2 - Soluciones numéricas de sistemas lineales:

**m=n (sistemas cuadrados):**

**Resolución por SVD:**

>>

function x=resuelve\_svd(A,b)

[U,S,V]=svd(A);

x=V\*diag(1./diag(S))\*U'\*b;

>>

**Descomposición por LUP:**

**Resuelve\_L:**

>>

function w=resuelve\_L(L,d)

w=zeros(size(d));

w(1,1)=d(1)/L(1,1);

for k=2:max(size(L))

w(k,1)=(d(k)-L(k,1:k-1)\*w(1:k-1))/L(k,k);

end

>>

**Resuelve\_U:**

>>

function x=resuelve\_U(U,w)

L=fliplr(flipud(U));

wud=flipud(w);

x=flipud(resuelve\_L(L,wud));

>>

**Resuelve\_LU:**

>>

function x=resuelve\_lu(A,b)

[L,U,P]=lu(A);

w=resuelve\_L(L,P\*b);

x=resuelve\_U(U,w);

>>

**Resolución por QR:**

>>

function x=resuelve\_qr(A,b)

[Q,R,P]=qr(A);

w=resuelve\_U(R,Q'\*b);

x=P\*w;

>>

**Resolución por Cholesky:**

>>

function x=resuelve\_chol(A,b)

R=chol(A);

x=resuelve\_U(R,resuelve\_L(R',b));

>>

**m>n (sistemas no cuadrados y sobredeterminados):**

**Reducir rango:**

>>

function kA=reduce\_rango(A,k)

[U,S,V]=svd(A);

kA=U(:,1:k)\*S(1:k,1:k)\*(V(:,1:k))';

>>

**Padefit:**

>>

function [P,Q]=padefit(x,y,n,m)

A=[];

for k=n:-1:0

A=[A x.^k];

end

for k=m:-1:1

A=[A -y.\*(x.^k)];

end

param=pinv(A)\*y;

P=param(1:n+1);

Q=param(n+2:end);

>>

**Padeval:**

>>

function y=padeval(P,Q,x)

y=polyval(P,x)./polyval([Q;1],x);

>>

**Busca grados Padé:**

>>

function [n,m]=busca\_grados\_pade(x,y,err\_rel\_propuesto)

n=0;

m=1;

gt=n+m;

[P,Q]=padefit(x,y,n,m);

err=norm(y-padeval(P,Q,x))/norm(y);

while err>=err\_rel\_propuesto

gt=gt+1;

for k=gt:-1:1

m=k;

n=gt-k;

[P,Q]=padefit(x,y,n,m);

err=norm(y-padeval(P,Q,x))/norm(y);

if err<err\_rel\_propuesto

return;

end

end

end

>>

**Integración:**

>>

function I=integra(f,a,b,n,m,k)

if nargin==3

n=20\*ceil(abs(b-a));

m=4;

k=3;

elseif nargin==4

m=4;

k=3;

elseif nargin==5

k=min([m-1,3]);

end

if a==b

I=0;

return

end

vab=linspace(a,b,n+1);

I=0;

for h=1:n

xsub=linspace(vab(h),vab(h+1),m);

ysub=f(xsub);

coefpk=polyfit(xsub,ysub,k);

coefprim=[coefpk./(k+1:-1:1) 0];

I=I+polyval(coefprim,vab(h+1))-polyval(coefprim,vab(h));

End

>>

3 - Soluciones numéricas de sistemas no lineales:

**Método de bisección (una variable):**

**Bisecante:**

>>

function sol\_ap=biseca(h,a,b,n)

if h(a)\*h(b)>=0

error('h(a) y h(b) deben tener signos opuestos');

end

m=(a+b)/2;

for k=1:n

if h(m)==0

sol\_ap=m;

return;

elseif h(a)\*h(m)>0

a=m;

else

b=m;

end

m=(a+b)/2;

end

sol\_ap=m;

>>

**Método de Newton (varias variables):**

**Método de Newton-Raphson:**

>>

function sol=N\_R(H,DH,V1,N)

sol=V1;

for k=1:N

sol=sol-inv(DH(sol))\*H(sol);

end

>>

**Secante:**

>>

function x=secante(h,x0,x1,N)

x=x1;

xant=x0;

for k=1:N

temp=x-h(x).\*(xant-x)./(h(xant)-h(x));

xant=x;

x=temp;

end

>>

**Aproximación jacobiana (derivación):**

>>

function J=JACOB\_APROX(H,X,epsi)

if nargin==2

epsi=sqrt(eps(1+norm(X)));

end

I=eye(max(size(X)));

for k=1:max(size(X))

J(:,k)=(H(X+epsi\*I(:,k))-H(X-epsi\*I(:,k)))/(2\*epsi);

End

>>

**Método de Newton-Raphson multivariado:**

>>

function sol=NR\_MULTI(H,x0,n,epsi)

if nargin==3

epsi=sqrt(eps(1+norm(x0)));

end

sol=x0;

for k=1:n

sol=sol-inv(JACOB\_APROX(H,sol,epsi))\*H(sol);

end

>>

4 - Soluciones numéricas de ecuaciones diferenciales ordinarias:

**Método de Euler (1, nivel de precisión más bajo):**

>>

function [t,X]=euler(F,t0,X0,T,h)

pasos=ceil(abs(T-t0)/h);

t=linspace(t0,T,pasos+1);

X=zeros(length(X0),length(t));

X(:,1)=X0;

hutil=t(2)-t(1);

for i=2:pasos+1

X(:,i)=X(:,i-1)+F(t(i-1),X(:,i-1))\*hutil;

End

>>

**Método H3 (5):**

>>

function [t,X]=H3(F,t0,X0,T,h)

pasos=ceil(abs(T-t0)/h);

t=linspace(t0,T,pasos+1);

X=zeros(length(X0),length(t));

X(:,1)=X0;

hutil=t(2)-t(1);

for i=2:pasos+1

s1=F(t(i-1),X(:,i-1));

s2=F(t(i-1)+hutil/3,X(:,i-1)+s1\*hutil/3);

s3=F(t(i-1)+hutil\*(2/3),X(:,i-1)+s2\*hutil\*(2/3));

X(:,i)=X(:,i-1)+(s1\*(1/4)+s3\*(3/4))\*hutil;

End

>>

**Método de Múltiples parámetros (3):**

>>

function [t,X]=MP(F,t0,X0,T,h)

pasos=ceil(abs(T-t0)/h);

t=linspace(t0,T,pasos+1);

X=zeros(length(X0),length(t));

X(:,1)=X0;

hutil=t(2)-t(1);

for i=2:pasos+1

s1=F(t(i-1),X(:,i-1));

s2=F(t(i-1)+0.5\*hutil,X(:,i-1)+s1\*0.5\*hutil);

X(:,i)=X(:,i-1)+s2\*hutil;

End

>>

**Métodos de Runge-Kutta:**

**RK2 (2):**

>>

function [t,X]=RK2(F,t0,X0,T,h)

pasos=ceil(abs(T-t0)/h);

t=linspace(t0,T,pasos+1);

X=zeros(length(X0),length(t));

X(:,1)=X0;

hutil=t(2)-t(1);

for i=2:pasos+1

s1=F(t(i-1),X(:,i-1));

s2=F(t(i),X(:,i-1)+s1\*hutil);

X(:,i)=X(:,i-1)+((s1+s2)/2)\*hutil;

End

>>

**RK3 (4):**

>>

function [t,X]=RK3(F,t0,X0,T,h)

pasos=ceil(abs(T-t0)/h);

t=linspace(t0,T,pasos+1);

X=zeros(length(X0),length(t));

X(:,1)=X0;

hutil=t(2)-t(1);

for i=2:pasos+1

s1=F(t(i-1),X(:,i-1));

s2=F(t(i-1)+hutil/2,X(:,i-1)+s1\*(hutil/2));

s3=F(t(i-1)+hutil,X(:,i-1)+(2\*s2-s1)\*hutil);

X(:,i)=X(:,i-1)+((s1+4\*s2+s3)/6)\*hutil;

End

>>

**RK4 (6):**

>>

function [t,X]=RK4(F,t0,X0,T,h)

pasos=ceil(abs(T-t0)/h);

t=linspace(t0,T,pasos+1);

X=zeros(length(X0),length(t));

X(:,1)=X0;

hutil=t(2)-t(1);

for i=2:pasos+1

s1=F(t(i-1),X(:,i-1));

s2=F(t(i-1)+hutil/2,X(:,i-1)+s1\*hutil/2);

s3=F(t(i-1)+hutil/2,X(:,i-1)+s2\*hutil/2);

s4=F(t(i),X(:,i-1)+s3\*hutil);

X(:,i)=X(:,i-1)+((s1+2\*s2+2\*s3+s4)/6)\*hutil;

End

>>