Gradient method with exact line search (es funzione)

```
Q=[6 0 -4 0;0 6 0 -4;-4 0  6 0;0 -4 0 6]; c = [ 1 -1 2 -3]';
disp('eigenvalues of  Q’) eig(Q)
x0 = [0 0 0 0]'; tolerance = 10^(-6);
x = x0 ;
X=[Inf,Inf,Inf,Inf,Inf,Inf,Inf];
for ITER=1:1000
   v = 0.5*x'*Q*x + c'*x;    g = Q*x + c ;    X=[X;ITER,x',v,norm(g)];   //parametri da aggiornare
   if norm(g) < tolerance
      break                                                              (ulteriori considerzioni)
   end
  d = -g;   %   search direction
  t = norm(g)^2/(d'*Q*d) ; %   exact line search
  x = x + t*d ; %   new point
end
disp('optimal solution')  x
disp('optimal value')   v
disp('gradient norm at the solution') norm(g)
ITER      //numero iterazioni (devo considerare questo -1
```

gradient method with inexact line search (es funzione)

```
alpha = 0.1; gamma = 0.9; tbar = 1;  //data
x0 = [ 10 ; -10]; tolerance = 10^(-3) ;
%% method
X=[Inf,Inf,Inf,Inf,Inf];
ITER = 0 ;
x = x0 ;
while true
  [v, g] = f(x);                                                         (ulteriori considerzioni)
   X=[X;ITER,x(1),x(2),v,norm(g)];
  if norm(g) < tolerance    % stopping criterion
     break
  end
  d = -g;  % search direction
  t = tbar ;
   while f(x+t*d) > v + alpha*g'*d*t   % Armijo inexact line search
      t = gamma*t ;
   end
  x = x + t*d ;   % new point
  ITER = ITER + 1 ;
End
x v norm(g)  ITER //data to show  disp('optimal solution')
function [v, g] = f(x)       //DICHIARAZIONE FUNZIONE
         v = x(1)^4 + x(2)^4 - 2*x(1)^2 + 4*x(1)*x(2)-2*x(2)^2 ;           //FUNZIONE
         g = [4*x(1)^3-4*x(1)+4*x(2);
                 4*x(2)^3+4*x(1)-4*x(2)];                                 //DERIVATA PRIMA
```

```
End
```

Conjugate Gradient method (es funzione)

```
  % Problem definition
  Q = [6 0 -4 0;0 6 0 -4;-4 0 6 0;0 -4 0 6] c = [ 1 -1 2 -3]';
  eig(Q) //autoval di Q
  x0 = [0,0,0,0]'; tolerance = 10^(-6);   %% Parameters
  x = x0; //starting val
  X=[Inf,Inf,Inf,Inf,Inf,Inf];
  for ITER=1:10
    v = 0.5*x'*Q*x + c'*x;
    g = Q*x + c ;
    X=[X;ITER,x',v,norm(g)];
    if norm(g) < tolerance % stopping criterion
      break
    end
    if ITER == 1
      d = -g;    %   search direction
    else
      beta = (g'*Q*d_prev)/(d_prev'*Q*d_prev);
      d = -g + beta*d_prev;   %   search direction
    end
    t = (-g'*d)/(d'*Q*d);    %   step size
    x = x + t*d;             %   new point
    d_prev = d ;
  end
  x v norm(g) ITER //valori da mostrare
```

Newton method with line search (es funzione)

```
alpha=0.1; gamma=0.9; tbar =1; %% data
x0 = [0;0];       tolerance = 10^(-3) ;     x = x0 ;
for ITER=0:100
  [v, g, H] = f(x);
   if norm(g) < tolerance % stopping criterion
       break
  end
  d = -inv(H)*g;  t=tbar;  % search direction e aggiorno t                    (altre considerazioni)
  while (f(x+t*d) > f(x)+alpha*t*d'*g)
    t=gamma*t;
  end
  % new point
  x = x + t*d;
end
x v norm(g)       //valori da mostrare
 function [v, g, H] = f(x)
 v = 2*x(1)^4 + 3*x(2)^4 + 2*x(1)^2 + 4*x(2)^2 + x(1)*x(2) - 3*x(1) - 2*x(2)  ;
 g = [ 8*x(1)^3 + 4*x(1) + x(2) - 3
```

```
    12*x(2)^3 + 8*x(2) + x(1) - 2];
H = [ 24*x(1)^2+4   1
        1       36*x(2)^2+8];
end
```