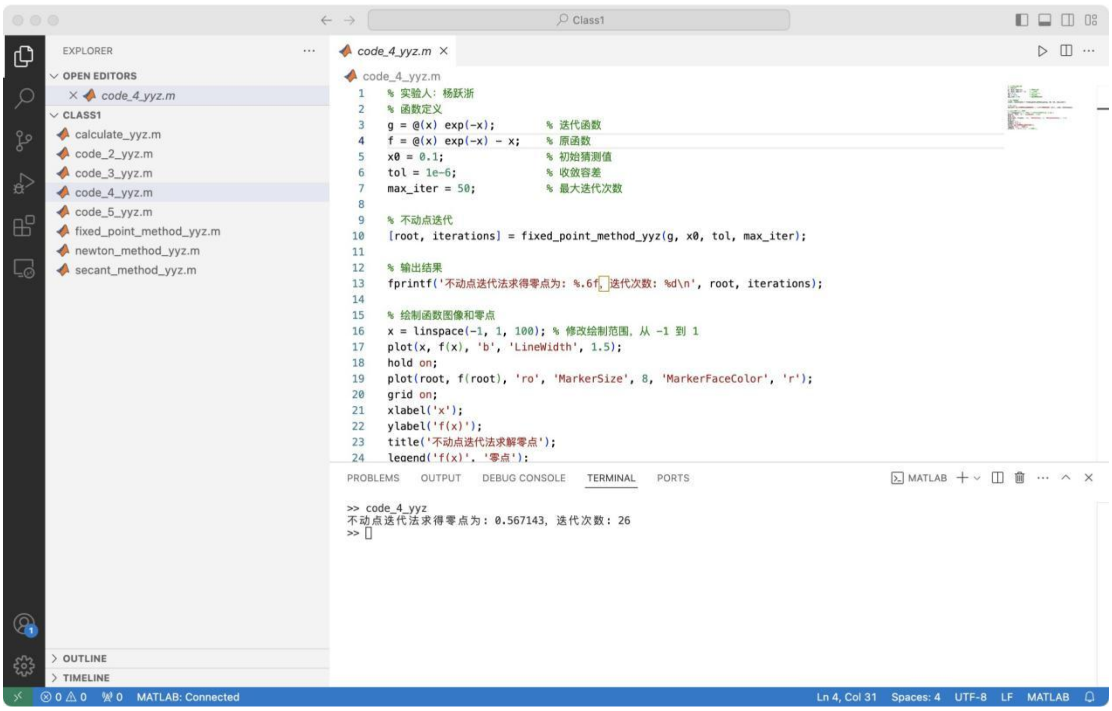
code_4_yyz.m

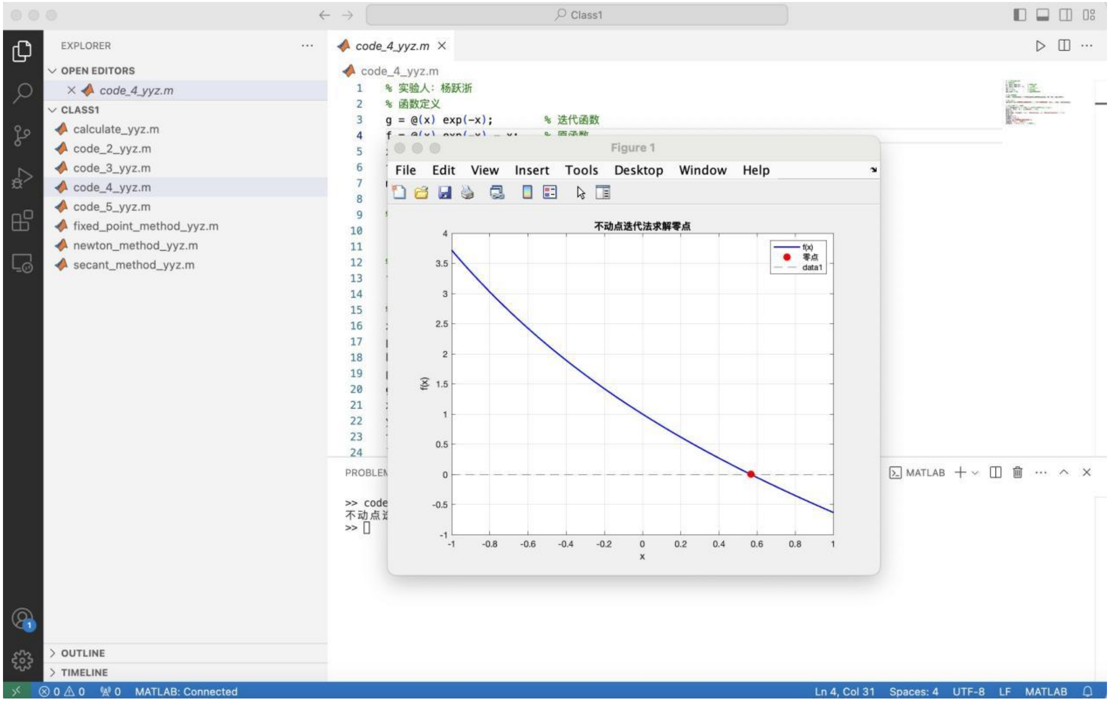
```
% 实验人: 杨跃浙
% 函数定义
g = @(x) exp(-x); % 迭代函数
f = @(x) exp(-x) - x; % 原函数
x0 = 0.1; % 初始猜测值
tol = 1e-6; % 收敛容差
max_iter = 50; % 最大迭代次数

% 不动点迭代
[root, iterations] = fixed_point_method_yyz(g, x0, tol,
max_iter);

% 输出结果
fprintf('不动点迭代法求得零点为: %.6f, 迭代次数: %d\n', root,
iterations);

% 绘制函数图像和零点
x = linspace(-1, 1, 100); % 修改绘制范围, 从 -1 到 1
plot(x, f(x), 'b', 'LineWidth', 1.5);
hold on;
plot(root, f(root), 'ro', 'MarkerSize', 8, 'MarkerFaceColor',
'r');
grid on;
xlabel('x');
ylabel('f(x)');
title('不动点迭代法求解零点');
legend('f(x)', '零点');
yline(0, '--k'); % 添加 x 轴辅助线
```

secant_method_yyz.m

```
function [root, iterations] = secant_method_yyz(f, x0, x1,
tol, max_iter)
for k = 1:max_iter
fx0 = f(x0);
fx1 = f(x1);
if abs(fx1 - fx0) < eps
error('割线法失败：分母接近零');
end
x_new = x1 - fx1 * (x1 - x0) / (fx1 - fx0);
if abs(x_new - x1) < tol
root = x_new;
iterations = k;
return;
end
x0 = x1;
x1 = x_new;
end
error('割线法未能在指定迭代次数内收敛');
end
```

结果：



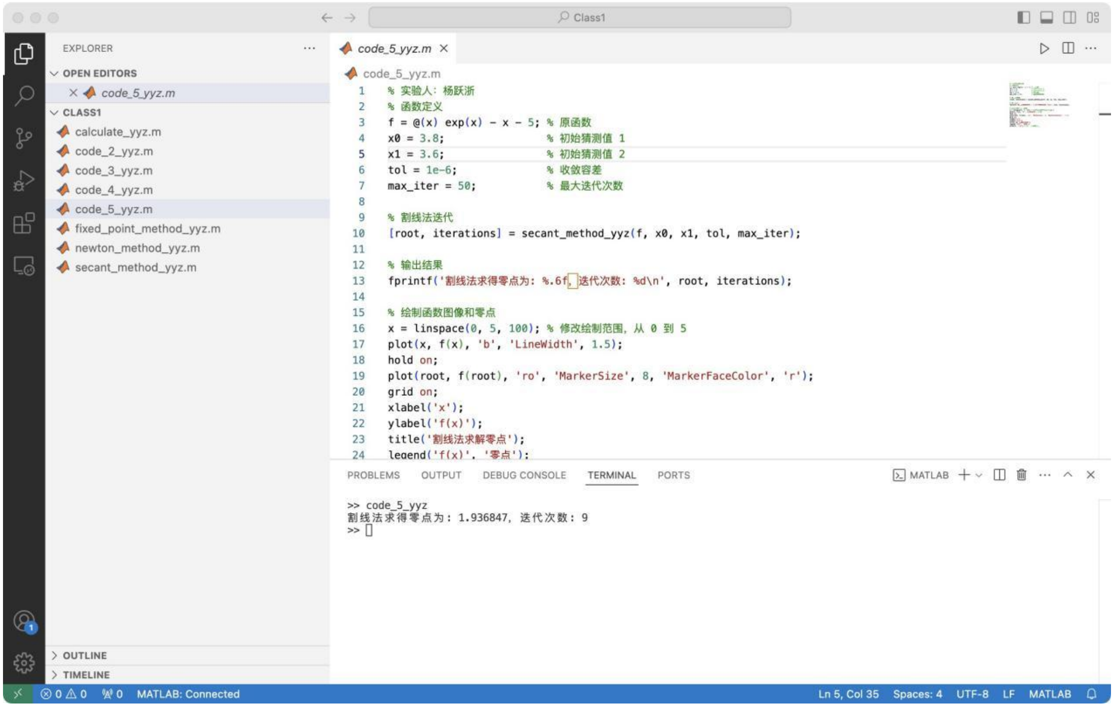


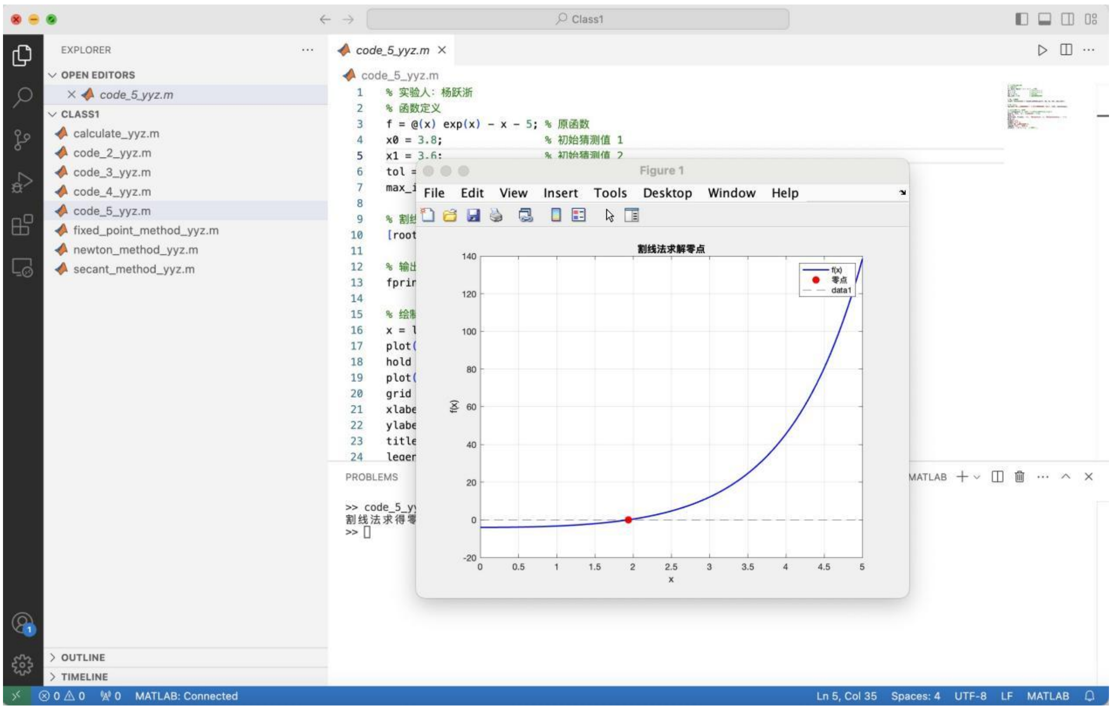
code_5_yyz.m

```
% 实验人：杨跃浙
% 函数定义
f = @(x) exp(x) - x - 5; % 原函数
x0 = 3.8; % 初始猜测值 1
x1 = 3.6; % 初始猜测值 2
tol = 1e-6; % 收敛容差
max_iter = 50; % 最大迭代次数

% 割线法迭代
[root, iterations] = secant_method_yyz(f, x0, x1, tol,
max_iter);

% 输出结果
fprintf('割线法求得零点为：%.6f，迭代次数：%d\n', root,
iterations);

% 绘制函数图像和零点
x = linspace(0, 5, 100); % 修改绘制范围，从 0 到 5
plot(x, f(x), 'b', 'LineWidth', 1.5);
hold on;
plot(root, f(root), 'ro', 'MarkerSize', 8, 'MarkerFaceColor',
'r');
```

```
grid on;
xlabel('x');
ylabel('f(x)');
title('割线法求解零点');
legend('f(x)', '零点');
yline(0, '--k'); % 添加 x 轴辅助线
```

结果:





【小结或讨论】

通过本次实验，我深入学习了有效数字与误差限的计算方法以及数值方法在求解方程零点中的应用。有效数字是描述数值精度的重要指标，从第一个非零数字开始，直到符合精度要求的最后一位数字。实验中，我们通过公式$|x^* - x| \leq 0.5 \times 10^{1-n}$判断近似值的有效数字位数。例如，当圆周率的近似值$\pi^* = 3.1415$时，其有效数字为 4 位，误差限为$9.265 \times 10^{-5}$;当$\pi^* = 3.141593$时，其有效数字为 7 位，误差限更小，为$3.464 \times 10^{-7}$。通过实验，我体会到有效数字和误差限之间的关系，它们共同衡量了数值计算的精度和可信度。

在数值方法的实现过程中，分别采用了牛顿法、不动点迭代法和割线法对方程的零点进行求解。牛顿法因其收敛速度快、迭代效率高而表现出良好的求解效果，但对初值选择和导数计算的依赖较高；不动点迭代法实现简单，但收敛性依赖于迭代函数的构造条件，实验中其收敛速度相对较慢；割线法不依赖导数信息，仅利用两点进行线性逼近，但需要两个初始点。在实验过程中，这三种方法均成功求解出零点，并通过图像直观展示了解的过程和结果。

此外，绘图范围的合理选择对实验结果的可视化起到了关键作用。例如，通过调整 的绘制范围，更直观地观察函数在零点附近的行为变

化。收敛容差和最大迭代次数的设定也影响了计算结果的效率和准确性，这些都是数值方法应用中的重要实践经验。

总的来说，本次实验让我深入理解了有效数字和误差限的理论意义，并掌握了三种常用数值迭代方法的基本原理和实现方式。这些方法各有优缺点，在不同场景中需根据问题特性选择适合的算法。通过实验，我进一步体会到数值方法在工程和科学计算中的重要性，为后续研究和实践打下了坚实基础。