sol=bisect(f, a, b, 0.0001)
bisect afla x a.i f(x)=0, a<=x<=b:

```
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;
```

sol=secant(f, x0, x1,k)
x0 si x1 sunt de fapt intervalul [a b]
calculeaza x a.i f(x)=0, k da precizia

```
function xc=secant(f,x0,x1,k)
   for i = 2:k
      if f(x1) - f(x0) == 0, break, end
      xc = x1 - (f(x1)*(x1-x0))/(f(x1) -
f(x0));
      x0 = x1;
      x1 = xc;
   end
```

sol=mfp(f, a, b, k)
mfp calculeaza x a.i f(x)=0, k este precizia

```
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
     if f(a) - f(b) == 0, break, end
     xc = (b*f(a)-a*f(b))/(f(a) - f(b));
     if f(xc) == 0, break, end
     if f(a) * f(b) < 0
        b = xc;
     else
        a = xc;
     end
   end
```

sol=newton(f, df, x0, k)
calculeaza x a.i f(x)=0, k da precizia
x0 spune in jurul carui pct cauta radacina

```
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);
```

sol=fpi(f, x0, k)
fpi face f(f(f(....f(x0))) de k ori

```
function xc=fpi(f, x0, k)
for i = 1:k
   xc = f(x0);
   x0 = xc;
end
```

fct predefinite:
fsolve(f, [a b])
returneaza x a.i. f(x)=0, a<x<b

[A,x]=gauss(A,b)
A este matricea coef, iar b este vector coloana cu val ecuatiilor

```
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
```

x=jacobi(A,b,x0,k)
A matrice, b vector coloana
x0=[0;0...;0]

```
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
end
```

FACTORIZARE LU

```
A=[2 1 5; 4 4 -4; 1 3 1];
[L,U,P]=lu(A)
```

Matrice simetrica<=>A==A'
Matrice pozitiv definita<=>eig(A)>0 este vector de 1

R=cholesky(A)
Sau R=chol(A) (chol e predefinit)

```
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
```

ai la lab5 exemplu

```
function x = gauss_newton(r,Dr,x0,n)
x= x0;
for k = 1:n
   A = Dr(x);
   v = (A'*A)\(-1*A'*r(x));
   x = x + v;
end
```

$F(u, v) = (v u^3, u^2 + v^2\ 1)$

```
F=@(x) [x(2)-x(1)^3;x(1)^2+x(2)^2-1];
x=broyden2(F,[1;1],10)
```

```
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
```

la fel ca jacobi

```
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);
end
```

ca jacobi, omega se da in enunt

```
function x=sor(A,b,x0,omega,k)
   [L,U,D]=lu(A);
   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
end
```

Ca jacobi

```
function x=conjgrad(A,b,x0,k)
d0 = b - Ax0;
r0 = d0;
for i=1:k
  if r0 == 0, stop, end
  a = (r0.' * r0) / (d0.' * A * d0);
  x = x0 + a * d0;
  r = r0 - a * A * d0;
  B = (r.' * r) / (r0.' * r0);
  d0 = r + B * d0;
  r0 = r;
end
```

ca broyden1

```
function x=broyden1(F,x0,k)
   n=length(x0);
   A=eye(n, n);
   for i=1:k
      x = x0 - inv(A) * F(x0);
      s = x - x0;
      d = F(x) - F(x0);
      A = A + ((d - A * s) * s.') / (s.' * s);
      x0 = x;
   end
end
```

fct predefinite: fsolve(F,[a b])
ca broyden1/2

x0=[1 2 3] si y0=[4 6 3]
c=newtondd(x0,y0,3)

```
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
end
```

dupa ce faci c=newtondd, continua coloana urmatoare

x-0:0.01:4 (sau ce interval ai tu)
y=nested(2, c, x, x0)
plot(x0, y0,'o', x, y)

```
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 lui sinus cu lagrange
x=0:0.01:2*pi;
y=sin(x);
y1=sin1(x);
plot(x,y1,x,y)

```
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);
```

sinus aprox prin cebisev, ca mai sus

```
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);
```

cebisev

```
function x = cebisev(a, b, k)
   for i = 1:k
      x(i)=(b+a)/2+(b-a)/2*cos((2*i-
1)*pi/(2*k));
   end
end
```

Fct care trece prin punctele date:
x=[0;1;2] si y=[3;1;5]
coeff=splinecoeff(x,y)
splineplot(x,y,10) (si grafic idk dc k=10)

```
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)
```

similar, avem curbe bezier(ca mai sus)
avem pct p1,p3,...pn, dar p1 si pn sunt „puncte", restul sunt pct de control

```
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;
```

Cele mai mici patrate

```
x0=[-1;0;1;2];
y0=[1;0;0;-2];
c=polyfit(x0,y0,2)
x=-1:0.01:2;
y=polyval(c,x);
plot(x0,y0,'o',x,y)
```

factorizare QR

```
x=(2+(0:10)/5)';
y=1+x+x.^2+x.^3+x.^4+x.^5+x.^6+x.^7;
A=[x.^0 x x.^2 x.^3 x.^4 x.^5 x.^6 x.^7];
[Q,R]=qr(A);
b=Q'*y;
c=R(1:8,1:8)\b(1:8)
```

Găsiți cea mai bună dreaptă care aproximează următorul set de puncte, şi calculați REMP-ul:

```
x0=[-3;-1;0;1;3];
y0=[3;2;1;-1;-4];
c = polyfit(x0,y0,1)
y0_interp = polyval(c,x0)
error = (y0 - y0_interp)
REMP =
norm(error)/sqrt(length(error))
x=[-3:0.01:3];
y=polyval(c,x);
plot(x0,y0,'o',x,y)
```

gauss-newton

```
% Ecuatia cercului:
% (x-a)^2 + (y-b)^2 = R^2
r=@(x) [norm(x-[0;1])-1;norm(x-[1;1])-
1;norm(x-[0;-1])-1];
Dr=@(x) [(x-[0;1])/norm(x-[0;1]),(x-
[1;1])/norm(x-[1;1]),(x-[0;-1])/norm(x-
[0;-1])]';
```

x = gauss_newton(r,Dr,[0;0],20)

```
function x = gauss_newton(r,Dr,x0,n)
x=x0;
for k=1:n
A=Dr(x);
v=-(A'*A)\(A'*r(x));
x=x+v
end
```

se apeleaza ca gauss-newton
lm este levenberg-marquardt

```
function x = lm(r,Dr,lambda,x0,n)
x=x0;
for k=1:n
A=Dr(x);
v=-
(A'*A+lambda*diag(diag(A'*A)))\(A'*r(
x));
x=x+v;
end
```

se apeleaza ca qr(A)

```
function [Q,R] = gram_schmidt(A)
[m,n]=size(A);
Q=zeros(m, n);
for j=1:n
   y=A(:,j);
   for i=1:j-1
      R(i,j)=Q(:,i)'*A(:,j);
      y=y-R(i,j)*Q(:,i);
   end
   R(j,j)=norm(y);
   Q(:j(=y/R(j,j);
End
```

Qr scris de mana daca cere(qr exista predefinit)

```
function [Q,R]=qr1(A)
[~,n]=size(A);
for j = 1:n
y = A(:,j);
for i = 1:j-1
R(i,j) = Q(:,i)'*A(:,j);
y = y - R(i,j)*Q(:,i);
end
R(j,j) = norm(y);
Q(:,j) = y/R(j,j);
end
```