学生学号	0121820390301	实验课成绩	

武漢理工大學

学 生 实 验 报 告 书

实验课程名称	计算机数值分析
开 课 学 院	计算机科学与技术学院
指导教师姓名	戚欣
学 生 姓 名	杜俊
学生专业班级	软件 1805

2019 —— 2020 学年 第 二 学期

课程名称：计算机数值分析

实验项目名称	插值方法			实验成绩	
实验者	杜俊	专业班级	软件 1805	实验日期	2020 年 4 月 23 日

第一部分　　实验概述

一、实验目的

通过设计、编制、调试 2~3 个多项式插值、拟合曲线的程序，加深对其数值计算方法及有关的基础理论知识的理解。

二、实验要求

用编程语言实现拉格朗日（Lagrange）插值多项式、牛顿（Newton）插值、用线性函数 $p(x)=a+bx$ 拟合给定数据 $(x_i, y_i), i=0,1,2,\cdots,m-1$ 的程序。

三、实验内容

用编程语言编程实现以下算法：

(1) 已知插值节点序列 $(x_i, y_i), i=0,1,2,\cdots,n$，用拉格朗日（Lagrange）插值多项式 $L_n(x)$ 计算的函数 $f(x)$ 在点 x_0 的近似值。

(2) 已知插值节点序列 $(x_i, y_i), i=0,1,2,\cdots,n$，用牛顿（Newton）插值值多项式 $N_n(x)$ 计算的函数 $f(x)$ 在点 x_0 的近似值。

(3) 用线性函数 $p(x)=a+bx$ 拟合给定数据 $(x_i, y_i), i=0,1,2,\cdots,m-1$。

一、实验器材

(1) PC 机。

(2) 编程语言：MATLAB

第二部分　　实验设计、测试与分析

一、问题：已知插值节点序列$(x_i, y_i), i = 0,1,2,\cdots,n$，用拉格朗日（Lagrange）插值多项式$L_n(x)$计算的函数$f(x)$在点$x_0$的近似值。

1.1　　算法描述

（1）给定插值点及其对应的函数值；

（2）两个循环，一个用来计算每一个插值基函数，另一个用来计算整个插值函数的值；

（3）输出变量 x 的对应插值结果的值。

1.2　　程序变量说明

```
%插值点
X=1:0.1:3;
%插值点数
n=(3-1)/0.1;
%插值
Y=sin(X);
%Lagrange 插值函数值
L=0;
%Lagrange 插值基函数
l=ones(1,n+1);
%所求的函数点
x=2;
```

1.3　　源程序代码及运行结果截图

```
%插值点
X=1:0.1:3;
%插值点数
n=(3-1)/0.1;
%插值
Y=sin(X);
%Lagrange 插值函数值
L=0;
%Lagrange 插值基函数
l=ones(1,n+1);
%所求的函数点
x=2;
```

```
%计算过程
for k=1:n+1
    for i=1:n+1
        if(i~=k)
            l(k)=l(k).*(x-X(i))./(X(k)-X(i));
        end
    end
    L=L+Y(k).*l(k);
end
fprintf('%d 次准确值为: %.6f\n',n, sin(x));
fprintf('%d 次插值的结果为: %.6f\n',n, L);
```

截图：

```
20次准确值为：0.909297
20次插值的结果为：0.909297
```

二、问题：已知插值节点序列 $(x_i, y_i), i = 0,1,2,\cdots,n$，用牛顿（Newton）插值值多项式 $N_n(x)$ 计算的函数 $f(x)$ 在点 x_0 的近似值。

2.1 算法描述

（1）给定插值点及其对应的值；

（2）构建差商表，先将插值点对应的函数值即 Y（i）赋给差商表的第一列，紧接着后面的每一列根据差商公式计算出值，完成表的构建；

（3）构建牛顿插值函数，即用 Newton 插值公式，用构建好的差商表里面的元素进行迭代计算，直到算到第 n 元，得到变量 x 的对应插值，输出。

2.2 程序变量说明

```
%插值点
X=1:0.1:3;
%插值点数
n=(3-1)/0.1;
%插值
Y=log(X);
%构建差商表
a=zeros(n+1,n+1);
%牛顿插值函数初值
N=0;
%增加的项初始值
w=1;
```

```
%所求的值
x=2;
```

2.3 源程序代码及运行结果截图

```
%插值点
X=1:0.1:3;
%插值点数
n=(3-1)/0.1;
%插值
Y=log(X);
%构建差商表
a=zeros(n+1,n+1);
for i=1:n+1
    a(i,1)=Y(i);
end
for j=2:n+1
    for i=j:n+1
        a(i,j)=(a(i,j-1)-a(i-1,j-1))/(X(i)-X(i-j+1));
    end
end
%构建牛顿插值函数
N=0;
w=1;
x=2;
for i=1:n+1
    N=N+a(i,i).*w;
    w=w.*(x-X(i));
end
fprintf('x 对应的准确值为: %.6f\n',log(x));
fprintf('x 对应的插值结果为: %.6f\n',N);
```

截图:

```
x对应的准确值为：0.693147
x对应的插值结果为：0.693147
```

三、问题：用线性函数 $p(x)=a+bx$ 拟合给定数据 $(x_i,y_i), i=0,1,2,\cdots,m-1$。

3.1 算法描述

（1）给定数据点及其对应的值；

（2）利用已知的数据点及其值分别计算出 xi、yi、xi 平方、xi*yi 的和；

（3）代入关于啊 a，b 的二元一次方程组，解得 a 和 b，得到拟合方程。

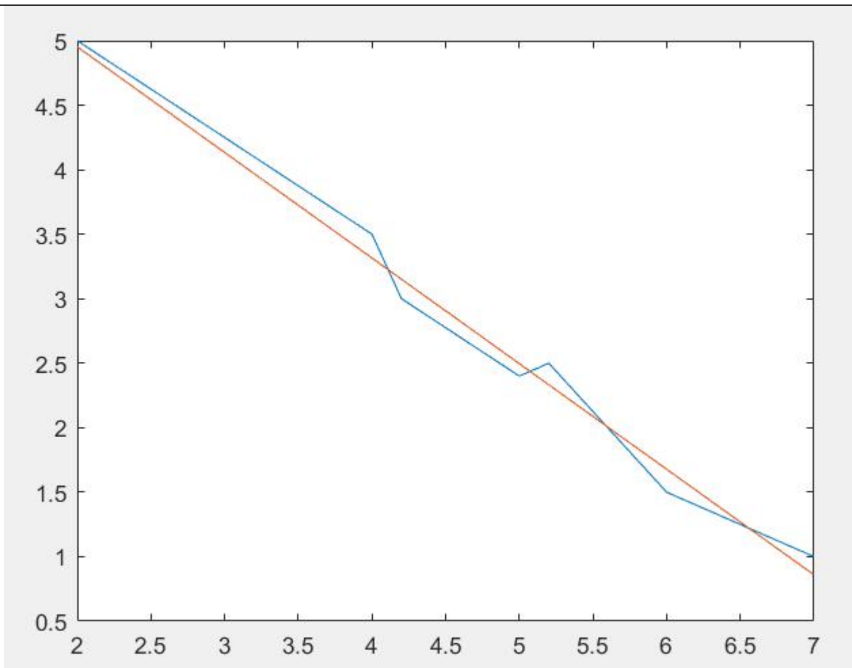
3.2 程序变量说明

```
%数据点
X=[2,4,4.2,4.6,5,5.2,5.6,6,6.6,7];
%节点数
n=10;
%对应的值
Y=[5,3.5,3,2.7,2.4,2.5,2,1.5,1.2,1];
%求解拟合方程需要的条件
p=sum(X);
q=sum(Y);
r=sum(X.^2);
s=sum(X.*Y);
%求解拟合方程的未知量
a，b;
```

3.3 源程序代码及运行结果截图

```
%数据点
X=[2,4,4.2,4.6,5,5.2,5.6,6,6.6,7];
n=10;
%对应的值
Y=[5,3.5,3,2.7,2.4,2.5,2,1.5,1.2,1];
plot(X,Y);
hold on;
%求解拟合方程
p=sum(X);
q=sum(Y);
r=sum(X.^2);
s=sum(X.*Y);
syms  a b
[a,b]=solve(n*a+p*b==q,p*a+r*b==s,a,b);
Z=a+b.*X;
plot(X,Z);
fprintf('最小二乘拟合一次函数为：y=%.4f%.4fx\n',a,b);
```

截图：（蓝色为数据点连成的折线，红色为拟合成的函数直线）



最小二乘拟合一次函数为：y=6.5900-0.8187x

第三部分　　实验小结与心得体会

利用差值的方法，让我们在无法得知函数方程的情况下，可以利用已知的有限个插值点及其对应的值来近似计算出原方程，从而达到最终目的。我认为这样的计算方法非常的巧妙和方便易懂，让我以后在计算一些数据分布规律的时候提供了非常大的启发。

课程名称：计算机数值分析

实验项目名称	数值积分与微分			实验成绩	
实验者	杜俊	专业班级	软件 1805	实验日期	2020 年 4 月 25 日

第一部分　实验概述

一、实验目的

通过设计、编制、调试 2~3 个数值积分与微分算法的程序，加深对其数值计算方法及有关的基础理论知识的理解。

二、实验要求

用编程语言实现复化梯形积分、Romberg 积分的程序。

三、实验内容

用编程语言编程实现以下算法：

(1) 用复化梯形公式 $T_n = h(\frac{1}{2}f(a) + \sum_{i=1}^{n} f(a+ih) + \frac{1}{2}f(b))$ 的自动控制误差算法求积分 $I = \int_a^b f(x)dx$；

(2) Romberg 积分算法求积分 $I = \int_a^b f(x)dx$。

四、实验器材

(1) PC 机。

(2) 编程语言：MATLAB

第二部分　实验设计、测试与分析

五、问题：用复化梯形公式 $T_n = h(\frac{1}{2}f(a) + \sum_{i=1}^{n} f(a+ih) + \frac{1}{2}f(b))$ **的自动控制误差算法求积分** $I = \int_a^b f(x)dx$ 。

5.1　算法描述

（1）确定积分区间两端 a 和 b，以及分成的小区间数 n 和步长 h;

（2）给出各个小区间端点对应的函数值，再带入复合梯形公式，求出积分的近似值，输出。

5.2　程序变量说明

```
%积分区间 a 和 b
a=0;
b=2;
%小区间数
n=20;
%步长
h=(b-a)./n;
%小区间及区间端点对应值
X=a:0.1:b;
Y=sin(X);
```

5.3　源程序代码及运行结果截图

```
%积分区间 a 和 b
a=0;
b=2;
%小区间数
n=20;
%步长
h=(b-a)./n;
%小区间及区间端点对应值
X=a:0.1:b;
Y=sin(X);
%复合梯形公式求积分
T=h/2*(Y(1)+2.*sum(Y(2:n))+Y(n+1));
fprintf("积分准确值为：%.6f\n",cos(a)-cos(b));
fprintf("复合梯形公式求得的积分近似值为：%.6f\n",T);
```

截图：

积分准确值为：1.416147
复合梯形公式求得的积分近似值为：1.414967

六、问题：Romberg 积分算法求积分 $I=\int_a^b f(x)dx$。

6.1 算法描述

（1）给定原函数以及积分区间；

（2）建立算法需要的数表，先利用复合梯形公式计算数表的第一列并填入；

（3）再利用加速算法计算出输报表的后面列数并填入；

（4）数表最后一个元素即为最终近似值。

6.2 程序变量说明

```
%给定原函数
f=@(x) x.^1.5;
%给定积分区间端点值
a=0;
b=1;
%建立算法表
k=6;
T=zeros(k,k);
%步长初始值
h=b-a;
```

6.3 源程序代码及运行结果截图

```
%给定原函数
f=@(x) x.^1.5;
%给定积分区间端点值
a=0;
b=1;
%建立算法表
k=6;
T=zeros(k,k);
%先计算第一列
T(1,1)=(b-a)/2*(f(a)+f(b));
h=b-a;
for i=2:k  T(i,1)=T(i-1,1)/2+h/2*sum(f(a+(0:2^(i-2)-1)*h)+0.5*h);
```

```
    h=h/2;
end
%计算后面的列
for j=2:k
    for i=j:k
        T(i,j)=1/(2^j-1)*(2^j*T(i,j-1)-T(i-1,j-1));
    end
end
fprintf("算法得到的数表如下：\n");
disp(T);
fprintf("积分准确值为：%.6f\n",2/5);
fprintf("Romberg 积分算法得到的近似值为：%.6f\n",T(k,k));
```

截图：

```
算法得到的数表如下：
   0.500000000000000                   0                   0                   0                   0                   0
   0.500000000000000   0.500000000000000                   0                   0                   0                   0
   0.463388347648318   0.451184463531091   0.444210815464104                   0                   0                   0
   0.435203229253110   0.425808189788040   0.422183007824747   0.420714487315456                   0                   0
   0.418507847026542   0.412942719617686   0.411104795307635   0.410366247806495   0.410032433628786                   0
   0.409485624164295   0.406478216543546   0.405554716104383   0.405184710824166   0.405017564469898   0.404937963372137

积分准确值为：0.400000
Romberg积分算法得到的近似值为：0.404938
```

第三部分　实验小结与心得体会

通过此次实验，我学会了在被积函数比较难以求原函数的情况下，利用特殊巧妙的算法一样可以求得积分的数值，并且可以自己掌握要达到的精度。

这对以后的数学计算又提供了一种简单而又实用的方法。

课程名称：计算机数值分析

实验项目名称	常微分方程初值问题的数值解法			实验成绩	
实验者	杜俊	专业班级	软件 1805	实验日期	2020 年 4 月 29 日

第一部分　　实验概述

一、实验目的

通过设计、编制、调试 1~2 个求常微分方程初值问题的数值解解的程序，加深对其数值计算方法及有关的基础理论知识的理解。

二、实验要求

用编程语言实现用改进的欧拉（Euler）公式求解常微分方程初值问题、用四阶龙格-库塔(Runge-Kutta)方法求解常微分方程初值问题的程序。

三、实验内容

用编程语言编程实现以下算法：

(1) 用改进的欧拉（Euler）公式求解常微分方程初值问题。

(2) 用四阶龙格-库塔(Runge-Kutta)方法求解常微分方程初值问题。

四、实验器材

(1) PC 机。

(2) 编程语言：MATLAB

第二部分　　实验设计、测试与分析

一、问题：用改进的欧拉（Euler）公式求解常微分方程初值问题。

1.1　　算法描述

（1）给定求解的微分方程、离散区间、等分段数及初始值；

（2）利用改进的欧拉方法进行迭代求解，即先利用 yi 的值计算 yi+1 的预告值，在进一步利用 yi 的值和 yi+1 的预告值计算出 yi+1 的迭代值，如此循环；

（3）迭代到最后一个时结束，输出最后结果。

1.2　　程序变量说明

```
%定义微分方程
f=@(x,y) 1/x;
%原方程
F=@(x,y) log(x)
%定义离散区间
a=1;b=3;
%定义等分段数
n=30;
%定义步长
h=(b-a)/n;
%定义自变量因变量
x=a:h:b;
y=zeros(1,n+1);
%定义初始值
y(1)=log(x(1))

%定义自变量因变量
x=a:h:b;
y=zeros(1,n+1);
%定义初始值
y(1)=log(x(1));
```

1.3　　源程序代码及运行结果截图

```
%定义微分方程
f=@(x,y) 1/x;
%原方程
F=@(x,y) log(x)
%定义离散区间
a=1;b=3;
```

```
%定义等分段数
n=30;
%定义步长
h=(b-a)/n;
%定义自变量因变量
x=a:h:b;
y=zeros(1,n+1);
%定义初始值
y(1)=log(x(1));
%开始迭代
for i=1:n+1
    t=y(i)+h*f(x(i),y(i));
    y(i+1)=y(i)+h/2*(f(x(i),y(i))+t);
end
fprintf("微分方程近似迭代结果为：y(%.1f)=%.6f\n",x(n+1),y(n+1));
fprintf("准确结果为：y(%.1f)=%.6f\n",x(n+1),F(x(n+1),y(n+1)));
```

截图：

```
微分方程近似迭代结果为：y(3.0)=1.092978
准确结果为：y(3.0)=1.098612
```

二、问题：用四阶龙格-库塔(Runge-Kutta)方法求解常微分方程初值问题。

2.1 算法描述

（1）给定求解的微分方程、离散区间、等分段数及初始值；

（2）利用四阶龙格库塔方法的公式，利用 yi 的值分别迭代计算出 K1~K4 的值，在代入公式计算出 yi+1 的值，如此循环迭代；

（3）迭代结束，输出最后结果。

2.2 程序变量说明

```
%定义微分方程
f=@(x,y) cos(x);
%原方程
F=@(x,y) sin(x)
%定义离散区间
a=3;b=4;
%定义等分段数
n=25;
%定义步长
```

```
h=(b-a)/n;
%定义自变量因变量
x=a:h:b;
y=zeros(1,n+1);
%定义初始值
y(1)=sin(x(1));
```

2.3 源程序代码及运行结果截图

```
%定义微分方程
f=@(x,y) cos(x);
%原方程
F=@(x,y) sin(x)
%定义离散区间
a=3;b=4;
%定义等分段数
n=25;
%定义步长
h=(b-a)/n;
%定义自变量因变量
x=a:h:b;
y=zeros(1,n+1);
%定义初始值
y(1)=sin(x(1));
%开始迭代
for i=1:n+1
    K1=f(x(i)+y(i));
    K2=f(x(i)+h/2,y(i)+h/2*K1);
    K3=f(x(i)+h/2,y(i)+h/2*K2);
    K4=f(x(i)+h,y(i)+h*K3);
    y(i+1)=y(i)+h/6*(K1+2*K2+2*K3+K4);
end
fprintf("微分方程近似迭代结果为：y(%.1f)=%.6f\n",x(n+1),y(n+1));
fprintf("准确结果为：y(%.1f)=%.6f\n",x(n+1),F(x(n+1),y(n+1)));
```

截图：

```
微分方程近似迭代结果为：y(4.0)=-0.772649
准确结果为：y(4.0)=-0.756802
```

第三部分　　实验小结与心得体会

通过这次实验，我学会了利用多种巧妙的方法来解决常微分方程的初值问题，即在微分方程难以求解的情况下，在得知一个初值的时候，可以利用迭代的方法从后往前不断计算的出最终的计算值。

这对我以后在处理微分方程的问题时提供了非常大的帮助。

课程名称：计算机数值分析

实验项目名称	方程求根的数值方法			实验成绩	
实验者	杜俊	专业班级	软件 1805	实验日期	2020 年 5 月 1 日

第一部分　　实验概述

一、实验目的

通过设计、编制、调试 2~3 个用数值方法求方程 $f(x)=0$ 根的程序，加深对方程求根的数值计算方法及有关的基础理论知识的理解。

二、实验要求

用编程语言实现二分法、Newton 迭代法、弦截法求方程 $f(x)=0$ 根的程序。

三、实验内容

用编程语言编程实现以下算法：

(1) 用二分法求 $f(x)=0$ 的根。

(2) 用牛顿(Newton)迭代法求 $f(x)=0$ 在 x_0 附近的根。

(3) 用弦截法求 $f(x)=0$ 的根。

四、实验器材

(1) PC 机。

(2) 编程语言：MATLAB

第二部分　　实验设计、测试与分析

一、问题：用二分法求 $f(x)=0$ 的根。

1.1　算法描述

（1）给定函数方程、初始端点以及压达到的精度；

（2）然后开始二分法求解，先判断中点处函数值是否为零，是则结束，不是则判断中点处函数值与 a 出函数值乘积是否<0，是则将中点赋给 b，否则将中点赋给 a，再循环，直到 a-b 小于精度值；

（3）输出近似根。

1.2　程序变量说明

```
%定义所求函数
f=@(x) x^2-3;
%定义初始两端点
a=0;
b=4;
%定义要达到的精度
u=1e-7;
%近似解初始值
x=0;
```

1.3　源程序代码及运行结果截图

```
%定义所求函数
f=@(x) x^2-3;
%定义初始两端点
a=0;
b=4;
%定义要达到的精度
u=1e-7;
%近似解
x=0;
%开始二分
while abs(a-b)>u
    x=(a+b)/2;
    if f(x)==0
        break;
    end
    if f(a)*f(x)<0
```

```
        b=x;
    else
        a=x;
    end
end
fprintf("二分法得到方程的跟近似根为%.6f\n",x);
```

截图：

二分法得到方程近似根为1.732051

二、问题：用牛顿(Newton)迭代法求 $f(x)=0$ 在 x_0 附近的根。

2.1 算法描述

（1）给定所求方程、它的导函数以及初始迭代值；

（2）开始 Newton 迭代，即将初始值代入迭代公式计算出新的迭代值，判断是否达到精度，若没有则继续用新的迭代值代入，如此循环；

（3）输出近似根。

2.2 程序变量说明

```
%定义所求方程
f=@(x) log(x)-1;
%方程导函数
df=@(x) 1/x;
%定义初始迭代值
x=5;
%定义精度
u=1e-7;
%下一个迭代值的初始值
t=x;
```

2.3 源程序代码及运行结果截图

```
%定义所求方程
f=@(x) log(x)-1;
%方程导函数
df=@(x) 1/x;
%定义初始迭代值
x=5;
%定义精度
```

```
u=1e-7;
%开始 Newton 迭代
t=x;%下一个迭代值
for i=1:20
    x=t;
    t=x-f(x)/df(x);
    if abs(t-x)<=u
        break;
    end
end
fprintf("Newton 迭代法得到方程的近似根为%.6f\n",t);
```

截图：

Newton迭代法得到方程的近似根为2.718282

三、问题：用弦截法求 $f(x)=0$ 的根。

3.1 算法描述

（1）给定所求方程、精度、以及迭代的两个初始值；

（2）开始迭代，即利用开始的两个值代入弦截法公式计算出一个新的迭代值，判断是否达到精度，若没有，则将计算出来的新的值作为迭代值假如计算，如此循环；

（3）输出方程的近似根。

3.2 程序变量说明

```
%定义所求方程
f=@(x) log(x)-2;
%定义初始迭代的两个点
x1=8;
x2=5;
%定义精度
u=1e-7;
%新的迭代值的初始值
X=x2;
```

3.3 源程序代码及运行结果截图

```
%定义所求方程
f=@(x) log(x)-2;
%定义初始迭代的两个点
```

```
x1=8;
x2=5;
%定义精度
u=1e-7;
%开始弦截法
X=x2;%新的迭代值
for i=1:20
    X=x2-(x2-x1)/(f(x2)-f(x1))*f(x2);
    if abs(X-x2)<=u
        break;
    end
    x1=x2;
    x2=X;
end
fprintf("弦截法得到方程的近似根为%.6f\n",X);
```

截图：

```
弦截法得到方程的近似根为7.389056
```

第三部分　　实验小结与心得体会

通过此次实验，我学会了在难以用求根公式或者因式分解计算方程的根时，可以利用二分法、Newton 迭代、弦截法等众多巧妙的方法来通过不断地迭代来逐渐逼近方程的根，并且可以自由控制所求的精度，从而在数学工具的协助下计算出复杂方程的根。

这对我以后的处理方程求根的问题提供了很方便的技巧。