Non Linear Optimisation I Assignment 4

Exercise 4.4

```
function [p, iters, flag] = steihaug_CG(B,g,radius,tol)

if (norm(B-B')~=0)
        fprintf('Matrix B is non symmetric: error ')
end

p_0 = zeros(size(g));
r_0 = g;
s_0 = -g;

for k = 1:1000

        if (k==1)
        p_k =p_0;
        r_k=r_0;
        s_k = s_0;
        end

        if(norm(r_k,2)> tol* norm(r_0,2))

        if (s_k'*B*s_k>0)
        alpha_k = (r_k'*r_k)/(s_k'*B*s_k);
        else
        tau = sqrt((radius^2 - norm(p_k,2)^2)/norm(s_k,2)^2);
        p_k = p_k +tau*s_k;
        p = p_k;
        iters =k;
        flag =-1;
        break;
        end

        if (norm(p_k+alpha_k*s_k,2)< radius)
        p_k = p_k +alpha_k*s_k;
        else
        tau = sqrt((radius^2 - norm(p_k,2)^2)/norm(s_k,2)^2);
        p_k = p_k +tau*s_k;
        p = p_k;
        iters =k;
        flag = 1;
        break;
        end
```

```
        r_k_1 = r_k + alpha_k*B*s_k;
        Beta = (r_k_1'*r_k_1)/(r_k'*r_k) ;
        s_k =  -r_k_1 + Beta*s_k;
        r_k = r_k_1;

        else

        iters = k;
        p = p_k;
        flag =0;
        break

        end
end
end
```

The function was tested on example 2.3 from the slides
Output:

```
B =

        1        0
        0        2

g =

        2
        4

radius =

        4

[p, iters, flag] = steihaug_CG(B,g,radius,10e-04)

p =

        -2
        -2


iters =

        3
```

flag =

0

Exercise 4.4

a)

```
function [x,F,G,H,iter,status] = unc_TR(fun,x0,maxit,printlevel,tol)

x_k = x0;
eta_vs = 0.9;
eta_s =0.1;
gam_d = 0.5;
gam_i =2;
status = 1;
radius = 0.5;

for iter = 1: maxit

    fh = str2func(fun);
    [F_k,G_k,H_k]  = fh(x_k);

    if (iter ==1)
    F_0 = F_k;
    G_0 =G_k;
    end

    if (printlevel)
    fprintf('\n Iter : % d || function value : %f ',iter, double(norm(F_k,'fro')/norm(F_0,'fro')))
    end

    % trial step using steihaug_CG and second derivative information
    [s_k, ~, ~] = steihaug_CG(H_k,G_k,radius,tol);

    [F_k_s,~,~] = fh(x_k+s_k);

    rho_k = -(F_k-F_k_s)/(G_k'*s_k+0.5*s_k'*H_k*s_k);

    if (rho_k>=eta_vs)
    % very successful iteration
    x_k =x_k +s_k;
    radius = gam_i*radius;
```

```
        elseif(rho_k>=eta_s)
        %  successful iteration

        x_k =x_k +s_k;

        else
        % unsuccessful iteration

        radius = gam_d*radius;

        end

        if(norm(G_k,2) <= 10e-06*max(1 ,norm(G_0,2)))
        x = x_k;
        F = F_k;
        G = G_k;
        H = H_k;
        status = 0;
        break;
        end
end

end
```

b)

```
function [F,G,H] = fun(x)

syms y1 y2
f = @(y) 10*(y2-y1^2)^2 +(y1-1)^2;

F = double(subs(f,[y1;y2],x));
grad = gradient(f,[y1;y2]);
G = double(subs(grad,[y1;y2],x));
hess = hessian(f,[y1;y2]);
H = double(subs(hess,[y1;y2],x));

end
```

Output:

```
[x,F,G,H,iter,status] = unc_TR('fun',[0,0]',100,1,10e-06)

 Iter :  1 || function value : 1.000000
 Iter :  2 || function value : 0.875000
```

```
 Iter :  3 || function value : 0.174093
 Iter :  4 || function value : 0.174093
 Iter :  5 || function value : 0.146829
 Iter :  6 || function value : 0.002905
 Iter :  7 || function value : 0.000081
 Iter :  8 || function value : 0.000000
 Iter :  9 || function value : 0.000000


x =

          1
          1


F =

   7.6285e-18


G =

    1.0e-07 * 0.3456
   -1.0e-07 * 0.1744


H =

   82.0000  -40.0000
  -40.0000   20.0000


iter =

          9


status =

          0
```

Converges to the minima [1;1] in 9 iterations