801.
```
function p=alfa(x)
r1=x:x+10;
M=eye(9);
N=ones(9,2)*x;
rn=r1.^2;
m=[r1;M,N;rn];
[p,d]=eig(m'*m);
```

La matrice **p** esiste in quanto la matrice $m^T m$ è una matrice simmetrica.

802.
```
function x=alfa(n)
v=n:-1:1;
for id=1:n
  m(id,:)=v;
end
p1=[eye(n)*n,m;ones(n)*n,zeros(n)];
p=eye(2*n)+p1;
x=p\(1:2*n)';
```

803.
```
function b=alfa(x,y)
r1=[2,x+2:2:x+20];
c1=(x+2:2:x+20)';
M=diag(y:-1:y-9);
N=M+ones(10)-eye(10);
a=[r1;c1,N];
if det(a)~=0
  b=inv(a);
else
  b=zeros(size(a));
end
```

804.
```
function m=alfa(x)
[r,c]=size(x);
if r~=1
  error('x non matrice riga')
end
a=[x,0;eye(c),x'];
s=eig(a+a');
m=diag(s);
```

805.
```
function x=alfa(v)
n=length(v);
w=(v(2:n))';
b=[v;w, diag(w)];
b(:,n)=ones(n,1)*v(n);
if det(b)~=0
  x=b\ones(n,1);
else
  x=zeros(n,1);
end
```

806.
```
function a=alfa(v)
[r,c]=size(v);
if r>1 & c==1
  v=v';
end
if r>1 & c>1
  v=v(1,:);
  warning('v ha più di una riga, verrà usata solo la prima')
end
n=length(v);
di=n:-1:1;
di1=1./di;
d=diag(di1);
a=v*v'+d;
```

807.
```
function b=alfa(a)
[r,c]=size(a);
if r==c
  b=[a,a';diag(1:r), zeros(r)];
else
  b=[a,eye(r);diag(1:c),a'];
end
```

808.
```
function b=alfa(v)
[r,c]=size(v);
if r>1 & c>1
  b=eig(v*v');
  return
end
if r>1 & c==1
  v=v';
end
n=length(v);
v1=[v(2:n),v(1)];
m=[v;v1;eye(n-2),zeros(n-2,2)];
b=eig(m*m');
```

809.
```
function b=alfa(a,b,n)
c1=(a:a+n-1)';
c2=(b+n-1:-1:b)';
M=ones(2,n-2)*a;
N=eye(n-2)*b;
b=[c1,c2,[M;N]];
```

810.
```
function b=alfa(a)
x=0:.1:2;
y=(a+x)./(x.^2+1);
b=[x',y'];
plot(x,y)
shg
```

811.
```
function a=alfa(n)
if n<4 | fix(n)~=n
  error('n non valido')
end
r1=2:2:2*n;
r2=n:-1:1;
M=eye(n-2)*pi;
N=exp(1)*ones(n-2,2);
a=[r1;r2;M,N];
```

812.
```
function a=alfa(n)
if n>=3
  m=floor(n);
else
  m=floor(6-n);
end
r1=m:-1:1;
M=eye(m-1);
cn=((2:m).^2)';
a=[r1;M,cn];
a(m,1)=n;
```

813.
```
function a=alfa(x)
[r,n]=size(x);
if r~=1
  a=x;
  warning('x non vettore riga')
  return
end
r1=x;
r2=[x(3:n),x(1:2)];
r3=x(n:-1:1);
M=zeros(n-3,3);
N=ones(n-3)*n;
N1=tril(N);
a=[r1;r2;r3;M,N1];
```

814.
```
function flag=alfa(x,n)
  r1=1:n;
  a=[x.^r1;n*ones(n-1,1),eye(n-1)];
  r=eig(a);
  flag=1;
  for id=1:n
     if real(r(id))~=r(id)
        flag=0;
     end
  end
```

815.
```
function b=alfa(z,n)
if n~=fix(n) | n<4
  error('n non valido')
end
v=z:z+n-1;
a=2*(v'*v)/norm(v)-eye(n);
b=a(1:4,1:4);
```

```
for z=0:.1:5
  b=alfa(z,5);
  x=b\[1,1,1,1]';
  if norm(x)>1
     disp(x)
  end
end
```

816.
```
function b=alfa(x,n)
if n<3
  error('n minore di 3')
end
if n~=fix(n)
  n=fix(n);
end
r1=n:-1:1;
c1=(n-1:-1:2)';
M=ones(n-2)*x;
cn=(2:n-1)';
rn=1:n;
b=[r1;c1,M,cn;rn];
```

```
for x=0:.1:5
b=alfa(x,5);
  s=eig(b);
  lambda=max(s);
  if abs(lambda)>10 & abs(lambda)<12
     disp(lambda)
  end
end
```

817.
```
function c=alfa(x,n)
r1=x-(0:0.1:(n-1)/10);
r2=r1.*exp(r1);
b=[r1;r2];
c=b'*b+eye(n);
```

```
ris=Inf;
for id=0:0.1:3
  c=alfa(id,5);
  d=det(c);
  if d<ris
     ris=d;
     c1=c;
  end
end
disp(c1)
```

818. La funzione è continua, quindi per il secondo pezzo si può evitare di ricalcolarla nel punto 1

```
x1=0:.1:1;
x2=1.1:.1:2;
y1=(x1+1)./(x1.^2+1);
y2=exp(1-x2^2);
x=[x1,x2];
y=[y1,y2];
plot(x,y)
```

```
for k=-3:0.1:3
  a=[2,2,k;2,3,k^2-2;k,k^2-2,3];
  d=eig(a);
  flag=1;
  for id=1:3
     if d(id)<0
        flag=0;
     end
  end
  if flag==1
     disp(k)
  end
end
```

819.

```
function v=alfa(a,n)
[r,c]=size(a);
if r~=1 | c~=1
  error('a non scalare')
end
[r,c]=size(n);
if r~=1 | c~=1
  error('n non numero')
end
if n~=fix(n) | n<4
  error('n non valido')
end
v(1)=a;
for id=2:n
  v(id)=v(id-1)^2-1;
end
```

820.

```
function u=alfa(v,w)
[rv,nv]=size(v);
if rv~=1
  error('v non vettore riga')
end
[rw,nw]=size(w);
if rw~=1
  error('w non vettore riga')
end
if norm(v)==0 | norm(w)==0
  error('uno dei vettori è nullo')
end
n=min(nv,nw);
v1=v(1:n);
w1=v(1:n);
u=v1/norm(v1)+w1/norm(w1);
end
```