```
% Initialization and model definition
run ../handout_files/init_files_2021_v2/init06.m
addpath(genpath("../handout_files/template_problem_2"))

% State Space model (x = [lambda r p p_dot]')
Ts    = 0.25; % sampling time

Ac = [
    0   1      0         0;
    0   0    -K_2        0;
    0   0      0         1;
    0   0  -K_1*K_pp  -K_1*K_pd];

Bc = [
        0;
        0;
        0;
    K_1*K_pp];

% Discrete model
A = eye(4) + Ts * Ac;
B = Ts*Bc;

% Initial value
x0 = [
        pi;                 % Lambda
        0;                  % r
        0;                  % p
        0;                  % p_dot
];

% Number of states and inputs
mx = size(A,2); % number of states
mu = size(B,2); % number of inputs

% Time horizon and initialization
N  = 100;                               % Time horizon for states
M  = N;                                 % Time horizon for inputs
z  = zeros(N*mx+M*mu,1);                % Initialize z for the whole horizon
z0 = z;                                 % Initial value for optimization

% Bounds |p_k| <= (60*pi)/360 = 1/6 * pi
ul        = -pi/6;                    % Lower bound on control
uu        = pi/6;                   % Upper bound on control

xl      = -Inf*ones(mx,1);              % Lower bound on states (no bound)
xu      = Inf*ones(mx,1);               % Upper bound on states (no bound)
xl(3)   = ul;                           % Lower bound on state x3
xu(3)   = uu;                           % Upper bound on state x3

% Constraints
[zlb,zub]       = gen_constraints(N,M,xl,xu,ul,uu);
```

```
zlb(N*mx+M*mu)  = 0;                     % We want the last input to be zero
zub(N*mx+M*mu)  = 0;                     % We want the last input to be zero

% Cost function
Q = zeros(mx,mx);
Q(1,1) = 1;                             % Weight on state x1
Q(2,2) = 0;                             % Weight on state x2
Q(3,3) = 0;                             % Weight on state x3
Q(4,4) = 0;                             % Weight on state x4

R = 12;                                  % Weight on input, use (0.12, 1.2, 12)
G = 2*gen_q(Q,R,N,M);                    % CostMatrix
c = [];                                  % linear constant term

% Liner model system matrices
Aeq = gen_aeq(A,B,N,mx,mu);
beq = zeros(size(Aeq, 1), 1);
beq(1:size(A, 1)) = A*x0;

% Solve QP problem with linear model
tic
[z,lambda] = quadprog(G,c,[], [], Aeq, beq, zlb, zub);
t1=toc;

% Calculate objective value
phi1 = 0.0;
PhiOut = zeros(N*mx+M*mu,1);
for i=1:N*mx+M*mu
  phi1=phi1+G(i,i)*z(i)*z(i);
  PhiOut(i) = phi1;
end

% Extract control inputs and states
u  = [z(N*mx+1:N*mx+M*mu);z(N*mx+M*mu)]; % Control input from solution

x1 = [x0(1);z(1:mx:N*mx)];               % State x1 from solution
x2 = [x0(2);z(2:mx:N*mx)];               % State x2 from solution
x3 = [x0(3);z(3:mx:N*mx)];               % State x3 from solution
x4 = [x0(4);z(4:mx:N*mx)];               % State x4 from solution

num_variables = 5/Ts;
zero_padding = zeros(num_variables,1);
unit_padding  = ones(num_variables,1);

u   = [zero_padding; u; zero_padding];
x1  = [pi*unit_padding; x1; zero_padding];
x2  = [zero_padding; x2; zero_padding];
x3  = [zero_padding; x3; zero_padding];
x4  = [zero_padding; x4; zero_padding];

% Export to simulink
t = 0:Ts:Ts*(length(u)-1);
u_simulink = timeseries(u, t);
```

```
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.
```

Published with MATLAB® R2023b