1 Problem 1 code

```
clear
syms x1 x2 f(x1,x2) g(x1,x2)

f(x1,x2) = x1^4 - 2*x2*x1^2 + x2^2 + x1^2 - 2*x1 + 5;
g(x1,x2) = -(x1 + 0.25)^2 + 0.75*x2;
dfx1 = diff(f,x1);
dfx2 = diff(f,x2);
dgx1 = diff(g,x1);
dgx2 = diff(g,x2);
% Starting point
xn = [0.2533, -0.1583];
hesslagrange = [1, 0; 0, 1];
fn = f(xn(1), xn(2));

x3 = [-0.3566, -0.0109];
hesslagrange = [5.4616,1.8157; 1.8157, 1.9805];
Pprev = 5.85;
% Step to calculate the hessian of the lagrange
[hl_l, xnew, P] = takeStepSQP(f,g,dfx1,dfx2,dgx1,dgx2,x3,hesslagrange,Pprev)

% First step
[hl_l, xnew, P] = takeStepSQP(f,g,dfx1,dfx2,dgx1,dgx2,xnew,hl_l,P)

% Second Step
[hl_l, xnew, P] = takeStepSQP(f,g,dfx1,dfx2,dgx1,dgx2,xnew,hl_l,P)

function [hl_1, xnew, P] = takeStepSQP(f,g,dfx1,dfx2,dgx1,dgx2,xn,hesslagrange,Pprev)
    syms cx1 cx2 lambda
    % Determine values of f and grad f at point
    fn = f(xn(1), xn(2));
    gradfn = [dfx1(xn(1), xn(2)); dfx2(xn(1), xn(2))];

    % Determine values of g and grad g at point
    gn = double(g(xn(1),xn(2)));
    gradgn = double([dgx1(xn(1), xn(2)); dgx2(xn(1), xn(2))]);

    % Make Taylor Series Expansion of function and constraints about point
    changex = [cx1; cx2];
    fa = fn + gradfn.'*changex + 1/2*changex.'*hesslagrange*changex;
    ga = gn + gradgn.'*changex;
    grad_fa_cx1 = diff(fa,cx1);
    grad_fa_cx2 = diff(fa,cx2);
    grad_ga_cx1 = diff(ga,cx1);
    grad_ga_cx2 = diff(ga,cx2);

    % Solve KKT conditions
    solution = solve(grad_fa_cx1 - lambda*grad_ga_cx1 == 0, ...
        grad_fa_cx2 - lambda*grad_ga_cx2 == 0, ...
        ga == 0);

    cx1 = double(solution.cx1);
```

```
    cx2 = double(solution.cx2);
    % If lambda is negative, drop the constraint from the equation
    lambda = double(solution.lambda);

    % Check to make sure penalty function decreased
    xnew = [(xn(1) + cx1), (xn(2) + cx2)];
    fnew = f(xnew(1),xnew(2));
    gnew = g(xnew(1),xnew(2));
    % Penalty is function + sum of penalty of violated constraints
    if gnew < 0
      P = fnew + lambda*abs(gnew);
    else
        P = fnew;
    end
    % Check if P is less than Pprev, if not reduce step size until it is.
    if P >= Pprev
        check = 0
    end

    % Update Lagrangian
    gradfnew = [dfx1(xnew(1), xnew(2)); dfx2(xnew(1), xnew(2))];
    gradgnew = [dgx1(xnew(1), xnew(2)); dgx2(xnew(1), xnew(2))];
    hl = hesslagrange;

    % Gamma is difference in grad lagrangians at x0 and x1 with updated
        lambda
    gradlagr0 = gradfn - lambda*gradgn;
    gradlagr1 = gradfnew - lambda*gradgnew;
    gamma = double(gradlagr1 - gradlagr0);
    cx = [cx1; cx2];
    hl_1 = double(hl + (gamma*gamma.')/(gamma.'*cx) - (hl*cx*cx.'*hl)/(cx
        .'*hl*cx));
end

function [hl_1, xnew, P] = takeStepSQP_noCon(f,dfx1,dfx2,xn,hesslagrange,
    Pprev)
    syms cx1 cx2 lambda
    % Determine values of f and grad f at point
    fn = f(xn(1), xn(2));
    gradfn = [dfx1(xn(1), xn(2)); dfx2(xn(1), xn(2))];

    % Make Taylor Series Expansion of function and constraints about point
    changex = [cx1; cx2];
    fa = fn + gradfn.'*changex + 1/2*changex.'*hesslagrange*changex;
    grad_fa_cx1 = diff(fa,cx1);
    grad_fa_cx2 = diff(fa,cx2);

    % Solve KKT conditions
    solution = solve(grad_fa_cx1 == 0, ...
        grad_fa_cx2 == 0);

    cx1 = double(solution.cx1);
    cx2 = double(solution.cx2);

    % Check to make sure penalty function decreased
    xnew = [(xn(1) + cx1), (xn(2) + cx2)];
```

```
    fnew = f(xnew(1),xnew(2));
    P = fnew;
    % Check if P is less than Pprev, if not reduce step size until it is.
    if P >= Pprev
        cx1 = cx1*0.5;
        cx2 = cx2*0.5;
        % Check to make sure penalty function decreased
        xnew = [(xn(1) + cx1), (xn(2) + cx2)];
        fnew = f(xnew(1),xnew(2));
        P = fnew
    end
    % Update Lagrangian
    gradfnew = [dfx1(xnew(1), xnew(2)); dfx2(xnew(1), xnew(2))];
    hl = hesslagrange;

    % Gamma is difference in grad lagrangians at x0 and x1 with updated
      lambda
    gradlagr0 = gradfn;
    gradlagr1 = gradfnew;
    gamma = double(gradlagr1 - gradlagr0);
    cx = [cx1; cx2];
    hl_1 = double(hl + (gamma*gamma.')/(gamma.'*cx) - (hl*cx*cx.'*hl)/(cx
      .'*hl*cx));
end
```

2 Problem 2 code

```
clear
syms x1 x2 f(x1,x2) g(x1,x2) lambda

f(x1,x2) = x1^4 - 2*x2*x1^2 + x2^2 + x1^2 - 2*x1 + 5;
g(x1,x2) = -(x1 + 0.25)^2 + 0.75*x2;
dfx1 = diff(f,x1);
dfx2 = diff(f,x2);
dgx1 = diff(g,x1);
dgx2 = diff(g,x2);

x1 = [-1.695,2.157];
xn = x1;
% Determine values of f and grad f at point
fn1 = f(xn(1), xn(2));
gradfn1 = [dfx1(xn(1), xn(2)); dfx2(xn(1), xn(2))];

% Determine values of g and grad g at point
gn1 = double(g(xn(1),xn(2)));
gradgn1 = double([dgx1(xn(1), xn(2)); dgx2(xn(1), xn(2))]);

x2 = [-0.592,-1.162];
xn = x2;
% Determine values of f and grad f at point
fn2 = f(xn(1), xn(2));
gradfn2 = [dfx1(xn(1), xn(2)); dfx2(xn(1), xn(2))];

% Determine values of g and grad g at point
gn2 = double(g(xn(1),xn(2)));
gradgn2 = double([dgx1(xn(1), xn(2)); dgx2(xn(1), xn(2))]);
```

```
hl = [20.762,5.629;5.629,1.910];
lambda = 0;

cx1_prev = 1.104;
cx2_prev = -3.319;
% Find update hessian
gradlagr1 = gradfn1 - lambda*gradgn1;
gradlagr2 = gradfn2 - lambda*gradgn2;
gamma = double(gradlagr2 - gradlagr1);
cx = [cx1_prev; cx2_prev];
hl_1 = double(hl + (gamma*gamma.')/(gamma.'*cx) - (hl*cx*cx.'*hl)/(cx.'*hl*
    cx));
s = 0.230;
syms cx1 cx2 clambda cs
A = [18.5612, 5.1258, 0, -0.6840; 5.1258, 2.1849, 0, -0.75; 0, 0, 0, 0.23;
    0.6840, 0.750, -1, 0];
b = -[-6.7655; -3.0249; -0.2; -1.2180];
x = mldivide(A,b)
xnew = [(x2(1) + x(1)), (x2(2) + x(2))];
f3 = double(f(xnew(1),xnew(2)))
g3 = double(g(xnew(1),xnew(2)))
```