数值分析实验

作者:Sandy

数值分析实验<一>

---- Matlab 绪论

一、实验目的

1) 熟悉 Matlab 的运行环境及各种窗口
2) 掌握 Matlab 的矩阵变量类型,矩阵输入和矩阵的基本运算
3) 掌握命令及函数文件的作用及区别,并编写简单的 M 文件
4) 能熟练的向查寻目录中添加新目录,掌握常用的 Matlab 系统命令

二、实验内容

一> Matlab 启动与环境设置

1)启动

双击桌面图标
开始>程序>Matlab
安装目录>bin>matlab

2)环境设置

命令窗口(Command Window)　　执行命令行,Matlab 主窗口;
窗口颜色及字体　　File>Preferences..
当前目录(Current Directory)
File>Set Path　用于将新文件夹加入搜索路径,设置当前文件默认目录;

3)Matlab 常用命令

上下箭头	调出最新用过的命令,重新执行
cd+目录名	改变当前目录
help	显示当前搜索路径中所有目录名称
help+函数(类)名	查找函数(类),给出函数用法及参数
lookfor+函数关键字	查询根据关键字搜索到的相关函数
exist+变量名	变量检验函数
what	目录中文件列表
who	内存变量列表
whos	内存变量详细信息
which	确定文件位置
clc	清屏
!	调用 Dos 命令

4)联机演示系统

Help>Demos..
输入命令:intro

二>Matlab 基本运算操作

1)数据类型

变量　　区分大小写,长度不超过 31,字母开头
常量

i,j	虚单位,定义 sqrt(-1)
pi	圆周率
eps	浮点运算的相对精度 exp(-52)
NaN	Not-a-Number,表示不定值

ans　系统缺省结果输出变量

数字格式　用 format 命令控制

short　long　hex　long g

2)向量及矩阵

输入

>>a=1:4:12

>>b=1:4　(默认间距 1)

>>c=linspace(1,12,6)

>>d=[1 2 3 4;2 3 4 5;5 6 7 8];　%';'使得屏幕上不显示操作结果)

>>d　%显示 d 内容

打开 Workspace 窗口(Veiw 菜单下),双击 d,并编辑修改

>>d　%显示修改后的 d

>>e=ones(3,3)

>>s=rand(5,6)

>>h=rand(size(s))

运算

>>a+b;

>>b=ones(size(d))+d;

>>a=b';

>>c=inv(e+eye(size(e))*a;　%inv 矩阵取逆

三>Matlab 的文件

1) 命令文件

相当于在 Command Window 中逐行输入并运行命令.

* 后缀名.m
* 常用于需经常调用的命令集
* 定义的变量及其值的改变在工作空间中有效

2) 函数文件

完成特定的带有参数(返回值)的计算的函数式文件

* 后缀名.m,第一句为 function 语句
* 定义的变量在调用结束后自动 free,不影响工作空间变量
* 保存文件名必须与定义函数名一致

3) 设置当前目录(Current Directory)

单击主窗口 Current Directory 列表框浏览按钮

选定要设置为当前目录的文件

例 1:添加新的查询目录

操作 1) 在选定位置新建文件夹

2) 在主窗口 File 菜单下选定 Set Path..选项

3) 在弹出对话框中单击 Add Folder

4) 在在弹出对话框中选定新建的文件夹

5) 单击确定并保存添加后的查询目录,退出

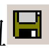
例 2:编写命令文件 demo1 完成以下操作

建立数组 a=[1,2,3,...,20],b=[1,3,5,...,39],并求 a,b 内积

操作 1） 主窗口点击新建按钮 

2） 在弹出的文本编辑窗口添加

```
a=1:20
b=1:2:39
sum=a*b'
```

3） 单击保存按钮 将文件命名为 demo1 保存在例 1 新建文件夹中

4） 在 Command Window 中输入 demo1 并回车

例 3:编写函数文件 demo2,返回输入变量的内积

操作:1） 新建 M 文件,编辑如下:

```
function sum=demo2(a,b)
    sum=a*b';
```

2） 保存文件在查询目录下,注意不要修改默认名

3） 在 Command Window 中输入

```
>>a=1:20;
>>b=1:2:39;
>>sum=demo2(a,b)
```

三、练习

1） 熟悉 Matlab 环境,进入 Demo.

2） 编写函数文件,要求返回输入矩阵的行列式(det()),秩(rank())及转置矩阵.

2004/2/28

数值分析实验<二>

-----Matlab 绘图及程序设计

一、实验目的

1） 掌握 Matlab 的控制语句

2） 熟悉数组运算

3） Matlab 图形处理功能

4） Matlab 程序初步设计

二、实验内容

一>数组运算(相同类型的运算)

1)':'引用

*A(:,n)　矩阵 A 的 n 列所有元素

```
>>A=rand(4,5);
>>A(:,3)=(1:4)'          %引用的为一列向量
```

*A(m,:)　矩阵 A 的 m 行所有元素

```
>>A(4,:)=2:6
```

*A(:)　　矩阵 A 所有元素

```
>>A(:)
```

2）变维

*reshape(X,M,N,P,..)将已知矩阵 X 变为 M*N*P..矩阵

```
>>a=1:12;
>>b=reshape(a,2,6)
```

*用':'引用

```
>>a=zeros(3,4);
>>a(:)=1:12            %Matlab 矩阵元素按列存储
>>a(4)
>>a(1,2)
```

3）'.'运算　同类型矩阵元素对应元素运算

*　".*"，"./"与".\'运算

```
>> a=[1 2 3;2 3 4;3 4 5];
>> b=[1 1 1;2 2 2;3 3 3];
>> a.*b                      %a,b 对应元素相乘
>> a*b                       %a,b 矩阵相乘
>> a.\b                      %a 对应元素做分母
>> a./b                      %b 对应元素做分母
```

* ".^"与^

```
>> b=[1 1 1;2 2 2;3 3 3];
>> b^3
>> b.^3
>> b*b*b                     %等于 b^3
```

二>程序设计

1）注释符 "%",不为 Matlab 执行,M 文件中第一段注释为文件的帮助文档,在 Commad Window 中输入 help 可见

2）控制语句

* 循环语句

```
for i=1:n              while expression
  ….                      statements
 end                    end
```

例:求 1^2+2^2+3^2+…+100^2

法 1:
```
>>  sum=0;
>> for i=1:100
      sum=sum+i*i;
    end
>> sum
```

法 2:
```
>>a=1:100;
>>sum=a*a'
```

法 3:
```
>>n=1;
>>sum=0;
>> while n<=100
      sum=sum+n*n;
      n=n+1;
   end
```

```
>> sum
```

* 选择语句

```
if expression()
    statements;
[else
    statements;]
 end
```

例:编写函数文件 demo3 实现 sgn 函数功能

操作:1)新建 M 文件,并编辑如下

```
function val=demo3(x)
 if x>0
   val=1;
 elseif x<0
         val=-1;
      else
         val=0;
     end
```

2) 将文件保存在查询目录内

3) >>demo3(0)

>>demo3(90)

>>demo3(-12)

3) 递归调用

例:编写函数文件 demo4,返回输入整数的阶乘

操作:1)新建 M 文件,并编辑如下

```
function val=demo4(n)
 if  n==1|n==0
    val=1;
 else
    val=n*demo3(n-1);    %递归
 end
```

或

```
function [val]=demo3(n)
 val=1;
 if n==0
    val=1;
 else
    for i=1:n
       val=val*i;
     end
 end
```

二>Matlab 图形处理初步

1) 二维图形

plot(x,y,s)

例 1):

```
>> x=rand(100,1);
>> y=rand(100,1);
>> z=x+y.*i;
>> plot(y)
>> plot(z);
```

例 2):

```
>> x=0.1:0.01*pi:pi;
>> y=sin(x).*cos(x);
>> plot(x,y);
```

注意: 当两个输入变量同为向量时,x,y 维数相同.x,y 为同阶矩阵时将按列或行进行.

例 3):

```
>> x=0.1:0.01*pi:pi;
>> y=[sin(x)',cos(x)'];
>> plot([x'],y)
>> plot([x',x'],y)
>> plot(x',y(:,1),x',y(:,2))
```

例 4):

```
>> x=0.1:0.1*pi:2*pi;
>> y=sin(x);
>> z=cos(x);
>> plot(x,y,'--k',x,z,'-.rd')
```

注:s 图形设置选项

选项	说明	选项	说明
-	实线	y	黄色
:	点线	r	红色
-.	点划线	g	绿色
..	虚线	k	黑色
o	圆号	+	+号
*	*号	d	菱形

2) 三维图形

* plot3(x,y,z,s)　　%其中 x,y 和 z 为 3 个相同维数的向量
* plot3(X,Y,Z,s)　　%其中 X,Y 和 Z 为 3 个相同阶数的矩阵,函数绘 3 矩阵的列向量曲线
* plot3(x1,y1,z1,s1,x2,y2,z2,s2,...)

例 1):

```
>> x=0:pi/50:10*pi;
>> y=sin(x);
>> z=cos(x);
>> plot3(x,y,z);
```

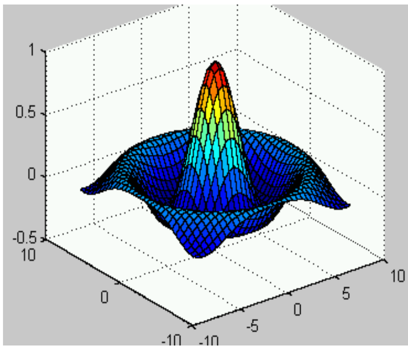
例 2):

```
>> [x,y]=meshgrid(-2:0.1:2,-2:0.1:2); %产生网格点
>> z=x.*exp(-x.^2-y.^2);
```

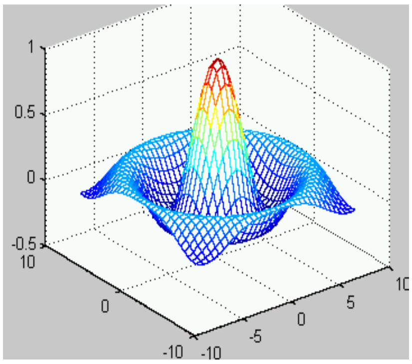
```
>> plot3(x,y,z);
```

例 3)：

```
>> x=-8:0.5:8;  y=x';
>> a=ones(size(y))*x;
>> b=y*ones(size(x));
>> c=sqrt(a.^2+b.^2)+eps;
>> z=sin(c)./c;
>> surf(x,y,z)
```



```
>> mesh(x,y,z)
```

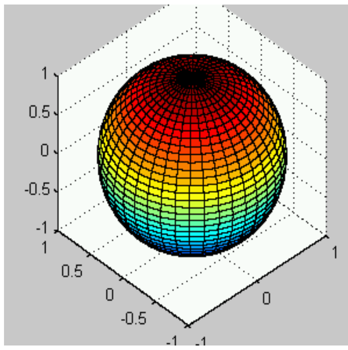


3) 特殊的三维图形函数

```
* [x,y,z]=sphere(n)        % 画球,n 默认值 20
```

例：

```
>> [a,b,c]=sphere(40);
>> surf(a,b,c)
>> axis('equal');
>> axis('square');
```

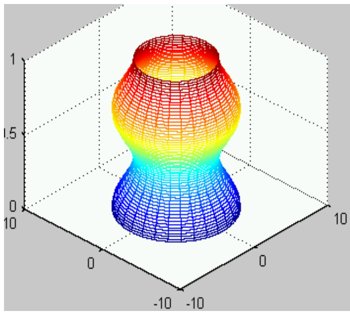


```
* [x,y,z]=cylinder(R,N)   %R 母线向量,N 分
```

格数

例：

```
>> x=0:pi/20:pi*3;
>> r=5+cos(x);
>> [a,b,c]=cylinder(r,30);
>> mesh(a,b,c)
```



2004/3/1

数值分析实验<三>

-------第二章插值法

一、实验目的

1) Matlab 中多项式的表示及多项式运算

2) 用 Matlab 实现拉格朗日及牛顿插值法

3) 用多项式插值法拟合数据

二、实验内容

一>多项式运算

1) 多项式表示

对多项式 $p = a_a x^n + a_1 x^{n-1} + \ldots. + a_{n-1} x + a_n$;用以下的行向量表示:

$$p = [a_0, a_1, \ldots., a_{n-1}, a_n]$$

这样把多项式转化为向量运算

2) 多项式表示

```
>> p=[1,-5,6,-33];
>> poly2sym(p)                          %符号表示多项式
ans =
 x^3-5*x^2+6*x-33
>> a=[1,2,3;2,3,4;3,4,5];
>> p1=poly(a)                           %矩阵 a 的特征多项式
p1 =
    1.0000    -9.0000    -6.0000    -0.0000
>> root=[-5 -3+4i -3-4i];
>> p=poly(root)                         %生成以 root 中元素为根的多项式
p =
1     11     55    125
>> poly2sym(p)
ans =x^3+11*x^2+55*x+125
```

3) 多项式运算

```
>> p=[1,11,55,125];                     %建立多项式 p
>> a=1:2:10;                            %向量 a
>> b=[1,2;2,1];                         %方阵 b
>> polyval(p,a)                         %计算 a 对应元素多项式值
ans =
        192          416          800          1392          2240
>> polyval(p,b)                         %矩阵 b 对应元素多项式值
ans =
   192   287
   287   192
>> polyvalm(p,b)                        %按矩阵多项式计算
ans =
   248   168
   168   248
>> roots(p)                             %多项式 p 的所有根
ans =
  -5.0000
  -3.0000 + 4.0000i
  -3.0000 - 4.0000i
>> p=[2 -5 6 -1 9];
>> d=[3 -90 -18];
>> pd=conv(p,d)                         %多项式 p,d 相乘
pd =
```

```
      6   -195   432   -453     9   -792   -162
>> p1=deconv(pd,d)                    %p1 为多项式 pd 除 d(等于 p)
p1 =
      2     -5      6     -1      9
>> p=[1 2 3 4 5 6];
>> q=[3 2 1];
>> [s,r]=deconv(p,q)                  %s 为商,r 余项
s =
   0.3333    0.4444    0.5926    0.7901
r =
         0          0          0          0    2.8272    5.2099
>> Dp=polyder(p)                      %对 p 求导所得多项式
Dp =
     5     8     9     8     5
```

二>拉格朗日插值法

1)算法实现

```
function [p]=Lag_polyfit(X,Y)
%Matlab 函数文件 Lag_polyfit.m
%拉格朗日插值法多项式拟合 P17
%[p]=Lag_polyfit(X,Y)
% X 拟合自变量
% Y 拟合函数值
% p 所得的拟合多项式系数
if size(X)~=size(Y)
   error('变量不匹配');
end                          %如果要拟合的函数值与自变量维数不一样,则退出报错
tic                          %开始记时
format long g                %设置最合适的数字格式
r=size(Y); n=r(2);           %n 为要拟合的数据长度
p=zeros(1,n);                %保存所得多项式系数
p0=p;  b=0;                  %工作变量
W=poly(X);                   %W 为以 X 为根的多项式
dW=polyder(W);               %dW 为对多项式 W 求导后的多项式系数
z=polyval(dW,X);             %z 为以 dW 为系数的多项式对 X 的值
A=[1 1]; r=A;                %A,r 为长度为 2 的(1,2)向量
for i=1:n                    %计算循环开始
    A=[1,-X(i)];             %A 为一次多项式 x-x(i)系数
    [p0,r]=deconv(W,A);      %进行多项式除法 W/A,p0 为商
    b=Y(i)/z(i);
    p0=b.*p0;                %p0 为累加项
    p=p+p0;
end                          %循环结束
disp(toc)                    %
```

2)例题

```
%Matlab 命令文件 LagSin.m
%用拉格朗日插值法拟合[0,2*pi]上的 sin 函数
 x0=linspace(0,2*pi,10);
y0=sin(x0);              %拟合数据采样
p=Lag_polyfit(x0,y0);    %调用 Lag_polyfit 计算拟合多项式
x=0:0.2:2*pi;
y1=sin(x);               %sin 函数值
y2=polyval(p,x);         %拟合多项式值
plot(x,y1,x,y2,'r')      %sin 函数(蓝色)与多项式(红色)效果图
```

命令窗口中输入 lagsin 观察图形输出窗口

```
>>logsin          %函数(文件)名不区分大小写
```

3)练习

＊ 已知 x,y 值,用线性插值及二次插值计算.x=0.54 的近似值

x: 0.4   0.5   0.6   0.7   0.8

y:-0.916291 -0.693147 -0.510826 -0.357765   -0.223144

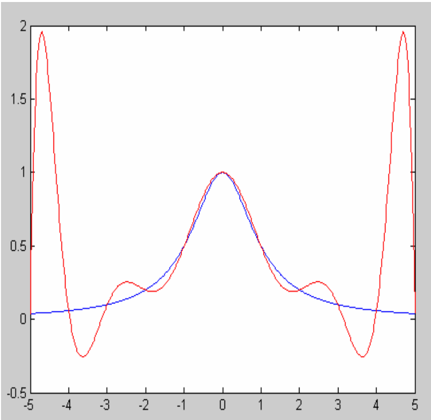
＊ 在[-5,5]区间上取 10 个采样点,用拉格朗日插值法拟合 Runge 函数 y=1/(1+x^2)+,并画出所得多项式与原函数图像,观察所得结果,思考出现的现象.

解:

```
%Matlab 命令文件 LagRunge.m
%用拉格朗日插值法拟合[0,2*pi]上的 sin 函数
x0=-5:5;
y0=ones(size(x0))./(ones(size(x0))+x0.^2);%拟合数据采样
p0=Lag_polyfit(x0,y0);      %调用 Lag_polyfit 计算拟合多项式
x=-5:0.02:5;
  y1=ones(size(x))./(ones(size(x))+x.^2);     %Runge 函数值
  y2=polyval(p0,x);                           %拟合多项式值
   plot(x,y1,x,y2,'r')        %Runge 函数  (蓝色)与多项式(红色)
```

效果图

```
>>LagRunge
```



三>牛顿插值法

1)均差表计算及算法实现

```
function [p]=Newton_Polyfit(X,Y)
 %Matlab 函数文件 Newton_polyfit.m
   %牛顿插值法多项式拟合 P23
   %[p]=Newton_Polyfit(X,Y)
   % X 拟合自变量
   % Y 拟合函数值
   % p 所得的拟合多项式系数
if size(X)~=size(Y)
  error('变量不匹配?');%如果要拟合的函数值与自变量维数不一样,则
```

退出报错

```
    end
    format long g                   %设置最合适的数字格式
    r=size(X); n=r(2);              %n 为要拟合的数据长度
    M=ones(n,n);                    %均差表工作矩阵
    M(:,1)=Y';                      %设置均差表第一列为函数值
    for i=2:n
        for j=i:n                   %高阶均差推算
           M(j,i)=(M(j,i-1)-M(j-1,i-1))/(X(j)-X(j-i+1));
        end
    end
    M                               %显示均差表
    p0=[zeros(1,n-1) M(1,1)];p=p0; %拟合多项式存储向量及初值
    for i=1:n-1
        p1=M(i+1,i+1).*poly(X(1:i)); %对应阶均差的多项式
        p0=[zeros(1,n-i-1) p1];
        p=p+p0;                     %对应项累加
    end
```

2)例题

```
  %Matlab 命令文件 NewtonSin.m
  %用牛顿插值法拟合[0,2*pi]上的 sin 函数
      x0=linspace(0,2*pi,10);
    y0=sin(x0);                 %拟合数据采样
    p=Newton_polyfit(x0,y0);    %调用 Newton_polyfit 计算拟合多项式
    x=0:0.2:2*pi;
    y1=sin(x);                  %sin 函数值
    y2=polyval(p,x);            %拟合多项式值
      plot(x,y1,x,y2,'r')         %sin 函数(蓝色)与多项式(红色)效果图
```

命令窗口中输入 newtonsin 观察图形输出窗口

```
>>newtonsin
```

3)练习

用牛顿插值法完成一>中练习.

2004/3/10

数值分析实验<四>

------第三章函数逼近

一、实验目的

4) 熟悉 Matlab 编程技巧及多项式表示

5) 用 Matlab 实现 Legendre,Chebisheff,Hermite 多项式

6) 实现正交多项式拟合

二、正交多项式及算法

1)Legendre 多项式

* [-1,1]上权函数为 1 时,由 1{1, x, x^2, x^3 , ……, x^n,…..}正交化得到正交多项式
* 递推关系式:

$P0=1 \qquad P1=x$

$P_{n+1}=(2n+1)xP_n-nP_{n-1}$

2)Chebisheff 多项式

* [-1 ,1]权函数为 $\frac{1}{\sqrt{1-x^2}}$ 时,由 1{1, x, x^2, x^3 , ……, x^n,…..}正交化得到正交多项式
* 递推关系式:

$P0=1 \qquad P1=x$

$P_{n+1}=2xP_n-P_{n-1}$

3)Hermite 多项式

* (-∞,+∞)权函数为时 e^{-x^2},由 1{1, x, x^2, x^3 , ……, x^n,…..}正交化得到正交多项式
* 递推关系式:

$P0=1 \qquad P1=x$

$P_{n+1}=2xP_n-2nP_{n-1}$

4)正交多项式实现

```
function [P]=ZhenJiao_Polies(n,flag)
 %   P 返回所得正交多项式
 %   n  所求多项式次数
 %   flag  标志正交多项式种类
 %        1: Legendre 多项式(默认)
 %        2: Chebisheff 多项式
 %      其他:Hermite 多项式
if nargin==1             %nargin  系统变量,检测函数输入参数个数
    flag=1;
end                      %如果输入参数为 1,设置默认值
if flag==1               %Legendre 多项式
   if n==0
      P=1;
   elseif n==1
      P=[1 0];
   else                                              %  递归调用
      P=([ZhenJiao_Polies(n-1,flag) 0]*(2*n-1)-...     % "…"为续行符
            [0 0 ZhenJiao_Polies(n-2,flag)]*(n-1))./(n);
   end
elseif flag==2                                        %Chebisheff 多项式
      if n==0
            P=1;
        elseif n==1
```

```
            P=[1 0];
        else
            P=[ZhenJiao_Polies(n-1,flag) 0]*2-...
                [0 0 ZhenJiao_Polies(n-2,flag)];
        end
    else                                              %Hermite 多项式
        if n==0
            P=1;
        elseif n==1
            P=[1 0];
        else
            P=[ZhenJiao_Polies(n-1,flag) 0]*2-...
                [0 0 ZhenJiao_Polies(n-2,flag)]*2*(n-1);
        end
    end
```

5)例 1: 输出前 9 次 Chebisheff 多项式表

```
>> for i=0:9
      disp([i Zhenjiao_Polies(i,2)]);
   end
0     1
1     1     0
2     2     0    -1
3     4     0    -3     0
4     8     0    -8     0     1
5    16     0   -20     0     5     0
6    32     0   -48     0    18     0    -1
7    64     0  -112     0    56     0    -7     0
8   128     0  -256     0   160     0   -32     0     1
9   256     0  -576     0   432     0  -120     0     9     0
```

例 2: 输出前 6 次 Hermite 多项式表

```
>> for i=0:6
      disp([i     zhenjiao_polies(i,3)]);
   end
 0     1
 1     1     0

 2     2     0    -2
 3     4     0    -8     0
 4     8     0   -28     0    12
 5    16     0   -88     0    88     0
 6    32     0  -256     0   456     0  -120
```

三\ 用正交函数作最小二乘拟合

1) 算法简介

如果 $g_0,g_1,g_2,g_3,..,g_n$ 是关于点集$\{x_i\}$带权 $w(x_i)$的正交函数族,即

$$(g_j,g_k)=\sum_{i=0}^{m} w(x_i)g_j(x_i)g_k(x_i)=\begin{cases}0, i\neq k;\\ A_k>0, j=k\end{cases}$$

由此,最小二乘法的发方程解为: $a_k=\dfrac{(f,g_k)}{(g_k,g_k)}=\dfrac{\sum_{i=0}^{m} w(x_i)f(x_i)g_k(x_i)}{\sum_{i=0}^{m} w_k^2(x_i)g_k^2(x_i)}$

且平方误差为: $\|\delta\|_2^2=\|f\|_2^2-\sum_{k=0}^{n} A_k a_k^2$

用这种算法编程,不用解方程组,只须用递推公式,并且当循环次数增加一次时,只要把循环增加一次.其余不变.这是目前用多项式作曲线拟合的最好算法,通用算法实现如下.

2) 算法实现

```
function [p,v]=ZXRC_Poly(X,Y,N)
    %    P72 用正交函数做最小二乘拟合
    %    [p, v]=ZXRC_Poly(X,Y,N)
    %    p:输出多项式系数
    %    v:均方误差
    %    X:拟合变量
    %    Y:拟合函数值
    %    N:多项式次数
  if size(X)~=size(Y)
      error('变量不匹配');
  end
  format long g
  p=zeros(1,N+1); p0=p; p1=p; M=p;
  a=0;b=0; c=0; d=0;v=0;
  r=size(X); q=r;
  p0=[zeros(1,N) 1]; q=polyval(p0,X);
  a=q*q'; p1=[p0(2:N+1) 0]-(X*ones(size(X'))/a).*p0;
  r=polyval(p1,X); d=r*Y'; b=r*r';
  p=((Y*q')/a).*p0+(d/b).*p1;
  if N>=2
    for i=2:N
      c=r.^2*X';
      M=[p1(2:N+1) 0]-(c/b).*p1-(b/a).*p0;
      p0=p1; p1=M;
      r=polyval(p1,X);
      a=b; b=r*r'; d=r*Y';
      p=p+(d/b).*p1;
    end
 end
```

```
if nargout>1                          %按要求输出误差
   v=sqrt(((Y-polyval(p,X)).^2)*ones(size(X')));
end
```

3）例 1: 用最小二乘法求一个形如 y=a+bx^2 的经验公式,使他与下列数据相拟合,并求均方误差

x	19	25	31	38	44
y	19.0	32.3	49.0	3.3	97.8

解:

```
>> x=[19 25 31 38 44];
>> y=[19.0 32.3 49.0 73.3 97.8];
>> [p,v]=ZXRC_Poly(x.^2,y,1)
p =
    0.0500351242191601          0.972578656906791
v =
    0.122569200640555
```

即得:

a=0.972578656906791　　　b=0.0500351242191601

均方误差为: 0.122569200640555

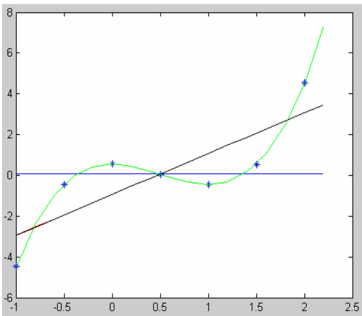
例 2: 用最小二乘法求一多项式,使他与下列数据相拟合

x	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0
y	-4.447	-0.452	0.551	0.048	-0.447	0.549	4.552

取权 $w(x_i)=1$,求拟合曲线 $y=P_n(x),(n=0,1,2,3)$并求均方误差$\|\delta\|_2^2$,并画出 $y=S_n(x)$ 的图像

解:

```
x0=[-1.0 -0.5 0.0 0.5 1.0 1.5 2.0];
 y0=[-4.47 -0.452 0.551 0.048 -0.447 0.549
4.552];
 v=zeros(1,4);
 [p1,v(1)]=zxrc_poly(x0,y0,0);
 [p2,v(2)]=zxrc_poly(x0,y0,1);
 [p3,v(3)]=zxrc_poly(x0,y0,2);
 [p4,v(4)]=zxrc_poly(x0,y0,3);
 x=-1.0:0.2:2.2;
 y1=polyval(p1,x);
 y2=polyval(p2,x);
 y3=polyval(p3,x);
 y4=polyval(p4,x);
 plot(x0,y0,'*',x,y1,'b',x,y2,'r',x,y3,'k',x,y4,'g')
```



误差: >>v

v =6.45856199165108　3.68200983548　3.68199555106945
0.00557417514717906

7) 练习

* 用上述算法计算 P70 例 8 与例 9

* 观察物体的直线运动,得出以下数据:

时间 t(秒)	0	0.9	1.9	3.0	3.9	5.0
距离 s(米)	0	10	30	50	80	110

求运动方程.

2004/3/13

数值分析实验<五>

---第四章数值积分

一、实验目的

1) 熟悉 Matlab 矩阵操作

2) 用 Matlab 实现数值积分 Cores, Romberg 及复化 Gauss 算法

3) 学会数值积分有关应用

二、试验内容

1) Cores 积分

在使用牛顿-柯斯特公式时,通过提高阶为提高精度的途径并不能取得满意的效果.通常采用复化积分法.以复化柯斯特公式为例:

将每个区间 4 等分,依此用柯斯特公式

```
function [val]=Cores_int(Y,u)
%  P89 复化 Cores 公式
%  [val]=Cores_int(Y,u)
%       Y        积分函数数值
%       u       区间间隔
%       val    返回积分值
r=size(Y); n=r(2); v=n; h=4*u;
m=mod(n-1,4);
if m==0                                %如果取值点恰巧满足复化区间数
   t=(n-1)/4; x=ones(t,1);
   val=(h/90)*(7*Y(1)+32*(Y(2:4:n-3)*x)+.....

+12*(Y(3:4:n-2)*x)+32*(Y(4:4:n-1)*x)+14*(Y(5:4:n-4)*ones(size((5:4:n-4)')))+.....
      7*Y(n));
elseif m==1
   n=n-1;
   t=(n-1)/4; x=ones(t,1);
   val=(h/90)*(7*Y(1)+32*(Y(2:4:n-3)*x)+......

+12*(Y(3:4:n-2)*x)+32*(Y(4:4:n-1)*x)+14*(Y(5:4:n-4)*ones(size((5:4:n-4)')))+...
      7*Y(n));
   n=v;
   val=val+(u/2)*(Y(n-1)+Y(n));                        %多余的区间用梯形
公式
```

```
elseif m==2
   n=n-2;
   t=(n-1)/4; x=ones(t,1);
   val=(h/90)*(7*Y(1)+32*(Y(2:4:n-3)*x)+...

12*(Y(3:4:n-2)*x)+32*(Y(4:4:n-1)*x)+14*(Y(5:4:n-4)*ones(size((5:4:n-4
)')))+...
      7*Y(n));
   n=v;
   val=val+(u/3)*(Y(n-2)+4*Y(n-1)+Y(n));           %Simpson 公式
elseif m==3
   n=n-3;
   t=(n-1)/4; x=ones(t,1);
   val=(h/90)*(7*Y(1)+32*(Y(2:4:n-3)*x)+...

12*(Y(3:4:n-2)*x)+32*(Y(4:4:n-1)*x)+14*(Y(5:4:n-4)*ones(size((5:4:n-4
)')))+...
      7*Y(n));
   n=v;
   val=val+(3*u/8)*(Y(n-3)+3*Y(n-2)+3*Y(n-1)+Y(n));     %3 阶
Newton-Cores 公式
end
```

例:对于函数 f(x)=sinx/x,利用上述算法计算[0,1]区间上积分.

```
    >>x=0:1/8:1;
    >>x=x+eps;
    >>y=sin(x)./x;
    >>[val]=Cores_int(y,1/8)
val =
       0.946083069350917
```

2) Romberg 算法:

* 复化梯形积分值递推公式:

$$T_{2n}=\frac{1}{2}T_n+\frac{h}{2}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}})$$

* Richardson 外推加速法递推公式

$$T_m^k=\frac{4^m}{4^m-1}T_{m-1}^{k+1}-\frac{1}{4^m-1}T_{m-1}^{k}\qquad(k=1,2,3,....)$$

```
function [val,M]=Romberg(f,ab,dalt)
% Romberg 算法计算积分
%  [val,M]=Romberg(f,ab,dalt)
%        val     返回积分值
%        M       T 数表
```

```
%       f       被积函数,函数文件名
%       ab      积分区间
%       dalt     精度
if nargin<3
   dalt=1e-7;                          %设置默认精度 10^-7
end
i=1; h=ab(2)-ab(1); t=0; j=0;
T(1,1)=(h/2)*(feval(f,ab)*ones(2,1));
while i<50                             %最大跌代次数为 50 次
    i=i+1; h=h/2;
    T=[T zeros(i-1,1);zeros(1,i)];
    x=ab(1)+h:2*h:ab(2)-h;
    y=feval(f,x)*ones(size(x')); % feval(f,x)求得 f 在 x 点值
    T(i,1)=T(i-1,1)/2+h*y;        % 细化区间,求得梯形值
    for t=2:i
        j=t-1;
        T(i,t)=(4^j*T(i,j)-T(i-1,j))/(4^j-1); %外推加速
    end
    if abs(T(i,i)-T(i-1,i-1))<=dalt      %控制精度
        break
    end
end
if nargout==2
   M=T;                                     %按需求返回 T 数表
end
val=T(i,i);                                 %积分值
```

例 1 利用上述算法计算积分: $\int_0^{\pi/2}\sqrt{1-\sin^2\varphi}\,d\varphi$

步骤:

1) 新建函数 fun1.m

```
function y=fun1(x)
 y=sqrt(1-sin(x).^2);
```

2) >> [val,M]=Romberg('fun1',[0 pi/2],1e-10)

```
val =
   1.00000000000000
 M =
Columns 1 through 4
0.78539816339745                    0                   0
0
0.94805944896852    1.00227987749221                    0
0
0.98711580097278    1.00013458497419    0.99999156547299
0
```

```
0.99678517188617    1.00000829552397    0.99999987622729
1.00000000814402
0.99919668048507    1.00000051668471    0.99999999809542
1.00000000002984
0.99979919432002    1.00000003226500    0.99999999997035
1.00000000000011
   Columns 5 through 6
                   0                   0
                     0                   0
                     0                   0
                     0                   0
     0.99999999999802                   0
     1.00000000000000    1.00000000000000
```

3) Gauss 公式

```
function [val]=Gauss_sum_int(f,ab,n,m)
     % 复化高斯公式积分 P98
     % [val]=Gauss_sum_int(f,ab,n,m)
     %   f    积分函数文件名
     %   ab   积分区间
     %   n    Gauss 点数
     %   m    复化区间数
 if nargin<3
      n=3;
 end
 if nargin<4
      m=4;
 end
 z=0; val=0;
 h=(ab(2)-ab(1))/(2*m);
 p=ZhenJiao_Polies(n,1);       %调用实验<四>,得到 Legendre 多项式
 r=roots(p)';                  %计算出 Gauss 点
 x=r;
 A=zeros(n,n);
 y=ones(1,n);
 for t=1:n
    A(t,:)=r.^(t-1);
    if mod(t-1,2)==1
       y(t)=0;
    else
       y(t)=2/t;
    end
 end
 y=inv(A)*y';
```

```
  d=ab(1):(ab(2)-ab(1))/m:ab(2);
  for t=1:m
     x=r*h+(d(t)+d(t+1))/2;
     z=feval(f,x);
  n=h*z*y; disp(n);                  %显示各区间Gauss积分值
  val=n+val;
end
```

例 2 利用上述算法计算积分: $\int_0^{\pi/2}\sqrt{1-\sin^2\varphi}\,d\varphi$

步骤:

1) 新建函数 fun1.m

```
function y=fun1(x)
 y=sqrt(1-sin(x).^2);
```

2)
```
>> [val]=Gauss_sum_int('fun1',[0 pi/2],4,7)

   0.22252093395631
   0.21136280516124
   0.18960606274117
   0.15834168060930
   0.11913738543439
   0.07395904427940
   0.02507208781818
val =
   1.00000000000000
```

三、练习

1) 用 Cores, Romberg 及复化 Gauss 算法计算下列积分,比较它们的精度及性能:

$$\int_0^1 \frac{x}{4+x^2}dx \qquad \int_0^1 \frac{(1-e^{-x})^{\frac{1}{2}}}{x}dx \qquad \frac{2}{\sqrt{\pi}}\int_0^1 e^{-x}dx$$

2) 应用 Cores 算法技术实现近似最佳一致逼近多项式—截断 Chebisheff 多项式(P63—6-3 式)

2004/3/14

数值分析实验<六>

---第四章数值微分第五章常微分方程数值解

一、实验目的

1)熟悉 Matlab 矩阵引用及高维矩阵操作

2)用 Matlab 实现数值微分 Richardson 外推算法

3)实现 4 阶 Runge_Kutta 常微分方程(组)数值解

4)应用 Runge_Kutta 方法解常微分方程(组)

二、试验内容

1）数值微分 Richardson 外推算法

＊ Richardson 外推加速法递推公式

$$T_m^k = \frac{4^m}{4^m - 1} T_{m-1}^{k+1} - \frac{1}{4^m - 1} T_{m-1}^k \qquad (k = 1,2,3,\ldots.)$$

＊ 算法实现

```
function [val]=Jacbi_div(f,X,n,h)
        % Richardson 外推加速法实现数值微分
        %[val]=Jacbi_div(f,X,n,h)
        %    val   返回求导(向量)函数的 Jacbi 矩阵
        %     f    求导(向量)函数名
        %     X    自变量(向量)值(不支持向量运算)
        %     n    最大跌代次数(默认 3)
        %     h    初始步长(默认 0.1)
if nargin<=3
   h=0.1;
end
if nargin==2
   n=3;
end
r=size(X); m=r(2);
b=size(feval(f,X)); b=b(2);
df=zeros(b,m);
for t=1:m
    Y=X; Y(t)=Y(t)+h;
    Z=X; Z(t)=Z(t)-h;
    df(:,t)=((feval(f,Y)-feval(f,Z))./(2*h))';
end
G(1,1,:,:)=df;
for t=2:n
    h=h/2;
    G=[G zeros(t-1,1,b,m);zeros(1,t,b,m)];
    for k=1:m
       Y=X; Y(k)=Y(k)+h;
       Z=X; Z(k)=Z(k)-h;
       df(:,k)=((feval(f,Y)-feval(f,Z))./(2*h))';
    end
    G(t,1,:,:)=df;
    for k=2:t
        r=k-1;
        G(t,k,:,:)=((4^r).*G(t,r,:,:)-G(t-1,r,:,:))./(4^r-1);
    end                                     % Richardson 外推加速
```

```
end
val=reshape(G(n,n,:),[b,m]);        %将数据整理成矩阵
```

例 1:求向量函数

$$y_1=\sqrt{1-\sin^2 x_1}+e^{x_2}$$

$$y_2=x_1^3+\cos(x_2+x_1)$$

$$y_3=x_1+x_2+x_3$$

在(0.9,2,1),(2.3,1.2,0.82)点的 Jacbi 矩阵

步骤:

1)新建函数 fun1.m

```
function y=fun1(x)
 y(1)=sqrt(1-sin(x(1)).^2)+exp(x(2));
 y(2)=x(1).^3+cos(x(2)+x(1));
 y(3)=x(1)+x(2)+x(3);
```

2)

```
>> [val]=Jacbi_div('fun1',[0.9 2 1],8,0.5)
val =
  -0.78332690962728   7.38905609893057                  0
   2.19075067078603  -0.23924932921398                  0
   1.00000000000000   1.00000000000000   1.00000000000000

>> [val]=Jacbi_div('fun1',[2.3,1.2,0.82],8,0.5)
val =
   0.74570521217667   3.32011692273652                  0
  16.22078322768936   0.35078322768953                  0
   1.00000000000000   1.00000000000000   1.00000000000000
```

2)常微分方程(组)数值解--- Runge_Kutta 方法

* 四阶 Runge_Kutta 经典方法

$$y_{n+1}=y_n+\frac{h}{6}(K_1+2K_2+2K_3+K_4);$$

$$K_1=f(x_n,y_n);$$

$$K_2=f(x_n+\frac{h}{2},y_n+\frac{h}{2}K_1);$$

$$K_3=f(x_n+\frac{h}{2},y_n+\frac{h}{2}K_2);$$

$$K_4=f(x_n+h,y_n+hK_1);$$

* 算法实现

```
function [T]=Runge_Kutta(f,x0,y0,h,n)
  %  [T]=Runge_Kutta(f,x0,y0,h,n)
  %    f        待解方程(组)
```

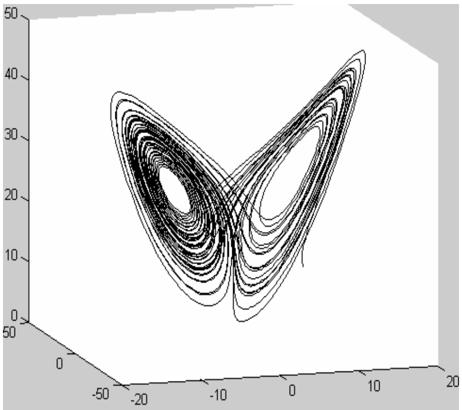
```
% x0 初试自变量值
% y0 初试函数值
% h 步长
% n 步数
if nargin<5
    n=100;
end
r=size(y0); r=r(1);
s=size(x0); s=s(1);
r=r+s;
T=zeros(r,n+1);
T(:,1)=[y0;x0];
for t=2:n+1
        k1=feval(f,T(1:r-1,t-1));
        k2=feval(f,[k1*(h/2)+T(1:r-1,t-1);x0+h/2]);
        k3=feval(f,[k2*(h/2)+T(1:r-1,t-1);x0+h/2]);
        k4=feval(f,[k3*h+T(1:r-1,t-1);x0+h]);
        x0=x0+h;
        T(:,t)=[T(1:r-1,t-1)+(k1+k2*2+k3*2+k4)*(h/6);x0];
end
```

例 2:求解下列著名的洛伦兹吸引子微分方程组

$$dx_1 = Q*(x_2 - x_1)$$
$$dx_2 = (R - x_3)*x_1 - x_2$$ 其中 Q,B,R 为常数.
$$dx_3 = x_1*x_2 - B*x_3$$

步骤:

1)新建函数 fun.m




```
function dy=fun(x)
 R=28.025;
 B=8.0/3;
 Q=10.0;
dy(1)=Q*(x(2)-x(1));
dy(2)=((R-x(3))*x(1)-x(2));
dy(3)=(x(1)*x(2)-B*x(3));
dy=dy';
```

2)运算:

```
>> T=Runge_Kutta('fun',0,[10.1;9.8;11.2],0.01,5000);
>> plot3(T(1,:),T(2,:),T(3,:))
```

3)观察图形窗口,改变角度,如图

三\练习

1) 参照 Jacbi_div.m 用 Richardson 外推算法或五点法实现支持向量运算的数值微分程序.

2) 用梯形或 Euler 公式实现常微分方程数值解

3) 改变例 2 初始值或参数,观察解的变化,你有何想法

```
>>T=Runge_Kutta('fun',0,[10.1;9.8;11.2],0.01,5000);
>>plot3(T(1,:),T(2,:),T(3,:))
>>hold on
>>T=Runge_Kutta('fun',0,[10.1;9.8;11.199],0.01,5000);
>>plot3(T(1,:),T(2,:),T(3,:),'r')
>>hold off
```

4) 求解下列微分方程(组)

* $\begin{matrix} dy = x + y \\ y(0) = 1 \end{matrix}$ * $\begin{matrix} dy_1 = y_2 - y_3 \\ dy_2 = y_1 \\ dy_3 = y_3 \end{matrix} \quad y(0) = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$

* $\begin{matrix} y'' - 3y' + 2y = 0 \\ y(0) = 1 \\ y'(0) = 1 \end{matrix}$

画出函数图象.

2004/3/16

数值分析实验<七>

---第六章方程求根

一、实验目的

1)熟悉用 Matlab 实现跌代

2)用 Matlab 实现二分,牛顿等求根算法

3)应用算法求解方程根

二、试验内容

1) 二分法

二分法不断搜索有根区间,最终收缩为一点.算法简单,且收敛性总能得到保障.实现如下:

```
function [val,n]=ErFen_root(f,x,dalt)
%   二分法求根算法 P144
%  [val,n]=ErFen_root(f,x,dalt)
%   f        要求根函数名
%   x        初试有根区间
%   dalt     精度,默认为 10^-5
%   val      返回所得根
%   n        跌代次数
if nargin<3
    dalt=1e-5;
end
```

```
ab=x;i=0;x0=sum(ab)/2;
f0=feval(f,ab(1));
f1=feval(f,x0);
while abs(f1)>=dalt
   if f0*f1<0
      ab(2)=x0;
   else
      ab(1)=x0;
   end
   x0=sum(ab)/2;
   f0=feval(f,ab(1));
   f1=feval(f,x0);
   i=i+1;
   disp(sym([ab(1) x0 ab(2) f0 f1 feval(f,ab(2))]));
                          % 显示跌代过程
   if (ab(2)-x0)<2*eps
       disp('May not be root!');
       break;
   end
end
val=x0;
if nargout==2
     n=i;
end
```

例 1：求方程 $f(x)=x^3-x-1=0$ 在区间[1.0 1.5]内的一实根,要求精确到第四位小数

步骤：

1)新建函数 F1.m

```
function f=F1(x)
 f=x.^3-x-1;
```

2)运算,显示跌代过程如下：

```
>> [val,n]=ErFen_root('F1',[1 1.5],1e-4)
    1.2500    1.3750    1.5000   -0.2969    0.2246    0.8750
    1.2500    1.3125    1.3750   -0.2969   -0.0515    0.2246
    1.3125    1.3438    1.3750   -0.0515    0.0826    0.2246
    1.3125    1.3281    1.3438   -0.0515    0.0146    0.0826
    1.3125    1.3203    1.3281   -0.0515   -0.0187    0.0146
    1.3203    1.3242    1.3281   -0.0187   -0.0021    0.0146
    1.3242    1.3262    1.3281   -0.0021    0.0062    0.0146
    1.3242    1.3252    1.3262   -0.0021    0.0020    0.0062
    1.3242    1.3247    1.3252   -0.0021   -0.0000    0.0020
val =     1.3247
```

```
n =      9
```

3)验证:

```
>> F1(val)
ans =
 -4.6595e-005
```

2) Newton 切线法求非线性方程(组)

* Newton 切线法跌代公式:

$$x_1 = x_0 - f_0 / f_0^{'}$$

* 算法实现:

```
function [val,n]=Newton_root(f,a,daltf,daltx)
      % Newton 切线法求非线性方程(组)  P151
  % [val,n]=Newton_root(f,a,daltf,daltx)
  %   val:输出解
  %      n:迭代次数(只有一个输出参数时不输出)
  %      f:待解方程(组)
  %      a:初始迭代值
  % dlatf:函数精度控制(默认为 1e-6)
  % daltx:解精度控制(默认为 1e-5)
tic                                         %记时开始
if nargin<4                                 %设置默认值
    daltx=1e-5;
end
if nargin<3
    daltf=1e-6;
 end
flag=0; i=size(a);
if i(2)==1;                                 %如果不为方程组,画图
   flag=1;
end
hold on
%初始参数设置
%  i:迭代记数
%  e:解精度
% x0:初始解
% x1:迭代一次
%df0:x0 点 Jacobi 矩阵
%df1:x1 点 Jacobi 矩阵
i=0; e=0; df1=0; x1=0; x0=a;
f0=feval(f,x0);d=ones(size(f0'));
df0=Jacbi_div(f,x0);                %调用实验<六> Jacbi_div 求 Jacobi 矩
阵
while sqrt((f0.^2)*d)>=daltf
```

```
    x1=x0-f0*inv(df0');                          %新迭代得到值,x0 为行向
量
    f1=feval(f,x1);
    df1=Jacbi_div(f,x1);
    i=i+1;
    if sqrt(x1.^2*d)<1
        e=sqrt((x0-x1).^2*d);
    else
        e=sqrt(((x0-x1).^2)*d./(x1.^2))*d;
    end
    if e<daltx
        break
     end                                         %满足精度,退出
    if i==100
        disp('No root find!');
        break
    end                                     %安全设置(最大迭代次数:100)
    if flag==1
      plot([x0 x1],[f0 0],'r',[x0 x0],[0,f0],'-.r');            %画图
    end
    x0=x1; df0=df1; f0=f1;                     %准备下次迭代
    disp(i)
    disp([x0' f0'])                            %输出跌代数据
end
if flag==1
   X=x1-(a-x1):(a-x1)/50:x1+(a-x1);
   plot(X,feval(f,X),'k');                     %画函数图象
end
hold off                                       %输出设置
val=x1;
if nargout==2
    n=i;
end                                            %记时结束
disp('耗时:')
disp(toc)
```

例 2: 用 Newton 切线法求非线性方程

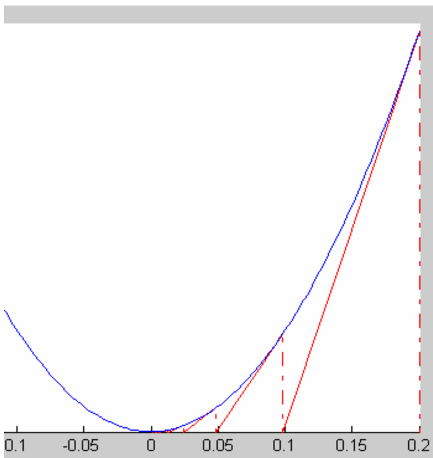
$y = 1 - e^{-x^2} = 0$ 在 0.2 附近的根

并画出跌代过程图形

解 步骤:

1)新建函数 fun1.m

```
function  y=fun1(x)
    %待界解方程
```



```
        y=1-exp(-x.^2);
```

2)运算,显示跌代过程如下:

```
>> [val,n]=Newton_root('fun1',0.2 ,1e-6,1e-8)
     1        0.0979730645190195       0.00955280068932718
     2        0.0487506741811492       0.00237380628825135
     3        0.0243463485729836      0.000592569050408831
     4         0.012169565781348      0.000148087365290039
     5       0.00608433229533475     3.70184142816088e-005
     6       0.00304210983785066     9.25438944343604e-006
     7       0.00152104788065106     2.31358397884129e-006
     8      0.000760523060572527     5.7839515843483e-007
耗时:          0.109999999999999
val =       0.000760523060572527
n =     8
```

例 2: 用 Newton 切线法求非线性方程组

$$y_1 = 3x_1^2 - x_2^2 = 0$$
$$y_2 = 3x_1x_2^2 - x_1^3 - 1$$

在[2,0.5]附近的根

解　步骤:

1)新建函数 fun2.m

```
    function y=fun2(x)
      %待界解方程组
      y(1)=3*x(1)^2-x(2)^2;
      y(2)=3*x(1)*x(2)^2-x(1)^3-1;
```

2)运算,显示跌代过程如下:

```
  >> [val,n]=Newton_root('fun2',[2,0.5] ,1e-9,1e-8)
     1    0.962962962962974           2.74408436213998
        -0.194444444444433          -1.78372834425651
     2    0.524848846327421           0.276521282534161
        -0.741537357076171         -0.278770222063188
     3    0.497758653672618        -0.0144211217636422
        -0.870466629849516        0.00814683001895977
     4    0.499993845199779     -4.80634978106131e-006
        -0.866017518334799     -2.97189108221518e-005
     5  0.500000000060973      5.14654985295238e-011
        -0.866025403860333      2.88638446477307e-010
耗时:      0.329999999999998
val =      0.500000000060973         -0.866025403860333
n =        5
```

三、练习

1) 参照 Newton_root.m 实现弦截及抛物法求方程根.

2) 用二分法和 Newton 切线法求根,注意初始区间或初始值

*　　$x^2 - x - 1 = 0$

* $1-x\sin x=0$

3) 试用跌代公式 $x_{k+1}=(2-e^{x_k})/10$ 求 $e^x+10x-2=0$ 的根.

2004/3/19

数值分析实验<八>

---第七章解线性方程组的直接方法

一、实验目的

1)熟悉用 Matlab 向量运算

2)用 Matlab 矩阵求逆及三角分解

3)掌握解线性方程组的直接方法

二、试验内容

1) Gauss_Jordan 列主元消去法求方阵逆

选择列主元,消去对角线上方和下方的元素.用其实现矩阵求逆如下:

```
function [A,det]=GaJo_inv(A)
% Gauss-Jordan 列主元消去法求方阵逆 P178
% [A,det]=GaJo_inv(A)
%       A          要求逆矩阵
%       det        按需求返回 A 的行列式
%       A          返回逆矩阵,放在 A 中
n=size(A);
if n(1)~=n(2)
   error('不是方阵!');
end
n=n(1);det=1;
flag=1:n;
for k=1:n
      t=find(abs(A(k:n,k))==max(abs(A(k:n,k))));  %寻找主元
      t=t(1)+k-1;
      flag(k)=t;
      if  t~=k
         p=A(k,:); A(k,:)=A(t,:); A(t,:)=p;        %交换行
         det=-det;
      end
      if A(k,k)==0
         error('矩阵不可逆!');
      end
      det=det*A(k,k);
      h=1/A(k,k); A(k,k)=h;
      A([1:k-1 k+1:n],k)=A([1:k-1 k+1:n],k)*(-h);
      for i=1:n
         if i~=k
            A(i,[1:k-1 k+1:n])=A(i,[1:k-1 k+1:n])+A(k,[1:k-1....
               k+1:n])*A(i,k);                             %消去
         end
```

```
            end
            A(k,[1:k-1 k+1:n])=A(k,[1:k-1 k+1:n])*h;
      end
      for k=n-1:-1:1
          t=flag(k);
          if k~=t            %调整A列(因为原矩阵换行)
              p=A(:,t);  A(:,t)=A(:,k); A(:,k)=p;
          end
       end
```

例 1: 产生 4,4 随机矩阵,求其逆和行列式,并作验证.

解:

```
>> A=rand(4,4)*10;
>> [B,det]=GaJo_inv(A)
B =
  -0.02351535074656 0.07531700884091 0.92750347291607
-0.40534881697594
   0.00202471257148 0.12120618957662-0.73647346208479 0.25918375866559
  -0.01484042304063-0.05935463795098-0.02379166719309
0.18823286310200

0.12131866286626-0.00724039036876-0.07729738792570-0.07634287303686
det =  3.599068227664570e+002
>> A*B
ans =
   1.00000000000000  -0.00000000000000  -0.00000000000000        0
   0.00000000000000   1.00000000000000   0.00000000000000   -0.0000
                  0  -0.00000000000000   1.00000000000000     0.000
                  0  -0.00000000000000   0.00000000000000     1.000
```

例 2: 解方程组

$$
\begin{aligned}
0.6428\ x_1 + 0.3475\ x_2 - 0.8468\ x_3 &= 0.4127 \\
0.3475\ x_1 + 1.4823\ x_2 + 0.4759\ x_3 &= 1.7321 \\
-0.8468\ x_1 + 0.4759\ x_2 + 1.2147\ x_3 &= -0.8621
\end{aligned}
$$

并验证.

解:

```
>> A=[0.6428 0.3475 -0.8468;0.3475 1.8423 0.4759;-0.8468 0.4759 1.2147];
>> b=[0.4127;1.7321;-0.8621];
>> x=GaJo_inv(A)*b
x =4.58668603133057
  -0.63152317337598
   2.73520013957385
>> A*x-b
ans =
  1.0e-015 *
```

0.05551115123126
0.22204460492503
0.55511151231258

2)Gauss 消去法变形----选主元的三角变形法

通过交换矩阵的行实现矩阵的 PA=LU 分解,采用与列主元相似的方法,避免一些奇异情况的情况下分解无法进行.具体实现如下:

```
function [L,A,P]=LU_P(A)
%选主元的三角变形法 P181
%[L,A,P]=LU_P(A)
%      A    待分解阵
%      L    返回单位下三角阵
%      A    返回上三角阵,存储在原矩阵中
%      P    返回排列阵
n=size(A);
if n(1)~=n(2)
     error('不是方阵!');
end
n=n(1);flag=1:n;
P=eye(n,n);L=P;
for k=1:n
   if k~=1
     A(k:n,k)=A(k:n,k)-A(k:n,1:k-1)*A(1:k-1,k);
   end
   t=find(abs(A(k:n,k))==max(abs(A(k:n,k))));
   t=t(1)+k-1;          %选主元
   flag(k)=t;
   if  t~=k
     p=A(k,:); A(k,:)=A(t,:); A(t,:)=p;
   end                      %交换行
  if abs(A(k,k))<eps
      error('Divided by zero!');
  end
  if k~=n
     A(k+1:n,k)=(1/A(k,k))*A(k+1:n,k);
     A(k+1:n,k)=A(k+1:n,k);
     A(k,k+1:n)=A(k,k+1:n)-A(k,1:k-1)*A(1:k-1,k+1:n);
  end
end
for k=1:n-1
    L(k+1:n,k)=A(k+1:n,k);
    A(k+1:n,k)=zeros(n-k,1);
end                           %得到 L,U
for k=n-1:-1:1
  t=flag(k);
```

```
        if k~=t                    %按主元配制排列阵
            p=P(:,t); P(:,t)=P(:,k); P(:,k)=p;
        end
    end
```

例 3: 产生 4,4 随机矩阵,做 PLU 分解,并作验证.

```
>> A=rand(4,4)*10;
>> [L,U,P]=LU_P(A)
L =
    1.00000000000000                 0                     0                  0
    0.07450685712708    1.00000000000000                   0                  0
    0.02101120108141    0.69070759931831    1.0000000000                       0
    0.65704199497446    0.15199566571131    0.12874801043619
1.00000000
U =
7.32460073511733 2.62755981990431 8.68838302205674 1.63010327966347
               0 8.82287890038379-0.60802733855546 2.55176073908108
               0                0 8.24709673417069-0.69241282842957
               0                0                0 6.76421347812075
P =
     0     0     0     1
     0     1     0     0
     1     0     0     0
     0     0     1     0
>> P*A-L*U
ans =
  1.0e-016 *
0                  0                  0                  0
0                  0  -0.41633363423443                  0
0                  0                  0                  0
0                  0                  0                  0
```

三、练习

1) 参考 LU_P.m 及 P181 算法实现基于 PLU 分解的线性方程组的解法与非奇异方阵的求逆.

2) 用 PLU 分解下列方阵:

$$A=\begin{bmatrix}1&2&3\\2&4&1\\4&6&7\end{bmatrix},\qquad B=\begin{bmatrix}1&1&1\\2&2&1\\3&3&1\end{bmatrix},\qquad C=\begin{bmatrix}1&2&6\\2&5&15\\6&15&46\end{bmatrix}$$

3) 求不同方法解方程组,比较它们的效率及性能:

$$\begin{bmatrix} 0 & 3 & 4 \\ 1 & -1 & 1 \\ 2 & 1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

2004/3/21

数值分析实验<九>

---追赶法及跌代法解线性方程组

一、实验目的

1)追赶法解三对角阵

2)掌握解线性方程组的跌代方法

3)用 Matlab 实现 Jacobi 及超松弛跌代法

二、试验内容

1) 追赶法解三对角矩阵方程

把 Gauss 消去法用到解三对角阵方程组,计算公式简单,算法稳定,具体实现如下:

```
function [x]=ZhuiGan(a,b,c,f)
% 用追赶法解三对角矩阵方程 P186
% [x]=ZhuiGan(a,b,c,f)
% a 为对角下的向量
% b 为对角向量
% c 为对角上向量
% f 为方程常数项
r=size(a);
m=r(2);
r=size(b);
n=r(2);
if size(a)~=size(b)|m~=n-1|size(b)~=size(f)
    error('变量不匹配,检查变量输入情况!');
end
p=ones(1,m);
Y=ones(1,n);
x=Y;
p(1)=a(1)/b(1);
Y(1)=f(1)/b(1);
t=0;
for i=2:m
    t=b(i)-a(i-1)*p(i-1);
    p(i)=c(i)/t;
    Y(i)=(f(i)-a(i-1)*Y(i-1))/t;
end
Y(n)=(f(n)-a(n-1)*Y(n-1))/(b(n)-a(n-1)*p(n-1));
x(n)=Y(n);
```

```
for i=n-1:-1:1
    x(i)=Y(i)-p(i)*x(i+1);
end
```

例 1 用追赶法解三对角矩阵方程 AX=b,其中:

$$A=\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix}, b=\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} .$$

解:

```
>> a=-1*ones(1,5);
>> c=a;
>> b=2*ones(1,6);
>> f=[1 0 1 0 0 1];
>> [x]=ZhuiGan(a,b,c,f)'
x =
   1.57142857142857
   2.14285714285714
   2.71428571428571
   2.28571428571429
   1.85714285714286
   1.42857142857143
```

2) Jacobi 跌代法解线性方程组

跌代公式: $x^{(0)}$初始向量,
$$x^{(k+1)} = B_0 x^{(k)} + f$$

```
function [x,n]=Jacobi_Solve(A,b,x0,dalt)
%  Jacobi 跌代法解线性方程组 P213
%[x,n]=Jacobi_Solve(A,b,x0,dalt)
%    A          方程组系数
%    b          常数项(列向量)
%    x0         初始值,默认为 0
%    dalt       精度,默认为 10^-8
%    x          返回跌代结果
%    n          返回跌代次数
e=1; i=0;
r=size(b); a=b;
if nargin<4
    dalt=1e-8;
```

```
end
if nargin<3
    x=zeros(r);
else
    x=x0;
end
r=r(1);
for t=1:r
    a(t)=A(t,t);
    A(t,t)=0;
    A(t,:)=A(t,:)/a(t);
end
b=b./a;
while e>=dalt
    Y=b-A*x;
    e=max(abs(Y-x));
    x=Y; i=i+1;
end
if nargout>1
    n=i;
end
```

例 3 对不同初值用 Jacobi 跌代法解例 1 中方程组 AX=b,比较结果.

```
>>x0=[1.5 2.0 3 1 1.5 1.8]';
>> [x,n]=Jacobi_Solve(A,f',x0)
x =
   1.57142857132519
   2.14285713577429
   2.71428571405342
   2.28571427688211
   1.85714285695657
   1.42857142464074
n =
   174
```

```
>> [x,n]=Jacobi_Solve(A,f')
x =
   1.57142854655303
   2.14285709549202
   2.71428565839088
   2.28571422665094
   1.85714281231868
   1.42857140228577
n =
   169
```

3) 超松弛跌代法解线性方程组

超松弛跌代法是 Gauss-Seidel 跌代法的一种加速,是解大型稀疏矩阵方程组的有效方法之一.但需要选择好的加速因子(最佳松弛因子).

```
function [x,n]=SOR_Solve(A,b,w,x0,dalt)
%  超松弛跌代法解线性方程组 P213
%[x,n]=SOR_Solve(A,b,w,x0,dalt)
%    A          方程组系数
%    b          常数项(列向量)
%    w          松弛因子
%    x0         初始值,默认为 0
```

```
%    dalt       精度,默认为 10^-8
%    x          返回跌代结果
%    n          返回跌代次数
r=size(b); a=b;
if nargin<5
    dalt=1e-8;
end
if nargin<4
    x=zeros(r);
else
    x=x0;
end
r=r(1); m=0; e=1;
for t=1:r
    a(t)=A(t,t);
    A(t,t)=0;
    A(t,:)=A(t,:)/a(t);
end
b=b./a;
while e>dalt
    e=0;
   for i=1:r
       t=x(i);
       x(i)=(1-w)*x(i)+w*(b(i)-A(i,:)*x);
       t=abs(x(i)-t);
       if t>e
           e=t;
       end
   end
   m=m+1;
end
if nargout>1
    n=m;
end
```

例 3 对不同松弛因子用 SOR 解例 1 中方程组 AX=b,并与例 2 比较结果.

解:

```
>> [x,n]=SOR_Solve(A,f',1.42)
x =
   1.57142856973855
   2.14285714102157
   2.71428571277593
   2.28571428491457
   1.85714285690737
   1.42857142843605
n =
   26
>> [x,n]=SOR_Solve(A,f',1)
x =
   1.57142855091549
   2.14285710955437
   2.71428567687039
   2.28571425200424
```

```
1.85714283278664                n =
1.42857141639332                     87
```

三、练习

1)实现 Gauss-Seidel 跌代法.

2) 用 SOR 方法解方程组(取 w=0.9) $\begin{cases} 5x_1 + 2x_2 + x_3 = -12; \\ -x_1 + x_2 + 2x_3 = 20; \\ 2x_1 - 3x_2 + 10x_3 = 3; \end{cases}$ 要求精度 10^{-4}.

数值分析实验<十>

---第九章矩阵的特征值

一、实验目的

1)幂法求矩阵的特征值

2)Jacobi 跌代法求矩阵的特征值

二、试验内容

1)幂法求计算矩阵的主特征值及主特征向量

幂法是一种计算矩阵的主特征值及主特征向量的一种跌代法.过程中矩阵不变,适合稀疏矩阵,单有时收敛过慢.

```
function [x0,v,n]=Mifa_eig(A,dalt,x0,p)
%     P223 幂法求主特征向量及特征值
%[x,v,n]=Mifa_eig(A,dalt,x0,p)
%    x:输出主特征向量
%    v:输出主特征向量
%    n:迭代次数
%    A:方阵
%dalt:精度
%   x0:初始向量
%    p:中心加速
% P223 幂法求主特征向量及特征值
n=size(A);
n=n(1); e=1;
v=0; m=0;
if nargin<4
   p=0;
end
if nargin<3
   x0=ones(n,1);
end
if nargin<2
   dalt=1e-6;
```

```
end
for i=1:n
   A(i,i)=A(i,i)-p;
end
while e>dalt
   x=A*x0;
   t=0;
   for i=1:n
      if abs(x(i))>t
         t=x(i);
      end
   end
   x=x/t;
   e=max(abs(t-v),max(abs(x-x0)));
   v=t; x0=x;
  m=m+1;
end
v=v+p;
if nargout>2
   n=m;
end
```

例 1 产生一对称矩阵,对不同的原点平移和初值用幂法求计算矩阵的主特征值及主特征向量

```
>>A=rand(4,4)*10;
>>A=A+A';
>>x0=[2 1 3 1]';
>>[x,v,n]=Mifa_eig(A)
x= 0.66938884032827
   0.85217987426127
   0.55024412397770
   1.00000000000000
v= 31.26897549924593
n= 17
>>[x,v,n]=Mifa_eig(A,1e-10,x0)
x= 0.66938884845095
   0.85217986205368
   0.55024412372418
   1.00000000000000
v= 31.26897572767276
n= 29
>>[x,v,n]=Mifa_eig(A,1e-10,[1 1 1 1]',1.5)
x= 0.66938884845270
   0.85217986205105
   0.55024412372412
   1.00000000000000
v= 31.26897572771655
n=  32
>>[x,v,n]=Mifa_eig(A,1e-10,x0,3)
x= 0.66938884845307
   0.85217986205049
   0.55024412372411
   1.00000000000000
v= 31.26897572771970
n=  42
```

2) Jacobi 方法计算实对称矩阵所有特征值和特征向量

Jacobi 方法通过一组平面旋转(正交相似变换)将对称矩阵化为对角阵,

得所有特征值和特征向量. 若 $A \in R^{n\times n}$ 为对角阵, 则存在正交阵 P, 使

$$PAP^T = diag[\lambda_1, \lambda_2, \lambda_3, ..., \lambda_n] \equiv D$$

```
function [v,R]=Jacobi_eigs(A,dalt)
%用 Jacobi 方法计算实对称矩阵所有特征值
%      和特征向量
% [v,R]=Jacobi_eigs(A,dalt)
%        v            所有特征值
%        R           阵交变换矩阵
%        A           对角化矩阵
%       dalt       精度
tic
if A~=A'
   error('输入对称方阵!');
end
if nargin<2
   dalt=1e-6;
end
n=size(A); n=n(2);
e=0; R=eye(n);
m=0;
for i=1:n
   for j=i+1:n
      e=e+A(i,j)^2;
   end
end
e=e;
while e>=dalt
   m=m+1;
   i=1; j=2;
   for c=1:n
      for s=c+1:n
         if abs(A(c,s))>abs(A(i,j))
            i=c; j=s;
        end
      end
   end
   e=e-A(i,j)^2;
   d=(A(i,i)-A(j,j))/(2*A(i,j));
   t=1/(abs(d)+sqrt(d^2+1));
   if d>0
      c=1/sqrt(1+t^2);
      s=c*t;
```

```
        else
            c=-1/sqrt(1+t^2);
            s=-c*t;
        end
        p=A(i,:);
        q=A(j,:);
        A(i,i)=p(i)*c^2+q(j)*s^2+2*p(j)*s*c;
        A(j,j)=p(i)*s^2+q(j)*c^2-2*p(j)*s*c;
        A(i,j)=(q(j)-p(i))*c*s+p(j)*(c^2-s^2);
        A(j,i)=A(i,j);
        for t=1:n
            if t~=i&t~=j
                A(i,t)=p(t)*c+q(t)*s;
                A(t,i)=A(i,t);
                A(j,t)=q(t)*c-p(t)*s;
                A(t,j)=A(j,t);
            end
        end
        p=R(:,i); q=R(:,j);
        for t=1:n
            R(t,i)=p(t)*c+q(t)*s;
            R(t,j)=-p(t)*s+q(t)*c;
        end
        if m>200
            %break;
        end
    end
    v=A(1,1);
    for i=2:n
        v=[v A(i,i)];
    end
    toc
```

例 2 产生一对称矩阵,用 Jacobi 方法求计算矩阵的所有特征值及所有特征向量,并做验证.

```
 >>A=rand(4,4)*10;
 >>A=A+A';
 >>[v,R]=Jacobi_eigs(A,1e-16)
v =
-2.57590902946911-0.90778403153066 3.36484699629322 48.22836753465661
R =
-0.75060637258137 0.01966820042378-0.50941062467560 0.42036180939750
-0.00875667868297 0.81701589320297 0.39274940382081 0.42208560343915
 0.65344434683384-0.05217120189433-0.49231400096801 0.57263913220689
```

```
-0.09759036637686-0.57391319712892 0.58640911121578 0.56322652355407
   >>R'*R
   ans =
 1.00000000000000                 0-0.00000000000000 0.00000000000000
                 0 1.00000000000000 0.00000000000000 0.00000000000000
-0.00000000000000 0.00000000000000 1.00000000000000 0.00000000000000
 0.00000000000000 0.00000000000000 0.00000000000000 1.00000000000000
   >>R'*A*R
   ans =
-2.57590902946911 0.00000000099639-0.00000000000000-0.00000000000031
 0.00000000099639-0.90778403153066 0.00000000000001 0.00000000000000
-0.00000000000000 0.00000000000001 3.36484699629322 0.00000000000001
-0.00000000000031 0.00000000000001 0.00000000000001 48.22836753465659
```

三、练习

1）幂法求计算矩阵的主特征值及主特征向量

$$A=\begin{bmatrix}7 & 3 & -2\\ 3 & 4 & -1\\ -2 & -1 & 3\end{bmatrix} \qquad B=\begin{bmatrix}3 & -4 & 3\\ -4 & 6 & 3\\ 3 & 3 & 1\end{bmatrix}$$

2）用 Jacobi 方法计算对称矩阵 $A=\begin{bmatrix}1.0 & 1.0 & 0.5\\ 1.0 & 1.0 & 0.25\\ 0.5 & 0.25 & 2.0\end{bmatrix}$ 所有特征值和特征向量