MATLAB数值计算代码

G. 编写

插入排序

```
function Y=INSERTION_SORT(X)
len=length(X);
for j=2:len
   key=X(j);
   i=j-1;
   while(i>0&&X(i)>key)
      X(i+1)=X(i);
      i=i-1;

   end

   X(i+1)=key;
end
Y=X;
```

冒泡排序

```
function Y=BUBBLESORT(X)
len=length(X);
for i=1:len-1
    for j=len:-1:(i+1)
        if X(j)<X(j-1)
            key=X(j);
            X(j)=X(j-1);
            X(j-1)=key;
        end
    end
end
Y=X;
```

快速排序

```
function Y=QUICKSORT(X,p,r)
len=length(X);
if nargin<3
    r=len;
end
if nargin<2

   p=1;
end
if p<r
    [q,X]=PARTITION(X,p,r);
    X=QUICKSORT(X,p,q-1);
    X=QUICKSORT(X,q+1,r);
end
Y=X;

function [q,Y]=PARTITION(X,p,r)
x=X(r);
i=p-1;
for j=p:(r-1)
    if X(j)<=x
        i=i+1;
        key=X(i);
        X(i)=X(j);
        X(j)=key;
    end
    key=X(i+1);
    X(i+1)=X(r);
    X(r)=key;
end
q=i+1;
Y=X;
```

归并排序

```
function Y=MERGE_SORT(X,p,r)
if nargin<3
r=length(X);
end
if nargin<2
 p=1;
end

if p<r

   q=floor((p+r)/2);
   X=MERGE_SORT(X,p,q);
   X=MERGE_SORT(X,q+1,r);
   X=MERGE(X,p,q,r);
end
Y=X;

function Y=MERGE(X,p,q,r)
n1=q-p+1;
n2=r-q;
L=zeros(1,n1+1);
R=zeros(1,n2+1);
for i=1:n1
    L(i)=X(p+i-1);
end
for i=1:n2
    R(i)=X(q+i);
end
L(n1+1)=inf;
R(n2+1)=inf;
i=1;
j=1;
for k=p:r
   if L(i)<=R(j)
       X(k)=L(i);
       i=i+1;
   else
       X(k)=R(j);
       j=j+1;
   end


end
Y=X;
```

秦九韶算法

```
function y=qin(A,x)
%A为系数向量，x为求值点
%系数由高阶向低阶排列
n=length(A);
y=A(1);

for i=2:n
 y=y.*x+A(i);
end
```

欧拉法求常微分方程数值解

```
function y=euler(f,x0,y0,h,x)
if x<x0
   h=-h;
end
while(abs(x-x0)>abs(h))
y0=y0+f(x0,y0)*h;
x0=x0+h;
end
h=x-x0;
y=y0+f(x0,y0)*h;
```

改进的欧拉法

```
function y=euler_mod(f,x0,y0,h,x)
if x<x0
   h=-h;
end
while(abs(x-x0)>abs(h))
y_p=y0+f(x0,y0)*h;
x0=x0+h;
y_c=y0+f(x0,y_p)*h;
y0=(y_p+y_c)/2;

end
h=x-x0;
y_p=y0+f(x0,y0)*h;
x0=x0+h;
y_c=y0+f(x0,y_p)*h;
y=(y_p+y_c)/2;
```

四阶经典龙格库塔方法

```
function [X,Y]=RK_4_class(f,x,y0,step)
%f为函数句柄，x=[x0,x1]为求解区间，y0为初值向量，step为步长
if nargin<4
    step=0.01;
end

num=round(abs(x(2)-x(1))/step);
h=(x(2)-x(1))/num;
X=linspace(x(1),x(2),num+1);
Y(:,1)=y0;
for i=1:num

K1=f(X(i),Y(:,i));
K2=f(X(i)+h/2,Y(:,i)+K1*h./2);
K3=f(X(i)+h/2,Y(:,i)+K2*h./2);
K4=f(X(i)+h,Y(:,i)+K3*h);
Y(:,i+1)=Y(:,i)+(K1+2*K2+2*K3+K4)*h./6;

end
```

勒让德多项式

```
function y=lege(x,n)
%Legendre polynomial
%
if n==0
    y=1;
else
    if n==1
        y=x;
    else
        y=((2*n-1)*x.*lege(x,n-1)-(n-1)*lege(x,n-2))./n;
    end
end
```

切比雪夫多项式

```
function y=cheb(x,n)
%Chebyshev polynomial
%Find the value of chebyshev polynomial at x
if n==0
    y=1;
else
    if n==1
        y=x;
    else
        y=2*x.*cheb(x,n-1)-cheb(x,n-2);
    end
end
```

理查德外推求微分

```
function df=diff_ric(f,x,h,m)
%微分的理查德外推方法
%f为求微分的函数，x为微分点，h为步长，m为外推步数
if m==0
    df=(f(x+h)-f(x-h))./(2*h);
else

df=(4^m*diff_ric(f,x,h/2,m-1)-diff_ric(f,x,h,m-1))./(4^m-1)
;
end
```

```
function S=inte_tra_rec(f,a,b,t)
%梯形公式的外推法
%f为被积函数，a为积分上限，b为积分下限，t为外推次数
h=b-a;
S=h*(f(a)+f(b))/2;
if t>0
for i=1:t
   h=h/2;
    S=S/2+h*sum(f(linspace(a+h,b-h,2^(i-1))));
end
end
```

自适应积分——辛普森方法

```
function y=inte_sim_ade(f,a,b,err)
if nargin<4
   err=0.000001;
end
delta=b-a;
S1=(f(a)+f(b)+4*f((a+b)/2))*delta/6;
delta=delta/2;
S2=(f(a)+4*(f(a+delta/2)+f(a+3*delta/2))+2*f(a+delta)+f(b))
*delta/6;
if abs(S1-S2)<err
   y=S2+(S2-S1)/15;
else

y=inte_sim_ade(f,a,(a+b)/2,err/2)+inte_sim_ade(f,(a+b)/2,b,
err/2);
end
```

自适应二重积分——辛普森方法

```
function y=inte_sim_ade2(f,a,b,c,d,err)
if nargin<6
   err=0.000001;
end

delta=d-c;
S1=(inte_sim_ade(f,c,a,b,err)+inte_sim_ade(f,d,a,b,err)+4*i
nte_sim_ade(f,(c+d)/2,a,b,err))*delta/6;
delta=delta/2;
S2=(inte_sim_ade(f,c,a,b,err)+4*(inte_sim_ade(f,c+delta/2,a
,b,err)+inte_sim_ade(f,c+3*delta/2,a,b,err))+2*inte_sim_ade
(f,c+delta,a,b,err)+inte_sim_ade(f,d,a,b,err))*delta/6;
if abs(S1-S2)<err
   y=S2+(S2-S1)/15;
else

y=inte_sim_ade2(f,a,(a+b)/2,c,(c+d)/2,err/4)+inte_sim_ade2(
f,(a+b)/2,b,c,(c+d)/2,err/4);

y=y+inte_sim_ade2(f,a,(a+b)/2,(c+d)/2,d,err/4)+inte_sim_ade
2(f,(a+b)/2,b,(c+d)/2,d,err/4);
end

function y=inte_sim_ade(f,y0,a,b,err)

delta=b-a;
S1=(f(a,y0)+f(b,y0)+4*f((a+b)/2,y0))*delta/6;
delta=delta/2;
S2=(f(a,y0)+4*(f(a+delta/2,y0)+f(a+3*delta/2,y0))+2*f(a+del
ta,y0)+f(b,y0))*delta/6;
if abs(S1-S2)<err
   y=S2+(S2-S1)/15;
else

y=inte_sim_ade(f,y0,a,(a+b)/2,err/2)+inte_sim_ade(f,y0,(a+b
)/2,b,err/2);
end
```

梯形公式

```
function S=inte_tra(f,a,b,N)
%Trapezoid Integration Method
% function f,lower bound a,upper bound b,step N (N=100 default)
if nargin<3
    N=100;
end
delta=(b-a)/N;
T1=sum(f(linspace(a,b,N+1)));
S=delta*(2*T1-f(a)-f(b))/2;
```

梯形公式外推法

```
function S=inte_tra_rec(f,a,b,t)
%梯形公式的外推法
%f为被积函数，a为积分上限，b为积分下限，t为外推次数
h=b-a;
S=h*(f(a)+f(b))/2;
if t>0
for i=1:t
   h=h/2;
    S=S/2+h*sum(f(linspace(a+h,b-h,2^(i-1))));
end
end
```

辛普森方法

```
function S=inte_sim(f,a,b,N)
%Simpton Integration Method
% function f,lower bound a,upper bound b,step N (N=100 default)
if nargin<3
    N=100;
end
delta=(b-a)/N;
T1=sum(f(linspace(a,b,N+1)));
T2=sum(f(linspace(a+delta/2,b-delta/2,N)));
S=delta*(2*T1+4*T2-f(a)-f(b))/6;
```

龙贝格积分

```
function S=inte_rom(f,a,b,m,k)
%龙贝格积分
%f为函数，a,b为积分上限限制，m为加速次数，k为二分次数
if m==0
    S=inte_tra_rec(f,a,b,k);
else

S=(4^m*inte_rom(f,a,b,m-1,k+1)-inte_rom(f,a,b,m-1,k))/(4^m-
1);
end

function S=inte_tra_rec(f,a,b,t)
h=b-a;
S=h*(f(a)+f(b))/2;
if t>0
for i=1:t
   h=h/2;
    S=S/2+h*sum(f(linspace(a+h,b-h,2^(i-1))));
end
end
```

反幂法求矩阵特征值

```
function [k,u]=inverse_pow(A,p,n)
%反幂法
[len,wide]=size(A);
u=ones(wide,1);
A=inv(A-p*diag(ones(wide,1)));
v=u;
k=0;
for i=1:n
v=A*u;
k=max(v);
u=v./k;
end
k=1/k+p;
```

QR分解

```
function [Q,R]=QR_divide(A)
[wide,len]=size(A);
y=A;
for i=1:wide
   for j=1:i-1
      y(:,i)=y(:,i)-A(:,i)'*y(:,j)/(y(:,j)'*y(:,j))*y(:,j);
   end
end

for i=1:wide
   y(:,i)=y(:,i)./sqrt(y(:,i)'*y(:,i));
end
Q=y;
R=Q^(-1)*A;
```

QR分解求特征值

```
function e=eigen(A,n)
%QR分解求特征值，n为迭代次数
if nargin<2
   n=100;

end
for i=1:n
 [Q,R]=QR_divide(A);
 A=R*Q;
end
e=diag(A);


function [Q,R]=QR_divide(A)
[wide,len]=size(A);
y=A;
for i=1:wide
   for j=1:i-1
      y(:,i)=y(:,i)-A(:,i)'*y(:,j)/(y(:,j)'*y(:,j))*y(:,j);
   end
end

for i=1:wide
   y(:,i)=y(:,i)./sqrt(y(:,i)'*y(:,i));
end
Q=y;
R=Q^(-1)*A;
```

LU分解

```
function [L,U]=lu_divide(A)
[wide,len]=size(A);
L=eye(wide,len);
U=zeros(wide,len);
U(1,:)=A(1,:);  %initialize
for i=2:wide
   L(i,1)=A(i,1)/U(1,1);
end

for k=2:wide
for j=k:wide
S1=0;
    for q=1:(k-1)
   S1=S1+L(k,q)*U(q,j);
    end
U(k,j)=A(k,j)-S1;
end
for i=(k+1):wide
S2=0;
    for q=1:(k-1)
   S2=S2+L(i,q)*U(q,k);
    end
L(i,k)=(A(i,k)-S2)/U(k,k);
end
end
```

多项式求根

```
function r=poly_roots(p)
n=length(p);
n=n-1;
   A = diag(ones(n-1,1),-1);
A(1,:) = -p(2:n+1)./p(1);
r=eigen(A);

function e=eigen(A,n)
if nargin<2
   n=1000;
end
for i=1:n
 [Q,R]=QR_divide(A);
 A=R*Q;
end
e=diag(A);

function [Q,R]=QR_divide(A)
[wide,len]=size(A);
y=A;
for i=1:wide
   for j=1:i-1
      y(:,i)=y(:,i)-A(:,i)'*y(:,j)/(y(:,j)'*y(:,j))*y(:,j);
   end
end
for i=1:wide
   y(:,i)=y(:,i)./sqrt(y(:,i)'*y(:,i));
end
Q=y;
R=Q^(-1)*A;
```

幂法求主特征值和主特征向量

```
function [k,u]=pow(A,n)
[len,wide]=size(A);
u=rand(wide,1);
v=u;
k=0;
for i=1:n
v=A*u;
k=max(v);
u=v./k;
end
```

线性方程组——LU分解法

```
function x=solve_linear(A,b)
[L,U]=lu_divide(A);
[wide,len]=size(A);
y=zeros(1,wide);
x=y;
y(1)=b(1);
for i=2:wide
    S=0;
    for j=1:(i-1)
        S=S+L(i,j)*y(j);
    end
    y(i)=b(i)-S;
end
x(wide)=y(wide)/U(wide,wide);
for i=(wide-1):-1:1
    S1=0;
    for j=(i+1):wide
S1=S1+U(i,j)*x(j); %
    end
    x(i)=(y(i)-S1)/U(i,i);
end
function [L,U]=lu_divide(A)
[wide,len]=size(A);
L=eye(wide,len);
U=zeros(wide,len);
U(1,:)=A(1,:);  %initialize
for i=2:wide
   L(i,1)=A(i,1)/U(1,1);
end
for k=2:wide
for j=k:wide
S1=0;
    for q=1:(k-1)
   S1=S1+L(k,q)*U(q,j);
    end
U(k,j)=A(k,j)-S1;
end
for i=(k+1):wide
S2=0;
    for q=1:(k-1)
   S2=S2+L(i,q)*U(q,k);
    end
L(i,k)=(A(i,k)-S2)/U(k,k);
end
end
```

线性方程组——雅克比迭代

```
function x=Jacobi(A,b,times,x0)
%线性方程组——雅克比迭代
[wide,length]=size(A);
x=zeros(length,1);
if nargin<4
    x0=zeros(length,1);
end
for i=1:times

   for j=1:length
       x(j)=(b(j)+A(j,j)*x0(j)-sum(A(j,:)*x0))/A(j,j);
   end
   x0=x;


end
```

布朗运动

```
function [X,Y]=Brown(n,h,sigma)
%模拟布朗运动
%h为时间间隔
%n为离散点个数
%[X,Y]为对应每一个离散点的坐标向量
%sigma为布朗运动的方差，方差越大则变化越剧烈
X_1=normrnd(0,1,n+1,1);
Y_1=normrnd(0,1,n+1,1);
X(1)=0;
Y(1)=0;
for k=2:n+1
   X(k)=X(k-1)+sigma.*h.*X_1(k);
    Y(k)=Y(k-1)+sigma.*h.*Y_1(k);
end
```

秩统计量

```
function y=zhi(x)
%求秩统计量
len=length(x);
x0=sort(x);
y=zeros(1,len);
for i=1:len
    k=0;
   for j=1:len
      if(x0(i)==x(j))
          y(j)=i;
          k=k+1;
      end
   end
   if(k>1)
      for j=1:len
          if(x0(i)==x(j))
            y(j)=(2*i-k+1)/2;

          end
      end

   end
end
```

层次分析

```
function [X,CI]=level(A)
%层次分析
[wide,length]=size(A);
T=zeros(1,wide);
B=eig(A);
for i=1:wide
T(i)=isreal(B(i))*B(i);
end
L=max(T);
k=find(L);

[V,D]=eig(A);
X=V(:,k);
X=X./sum(X);
CI=(L-wide)/(wide-1);
```

因子分析

```
function [A,D,F]=factor_analysis(X,contribution)
%因子分析，标准化主成分法
%X为样本矩阵，contribution为所需的累计贡献率
%A为因子载荷矩阵，D为特殊因子方差阵，F为因子得分

[wide,len]=size(X);
[C,R]=covar(X);
[Ve,Va]=eig(R);
Ve=fliplr(Ve);
Va=fliplr(Va);
Va=rot90(Va,3);
i=0;s=0;
while(s/sum(sum(Va))<contribution)
    i=i+1;
    s=s+Va(i,i);
end

Ve=Ve(:,1:i);
Va=Va(1:i,1:i);

A=Ve*sqrt(Va);
D=R-A*A';

F=A'*R\X;

function [C,R]=covar(x)
[wide,len]=size(x);
aver=sum(x)./wide;
C=zeros(len,len);
R=C;
for i=1:len
   for j=1:len
      C(i,j)=(x(:,i)-aver(i))'* (x(:,j)-aver(j));
   end
end
for i=1:len
   for j=1:len
      R(i,j)=C(i,j)/sqrt(C(i,i)*C(j,j));
   end
end
C=C./(wide-1);
```

Floyd算法

```
function [D,W]=floyd(X)
%D为距离，W为路径
[wide,len]=size(X);
if wide~=len
   disp('dimension error!');
end
D=X;
W=zeros(wide,wide,wide);
for i=1:wide
  for j=1:wide
   W(i,j,1)=i;
   W(i,j,2)=j;
    end

end
for k=1:wide
    for i=1:wide
        for j=1:wide
           if D(i,j)>D(i,k)+D(k,j)
            W(i,j,:)=copy(W(i,k,:),W(k,j,:));
          D(i,j)=D(i,k)+D(k,j);
      end
        end
    end
end


function z=copy(x,y)
len=length(x);
mark=0;

for i=1:len

    if x(i)==0
    mark=i;
break;
end
end

for i=mark:len
    x(i-1)=y(i-mark+1);
 end
z=x;
```

Prim算法

```
function TE=prim(X)
[wide,len]=size(X);
if wide~=len
   disp('dimension error!');
end
TE=zeros(wide,len);
U=zeros(1,wide);
U(1)=1;
while(~all(U))
   P=(1:wide)-U;
   mark=zeros(1,3);
   mark(3)=inf;
    for i=1:wide
      if U(i)~=0
        for j=1:wide
           if P(j)~=0
               if X(i,j)<mark(3)
                  mark(1)=i;mark(2)=j;mark(3)=X(i,j);
               end
           end
        end

      end
   end
 TE(mark(1),mark(2))=X(mark(1),mark(2));
 U(mark(2))=mark(2);

end
```