实验一　插值方法

严禁抄袭！

No plagiarism!

一. 实验目的

（1）熟悉数值插值方法的基本思想，解决某些实际插值问题，加深对数值插值方法的理解。

（2）熟悉 Matlab 编程环境，利用 Matlab 实现具体的插值算法，并进行可视化。

二. 实验要求

用 Matlab 软件实现 Lagrange 插值、分段线性插值、Hermite 插值、Aitken 逐步插值算法，并用实例在计算机上计算和作图。

三. 实验内容

1. 实验题目

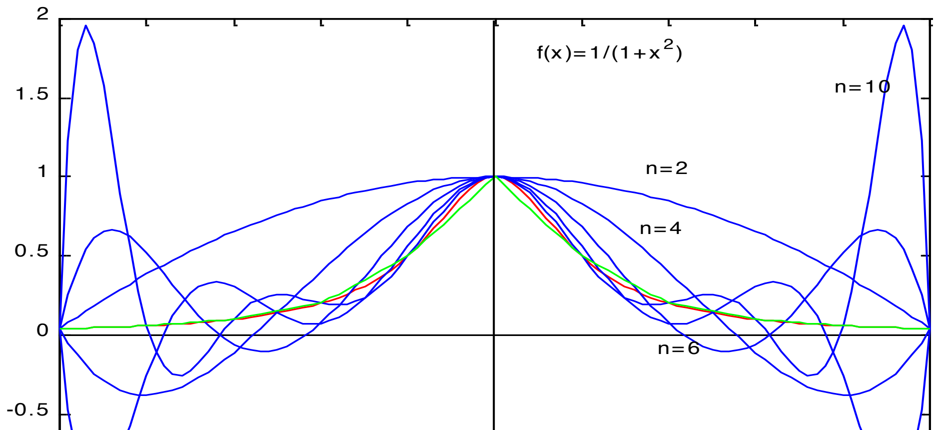
3-1：已知正弦积分$f(x)=-\int_x^{\infty}\frac{sint}{t}dt$的数据表

x	0.3	0.4	0.5	0.6	0.7
y	0.29850	0.39646	0.49311	0.58813	0.68122

构造适合该数据表的一次、二次和三次 Lagrange 插值公式，计算 x=0.358, 0.462, 0.514, 0.635 时 $f(x)$ 的值，比较不同次数的插值公式的计算结果。

3-2：仿照附录 C 中“文件 1.2 逐步插值”程序（Neville 算法）编写相应的 Aitken 逐步插值算法的程序，根据实验题目 3-1 中所给数据，分别利用上述两种算法求正弦积分 $f(x)$ 在 x=0.358, 0.462, 0.514, 0.635 处的值，比较两种算法的计算结果，并与 3-1 中的计算结果进行比较。

3-3：对于函数$f(x)=\frac{1}{1+x^2}$，在利用 Lagrange 插值方法进行插值时，随着插值次数的增大，会出现如下图中所示的 Runge 现象：



要求：

（1）利用 Lagrange 插值方法验证 Runge 现象；

（2）将区间[-5,5]分为 n 等份（n=5,10,20），做$f(x)=\frac{1}{1+x^2}$的 Lagrange 分段线性插值函数 $L_5(x)$、$L_{10}(x)$、$L_{20}(x)$，考察上述三种插值在 x=-4.8、4.8 处的误差，并分析。

2. 设计思想

要求针对上述题目，详细分析每种算法的设计思想。

3-1:

Lagrange 具有累加的嵌套结构，容易编制其计算程序。在逻辑上表现为二重循环，内循环累乘求得系数，然后再通过外循环累加得出插值结果 y。

3-2:

Aitken 插值是对三步插值转化为两步插值的重复，先将前两个插值点插值生成新的数据，然后与第三个插值点进行新的两点插值，不断重复这个插值过程，每一步增加一个新的节点，直到遍历所有节点为止，最终获得与原函数更加接近的插值函数。

Neville 插值的基本思想和 Aitken 插值一样，不同的是 Neville 插值每次选取的两个插值节点都是上一步相邻节点插值后得到的，而不是新的插值节点，这样得到的插值函数和原函数更加接近。

3-3:

分段线性插值： 分段插值是将被插值函数逐步多项式化。分段插值的处理过程分两步，将区间分成几个子段，并在每个子段上构造插值多项式装配在一起，作为整个区间的插值函数。在分化的每个节点给出数据，连接相邻节点得一折线，该折线函数可以视作插值问题的解。

3. 对应程序

列出每种算法的程序。

3-1:

```
function[result] = Lagrange_eval(X,Y,x0)
m = length(X);
N = zeros(m,1);
```

```
len = length(x0);
result = [];
for q = 1:len
    y0 = 0;
    for i = 1:m
        N(i)=1;
        for j = 1:m
            if j ~=i
                N(i)=N(i)*(x0(q)-X(j))/(X(i)-X(j));
            end
        end
    y0 = y0 + Y(i)*N(i);
    end
    result = [result y0];
end

X = [0.3 ,0.4];
Y = [0.29850,0.39646];
x0 = [0.358,0.462,0.514,0.635];
disp('一次插值：y0=')
disp(Lagrange_eval(X,Y,x0));

X = [0.3,0.4,0.5];
Y = [0.29850,0.39646,0.49311];
x0 = [0.358,0.462,0.514,0.635];
disp('二次插值：y0=')
disp(Lagrange_eval(X,Y,x0));

X = [0.3 ,0.4,0.5,0.6];
Y = [0.29850,0.39646,0.49311,0.58813];

x0 = [0.358,0.462,0.514,0.635];
disp('三次插值：y0=')
disp(Lagrange_eval(X,Y,x0));
```

3-2:

```
function y0 = Aitken_eval(X,Y,x0)
m=length(X);
n=length(x0);
y0=[];
for q = 1:n
    P=Y;
    ct = 0;
    for i = 1:m
        ct = ct+1;
```

```
        P1=P;
        for j=i+1:m
            P(j)=((x0(q)-X(i))*P1(j)-((x0(q)-X(j))*P1(i)))/(X(j)-X(i));
        end
        if abs(P(m)-P(m-1))<10^-6
            a = P(m);
            break;
        end
    end
    if ct == m
        y0=[y0 P(m)];
    else
        y0=[y0 a];
    end
end
function y0 = Neville_eval(X,Y,x0)
m=length(X);
n=length(x0);
y0=[];
for q = 1:n
    P=Y;
    ct = 0;
    for i = 1:m
        ct = ct + 1;
        k=1;
        P1=P;
        for j=i+1:m
            k=k+1;
            P(j)=P1(j-1)+(P1(j)-P1(j-1))*(x0(q)-X(k-1))/(X(j)-X(k-1));
        end
        if abs(P(m)-P(m-1))<10^-6
            a = P(m);
            break;
        end
    end
    if ct == m
        y0=[y0 P(m)];
    else
        y0=[y0 a];
    end
end
X = [0.3 ,0.4,0.5,0.6,0.7];
Y = [0.29850,0.39646,0.49311,0.58813,0.68122];
x0 = [0.358,0.462,0.514,0.635];
```

```
disp('Aitken algorithm: y0=')
disp(Aitken_eval(X,Y,x0))
disp('Neville algorithm: y0=')
disp(Neville_eval(X,Y,x0))
```

3-3:

```
x0=linspace(-5,5,1000);
x = linspace(-5,5,3);
y = 1./(1+x.^2);
for i=1:1000
   y0(i)=Lagrange_eval(x,y,x0(i));
end
plot(x0,y0,'-')
hold on

x = linspace(-5,5,5);
y = 1./(1+x.^2);
for i=1:1000
   y0(i)=Lagrange_eval(x,y,x0(i));
end
plot(x0,y0,'-')
hold on

x = linspace(-5,5,7);
y = 1./(1+x.^2);
for i=1:1000
   y0(i)=Lagrange_eval(x,y,x0(i));
end
plot(x0,y0,'-')
hold on

x = linspace(-5,5,9);
y = 1./(1+x.^2);
for i=1:1000
   y0(i)=Lagrange_eval(x,y,x0(i));
end
plot(x0,y0,'-')
hold on

x = linspace(-5,5,11);
y = 1./(1+x.^2);
for i=1:1000
   y0(i)=Lagrange_eval(x,y,x0(i));
end
plot(x0,y0,'-')
```

```
legend('n=2','n=4','n=6','n=8','n=10')
hold off

clear
clc
disp('exact value: ')
disp(1/(1+4.8^2))

x = [-5,-3];
y = 1./(1+x.^2);
disp('n=5, x=-4.8, y=:')
disp(Lagrange_eval(x,y,-4.8))
x = [3,5];
y = 1./(1+x.^2);
disp('n=5, x=4.8, y=:')
disp(Lagrange_eval(x,y,4.8))
x = [-5,-4];
y = 1./(1+x.^2);
disp('n=10, x=-4.8, y=:')
disp(Lagrange_eval(x,y,-4.8))
x = [4,5];
y = 1./(1+x.^2);
disp('n=10, x=4.8, y=:')
disp(Lagrange_eval(x,y,4.8))
x = [-5,-4.5];
y = 1./(1+x.^2);
disp('n=20, x=-4.8, y=:')
disp(Lagrange_eval(x,y,-4.8))
x = [4.5,5];
y = 1./(1+x.^2);
disp('n=20, x=4.8, y=:')
disp(Lagrange_eval(x,y,4.8))
```

4. **实验结果**

列出相应的运行结果截图，如果要求可视化，则同时需要给出相应的图形。

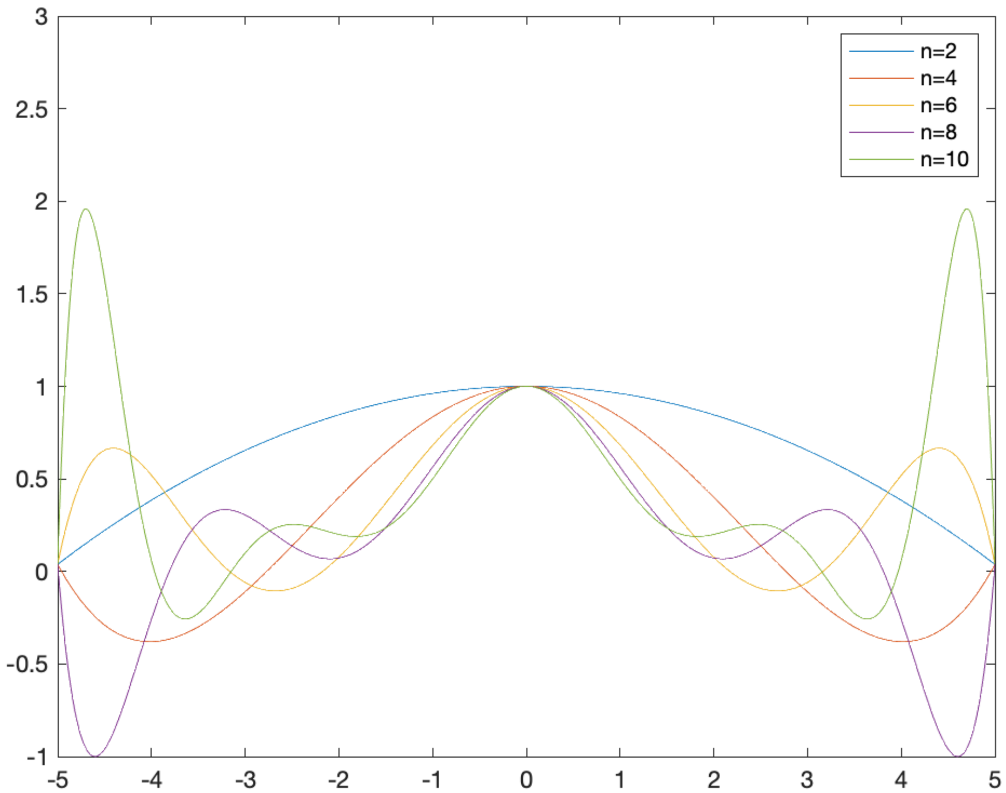
3-1

```
一次插值：y0=
   0.355316800000000   0.457195200000000   0.508134400000000   0.626666000000000

二次插值：y0=
   0.355476358000000   0.456537318000000   0.506536462000000   0.621509512500000

三次插值：y0=
   0.355457909360000   0.456557673840000   0.506518246320000   0.620942692500000
```

3-2

```
Aitken algorithm: y0=
   0.355458197620000   0.456557355780000   0.506518530940000   0.620945792296875

Neville algorithm: y0=
   0.355457211770800   0.456558112762800   0.506517788800000   0.620945792296875
```

3-3



```
exact value:
   0.041597337770383

n=5, x=-4.8, y=:
   0.044615384615385

n=5, x=4.8, y=:
   0.044615384615385

n=10, x=-4.8, y=:
   0.042533936651584

n=10, x=4.8, y=:
   0.042533936651584

n=20, x=-4.8, y=:
   0.041900452488688

n=20, x=4.8, y=:
   0.041900452488688
```

四. 实验体会

对实验过程进行总结，分析比较各插值算法的效率和精度差异，指出每种算法的设计要点及应注意的事项，以及自己通过实验所获得的对插值方法的理解。

Lagrange 插值模型简单，结构紧凑，但在高次插值时的误差比较大，并且由于拉格朗日的插值多项式和每个节点都有关，当改变节点个数时，需要重新计算。且当增大插值阶数时容易出现 Runge 现象。

Neville 插值的基本思想和 Aitken 插值一样，不同的是 Neville 插值每次选取的两个插值节点都是上一步的相邻节点。Aitken 插值是对三步插值转化为两步插值的重复，先将前两个插值点插值生成新的数据，然后与第三个插值点进行新的两点插值，不断重复这个插值过程，每一步增加一个新的节点，直到遍历所有节点为止，最终获得与原函数更加接近的插值函数。

分段线性插值是将整个区间分成许多小段，运用一次插值，从而提高精度。分段线性插值算法简单，计算量小，但精度不高。

（注：不要改变实验报告的结构，写清页码和题号，源程序以自己的中文姓名命名，如 3-1 题可命名为“张三_3-1.m”，运行截图中应出现自己的姓名和题号）