实习题目	
算法程序	
数值结果	
理论分析	
总　　分	

矩阵与数值分析课程数值试验报告

学　　院：软件学院

姓　　名：李 荣 达

学　　号：32017019

任课教师：程 明 松

2020 年 12 月

二.线性方程组求解

1.分别用Gauss消元法和列主元消去法编程求解方程组Ax=b，其中

```
A=[32,-13,0,0,0,-10,0,0,0;

  -13,35,-9,0,-11,0,0,0,0;

  0,-9,31,-10,0,0,0,0,0;

  0,0,-10,79,-30,0,0,0,-9;

  0,0,0,-30,57,-7,0,-5,0;

  0,0,0,0,-7,47,-30,0,0;

  0,0,0,0,0,30,41,0,0;

  0,0,0,0,-5,0,0,27,-2;

  0,0,0,-9,0,0,0,-2,29]

b=[-15;27;-23;0;-20;12;-7;7;10]
```

并求出矩阵A的LU分解及列主元的LU分解（求出L，U和P），并用LU分解的方法求A的逆矩阵及A的行列式.

解:

（1） Gauss消元法:

算法公式：$(A|b) \longrightarrow (U|c)$

Matlab代码:

```
%% Gauss消元法
function [res] = GaussElimination(A,b)
%  inputs:
%         A:系数矩阵，为n*n维方阵
%         b:载荷矩阵，为n*1维矩阵
%  outputs:
%         res:计算结果向量,为n*1为矩阵

% 判断输入矩阵维度是否满足要求
[row_A,col_A] = size(A);
[row_b,~] = size(b);

% 初始化r_matrix矩阵
res = zeros(row_b,1);
% 判断输入的维度是否满足要求
if (row_A ~= col_A) || (row_A ~= row_b)
    % 不满足则输出错误提示
    print('输入错误! ');
else
    % 满足则进行下一步运算
    n=row_A;
    for k = 1:n-1
        % 检查主对角元素第k行的第k个元素是否为0
        if A(k,k) == 0
            print('主对角元素错误! ');
```

```
        else
            % 循环计算第k+1行到最后一行
            for i = k+1:n
                beishu = A(i,k) / A(k,k); %更新倍数
                % 更新每一行从第i个元素开始后的所有元素
                for j = k:n
                    A(i,j) = A(i,j) - beishu*A(k,j); % 更新a(i,j)
                end
                b(i,1) = b(i,1) - b(k,1)*beishu; % 更新b(i)
            end %for循环结束
        end % 条件2结束
    end % for循环结束
    % 回代过程,从最后一行开始
    res(n,1) = b(n,1)/A(n,n);
    for k = row_A-1:-1:1%倒着走
        sum_temp = 0;
        for j = k+1:col_A
            sum_temp = sum_temp + A(k,j)*res(j);
        end
        res(k,1) = (b(k,1) - sum_temp)/A(k,k);
    end
end % 条件判断结束
end % 函数结束
```

运行结果截图：

```
调用Gauss消去法:

res_Gauss =

   -0.3188
    0.3273
   -0.7205
   -0.2282
   -0.4469
    0.0544
   -0.2105
    0.1978
    0.2877
```

（2）列主元Gauss消去法：

算法公式：$(A|b) \longrightarrow (U|c)$，消元的时候选择主对角元及其以下的元素中绝对值最大的元素，进行行变换，继续消元.

Matlab代码：

```
%% 列主元Gauss消元法
function [res] = L_GaussElimination(A,b)
%  inputs:
%         A:系数矩阵，为n*n维方阵
%         b:载荷矩阵，为n*1维矩阵
%  outputs:
%         res:计算结果向量,为n*1为矩阵

% 判断输入矩阵维度是否满足要求
[row_A,col_A] = size(A);
[row_b,~] = size(b);

% 初始化r_matrix矩阵
res = zeros(row_b,1);
% 判断输入的维度是否满足要求
if (row_A ~= col_A) || (row_A ~= row_b)
    % 不满足则输出错误提示
```

```
    print('输入错误! ');
else
    % 满足则进行下一步运算
    for k = 1:row_A-1
        % 找主对角元及以下的最大值进行交换
        [~,rows]=max(abs(A(k:row_A,k)));
        rows=rows+k-1; %rows转换为在矩阵中的行号
        A([k rows],:)=A([rows k],:);%交换矩阵A的两行
        b([k,rows],1)=b([rows,k],1); %交换向量b的两行
        % 循环计算第k+1行到最后一行
        for i = k+1:row_A
           beishu = A(i,k) / A(k,k); %更新倍数
           % 更新每一行从第i个元素开始后的所有元素
           for j = k:col_A
             A(i,j) = A(i,j) - beishu*A(k,j); % 更新a(i,j)
           end
             b(i,1) = b(i,1) - b(k,1)*beishu; % 更新b(i)
        end %for循环结束
    end % for循环结束
    % 回代过程,从最后一行开始
    res(row_A) = b(row_A)/A(row_A,col_A);
    for k = row_A-1:-1:1%倒着走
        sum_temp = 0;
        for j = k+1:col_A
            sum_temp = sum_temp + A(k,j)*res(j);
        end
        res(k,1) = (b(k,1) - sum_temp)/A(k,k);
    end
end % 条件判断结束
end % 函数结束
```

运行结果截图：

调用列主元的高斯消去法：

```
res_L_Gauss =

   -0.3188
    0.3273
   -0.7205
   -0.2282
   -0.4469
    0.0544
   -0.2105
    0.1978
    0.2877
```

(3) LU分解：

算法公式：$A = LU$，L为单位下三角，U为上三角

Matlab代码：

```
%%LU分解
function [L,U] = LU(A)
%  inputs:
%         A:系数矩阵，为n*n维方阵
%  outputs:
%         L:单位下三角矩阵，n*n矩阵
%         U:上三角矩阵，n*n矩阵
%获得矩阵的维度
[row_A,col_A] = size(A);
% 判断输入的维度是否满足要求
if (row_A ~= col_A)
```

```
    % 不满足则输出错误提示
    print('输入错误! ');
else
    % 满足则进行下一步运算
     n=row_A;
     L=zeros(n,n);
     U=zeros(n,n);
     for i=1:row_A
         L(i,i)=1;
     end
     for k = 1:n-1
        % 检查主对角元素第k行的第k个元素是否为0
        if A(k,k) == 0
            print('主对角元素错误! ');
        else
            % 循环计算第k+1行到最后一行
            for i = k+1:n
               beishu = A(i,k) / A(k,k); %更新倍数
               L(i,k)=beishu;
               % 更新每一行从第i个元素开始后的所有元素
               for j = k:n
                   A(i,j) = A(i,j) - beishu*A(k,j); % 更新a(i,j)
               end
            end %for循环结束
        end % 条件2结束
    end % for循环结束
    U=A;
end
```

运行结果截图：

```
调用LU分解:

L =

  列 1 至 7

    1.0000         0         0         0         0         0         0
   -0.4063    1.0000         0         0         0         0         0
         0   -0.3028    1.0000         0         0         0         0
         0         0   -0.3537    1.0000         0         0         0
         0         0         0   -0.3975    1.0000         0         0
         0         0         0         0   -0.1569    1.0000         0
         0         0         0         0         0    0.6540    1.0000
         0         0         0         0   -0.1121   -0.0175   -0.0087
         0         0         0   -0.1193   -0.0834   -0.0142   -0.0070

  列 8 至 9

         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
    1.0000         0
   -0.0917    1.0000

U =

  列 1 至 7

   32.0000  -13.0000         0         0         0  -10.0000         0
         0   29.7188   -9.0000         0  -11.0000   -4.0625         0
         0         0   28.2744  -10.0000   -3.3312   -1.2303         0
         0         0         0   75.4632  -31.1782   -0.4351         0
         0         0         0         0   44.6053   -7.1730         0
         0         0         0         0         0   45.8743  -30.0000
         0         0         0         0         0         0   60.6188
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  列 8 至 9

         0         0
         0         0
         0         0
         0   -9.0000
   -5.0000   -3.5779
   -0.7847   -0.5615
    0.5131    0.3672
   26.4302   -2.4077
         0   27.4021
```

（4）列主元LU分解：

算法公式：$PA = LU$，L为单位下三角，U为上三角，P为初等置换矩阵

Matlab代码：

```
%% 列主元的LU分解，PA=LU
function [L,U,P] = LUP(A)
%  inputs:
%         A:系数矩阵，为n*n维方阵
%  outputs:
%         L:单位下三角矩阵，n*n矩阵
%         U:上三角矩阵，n*n矩阵
```

```
%          P:置换矩阵
% 判断输入矩阵维度是否满足要求
[row_A,col_A] = size(A);

% 判断输入的维度是否满足要求
if (row_A ~= col_A)
    % 不满足则输出错误提示
    print('输入错误! ');
else
    % 满足则进行下一步运算
    %初始化L、U、P
    n=row_A;
    L=zeros(n,n);
    U=A;
    P=eye(n);
    for i=1:row_A
         L(i,i)=1;
    end
    for k = 1:row_A-1
        % 找主对角元及以下绝对值最大的进行交换
        [~,rows]=max(abs(U(k:row_A,k)));
        rows=rows+k-1; %rows转换为在矩阵中的行号
        U([k rows],:)=U([rows k],:);%交换矩阵A的两行
        P([k,rows],:)=P([rows,k],:);%交换矩阵P的两行
        %根据计算过程，L同时进行行列变换，最后仍然是单位下三角
        L([k,rows],:)=L([rows,k],:);%交换矩阵L的两行
        L(:,[k,rows])=L(:,[rows,k]);%交换矩阵L的两列
        % 循环计算第k+1行到最后一行
        for i = k+1:row_A
           beishu = U(i,k) / U(k,k); %更新倍数
           L(i,k)=beishu;
           % 更新每一行从第i个元素开始后的所有元素
           for j = k:col_A
             U(i,j) = U(i,j) - beishu*U(k,j); % 更新U(i,j)
           end
        end %for循环结束
    end % for循环结束
end % 条件判断结束
end % 函数结束
```

运行结果截图：

调用带有列主元的LU分解：

```
L =

  列 1 至 7

    1.0000         0         0         0         0         0         0
   -0.4063    1.0000         0         0         0         0         0
         0   -0.3028    1.0000         0         0         0         0
         0         0   -0.3537    1.0000         0         0         0
         0         0         0   -0.3975    1.0000         0         0
         0         0         0         0   -0.1569    1.0000         0
         0         0         0         0         0    0.6540    1.0000
         0         0         0         0   -0.1121   -0.0175   -0.0087
         0         0         0   -0.1193   -0.0834   -0.0142   -0.0070

  列 8 至 9

         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
    1.0000         0
   -0.0917    1.0000

U =

  列 1 至 7

   32.0000  -13.0000         0         0         0  -10.0000         0
         0   29.7188   -9.0000         0  -11.0000   -4.0625         0
         0         0   28.2744  -10.0000   -3.3312   -1.2303         0
         0         0         0   75.4632  -31.1782   -0.4351         0
         0         0         0         0   44.6053   -7.1730         0
         0         0         0         0         0   45.8743  -30.0000
         0         0         0         0         0         0   60.6188
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  列 8 至 9

         0         0
         0         0
         0         0
         0   -9.0000
   -5.0000   -3.5779
   -0.7847   -0.5615
    0.5131    0.3672
   26.4302   -2.4077
         0   27.4021

P =

     1     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0
     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     1
```

（5）LU分解的方法求A的逆矩阵及A的行列式：

算法公式：$det(A)=det(L)det(U)=det(U);AA^{-1}=I,A^{-1}=X,LUX=I,LY=I,UX=Y.$

Matlab代码：

```
%利用LU分解求A的行列式和逆矩阵
function [rever_A,det_A]=reverse_and_det(A)
%%LU分解
%  inputs:
%         A:系数矩阵，为n*n维方阵
%  outputs:
%         A的逆
%         det(A)
%获得矩阵的维度
[row_A,col_A] = size(A);
% 判断输入的维度是否满足要求
if (row_A ~= col_A)
    % 不满足则输出错误提示
    print('输入错误！');
else
    % 满足则进行下一步运算
     n=row_A;
     L=zeros(n,n);
     U=zeros(n,n);
     for i=1:row_A
         L(i,i)=1;
     end
     for k = 1:n-1
        % 检查主对角元素第k行的第k个元素是否为0
        if A(k,k) == 0
            print('主对角元素错误！');
        else
            % 循环计算第k+1行到最后一行
            for i = k+1:n
               beishu = A(i,k) / A(k,k); %更新倍数
               L(i,k)=beishu;
               % 更新每一行从第i个元素开始后的所有元素
               for j = k:n
                   A(i,j) = A(i,j) - beishu*A(k,j); % 更新a(i,j)
               end
            end %for循环结束
        end % 条件2结束
    end % for循环结束
    U=A;
    det_A=1;
    %A的行列式等于U的对角元的乘积
    for i=1:n
        det_A=det_A*U(i,i);
    end
    I=eye(n);
    Y=zeros(n,n);
     % 回代过程,从最后一行开始
    L=L'%转置变为上三角求解将结果转置即可得到答案I=LY
    for i=1:n
        Y(n,i) = I(n,i)/L(n,n);
        for k = n-1:-1:1%倒着走
            sum_temp = 0;
            for j = k+1:n
                sum_temp = sum_temp + L(k,j)*Y(j,i);
            end
            Y(k,i) = (I(k,i) - sum_temp)/L(k,k);
        end
```

```
    end
    Y=Y';%Y=UX Y转换为正确的下三角
    X=zeros(n,n);
    %求A的逆矩阵
    for i=1:n
        X(n,i) = Y(n,i)/U(n,n);
        for k = n-1:-1:1%倒着走
            sum_temp = 0;
            for j = k+1:n
                sum_temp = sum_temp + U(k,j)*X(j,i);
            end
            X(k,i) = (Y(k,i) - sum_temp)/U(k,k);
        end
    end
    rever_A=X;
end%函数结束
```

运行结果截图：

```
rever_A =

  列 1 至 7

    0.0375    0.0153    0.0053    0.0028    0.0052    0.0060    0.0044
    0.0153    0.0375    0.0128    0.0060    0.0110    0.0033    0.0024
    0.0047    0.0115    0.0380    0.0076    0.0065    0.0013    0.0010
    0.0008    0.0019    0.0064    0.0180    0.0102    0.0011    0.0008
    0.0004    0.0010    0.0035    0.0098    0.0236    0.0025    0.0018
    0.0000    0.0001    0.0004    0.0010    0.0024    0.0148    0.0108
   -0.0000   -0.0001   -0.0003   -0.0007   -0.0018   -0.0108    0.0165
    0.0001    0.0002    0.0008    0.0022    0.0046    0.0005    0.0004
    0.0002    0.0006    0.0020    0.0057    0.0035    0.0004    0.0003

  列 8 至 9

    0.0010    0.0009
    0.0022    0.0020
    0.0014    0.0024
    0.0023    0.0057
    0.0046    0.0034
    0.0005    0.0003
   -0.0003   -0.0002
    0.0381    0.0033
    0.0033    0.0365


det_A =

   1.8229e+14
```

（6）函数调用部分的代码：

```
%调用函数
a=[32,-13,0,0,0,-10,0,0,0;
   -13,35,-9,0,-11,0,0,0,0;
   0,-9,31,-10,0,0,0,0,0;
   0,0,-10,79,-30,0,0,0,-9;
   0,0,0,-30,57,-7,0,-5,0;
   0,0,0,0,-7,47,-30,0,0;
   0,0,0,0,0,30,41,0,0;
   0,0,0,0,-5,0,0,27,-2;
   0,0,0,-9,0,0,0,-2,29]
b=[-15;27;-23;0;-20;12;-7;7;10]
disp('调用Gauss消去法:')
res_Gauss=GaussElimination(a,b)
```

```
disp('调用列主元的高斯消去法:')
res_L_Gauss=L_GaussElimination(a,b)
disp('调用LU分解:')
[L,U]=LU(a)
disp('调用带有列主元的LU分解:')
[L,U,P]=LUP(a)
disp('调用reverse_and_det输出矩阵的逆和行列式:')
[rever_A,det_A]=reverse_and_det(a)
```

（7）误差分析：误差来自计算过程中不能整除产生。

2.编制程序求解矩阵A的Cholesky分解，并用程序求解方程组Ax=b，其中

```
A=[7 1 -5 1;1 9 2 7;-5 2 7 -1;1 7 -1 9]
b=[13;-9;6;0]
```

（1）算法公式：$A=LL^T, LL^T=b, Ly=b, L^Tx=y$

（2）Matlab代码

```
%求矩阵A的Cholesky分解
function [L,x]=cholesky(A,b)
%  inputs:
%         A:系数矩阵，为n*n维方阵
%         b:n*1
%  outputs:
%         L:下三角矩阵，n*n矩阵
%         x:线性方程组的解
% 判断输入矩阵维度是否满足要求
[row_A,col_A] = size(A);

% 判断输入的维度是否满足要求
if (row_A ~= col_A)
    % 不满足则输出错误提示
    print('输入错误! ');
else
    % 满足则进行下一步运算
    %初始化L、U、P
    n=row_A;
    L=zeros(n,n);
    L(1,1)=sqrt(A(1,1));
    for i=2:n
        L(i,1)=A(i,1)/L(1,1);
    end
    for i=2:n%逐列计算
        sum=0;
        for k=1:i-1
            sum=sum+L(i,k)*L(i,k);
        end
        L(i,i)=sqrt(A(i,i)-sum);
        for j=i+1:n %从对角元后一个元素计算到该列末尾
            sum=0;
            for k=1:i-1
                sum=sum+L(i,k)*L(j,k);
            end
            L(j,i)=(A(i,j)-sum)/L(i,i);
        end
    end  % end cholesky分解
```

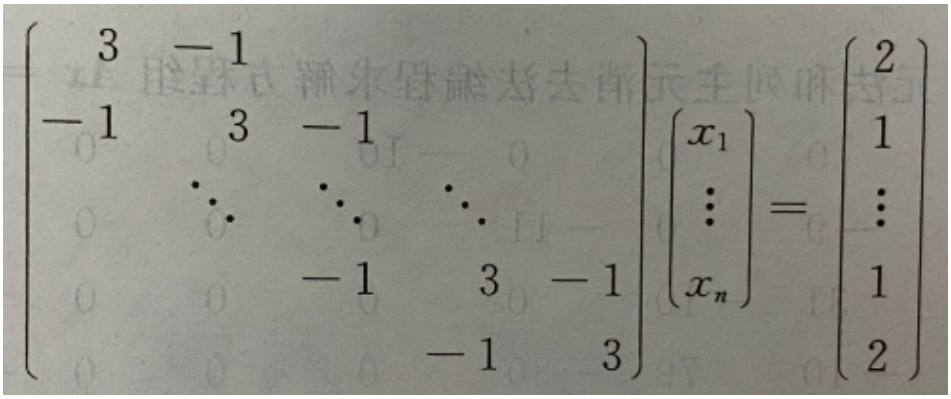
```
    %求解方程组
    Y=zeros(n,1);
    %LY=b
    Y(1) = b(1)/L(1,1);
    for k = 2:n
        sum_temp = 0;
        for j = 1:k-1
            sum_temp = sum_temp + L(k,j)*Y(j);
        end
        Y(k) = (b(k) - sum_temp)/L(k,k);
    end
    %求x
    L_=L'%L_*x=Y
    x=zeros(n,1);
    x(n) = Y(n)/L_(n,n);
    for k = n-1:-1:1%倒着走
        sum_temp = 0;
        for j = k+1:n
        sum_temp = sum_temp + L_(k,j)*x(j);
        end
        x(k) = (Y(k) - sum_temp)/L_(k,k);
    end
end
%调用Cholesky函数:
disp('Cholesky分解')
d=[7 1 -5 1;1 9 2 7;-5 2 7 -1;1 7 -1 9]
e=[13;-9;6;0]
[L,x_]=cholesky(d,e)
```

运行结果截图:

```
L_ =

    2.6458    0.3780   -1.8898    0.3780
         0    2.9761    0.9120    2.3041
         0         0    1.6115   -1.4813
         0         0         0    1.1636


L =

    2.6458         0         0         0
    0.3780    2.9761         0         0
   -1.8898    0.9120    1.6115         0
    0.3780    2.3041   -1.4813    1.1636


x_ =

   19.0780
  -21.8716
   23.2294
   17.4725
```

> 其中x_为方程组的解

7.已知方程组



分别用Jacobi迭代法和Gauss-Seidel迭代法求解方程组，精确到小数点后6位，分别就n=10，20，30，50，100给出相应的计算结果。

（1）Jacbi迭代法

算法公式：$x^{(k+1)} = D^{-1}(L+U)x^{(k)} + D^{-1}b, k = 0, 1, 2...$

Matlab代码：

```
function x=jacobi(A,b)
%  inputs:
%          A:系数矩阵，为n*n维方阵
%          b:载荷矩阵，为n*1维矩阵
%  outputs:
%          x:计算结果向量,为n*1为矩阵

% 判断输入矩阵维度是否满足要求
[row_A,col_A] = size(A);
[row_b,~] = size(b);
format long %为了输出6位小数
% 初始化r_matrix矩阵
x = zeros(row_b,1);
% 判断输入的维度是否满足要求
if (row_A ~= col_A) || (row_A ~= row_b)
    % 不满足则输出错误提示
    print('输入错误! ');
else
    % 满足则进行下一步运算
    n=row_A;
    D=zeros(n,n);%对角矩阵
    B_j=zeros(n,n);
    for i=1:n
        D(i,i)=1/A(i,i)
    end %D的对角元为A的对角元
    f_j=D*b; %完成对f的赋值
    for i=1:n
        for j=1:n
            B_j(i,j)=-A(i,j);
        end
    end
    for i=1:n
        B_j(i,i)=0;
    end
    B_j=D*B_j; %完成对B的赋值
    norm=Inf;
    e=1e-5;
    while norm>e %退出循环的条件
        x_=x;
        x=B_j*x+f_j;
```

```
        norm=abs(x-x_);
    end
    x=vpa(x,6) ; %小数点后6位
end
```

(2) Gauss-Seidel迭代法

算法公式：$x^{(k+1)}=(D-L)^{-1}Ux^{(k)}+(D-L)^{-1}b$

Matlab代码：

```
%Gauss-Seidel迭代法
function x=Gauss_Seidel(A,b)
%  inputs:
%         A:系数矩阵，为n*n维方阵
%         b:载荷矩阵，为n*1维矩阵
%  outputs:
%         x:计算结果向量,为n*1为矩阵

% 判断输入矩阵维度是否满足要求
[row_A,col_A] = size(A);
[row_b,~] = size(b);
format long %为了输出6位小数
% 初始化r_matrix矩阵
x = zeros(row_b,1);
% 判断输入的维度是否满足要求
if (row_A ~= col_A) || (row_A ~= row_b)
    % 不满足则输出错误提示
    print('输入错误! ');
else
    % 满足则进行下一步运算
    n=row_A;
    U=zeros(n,n);%对角矩阵
    B_g=zeros(n,n);
    for i=1:n
        for j=i+1:n
            U(i,j)=-A(i,j);
        end
    end

    for i=1:n
        for j=1:n
            B_g(i,j)=-A(i,j);
        end
    end
    for i=1:n
        for j=1:i
            B_g(i,j)=A(i,j);
        end
    end
    f_g=B_g\b %完成对f的赋值
    B_g=B_g\U %完成对B的赋值
    norm=Inf;
    e=1e-5;
    while norm>e %退出循环的条件
        x_=x;
        x=B_g*x+f_g;
        norm=abs(x-x_);
    end
```

```
    x=vpa(x,6) ; %小数点后6位
end
```

（3）函数调用：

```
n=10 %20,30,50,100
a=zeros(n,n);
a(1,1)=3;
a(1,2)=-1;
a(n,n-1)=-1;
a(n,n)=3
for i=2:n-1
    a(i,i-1)=-1;
    a(i,i)=3;
    a(i,i+1)=-1;
end
b=zeros(n,1);
b(1,1)=2;
b(n,1)=2;
for i=2:n-1
    b(i,1)=1;
end
print('Jacobi迭代输出结果：')
x_j=jacobi(a,b)
print("Gauss-Seidel迭代输出结果：")
x_g=Gauss_Seidel(a,b)
```

（4）输出结果截图

x_j表示Jacobi迭代法输出的结果，x_g表示Gauss-Seidel迭代法的输出结果.

n=10：

```
x_j =      x_g =

0.999988   0.666656
0.999976   0.555538
0.999967   0.518494
 0.99996   0.506145
0.999957   0.502031
0.999957   0.500665
 0.99996   0.500215
0.999967   0.500068
0.999976   0.500022
0.999988   0.833341
```

n=20:

```
x_j =      x_g =

0.999986   0.666652
0.999971   0.555537
0.999959   0.518493
0.999947   0.506152
0.999938   0.502028
 0.99993   0.500666
0.999925   0.500198
 0.99992   0.500055
0.999918   0.499999
0.999916   0.499981
0.999916   0.499979
0.999918   0.499977
 0.99992   0.499973
0.999925   0.499971
 0.99993   0.499973
0.999938   0.499979
0.999947   0.499986
0.999959   0.499992
0.999971   0.499996
0.999986   0.833332
```

n=30:

```
x_j =      x_g =

0.999986   0.666653
0.999971   0.555536
0.999959   0.518496
0.999947   0.506149
0.999938   0.502033
 0.99993   0.500661
0.999924   0.500203
0.999919   0.500051
0.999917         0.5
0.999914   0.499983
0.999913   0.499977
0.999912   0.499977
0.999911   0.499973
0.999911   0.499978
0.999911    0.49997
0.999911    0.49998
0.999911   0.499969
0.999911   0.499978
0.999912   0.499974
0.999913   0.499973
0.999914   0.499976
0.999917   0.499976
0.999919   0.499973
0.999924   0.499971
 0.99993   0.499973
0.999938   0.499979
0.999947   0.499986
0.999959   0.499992
0.999971   0.499996
0.999986   0.833332
```

n=50:

```
x_j =

0.999986
0.999971
0.999959
0.999947
0.999938
 0.99993
0.999924
```

```
0.999919
0.999917
0.999914
0.999913
0.999912
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999912
0.999913
0.999914
0.999917
0.999919
0.999924
 0.99993
0.999938
0.999947
0.999959
0.999971
0.999986


x_g =

0.666653
0.555536
0.518496
0.506149
0.502033
0.500661
0.500203
0.500051
     0.5
0.499983
0.499977
0.499976
0.499975
0.499975
```

```
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499974
0.499975
0.499974
0.499976
0.499972
0.499978
 0.49997
 0.49998
0.499969
0.499978
0.499974
0.499973
0.499976
0.499976
0.499973
0.499971
0.499973
0.499979
0.499986
0.499992
0.499996
0.833332
```

n=100：

```
x_j =

0.999986
0.999971
0.999959
0.999947
0.999938
 0.99993
0.999924
0.999919
0.999917
0.999914
0.999913
0.999912
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
```

```
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
```

```
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999911
0.999912
0.999913
0.999914
0.999917
0.999919
0.999924
 0.99993
0.999938
0.999947
0.999959
0.999971
0.999986


x_g =

0.666653
0.555536
0.518496
0.506149
0.502033
0.500661
0.500203
0.500051
     0.5
0.499983
0.499977
0.499976
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
```

```
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499975
0.499974
0.499975
0.499974
0.499976
0.499972
0.499978
 0.49997
 0.49998
0.499969
0.499978
0.499974
0.499973
0.499976
0.499976
0.499973
0.499971
0.499973
0.499979
0.499986
0.499992
```

```
0.499996
0.833332
```

（5）误差分析：误差来自计算过程中不能整除产生。

8.用共轭梯度法求解上题中的方程组.

（1）算法公式

（2）Matlab代码

```
%共轭梯度法求解方程组
function x=CG(A,b)
%  inputs:
%          A:系数矩阵，为n*n维方阵
%          b:载荷矩阵，为n*1维矩阵
%  outputs:
%          x:计算结果向量,为n*1为矩阵

% 判断输入矩阵维度是否满足要求
[row_A,col_A] = size(A);
[row_b,~] = size(b);
format long %为了输出6位小数
% 初始化r_matrix矩阵
x = zeros(row_b,1);
% 判断输入的维度是否满足要求
if (row_A ~= col_A) || (row_A ~= row_b)
    % 不满足则输出错误提示
    print('输入错误! ');
else
    % 满足则进行下一步运算
    n=row_A;
    r=x;
    r1=x;
    p=x;
    c=0;
    while 1
        r=b-A*x;
        p=r;
        t=A*p;
        if dot(p,t)==0
            break
        end
        c=dot(r,r)/dot(p,t);
        x=x+c*p;
        r=r-c*t;
        e=dot(r,r)/dot(p,p);
        p=r+e*p;
        if r==zeros(n,1)
            break
        end
    end
    x=vpa(x,6) ; %结果保留6位小数
end
```

> 函数调用部分和上题中类似

（3）**运行结果截图**

n=10

```
x_cg =

     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
```

n=20

```
x_cg =

     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
```

```
x_cg =

     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
```

n=30

n=50和n=100均为全1列向量

（4）误差分析：误差来自计算过程中不能整除产生。

三.非线性方程组求解

4.用Newton迭代法求解方程$x^3+x^2+x-3=0$的根，初值选择$x_0=-0.7$，迭代7步并与真值$x^*=1$相比较，并列出数据表（表1）.

表 1　　数据表

i	x_i	$e_i=\|x_i-x^*\|$	$\frac{e_i}{e_{i-1}^2}$
0			
1			
⋮			

从最后一列能得到什么结论（从收敛阶的角度回答）？把最后一列的值与$\frac{f''(x^*)}{2f'(x^*)}$比较能得到什么结论?

（1）算法公式：$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)},k=0,1,2,\ldots$

（2）Matlab代码：

```
%Newton迭代法,x0=-0.7,迭代7步
x=-0.7
for i=1:7
    e=abs(x-1)
    k=e/(e^2-1)
    f=x^3+x^2+x-3; %f(x)
    f_=3*x^2+2*x+1;%f(x)的导数
    x=x-f/f_ %x取得新值
end
e=abs(x-1)
k=e/(e^2-1)
```

（3）填表

i	x_i	$e_i = \|x_i - x^*\|$	$\frac{e_i}{e_i^2-1}$
0	-0.7	1.7	0.89947
1	2.62056	1.62056	0.99652
2	1.70844	0.70844	-1.42225
3	1.20637	0.20638	-0.21556
4	1.02416	0.02416	-0.02418
5	1.00038	3.8149e-04	-3.8149e-04
6	1.0000	9.6993e-08	-9.6993e-08
7	1.0000	6.2172e-15	-6.2172e-15

$\frac{f''(x^*)}{2f'(x^*)} = 0.667$

（4）误差分析：误差来自计算过程中不能整除产生。

5.取不同的初值用弦截法求求方程$x^3 + 2x^2 + 10x - 100 = 0$的实根，列表或者画图说明收敛速度.

（1）算法公式：

弦截法$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k)-f(x_{k-1})}(x_k - x_{k-1}), k = 0, 1, 2, \ldots$

（2）Matlab代码

```
%弦截法求分线性方程的根
x=0
x1=0.1
for i=1:7
    fx=x^3+2*x^2+10*x-100; %f(x)
    fx1=x1^3+2*x1^2+10*x1-100; %f(x1)
    t=x1-fx1/(fx1-fx)*(x1-x);
    x=x1
    x1=t
end
```

弦截法循环迭代10次

x0	x1	i	0	1	2	3	4	5	6	7	8	9	10
0	0.1	x	0.1000	9.7943	0.8812	1.5319	5.4902	2.7082	3.1957	3.5078	3.4579	3.4606	3.4606
1	2		2	4.7826	3.0666	3.3652	3.4689	3.4604	3.4606	3.4606	3.4606	3.4606	3.4606
4	5		5	3.5955	3.4957	3.4615	3.4606	3.4606	3.4606	3.4606	3.4606	3.4606	3.4606
10	16		16	7.9239	6.7274	4.9220	4.0617	3.6037	3.4768	3.4611	3.4606	3.4606	3.4606

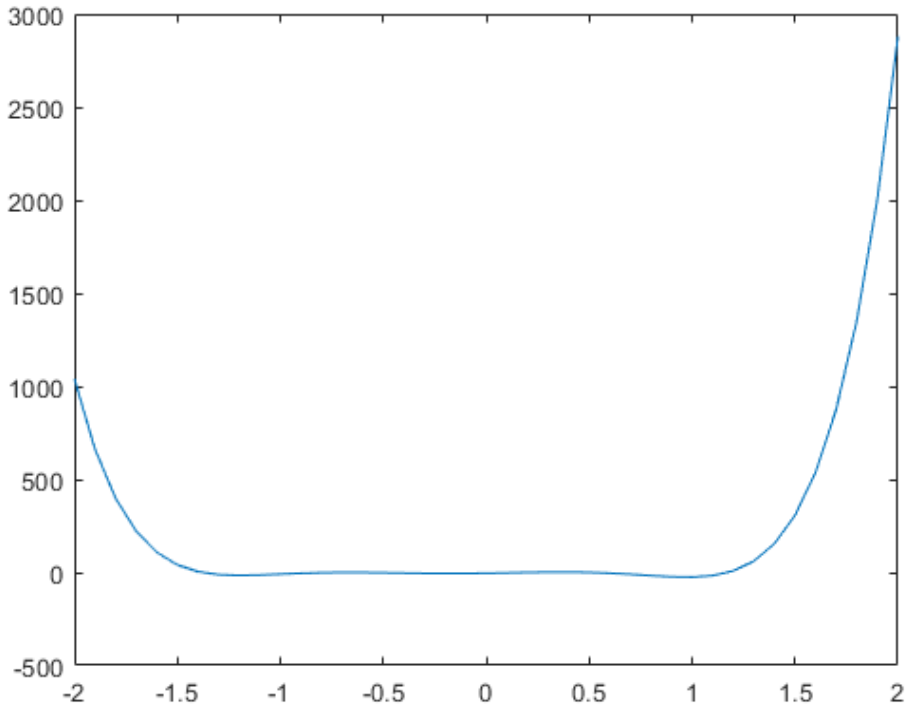
由表格可知，离根越近收敛越快，根为3.4606。

（3）误差分析：误差来自计算过程中不能整除产生。

6.设$f(x)=54x^6+45x^5-102x^4-69x^3+35x^2+16x-4$.在区间[-2,2]上画出函数，（1）使用Newton迭代法找出该区间上的5个根，并计算e_{i+1}/e_i^2和e_{i+1}/e_i，由此判断哪个根是1阶收敛，哪个根是2阶收敛?
（2）使用割线法计算这5个根，并判断哪个根是线性收敛，哪个是超线性收敛?

（1）函数图像代码

```
x=-2:0.1:2;
y=54*x.^6+45*x.^5-102*x.^4-69*x.^3+35*x.^2+16*x-4;
plot(x,y);
```



（2）New迭代法计算该区间的5个根

Matlab代码

```
%Newton迭代法,x0不同,迭代8步
x=-1.5 %设置初始值
f=54*x^6+45*x^5-102*x^4-69*x^3+35*x^2+16*x-4;%f(x)
f_=324*x^5+225*x^4-408*x^3-207*x^2+70*x+16;%f(x)的导数
t=x;
x=x-f/f_ %x取得新值
e=abs(x-t)
for i=1:7
    f=54*x^6+45*x^5-102*x^4-69*x^3+35*x^2+16*x-4;%f(x)
    f_=324*x^5+225*x^4-408*x^3-207*x^2+70*x+16;%f(x)的导数
    t=x;
    x=x-f/f_ %x取得新值
```

```
    e_=e;
    e=abs(x-t)
    k1=e/(e_^2)
    k2=e/e_
end
```

x1=-1.3813，观察$k1=e_{i+1}/e_i^2$和$k2=e_{i+1}/e_i$可知，x1是2阶收敛；

x2=-0.6667，观察$k1=e_{i+1}/e_i^2$和$k2=e_{i+1}/e_i$可知，x2是1阶收敛；

x3=0.2052，观察$k1=e_{i+1}/e_i^2$和$k2=e_{i+1}/e_i$可知，x3是2阶收敛；

x4=0.5000，观察$k1=e_{i+1}/e_i^2$和$k2=e_{i+1}/e_i$可知，x4是2阶收敛；

x5=1.1761，观察$k1=e_{i+1}/e_i^2$和$k2=e_{i+1}/e_i$可知，x5是2阶收敛；

（3）使用割线法计算该区间的5个根

Matlab代码

```
%割线法迭代法,x0不同,迭代8步
x=-2;
x1=-1.7 %设置初始值
e=abs(x1-x)
for i=1:12
    fx=54*x^6+45*x^5-102*x^4-69*x^3+35*x^2+16*x-4;%f(x)
    fx1=54*x1^6+45*x1^5-102*x1^4-69*x1^3+35*x1^2+16*x1-4;%f(x)
    t=x1-fx1/(fx1-fx)*(x1-x);
    x=x1;
    x1=t
    e_=e; %旧的
    e=abs(x1-x);%新的
    k1=e/(e_^2)
    k2=e/e_
end
```

x1=-1.3813，超线性收敛；

x2=-0.6667，线性收敛；

x3=0.2052，超线性收敛；

x4=0.5000，超线性收敛；

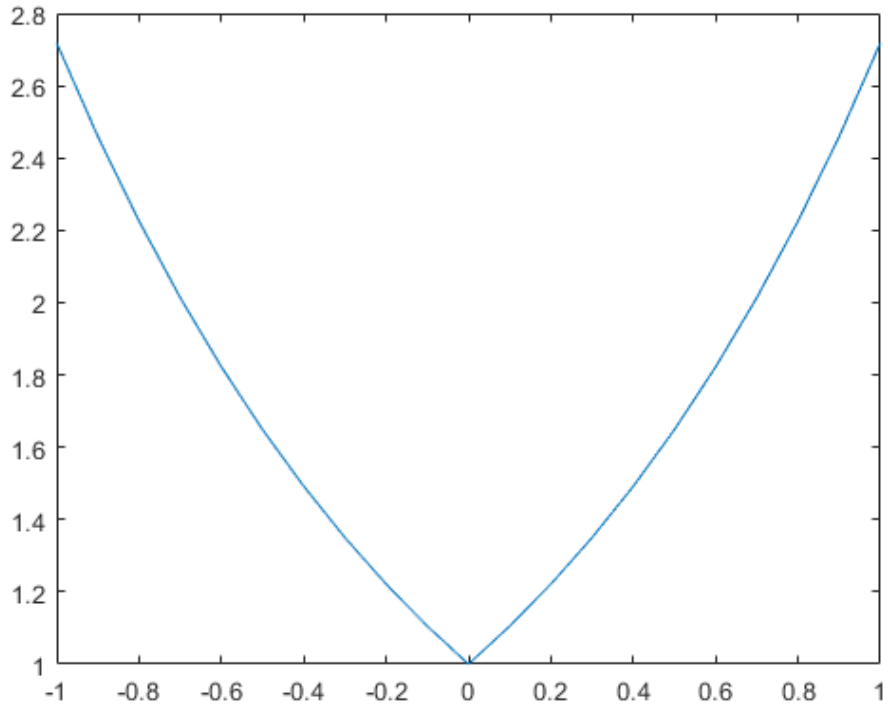
x5=1.1761，超线性收敛；

（4）误差分析：误差来自计算过程中不能整除产生。

四.插值与逼近

3.令$f(x)=e^{|x|}$，x∈[-1，1]，分别用等距节点和Chebyshev的零点去插值f(x)，等距节点包括左右两个端点，分别取n=5，10，15，20，画出插值函数以及原函数的图并比较，观察有没有Runge现象发生.

（1）原函数图像



Matlab代码

```
x=-1:0.1:1;
y=exp(abs(x));
plot(x,y)%原函数的图像
```

（2）用等距节点和Chebyshev的零点使用拉格朗日插值f(x)

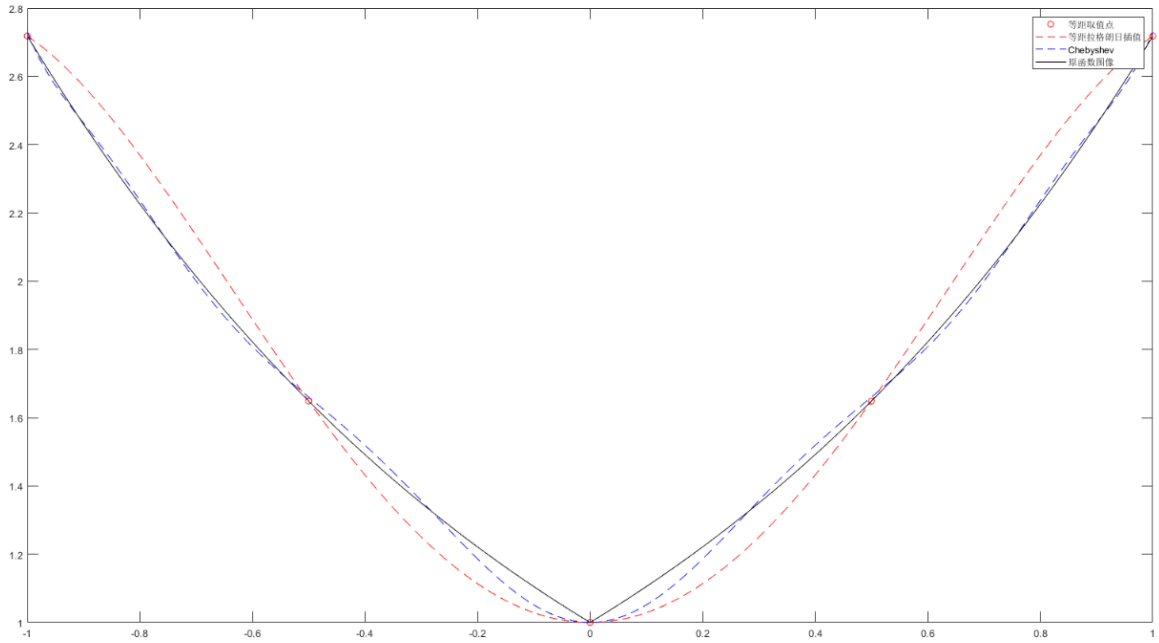
Matlab代码

```
%拉格朗日插值
function f=Lagrange(x,y,x0)
syms t;
if (length(x)==length(y))
    n=length(y);
else
    disp('x和y的维数不相等! ');
end %检错
f=0.0
for i=1:n
    l=y(i);
    for j=1:i-1
        l=l*(t-x(j))/(x(i)-x(j));
    end
    for j=i+1:n
        l=l*(t-x(j))/(x(i)-x(j)); %计算拉格朗日基函数
    end
    f=f+l; %计算拉格朗日插值函数
    simplify(f); %化简
    if i==n
        if nargin==3
            f=subs(f,'t',x0); %计算插值点函数
        else
            f=collect(f); %将插值多项式展开
            f=vpa(f,6); %将插值多项式的系数化成6位精度的小数
        end
    end
end
```

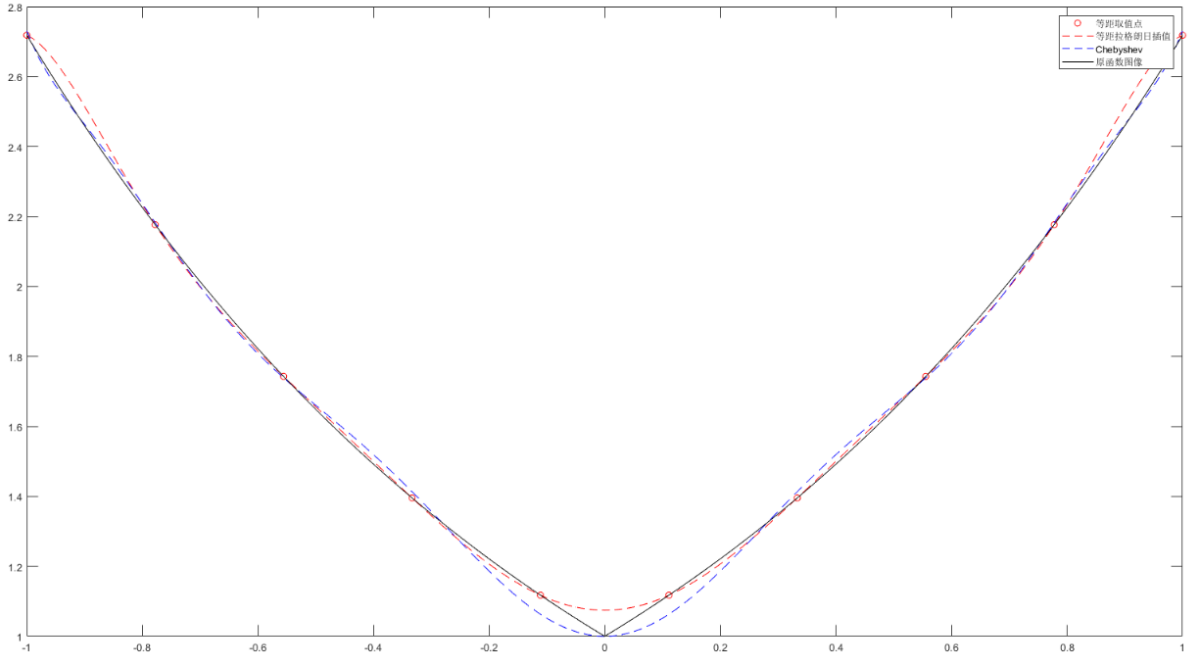
```
%画图
f=@(x) exp(abs(x)); %函数表达式

u= -1:0.01:1;
v=f(u); %原函数图像

n=20;
x =linspace(-1,1,n); %等距取值
y = f(x);    %插值节点
v1 = Lagrange(x,y,u); %拉格朗日插值多项式

%Chebyshev
x_0 = 0:10;
x_0 = cos((21-2*x_0)*pi/22);
y_0 = f(x_0);
v2 = Lagrange(x_0,y_0,u);

%set(gcf,'outerposition',get(0,'screensize'));%图像最大化
plot(x,y,'ro',u,v1,'r--',u,v2,'b--',u,v,'k-');
legend('等距取值点','等距拉格朗日插值','Chebyshev','原函数图像')
```

图像：

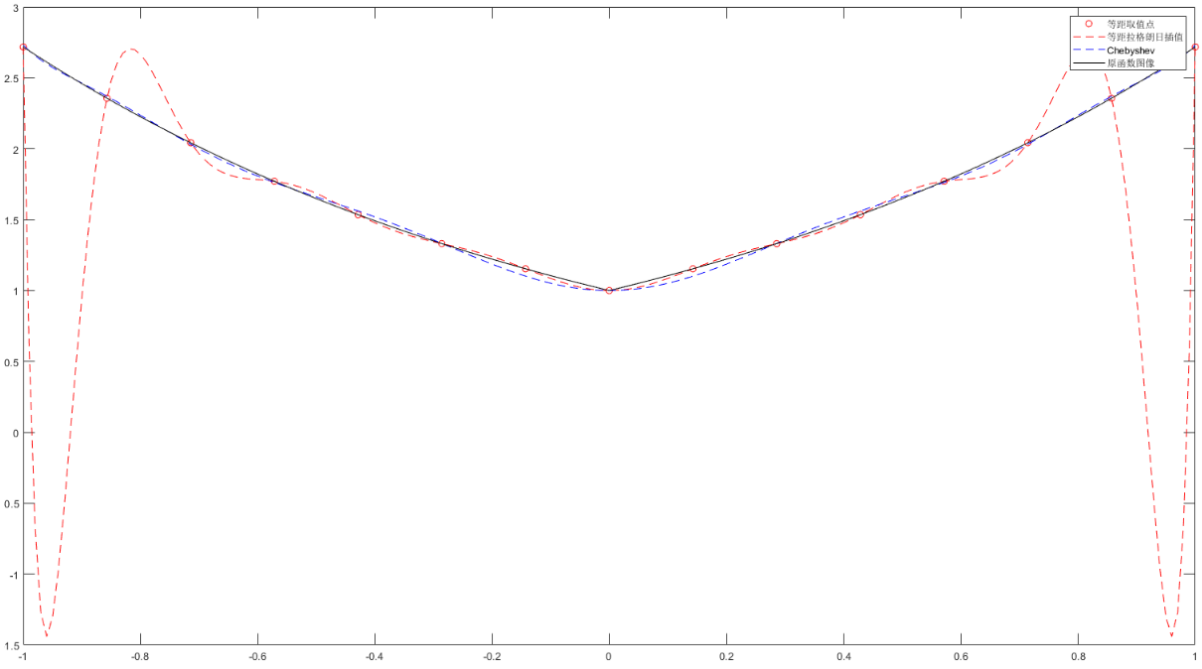
当n=5时



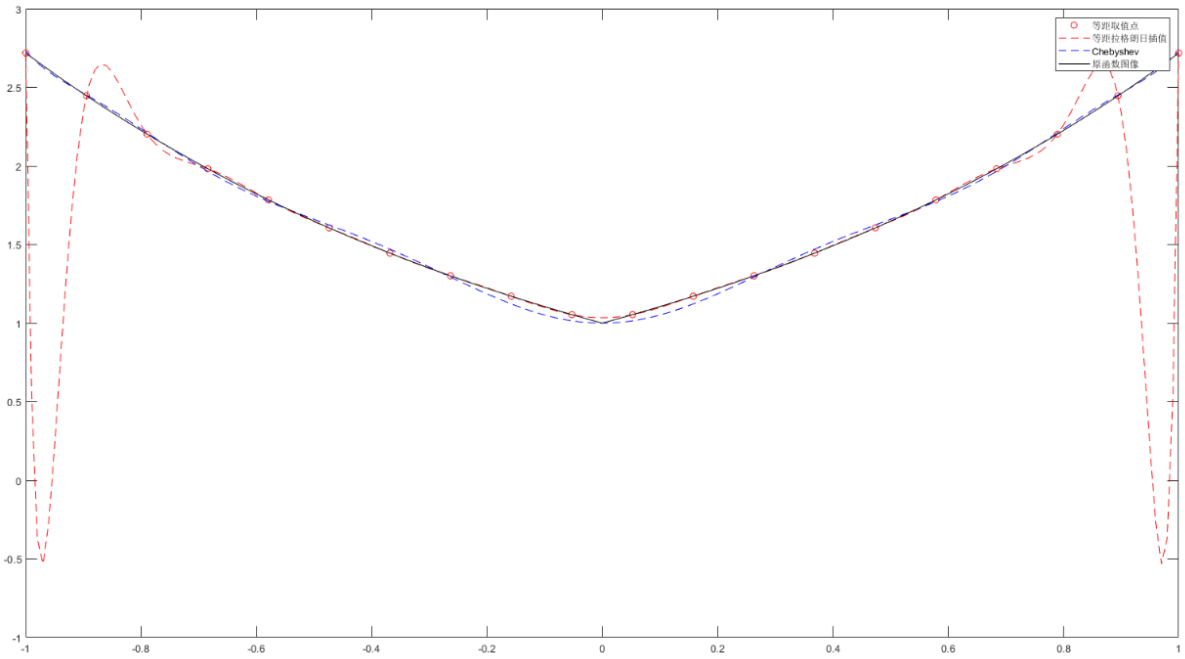
当n=10时


等距取值点
等距拉格朗日插值
Chebyshev
原函数图像

当n=15时


等距取值点
等距拉格朗日插值
Chebyshev
原函数图像

当n=20时



观察图像可知，当等距节点n增大时，拉格朗日插值出现了Runge现象；Chebyshev零点插值可以避免Runge现象。

4.（1）给定数据点（x_i，x_i^2），$x_i = 0, 1/n, 2/n, \ldots, 1$，当n=5，10，15，20，25，30时分别用直线拟合这组数据点并注意观察当点数逐渐增加时直线的表达式的变化.（2）计算函数$f(c_1, c_2) = \int_0^1 (x^2 - c_1 - c_2 x)^2 dx$的最小值，并解释与（1）的关系.

（1）Matlab代码

```
n=5
format short
x=zeros(1,n);
y=zeros(1,n);
for i=1:6
    x(i)=(i-1)/n;
    y(i)=x(i)*x(i);
end
a=polyfit(x,y,1) %自变量，因变量，阶数
```

（2）直线表达式

- 当n=5时，直线表达式：$y = 1.0000x - 0.1333$
- 当n=10时，直线表达式：$y = 0.4385x - 0.0108$
- 当n=15时，直线表达式：$y = 0.2833x - 0.0026$
- 当n=20时，直线表达式：$y = 0.2100x - 0.0010$
- 当n=25时，直线表达式：$y = 0.1670x - 0.0005$
- 当n=30时，直线表达式为：$y = 0.1386x - 0.0003$

（3）$f(c_1, c_2) = \int_0^1 (x^2 - c_1 - c_2 x)^2 dx = c_1^2 + c_1 * c_2 - (2 * c_1)/3 + c_2^2/3 - c_2/2 + 1/5$

6.全世界石油产量（单位：百万桶/日）见表2所示，确定并画出经过这些点的9阶多项式，并使用该多项式估计2010年的石油产量.Runge现象在这个例子中出现了吗？以你的观点，插值多项式是描述这些数据好的模型吗？请解释。

表 2　　全世界石油产量

年	产量/(百万桶/日)	年	产量/(百万桶/日)
1994	67.052	1999	72.063
1995	68.008	2000	74.669
1996	69.803	2001	74.487
1997	72.024	2002	74.065
1998	73.400	2003	76.777

用最小二乘法拟合题目中的10个数据点，拟合曲线为（1）直线；（2）抛物线，以及（3）三次曲线，并计算它们的均方误差。使用所得到的拟合曲线估计2010年的产量，在均方误差意义下，哪个拟合最好?

（1）9阶多项式

$$P_9(x) = -(1695361414010883 * x^9)/2305843009213693952 + (4227396721652201 * x^8)/144115188075855872 - (8819278816003891 * x^7)/18014398509481984 + (4994493159566287 * x^6)/1125899906842624 - (3333749145844049 * x^5)/140737488355328 + (5324582419842471 * x^4)/70368744177664 - (2460605999175011 * x^3)/17592186044416 + (4766208684428403 * x^2)/35184372088832 - (7120293268944661 * x)/140737488355328 + 16763/250$$

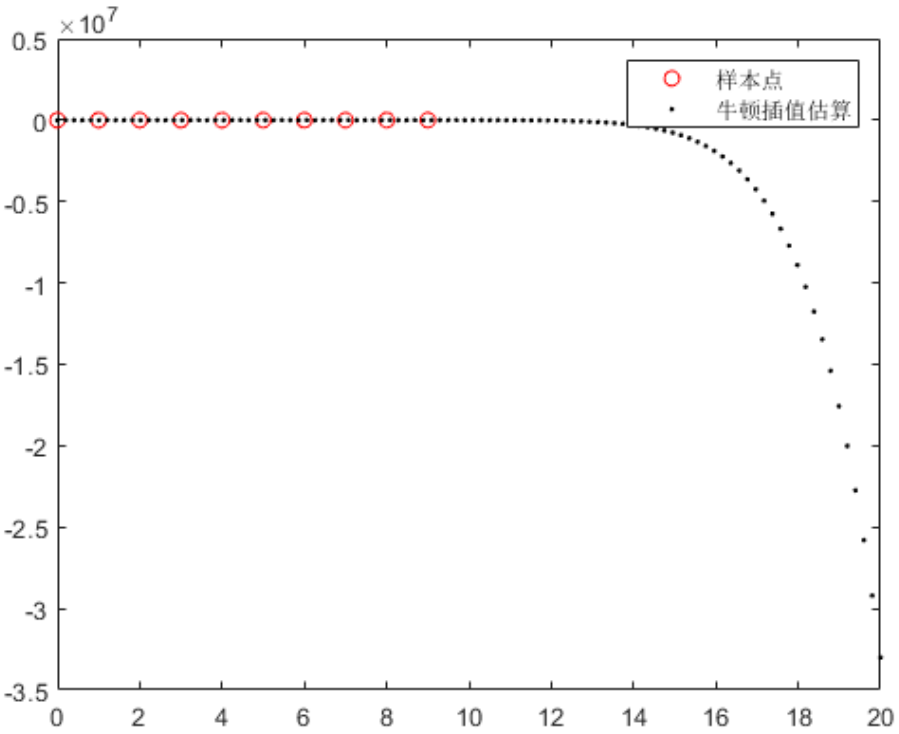
Matlab代码

```
%newton.m
%求牛顿插值多项式、差商、插值及其误差估计的MATLAB主程序
%输入的量:X是n+1个节点(x_i,y_i)(i = 1,2, ... , n+1)横坐标向量，Y是纵坐标向量，
%x是以向量形式输入的m个插值点，M在[a,b]上满足|f~(n+1)(x)|≤M
%注：f~(n+1)(x)表示f(x)的n+1阶导数
%输出的量：向量y是向量x处的插值，误差限R，n次牛顿插值多项式L及其系数向量C，
%差商的矩阵A
function[y,R,A,C,L] = newton(X,Y,x,M)
n = length(X);
m = length(x);
for t = 1 : m
    z = x(t);
    A = zeros(n,n);
    A(:,1) = Y';
    s = 0.0; p = 1.0; q1 = 1.0; c1 = 1.0;
        for j = 2 : n
            for i = j : n
                A(i,j) = (A(i,j-1) - A(i-1,j-1))/(X(i)-X(i-j+1));
            end
            q1 = abs(q1*(z-X(j-1)));
            c1 = c1 * j;
        end
        C = A(n, n); q1 = abs(q1*(z-X(n)));
        for k = (n-1):-1:1
            C = conv(C, poly(X(k)));
            d = length(C);
            C(d) = C(d) + A(k,k);%在最后一维，也就是常数项加上新的差商
        end
        y(t) = polyval(C,z);
        R(t) = M * q1 / c1;
end
L = poly2sym(C);

%调用newton插值法确定经过这些点的9阶多项式
X=(0:1:9);%0表示1994，9表示2003
Y=[67.052,68.008,69.803,72.024,73.400,72.063,74.669,74.487,74.065,76.777];
```

```
x=linspace(0,20);
M=1;
[y,R,A,C,L] = newton(X, Y, x, M)
%y是在向量x处的插值，误差限R，n次newton多项式L以及系数常量C
plot(X, Y, 'or', x, y, '.k');
legend('样本点','牛顿插值估算');
```

根据推测的多项式代入求值得2010年全世界石油产量 -9.0236e+06百万桶/日， 明显不符合实际。

函数图像：



由图像可知，出现了Runge现象。

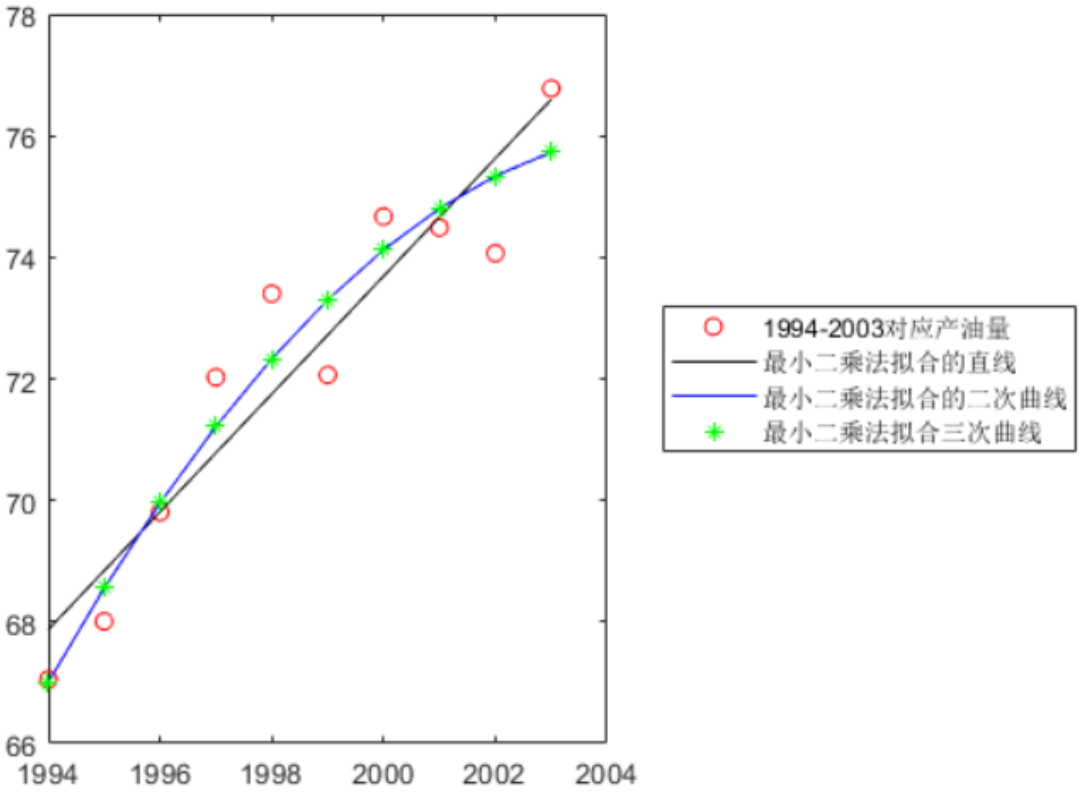
（2）最小二乘法拟合

- Matlab代码

```
%最小二乘法拟合为直线、抛物线、三次曲线
%直线y=a1+a2*x
X=[1994:1:2003];%0表示1994，9表示2003
Y=[67.052,68.008,69.803,72.024,73.400,72.063,74.669,74.487,74.065,76.777];
plot(X,Y,'ro');
hold on
sx=sum(X);
sx2=sum(X.*X);
sy=sum(Y);
sxy=sum(X.*Y);
A=[10,sx;sx,sx2];
b=[sy;sxy];
a1=GaussElimination(A,b)%Gauss消去法求解
f1=a1(1)+a1(2).*X;
plot(X,f1,'k-')
%二次曲线y=a0+a1*x+a2*x^2
sx3=sum(X.^3);
sx4=sum(X.^4);
sx2y=sum(X.^2.*Y);
A=[10,sx,sx2;sx,sx2,sx3;sx2,sx3,sx4];
b=[sy;sxy;sx2y];
a2=GaussElimination(A,b)%Gauss消去法求解
f2=a2(1)+a2(2).*X+a2(3).*X.^2;
```

```
plot(X,f2,'b-')
%三次曲线拟合y=a0+a1*x+a2*x^2+a3*x^3
sx5=sum(X.^5);
sx6=sum(X.^6);
sx3y=sum(X.^3.*Y);
A=[10,sx,sx2,sx3;sx,sx2,sx3,sx4;sx2,sx3,sx4,sx5;sx3,sx4,sx5,sx6];%系数矩阵
b=[sy;sxy;sx2y;sx3y];
a3=GaussElimination(A,b)%Gauss消去法求解
f3=a3(1)+a3(2).*X+a3(3).*X.^2+a3(4).*X.^3;
plot(X,f3,'g*')
l=legend('1994-2003对应产油量','最小二乘法拟合的直线','最小二乘法拟合的二次曲线','最小二乘法拟合三次曲线','orientation','vertical','location','eastoutside');%图例放在右侧
```

- 图像



- 均方误差
  - 直线：3.0874
  - 抛物线：2.6015
  - 三次曲线：2.5950
- 估计2010年的产量
  - 直线：83.3823百万桶/日
  - 抛物线：74.4106百万桶/日
  - 三次曲线：74.8245百万桶/日

在均方误差的意义下，三次曲线和二次曲线拟合较好。

7.编程计算三次样条S，满足S（0）=1，S（1）=3，S（2）=3，S（3）=4，S（4）=2，其中边界条件S"（0）=S"（4）=0.

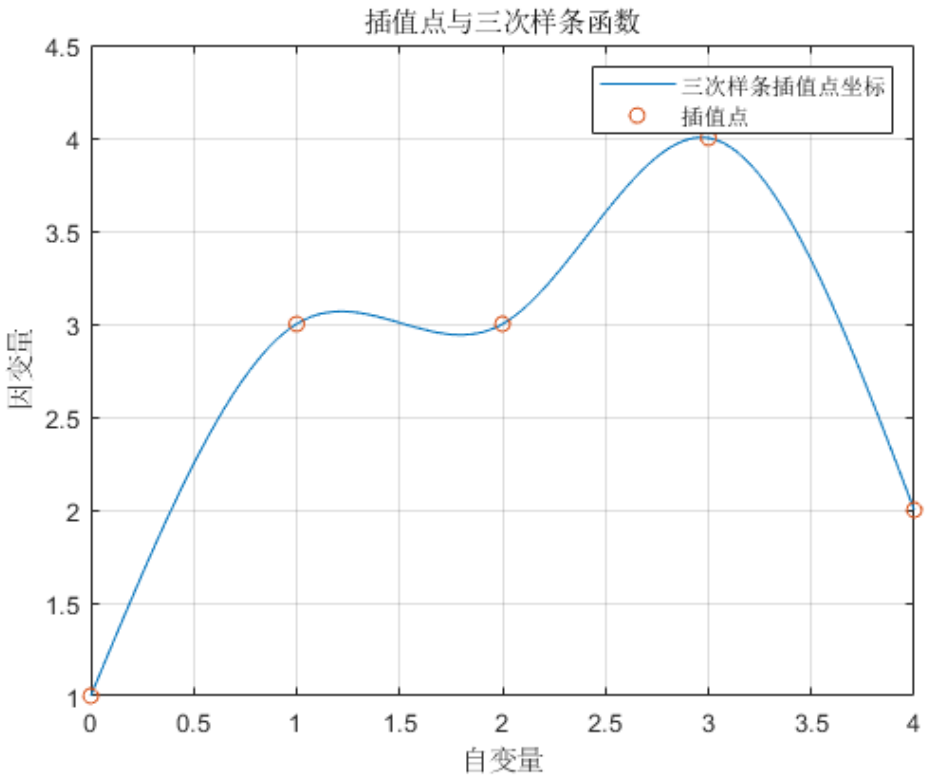
（1）Matlab代码

```
% 第二边界条件的三次样条函数(包含自然边界条件)
% y0,yn表示的是S''(x0)=f''(x0)=y0，S''(xn)=f''(xn)=yn,自然边界即条件值为0
% 此函数为M值求值函数
```

```
% D,h,A,g,M输出量分别为系数矩阵D，插值宽度h，差商表A，g值,M值
function [D,h,A,g,M]=th_sec(X,Y,y0,yn)
         n=length(X);
         A=zeros(n,n);A(:,1)=Y';D=zeros(n-2,n-2);g=zeros(n-2,1);
         for  j=2:n
            for i=j:n
                A(i,j)=(A(i,j-1)- A(i-1,j-1))/(X(i)-X(i-j+1));
            end
         end

         for i=1:n-1
             h(i)=X(i+1)-X(i);
         end
         for i=1:n-2
             D(i,i)=2;
             if (i==1)
                 g(i,1)=(6/(h(i+1)+h(i)))*(A(i+2,2)-A(i+1,2))-h(i)/(h(i)+h(i+1))*y0;
             elseif (i==(n-2))
                 g(i,1)=(6/(h(i+1)+h(i)))*(A(i+2,2)-A(i+1,2))-(1-
h(i)/(h(i)+h(i+1)))*yn;
             else
                 g(i,1)=(6/(h(i+1)+h(i)))*(A(i+2,2)-A(i+1,2));
             end
         end
         for i=2:n-2
             u(i)=h(i)/(h(i)+h(i+1));
             n(i-1)=h(i)/(h(i-1)+h(i));
             D(i-1,i)=n(i-1);
             D(i,i-1)=u(i);
         end
         M=D\g;
         M=[y0;M;yn];
end

function s=t_simple(X,Y,x,y0,yn)
% 第二边界条件函数
% s函数表示三次样条插值函数插值点对应的函数值
% 根据三次样条参数函数求出的D,h,A,g,M
% x表示求解插值点函数点，X为已知插值点
         [D,h,A,g,M]=th_sec(X,Y,y0,yn)
         n=length(X);
         m=length(x);
         for t=1:m
            for i=1:n-1
               if (x(t)<=X(i+1))&&(x(t)>=X(i))
                  p1=M(i,1)*(X(i+1)-x(t))^3/(6*h(i));
                  p2=M(i+1,1)*(x(t)-X(i))^3/(6*h(i));
                  p3=(A(i,1)-M(i,1)/6*(h(i))^2)*(X(i+1)-x(t))/h(i);
                  p4=(A(i+1,1)-M(i+1,1)/6*(h(i))^2)*(x(t)-X(i))/h(i);
                  s(t)=p1+p2+p3+p4;
                  break;
               else
                   s(t)=0;
               end
            end
         end
end

%usefunction
%调用三次样条函数中满足第二类边界条件
x=[0 1 2 3 4];
```

```
y=[1 3 3 4 2];
y0=0;           %  S''(x0)=f''(x0)=y0
yn=0;           %  S''(xn)=f''(xn)=yn
x0=0:0.01:4;
s=t_simple(x,y,x0,y0,yn)
plot(x0,s)          %绘制第二边界条件插值函数图像
hold on
grid on
plot(x,y,'o')
xlabel('自变量'), ylabel('因变量')
title('插值点与三次样条函数')
```

（2）运行结果



五.数值积分

1.已知$f(x)=x^2sinx$，分别用复化梯形公式和复化Simpson公式计算积分$\int_{-2}^{2}f(x)dx$，区间分为20，40，80，200个小区间，并计算其精确值，比较计算精度情况。

（1）Matlab代码

```
%f(x)=x^2*sinx
function num=f(x)
    num=x^2*sin(x);
end
%复化的梯形求积公式
n=20;%分成多少个小区间
a=0;%积分下界
b=pi;%积分上界
h=(b-a)/n;%步长
x=zeros(1,n+1);
res=0;
for i=1:n+1
    x(i)=a+h*(i-1);
end
for i=2:n
    res=res+f(x(i));
```

```
end
Tn=h/2*(f(x(1))+2*res+f(x(n+1)))

%复化的Simpson求积公式
xx=zeros(1,2*n+1);%有2n+1个数
for i=1:2*n+1
    xx(i)=a+h/2*(i-1);
end
res1=0;
for i=2:2:2*n
    res1=res1+f(xx(i));
end
res1=res1*4;%xk+1/2部分
res2=0;
for i=3:2:2*n-1
    res2=res2+f(xx(i));
end
res2=2*res2;%除了首尾的整数部分
Sn=h/6*(f(xx(1))+res1+res2+f(xx(2*n+1)))
```

（2）运行结果（Tn表示复化梯形求积公式，Sn表示复化Simpson求积公式）

- n=20，Tn与Sn相差e = 0.0203

```
n =

    20


Tn =

    5.8493


Sn =

    5.8696
```

- n=40，Tn与Sn相差e =0.0051

```
n =

    40


Tn =

    5.8645


Sn =

    5.8696
```

- n=80，Tn与Sn相差e =0.0013

```
n =

    80

Tn =

    5.8683

Sn =

    5.8696
```

- n=200，Tn与Sn相差e =2.0294e-04

```
n =

   200

Tn =

    5.8694

Sn =

    5.8696
```

根据运行结果可知，随着n的增大，复化的梯形求积公式和复化的Simpson求积公式求得的结果越接近。

六.微分方程数值解法

1.已知常微分方程

$$\begin{cases} \dfrac{du}{dx} = \dfrac{2}{x}u + x^2 e^x, \\ x \in [1,2], u(1) = 0 \end{cases}$$

分别用Euler法，改进的Euler法，Runge-Kutta法去求解该方程，步长选为0.1，0.05，0.01.画图观察求解效果。

（1）Euler法

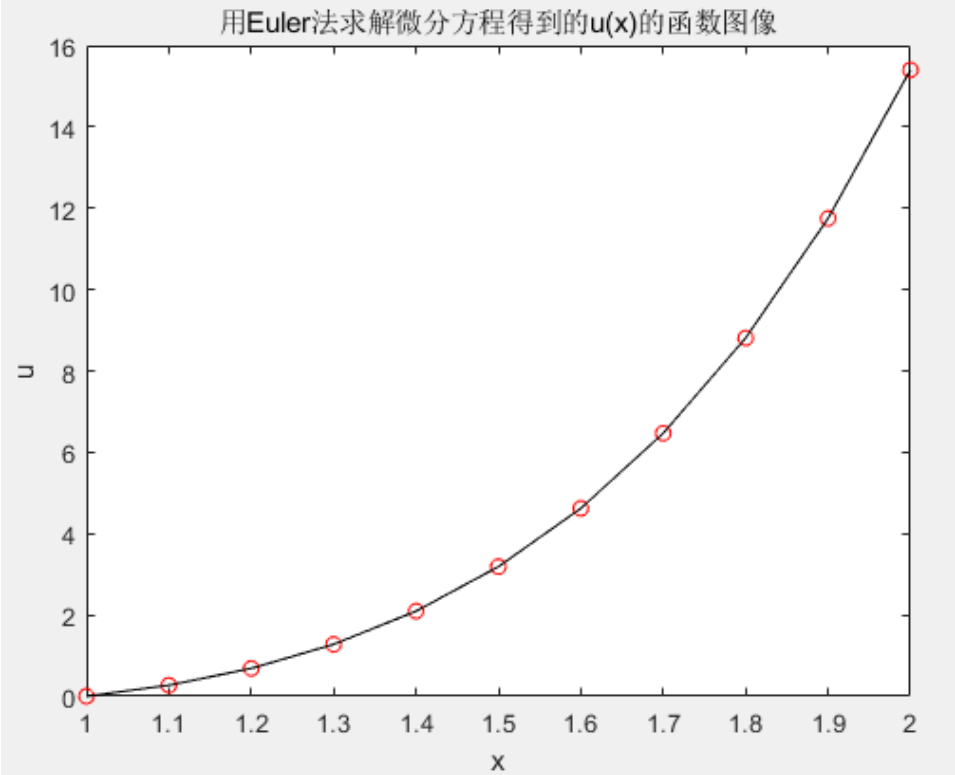
Matlab代码

```
%定义u(t)的导数
function num=ut_(t,u)
    num=2*u/t+t^2*exp(t);
end

%Euler法求微分方程的解
i=2;
h=0.1;%步长
u=zeros(1,1/h+1);
u(1,1)=0;
for t=1:h:1.9
```

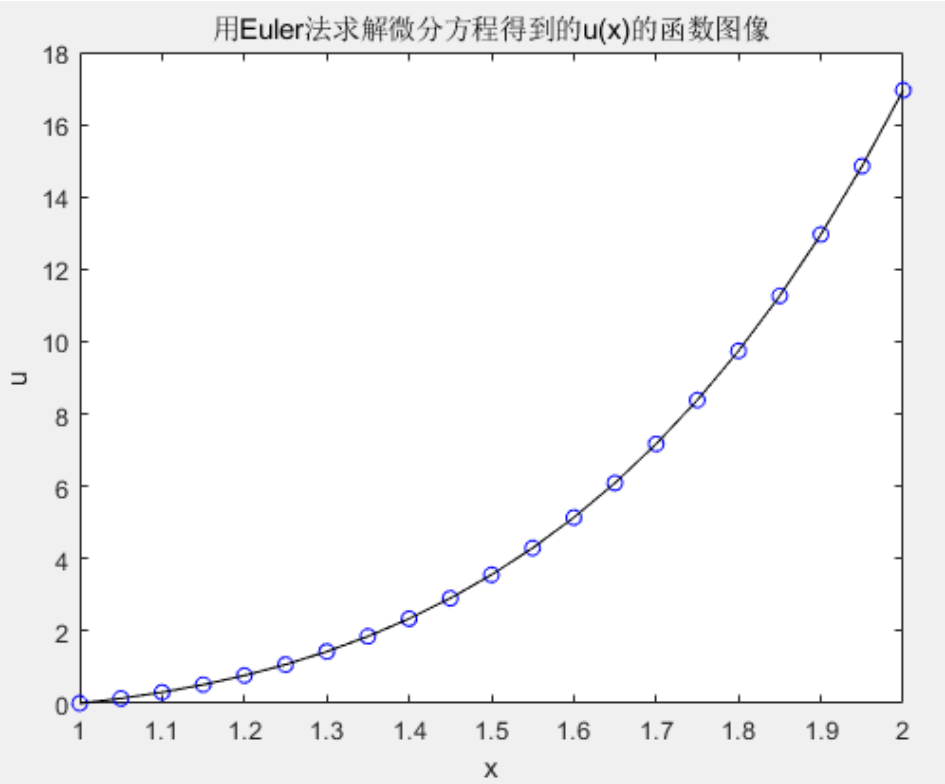
```
    u(1,i)=u(1,i-1)+ut_(t,u(1,i-1))*h;
    i=i+1;
end
t=(1:h:2);
plot(t,u,'ro')
hold on
plot(t,u,'k-')
title("用Euler法求解微分方程得到的u(x)的函数图像");
xlabel('x');
ylabel('u');
```

运行结果截图：

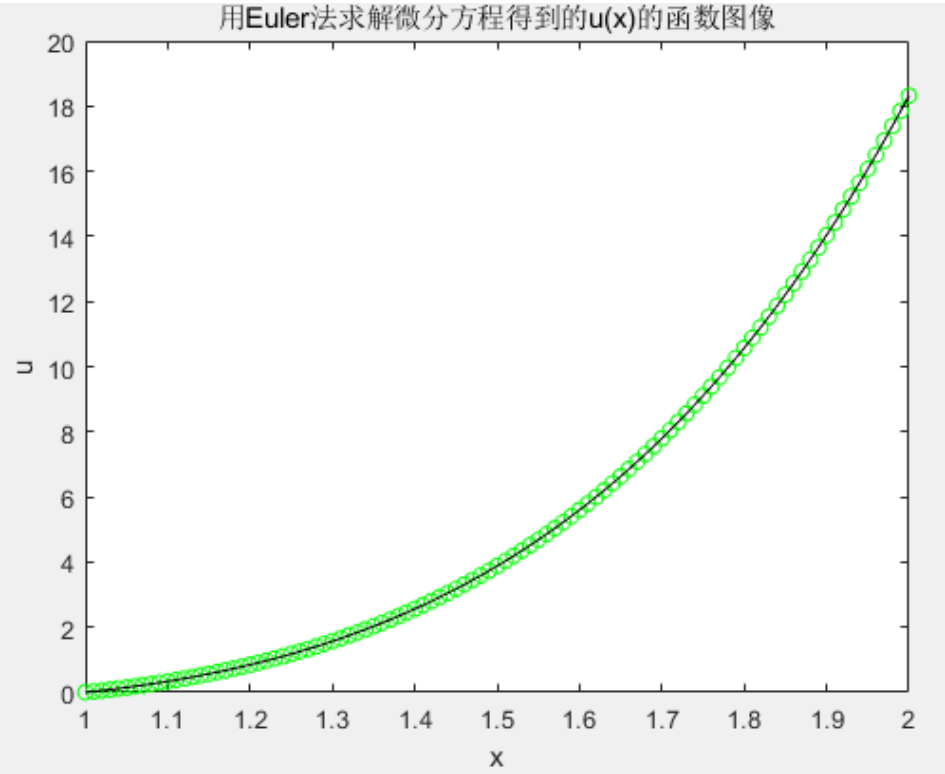
- 步长h=0.1



- 步长h=0.05



- 步长h=0.01



（2）改进的Euler法

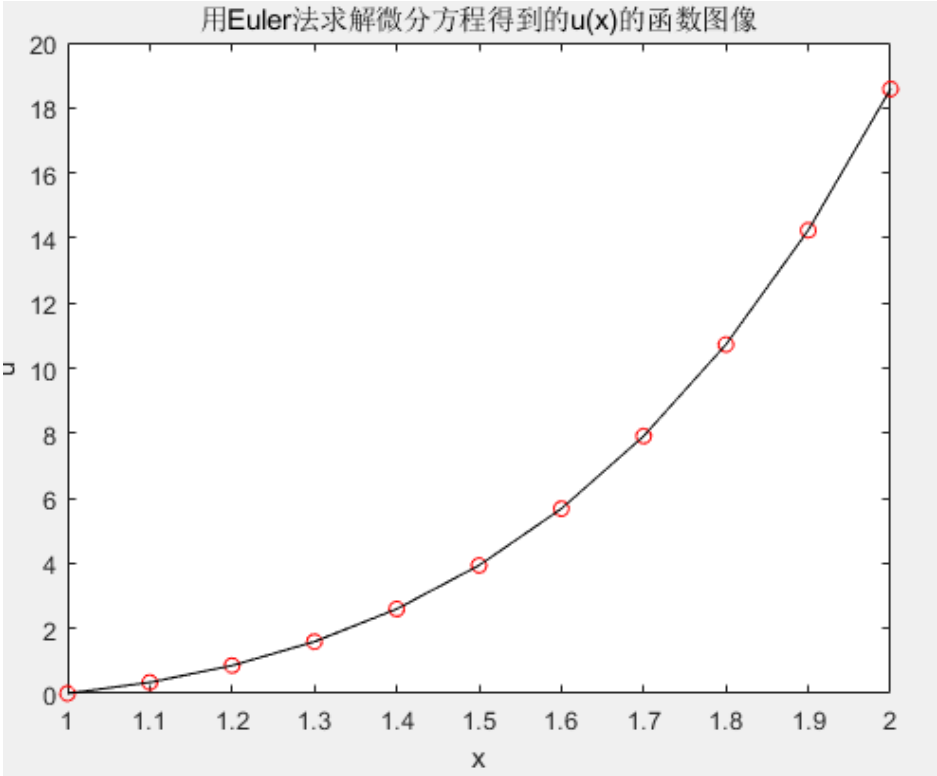
Matlab代码：

```
%定义u(t)的导数
function num=ut_(t,u)
    num=2*u/t+t^2*exp(t);
end

%改进的Euler法求微分方程的解
h=0.01;%步长
```

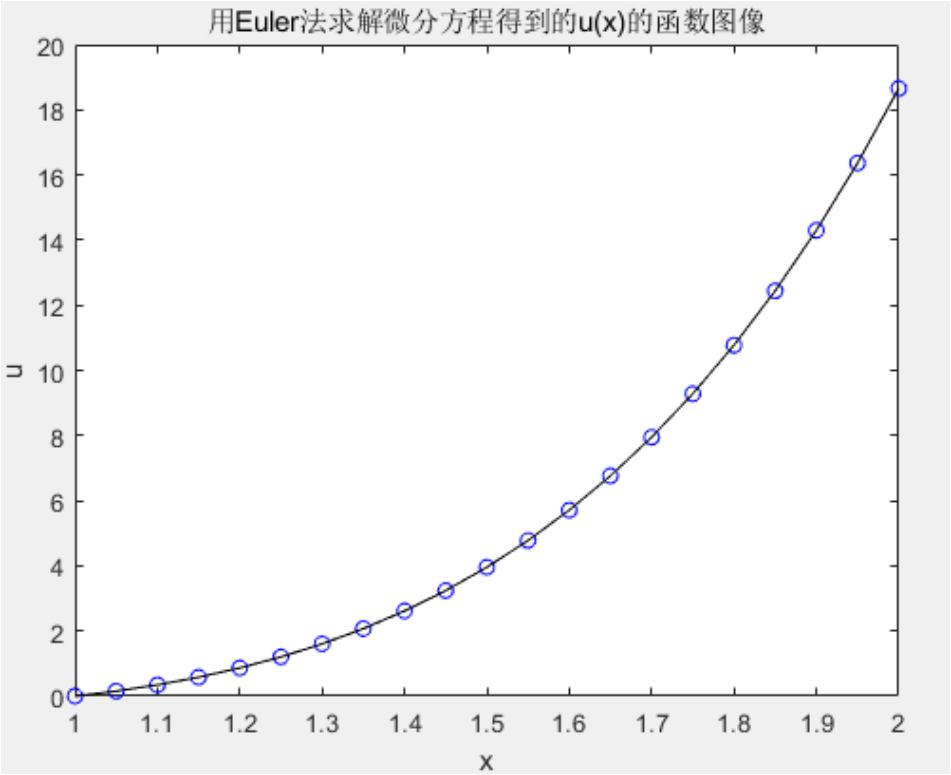
```
u=zeros(1,1/h+1);
u(1,1)=0;
i=2;
for t=1:h:1.99
    u(1,i)=u(1,i-1)+ut_(t,u(1,i-1))*h;
    u(1,i)=u(1,i-1)+h/2*(ut_(t,u(1,i-1))+ut_(t+h,u(1,i)));
    i=i+1;
end
t=(1:h:2);
plot(t,u,'g*')
hold on
plot(t,u,'k-')
title("用Euler法求解微分方程得到的u(x)的函数图像");
xlabel('x');
ylabel('u');
```

运行结果截图：

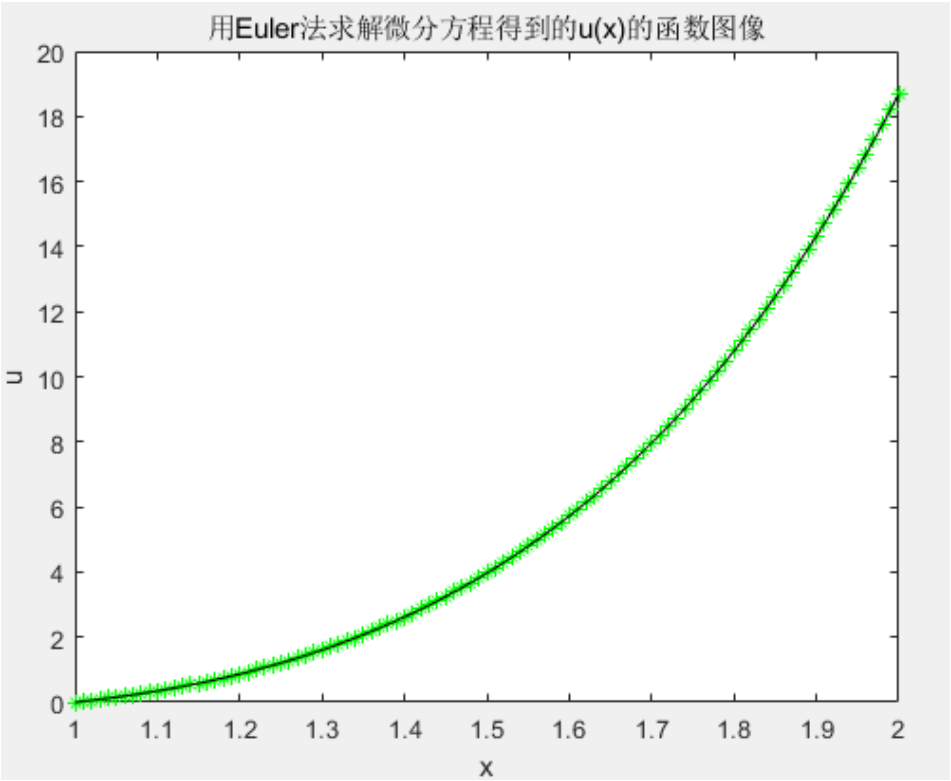
- h=0.1



- h=0.05



- h=0.01



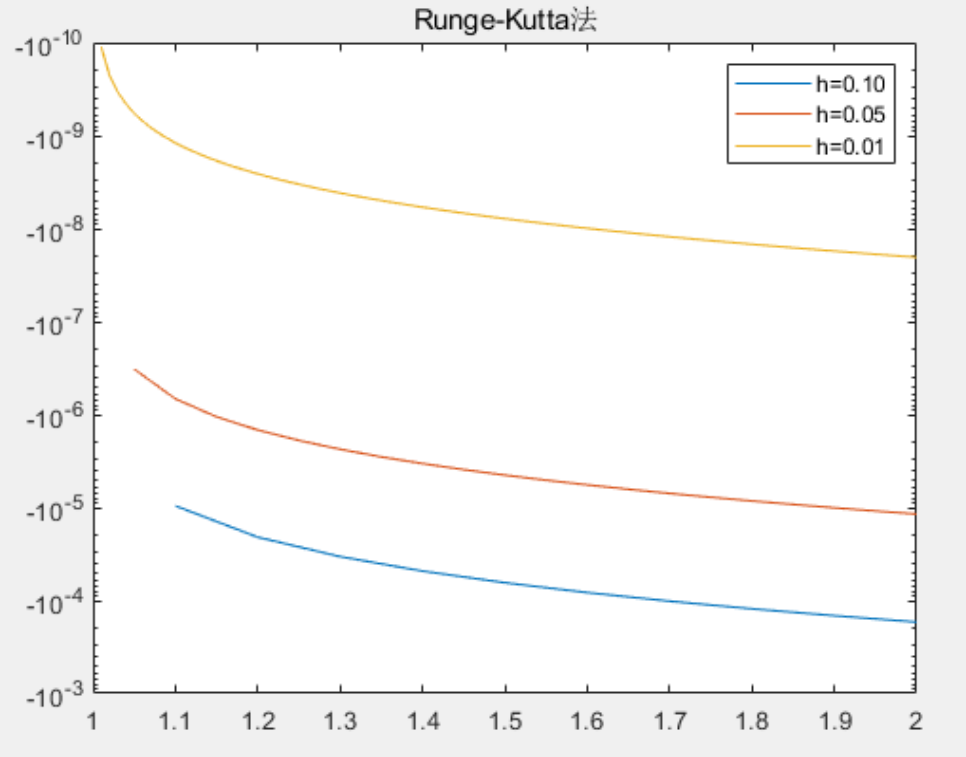
(3) Runge-Kutta法

Matlab代码:

```
clear all
y=@(x,u)2./x.*u+x.^2.*exp(x);
h=[0.1,0.05,0.01];
ty=matlabFunction(dsolve('Dy=2/x*y+x^2*exp(x)','y(1)=0','x'));
for i=1:3
    RK(1)=0;
    x=1:h(i):2;
    for j=2:length(x)
        kR1=y(x(j-1),RK(j-1));
```

```
        kR2=y(x(j-1)+h(i)/2,RK(j-1)+h(i)/2*kR1);
        kR3=y(x(j-1)+h(i)/2,RK(j-1)+h(i)/2*kR2);
        kR4=y(x(j),RK(j-1)+h(i)*kR3);
        RK(j)=RK(j-1)+h(i)/6*(kR1+2*kR2+2*kR3+kR4);
    end
    figure(1)
    semilogy(x(2:end),RK(2:end)-ty(x(2:end)))
    hold on
end
figure(1)
title('Runge-Kutta法')
legend(sprintf('h=%.2f',h(1)),sprintf('h=%.2f',h(2)),sprintf('h=%.2f',h(3)))
```

运行结果截图：



P110 例4

求定解问题 $\begin{cases} dX(t) = AX(t) \\ X(0) = (1,1,1)t \end{cases}$

其中 A=$\begin{matrix} 3 & -1 & 1 \\ 2 & 0 & -1 \\ 1 & -1 & 2 \end{matrix}$

精细积分法：

```
function x = jingxijifen(A,x0,t0,t1)
% 计算dx(t)/dt=Ax(t)的近似解，初值条件为x(t0)=x0
% 输入：矩阵A,初值x0,时间步长h
% 输出：近似解x
    n=length(x0);% 未知数个数
    I=eye(n);
    x(1:n,1)=x0;%初始值
    N=20;
    dt=h/2^N;
    At=A*dt;
```

```
    BigT=At*(I+At*(I+At*(I+At/4)/3)/2);
    for k=1:N
        BigT=BigT*(I+2*BigT);
    end
    BigT=I+BigT
    m=2^N;
    for i=1:1:m
        x0=BigT*x0;
    end
end
```

P243 例1

某化学反应方程式

$$\begin{cases} du/dx = 2/xu + x^2e^x \\ x \in [1,2], u(1) = 0 \end{cases}$$

方程右端矩阵为A=$\begin{matrix} -2000 & 999.75 \\ 1 & -1 \end{matrix}$

四阶R-K法:

```
A=[-2000,999.75;1,-1];
[u,v]=dsolve('Du=-2000*u+999.75*v+1000.25,Dv=u-v','u(0)=0,v(0)=-2');
uf=matlabFunction(u);
vf=matlabFunction(v);
uh=zeros(1,100);
vh=zeros(1,100);
uh(1)=0;
vh(1)=-2;
h=0.001;
x=0:h:1;
for j=2:length(x)
    k11=A(1,:)*[uh(j-1);vh(j-1)]+1000.25;
    k21=A(2,:)*[uh(j-1);vh(j-1)];
    k12=A(1,:)*[uh(j-1)+k11*h/2;vh(j-1)+k21*h/2]+1000.25;
    k22=A(2,:)*[uh(j-1)+k11*h/2;vh(j-1)+k21*h/2];
    k13=A(1,:)*[uh(j-1)+k12*h/2;vh(j-1)+k22*h/2]+1000.25;
    k23=A(2,:)*[uh(j-1)+k12*h/2;vh(j-1)+k22*h/2];
    k14=A(1,:)*[uh(j-1)+k13*h;vh(j-1)+k23*h]+1000.25;
    k24=A(2,:)*[uh(j-1)+k13*h;vh(j-1)+k23*h];
    uh(j)=uh(j-1)+h/6*(k11+2*k12+2*k13+k14);
    vh(j)=vh(j-1)+h/6*(k21+2*k22+2*k23+k24);
end
figure(1)
semilogy(x,abs(uh-uf(x))./abs(uf(x)))
title('u的相对误差')
figure(2)
semilogy(x,abs(vh-vf(x))./abs(vf(x)))
title('v的相对误差')
```