电子科技大学

课程实验报告

课程名称：	最优化理论与应用
实验题目：	精确一维搜索方法
姓　　名：	徐宇健
学　　号：	202422280813
专　　业：	085400 电子信息工程
学　　院：	（深圳）高等研究院
完成日期：	2024 年 10 月 12 日

实验目的：

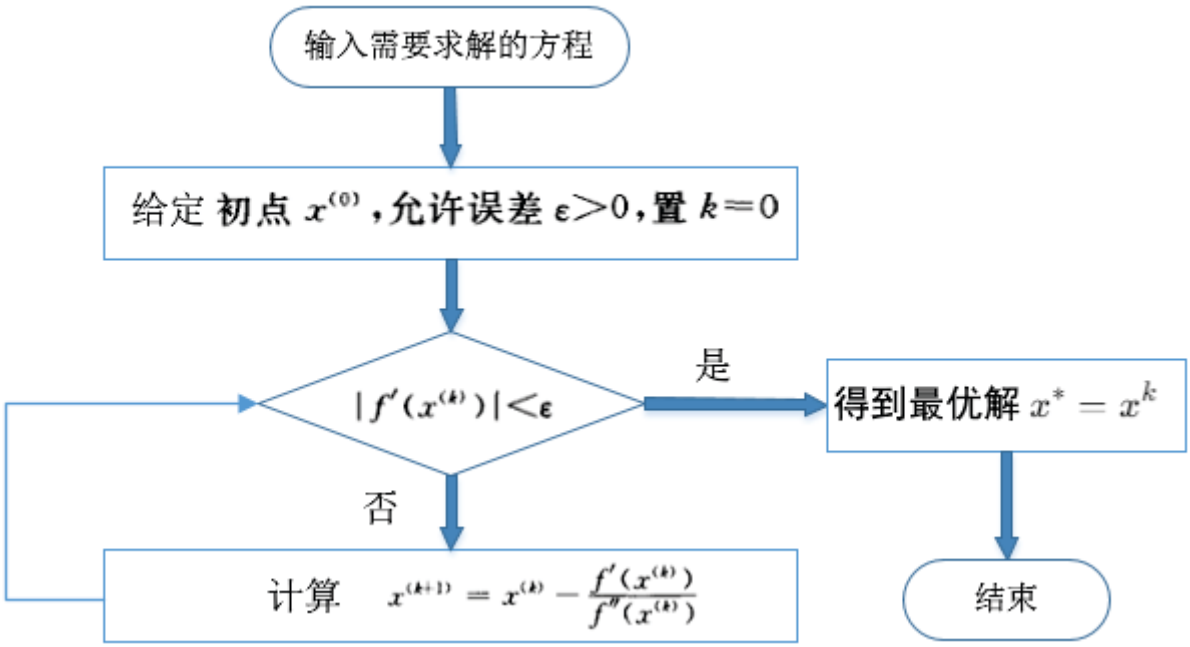
1. 以本次实验为案例，了解算法复现流程；

2. 掌握 matlab 的基本操作，学习编写 matlab 程序，提高编程能力和技巧；

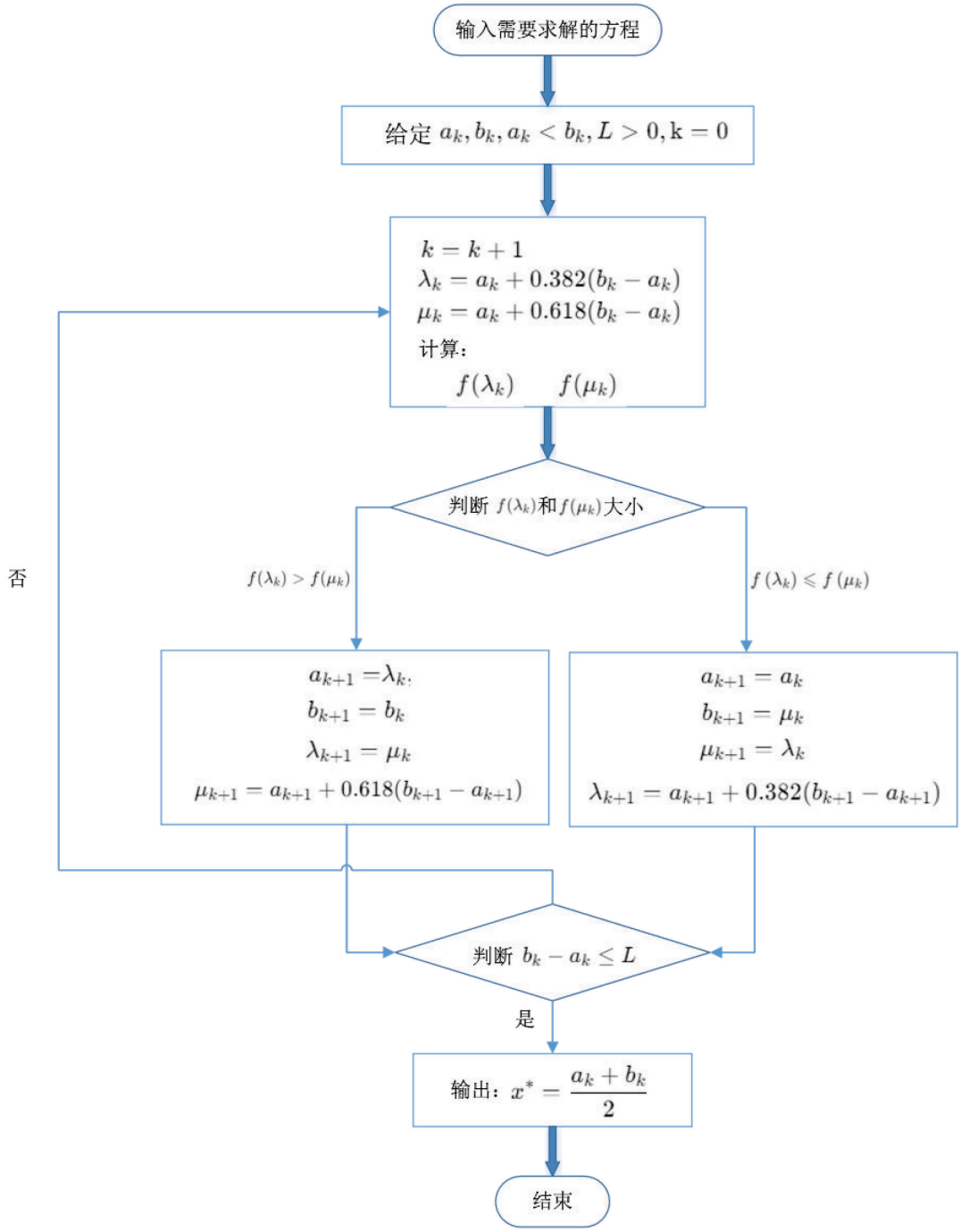
3. 学习用已学的知识解决实际问题，将理论应用于实际。

算法步骤：

牛顿法：



0.618 法：


输入需要求解的方程
给定 $a_k, b_k, a_k < b_k, L > 0, \mathrm{k} = 0$
$k = k + 1$
$\lambda_k = a_k + 0.382(b_k - a_k)$
$\mu_k = a_k + 0.618(b_k - a_k)$
计算：
$f(\lambda_k)$ $f(\mu_k)$
判断 $f(\lambda_k)$和$f(\mu_k)$大小
$f(\lambda_k) > f(\mu_k)$
$f(\lambda_k) \leqslant f(\mu_k)$
否
$a_{k+1} = \lambda_k$
$b_{k+1} = b_k$
$\lambda_{k+1} = \mu_k$
$\mu_{k+1} = a_{k+1} + 0.618(b_{k+1} - a_{k+1})$
$a_{k+1} = a_k$
$b_{k+1} = \mu_k$
$\mu_{k+1} = \lambda_k$
$\lambda_{k+1} = a_{k+1} + 0.382(b_{k+1} - a_{k+1})$
判断 $b_k - a_k \leq L$
是
输出：$x^* = \frac{a_k + b_k}{2}$
结束

Matlab 程序：

牛顿法：

```
clc,clear;
% 定义符号变量
syms x
% 定义目标函数 f(x) 及其一阶和二阶导数
f = x^4 - 4*x^3 - 6*x^2 - 16*x + 4;  % 目标函数 f(x)
df = diff(f, x);
ddf = diff(df, x);
% 将符号表达式转换为匿名函数
f = matlabFunction(f);
df = matlabFunction(df);
ddf = matlabFunction(ddf);
% 定义目标函数 f(x) 及其一阶和二阶导数
% f = @(x) x^4 - 4*x^3 - 6*x^2 - 16*x + 4;  % 目标函数 f(x)
% df = @(x) 4*x^3 - 12*x^2 - 12*x - 16;       % 一阶导数 f'(x)
% ddf = @(x) 12*x^2 - 24*x -12;             % 二阶导数 f''(x)

% 初始猜测值
x0 = 3.0;

% 精度要求
tolerance = 1e-3;
max_iterations = 100;

% 牛顿法迭代
x = x0;
for i = 1:max_iterations
    numerator = df(x);%分子
    denominator = ddf(x);%分母
    x_new = x - df(x)/ddf(x);

    % 检查是否满足收敛条件
    if abs(df(x)) < tolerance
        break;
    end

    x = x_new;
end

% 输出结果
fprintf('在牛顿法%d 次迭代后，最小值点的近似值为: %.6f\n', i, x);
fprintf('函数最小值约为: %.6f\n', f(x));
```

0.618 法：

```
clc,clear;
% 定义符号变量
%syms x
%定义函数和相关参数
mf = @(x) 3*x^3 - 4*x + 2;  % 目标函数 f(x)
% f = exp(-x) + x^2;  % 目标函数 f(x)
% f = 2*x^2 - x - 1;  % 目标函数 f(x)

% 使用 fplot 绘制符号表达式的图像
% fplot(f, [-5 5]) % 指定 x 轴范围为-5 到 5
% grid on; % 显示网格
% title('Graph of f(x)'); % 设置标题
% xlabel('x'); % 设置 x 轴标签
% ylabel('f(x)'); % 设置 y 轴标签

% 将符号表达式转换为匿名函数
%mf = matlabFunction(f);
x1 = 0;%初始点
h0 = 1;%初始步长
k = 2;%步长放大倍数
max_iterations = 100;%最大迭代次数

% 精度要求
tolerance = 0.2;

%寻找极小值所在区间
f1 = mf(x1);
f2 = mf(x1 + h0);
%前进运算
if f1 > f2
    for i = 1:max_iterations
        f1_new = mf(x1 + h0 + h0*k*(i-1));
        f2_new = mf(x1 + h0 + h0*k*i);
        if f1_new <= f2_new  %得到初始化区间
            a = x1 + h0*k*(i-1);
            b = x1 + h0 + k*h0*i;
            break;
        end
    end
else %后退运算
    for i = 1:max_iterations
        f1_new = mf(x1 - h0*k*i);
        f2_new = mf(x1 - h0*k*(i-1));
        if f1_new >= f2_new  %得到初始化区间
```

```
            a = x1 - h0*k*i;
            b = x1 - h0*k*(i-1) + h0;
            break;
        end
    end
end
%disp(a);
%disp(b);

%0.618法
ma  = a;
mb = b;
c = ma + 0.382*(mb - ma);
d = ma + 0.618*(mb - ma);
for i = 1:max_iterations
    if mb - ma < tolerance
        my_ans = 0.5*(ma + mb);
        % 输出结果
        fprintf('在经过0.618法%d次迭代后，最小值点的近似值为: %.6f\n', i, my_ans);
        fprintf('函数最小值约为: %.6f\n', mf(my_ans));
        break;
    end
    if mf(c) > mf(d)
        ma = c;
        c = d;
        d = ma + 0.618*(mb - ma);
    else
        mb = d;
        d = c;
        c = ma + 0.382*(mb - ma);
    end
end
```

实验结果：

题目 1：

牛顿法：

$$f(\alpha)=\alpha^4-4\alpha^3-6\alpha^2-16\alpha+4$$
$$\alpha_0=3 \quad \varepsilon=0.001$$

命令行窗口

在牛顿法6次迭代后:
最小值点的近似值为: 4.000000
函数最小值约为: -156.000000

题目 2：

0.618 法：

$$f(x)=3x^3-4x+2$$
$$x_1=0, h_0=1, \varepsilon=0.2$$

命令行窗口

在0.618法7次迭代后:
最小值点的近似值为: 0.688549
函数最小值约为: 0.225127

实验收获：

牛顿法是一种求解无约束最优化问题的有效方法，尤其适用于目标函数为二次可微的情况。它基于泰勒展开式的一阶和二阶导数信息来逼近最优解。在实验中，我发现当初始点选择得当且函数满足一定条件时，**牛顿法**可以非常快速地收敛到局部最小值点。然而，该方法对于非凸函数可能不会总是有效，并且对初始猜测值敏感。

对于 **0.618 法**，这是一种单变量函数极小化的方法，特别适合于不需要求导或难以求导的情形。这种方法利用了黄金比例的特性，能够在每次迭代中以一种高效的方式缩小搜索区间。实验中使用 **0.618 法**寻找函数极小值的过程相对直观简单，但其局限性在于仅适用于一维问题以及要求目标函数在给定区间内是单峰的。尽管如此，通过这次练习，我还是学到了如何有效地设置参数并调整算法流程来提高搜索效率。

通过本次实验，不仅加深了我对这些经典优化算法工作原理的理解，更重要的是学会了如何根据具体问题特点选择合适的工具和技术手段。同时，也认识到即便是看似简单的数学模型背后也可能隐藏着复杂的考量因素。