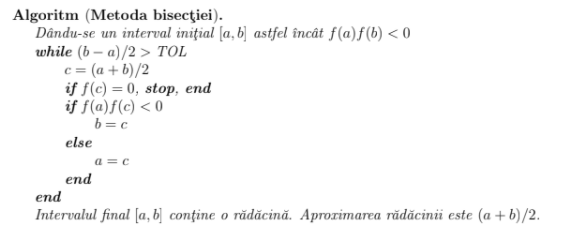
Metoda BISECTIEI



function xc=bisect(f,a,b,tol)

fa=f(a);

fb=f(b);

if sign(fa)\*sign(fb) >= 0

error(’f(a)f(b)<0 not satisfied!’)

end

while (b-a)/2>tol

c=(a+b)/2;

fc=f(c);

if fc == 0

break;

end

if sign(fc)\*sign(fa)<0

b=c;

fb=fc;

else

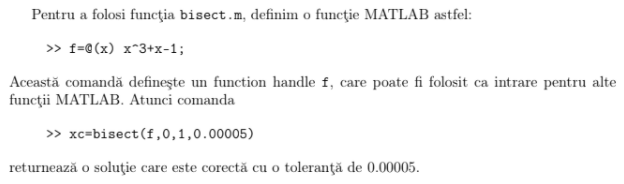
a=c;

fa=fc;

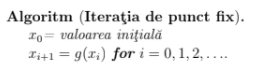
end

end

xc=(a+b)/2;



Iteratia de punct fix (IPF)



function xc=fpi(g,x0,k)

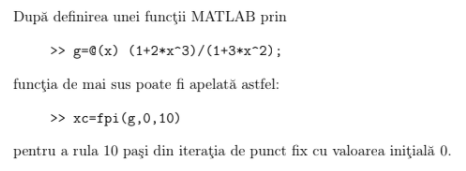
x(1)=x0;

for i=1:k

x(i+1)=g(x(i));

end

xc=x(k+1);



Metoda lui NEWTON



function xc=newton(f,df,x0,k)

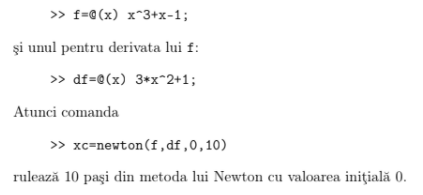
x(1)=x0;

for i=1:k

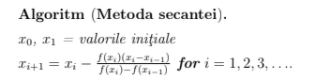
x(i+1)=x(i)-f(x(i))/df(x(i));

end

xc=x(k+1);



Metoda SECANTEI



function xc = secant(f,x0,x1,k)

x(1) = x0;

x(2) = x1;

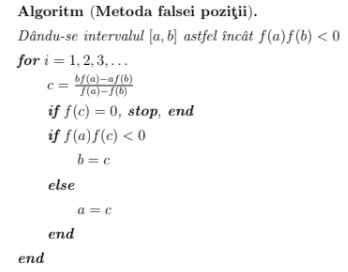
for i = 2:k

x(i+1) = x(i) - (f(x(i))\*(x(i)-x(i-1)))/(f(x(i))-f(x(i-1)));

end

xc = x(k+1);

Metoda FALSEI POZITII



function xc = mfp(f,a,b,k)

fa = f(a);

fb = f(b);

if sign(fa)\*sign(fb) >= 0

error('f(a)f(b)<0 not satisfied')

end

for i = 1:k

c = (b\*fa-a\*fb)/(fa-fb);

fc = f(c);

if fc == 0

break;

end

if sign(fa)\*sign(fc) < 0

b = c;

fb = fc;

else

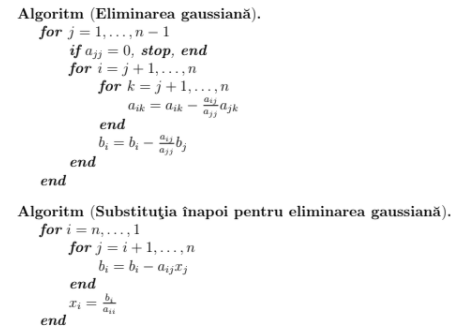
a = c;

fa = fc;

end

end

xc = c

Eliminarea Gaussiana 

function [A,x]=gauss(A,b)

n=length(A);

for j=1:n-1

if abs(A(j,j)) == 0

error(’zero pivot encountered!’);

end

for i=j+1:n

mult=A(i,j)/A(j,j);

for k=j:n

A(i,k)=A(i,k)-mult\*A(j,k);

end

b(i)=b(i)-mult\*b(j);

end

end

for i=n:-1:1

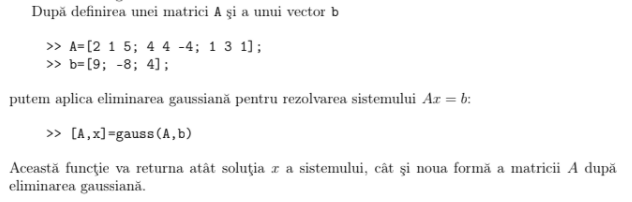
for j=i+1:n

b(i)=b(i)-A(i,j)\*x(j);

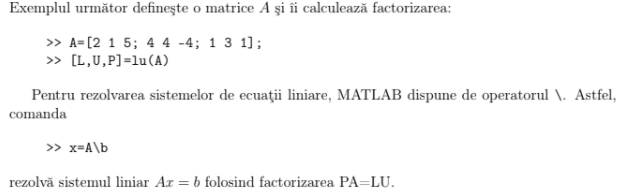
end

x(i)=b(i)/A(i,i);

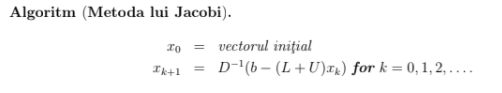
end



Factorizarea PA=LU (PALU)



Metoda lui JACOBI



function x=jacobi(A,b,x0,k)

D=diag(diag(A));

L=tril(A)-D;

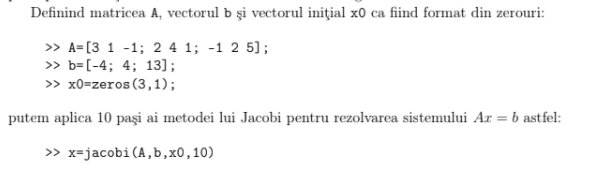
U=triu(A)-D;

x=x0;

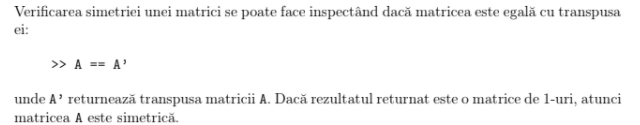
for j=1:k

x = inv(D)\*(b-(L+U)\*x);

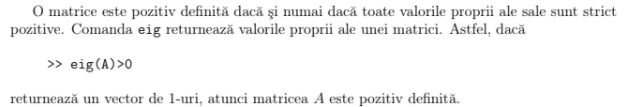
end



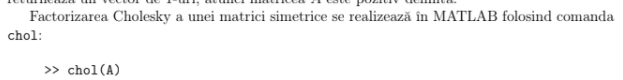
Matrice SIMETRICA



Matrice POZITIV DEFINITA



Factorizarea CHOLESKY (CHOL)



function R = cholesky(A)

n = length(A);

for k=1:n

if A(k, k) < 0

error('pivot < 0');

end

R(k, k) = sqrt( A(k, k) );

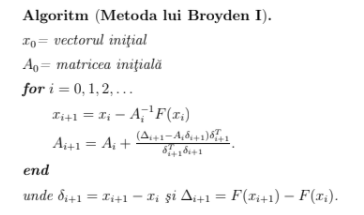
u = ( A(k, (k+1):n ) / R(k, k) )';

R(k, (k+1):n) = u;

A((k+1):n, (k+1):n) = A((k+1):n, (k+1):n) - u.\*u;

end

Metoda lui BROYDEN 1



function x=broyden1(F,x0,k)

n=length(x0);

A=eye(n,n);

for i=1:k

x=x0-inv(A)\*F(x0);

delta=x-x0;

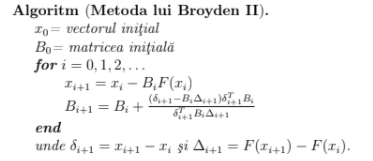
Delta=F(x)-F(x0);

A=A+(Delta-A\*delta)\*delta’/(delta’\*delta);

x0=x;

end

Metoda lui BROYDEN 2



function x=broyden2(F,x0,k)

n=length(x0);

B=eye(n,n);

for i=1:k

x=x0-B\*F(x0);

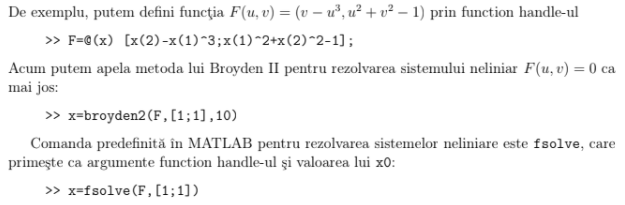
delta=x-x0;

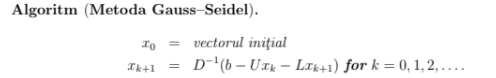
Delta=F(x)-F(x0);

B=B+(delta-B\*Delta)\*delta’\*B/(delta’\*B\*Delta);

x0=x;

end



Metoda GAUSS-SEIDEL 

function x=gauss\_seidel(A,b,x0,k)

D = diag(diag(A));

L = tril(A)-D;

U = triu(A)-D;

x = x0;

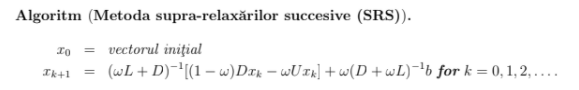
for i = 1:k

x = inv(L+D)\*(b-U\*x);

x0 = x;

end

Metoda SUPRA-RELAXARILOR SUCCESIVE (SRS)



function x=sor(A,b,x0,omega,k)

D=diag(diag(A));

L=tril(A)-D;

U=triu(A)-D;

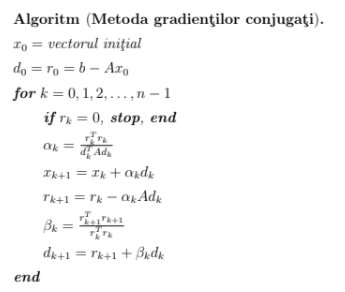
for i = 1:k

x = inv(omega\*L+D)\*[(1-omega)\*D\*x0-omega\*U\*x0]+omega\*(inv(D+omega\*L))\*b;

x0 = x;

end

Metoda GRADIENTILOR CONJUGATI (CONJGRAD)



function x=conjgrad(A,b,x0,k)

d=b-A\*x0;

r=b-A\*x0;

x=x0;

for j=1:k

if r==0

break;

end

alpha=(r’\*r)/(d’\*A\*d);

x=x+alpha\*d;

r0=r;

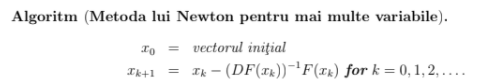
r=r-alpha\*A\*d;

beta=(r’\*r)/(r0’\*r0);

d=r+beta\*d;

end

Metoda lui NEWTON pentru MAI MULTE VARIABILE



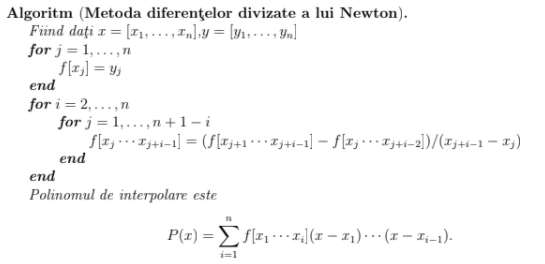
function x = newton(F, DF, x0, k)

for i= 1:k

x = x0 - inv(DF(x0)) \* F(x0);

x0 = x;

end

Metoda DIFERENTELOR DIVIZATE a lui NEWTON

function c=newtondd(x,y,n)

for j=1:n

v(j,1)=y(j);

end

for i=2:n

for j=1:n+1-i

v(j,i)=(v(j+1,i-1)-v(j,i-1))/(x(j+i-1)-x(j));

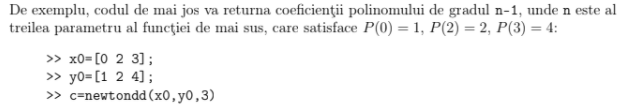
end

end

for i=1:n

c(i)=v(1,i);

end



NESTED

function y=nested(d,c,x,b)

if nargin<4

b=zeros(d,1);

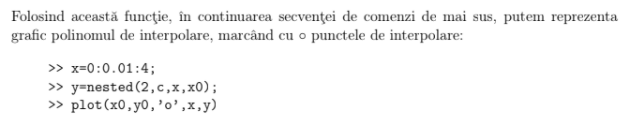
end

y=c(d+1);

for i=d:-1:1

y = y.\*(x-b(i))+c(i);

end



Aproximarea funcţiei sinus folosind interpolarea Lagrange

function y=sin1(x)

b=pi\*(0:3)/6;

yb=sin(b);

c=newtondd(b,yb,4);

s=1;

x1=mod(x,2\*pi);

if x1>pi

x1 = 2\*pi-x1;

s = -1;

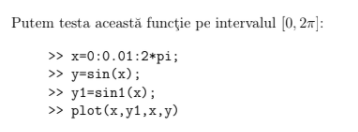
end

if x1 > pi/2

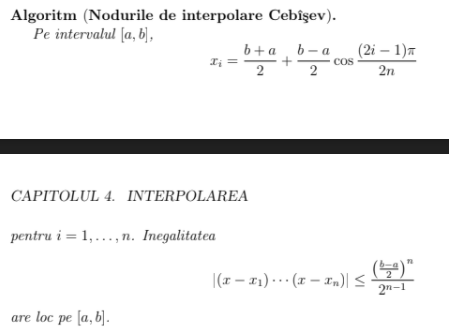
x1 = pi-x1;

end

y = s\*nested(3,c,x1,b);



Interpolarea CEBISEV



function y=sin2(x)

n=10;

b=pi/4+(pi/4)\*cos((1:2:2\*n-1)\*pi/(2\*n));

yb=sin(b);

c=newtondd(b,yb,n);

s=1;

x1=mod(x,2\*pi);

if x1>pi

x1 = 2\*pi-x1;

s = -1;

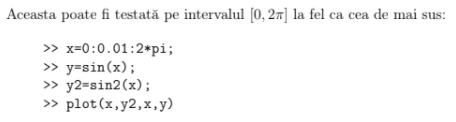
end

if x1 > pi/2

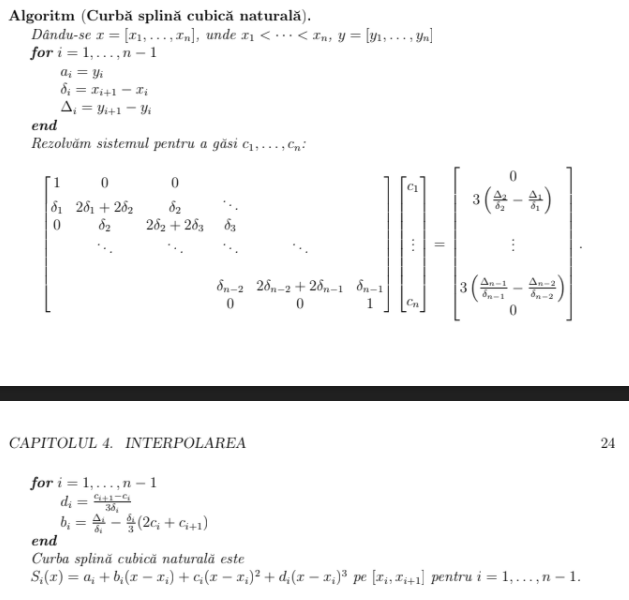
x1 = pi-x1;

end

y = s\*nested(n-1,c,x1,b);



Curbe SPLINE CUBICE



function coeff=splinecoeff(x,y)

n=length(x);

A=zeros(n,n);

r=zeros(n,1);

for i=1:n-1

dx(i)=x(i+1)-x(i);

dy(i)=y(i+1)-y(i);

end

for i=2:n-1

A(i,i-1:i+1)=[dx(i-1) 2\*(dx(i-1)+dx(i)) dx(i)];

r(i)=3\*(dy(i)/dx(i) - dy(i-1)/dx(i-1));

end

A(1,1) = 1;

A(n,n) = 1;

coeff=zeros(n,3);

coeff(:,2)=A\r;

for i=1:n-1

coeff(i,3)=(coeff(i+1,2)-coeff(i,2))/(3\*dx(i));

coeff(i,1)=dy(i)/dx(i)-dx(i)\*(2\*coeff(i,2)+coeff(i+1,2))/3;

end

coeff=coeff(1:n-1,1:3);



function splineplot(x,y,k)

n=length(x);

coeff=splinecoeff(x,y);

x1=[];

y1=[];

for i=1:n-1

xs=linspace(x(i),x(i+1),k+1);

dx=xs-x(i);

ys=coeff(i,3)\*dx;

ys=(ys+coeff(i,2)).\*dx;

ys=(ys+coeff(i,1)).\*dx+y(i);

x1=[x1;xs(1:k)’];

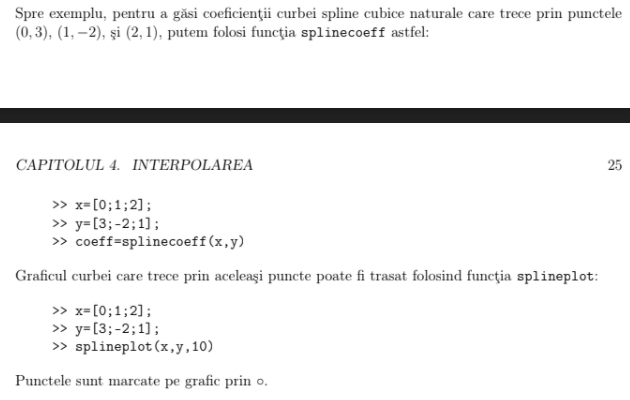
y1=[y1;ys(1:k)’];

end

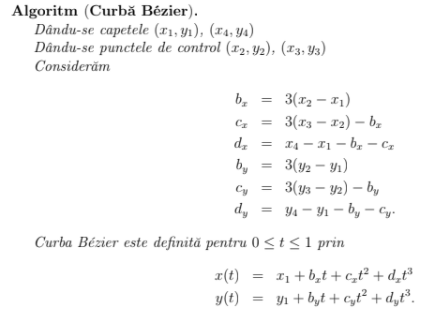
x1=[x1;x(end)];

y1=[y1;y(end)];

plot(x,y,’o’,x1,y1)



Curbe BEZIER



function coeff=beziercoeff(x,y)

bx=3\*(x(2)-x(1));

by=3\*(y(2)-y(1));

cx=3\*(x(3)-x(2))-bx;

cy=3\*(y(3)-y(2))-by;

dx=x(4)-x(1)-bx-cx;

dy=y(4)-y(1)-by-cy;

coeff=zeros(2,4);

coeff(1,:)=[x(1),bx,cx,dx];

coeff(2,:)=[y(1),by,cy,dy];



function bezierplot(x,y)

hold on;

t=0:0.02:1;

plot([x(1) x(2)],[y(1) y(2)],’r:’,x(2),y(2),’rs’);

plot([x(3) x(4)],[y(3) y(4)],’r:’,x(3),y(3),’rs’);

plot(x(1),y(1),’bo’,x(4),y(4),’bo’);

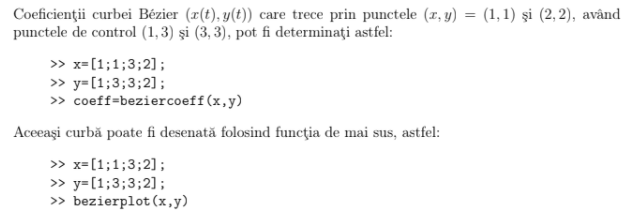
coeff=beziercoeff(x,y);

xp=coeff(1,1)+t.\*(coeff(1,2)+t.\*(coeff(1,3)+t\*coeff(1,4)));

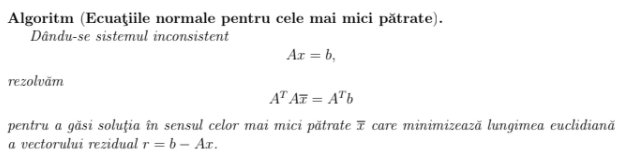
yp=coeff(2,1)+t.\*(coeff(2,2)+t.\*(coeff(2,3)+t\*coeff(2,4)));

plot(xp,yp)

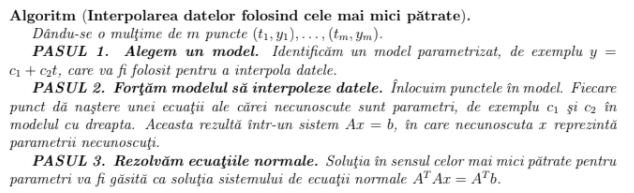
hold off;

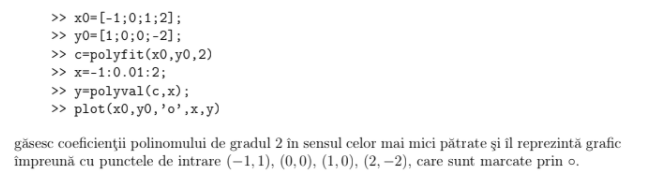


Ecuatiile NORMALE pentru CELE MAI MICI PATRATE

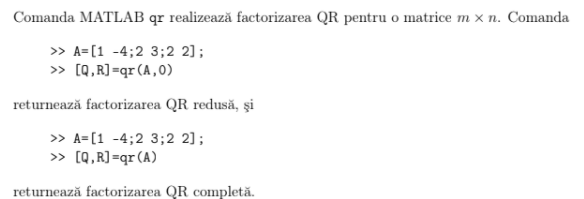


Interpolarea DATELOR folosind cele mai mici patrate





Factorizarea QR



Cele mai MICI PATRATE folosind factorizarea QR

