~\Documents\documents_general\structured_courses\math565\resources\optCQP.txt

```
function [out]=optCQP(pr)

% fnction optCCP solves a user-defined convex quadratic program
% Author: Tom Asaki
% Version: January 28, 2024
%
% The problem class is
%    min  f(x) = (1/2)x'Gx + x'c
%    s.t.   Ae*x = be
%           A*x >= b
%    with   G p.s.d.
%
% If there are no constraints, return the analytic minimizer.
% Else, if G=0, then the solution is optained by a call to linprog.
% Else, if there are no inequality constraints, then use projected CG
% Otherwise, use an active set method with recursive calls.
%
% USAGE:
%
%   [out]=optCQP(pr)
%
% INPUTS:
%
%   pr  structure variable containing all problem information.
%       .G      symmetric positive definite quadratic term matrix
%       .c      linear term vector of the objective function
%       .Ae     matrix of equality constraint coefficients
%       .be     vector of rhs equality constraint values
%       .A      matrix of inequality constraint coefficients (Ax>=b)
%       .b      vector of rhs inequality constraint values (Ax>=b)
%       .x0     (optional) initial *feasible* decision variable vector.
%       .etol   [1E-8] stop tolerance on PCG iterations (equality problem)
%       .itol   [1E-8] stop tolerance on AS method (general problem)
%
% OUTPUTS:
%
%   out structure variable of output quantities
%       .pr     copy of input problem description with addtional items:
%               .I  number of inequality constraints
%               .E  number of equality constraints
%               .iter   number of PCG iterations for equality constrained
%                       problem or number of active constraint iterations
%                       for the general problem
%       .x      optimal decision variable vector
%       .f      optimal objective value
%

% set any default values as needed
pr=SetDefaults(pr);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The first case is the unconstrained problem.
if pr.I+pr.E==0

      (take the unconstrained newton step)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The second case is a linear program
```

```
elseif ~isfield(pr,'G') || isempty(pr.G)

    [x,f]=linprog(pr.c,-pr.A,-pr.b,pr.Ae,pr.be);
    out.pr=pr;
    out.x=x;
    out.f=f;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The third case is an equality constrained problem
elseif pr.I==0

        (get an initial point if the user did not supply one)
    if ~isfield(pr,'x0') || isempty(pr.x0)
        pr.x0=pr.Ae'*((pr.Ae*pr.Ae')\pr.be);
    end
    out.pr=pr;


    (solve using the equality constrained algorithm 16.2)
    (with equation 16.31 and the formula just previous)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The fourth case is the general CQP with inequality constraints
else

    n=size(pr.A,2);

    % Check to see if a provided initial point is feasible.
    % Set getx0 to true if an initial feasible point is needed.


    % Find an initial feasible point if needed (if getx0==true).
    % The method is a solution to a linear program

    % Setup for the active set method


    % Active Set Method iteration
    goflag=true;
    while goflag

        % solve for step p by calling ECQP (eq. 16.39)
        % epr is the equality constrained subproblem


        % If step is zero then we may be optimal or simply need to
        % remove a constraint which prevents progress
        if norm(p)<eps

            % Compute Lagrange multipliers for all active
            % inequality constraints (16.42).

            % check for optimality or remove the active constraint of minimum lambda

        % Here the step is nonzero, so we will either take the step
        % if it is feasible or step up to a blocking constraint.
        else

            % compute alpha (16.41)

            % Take the step.
            x=x+alpha*p;
            % Add blocking constraint to the active set.

        end  % |p|=0 or |p|>0 decisions

    end  % Main Active Set iteration

end  % CQP cases


return


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function pr=SetDefaults(pr)
df.etol=1E-8;
df.itol=1E-8;
df.Ae=[];
df.be=[];
df.A=[];
df.b=[];
fn=fieldnames(df);
for k=1:length(fn)
    if ~isfield(pr,fn{k}),pr.(fn{k})=df.(fn{k});end
end
pr.I=length(pr.b);
pr.E=length(pr.be);
return
```