Table of Contents


```
% TTK4135 - Helicopter lab
% Hints/template for problem 2.
% Updated spring 2018, Andreas L. Flåten
```

Initialization and model definition

```
close
clear

init_simulator; % Change this to the init file corresponding to your
 helicopter

% Discretize model
% Discrete time system model. x = [lambda r p p_dot]'
delta_t = 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];

A1 = eye(4)+delta_t*Ac;
B1 = delta_t*Bc;

% A1 = [1 0.25 0 0;
%       0 0.9925 -0.0975 0;
%       0 0 1 0.25;
%       0 0 -1.7825 0.1];
% B1 = [0; 0; 0; 1.685];
q1 = 4;
q2 = 2;
q3 = 0;
q4 = 0;
Q = diag([q1, q2, q3, q4]);
R = 0.1;

[~,P] = dlqr(A1,B1,Q,R);
```

```
K = 1/R * B1'*P*((eye(4)+B1*(1/R)*B1'*P)\A1);
```

Number of states and inputs

```
mx = size(A1,2); % Number of states (number of columns in A)
mu = size(B1,2); % Number of inputs(number of columns in B)

% Initial values
x1_0 = pi;                                % Lambda
x2_0 = 0;                                % r
x3_0 = 0;                                % p
x4_0 = 0;                                % p_dot
x0 = [x1_0 x2_0 x3_0 x4_0]';           % Initial values

% 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
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

% Generate constraints on measurements and inputs
[vlb,vub]       = gen_constraints(N,M,xl,xu,ul,uu); % hint:
 gen_constraints
vlb(N*mx+M*mu)  = 0;                     % We want the last input to be
 zero
vub(N*mx+M*mu)  = 0;                     % We want the last input to be
 zero

% Generate the matrix Q and the vector c (objecitve function weights
 in the QP problem)
Q1 = zeros(mx,mx);
Q1(1,1) = 1;                             % Weight on state x1
Q1(2,2) = 0;                             % Weight on state x2
Q1(3,3) = 0;                             % Weight on state x3
Q1(4,4) = 0;                             % Weight on state x4
P1 = 10;                                % Weight on input
Q1 = gen_q(Q1,P1,N,M);                    % Generate Q, hint: gen_q
c = 0;                                   % Generate c, this is the
 linear constant term in the QP
```

Generate system matrixes for linear model

```
Aeq = gen_aeq(A1,B1,N,mx,mu);            % Generate A, hint: gen_aeq
beq = zeros(size(Aeq,1),1);                  % Generate b
beq(1) = pi;
```

Solve QP problem with linear model

```
tic
[z,lambda] = quadprog(Q1,[],[],[],Aeq,beq,vlb,vub); % hint: quadprog.
 Type 'doc quadprog' for more info
t1=toc;

% Calculate objective value
phi1 = 0.0;
PhiOut = zeros(N*mx+M*mu,1);
for i=1:N*mx+M*mu
  phi1=phi1+Q1(i,i)*z(i)*z(i);
  PhiOut(i) = phi1;
end
```

Minimum found that satisfies the constraints.

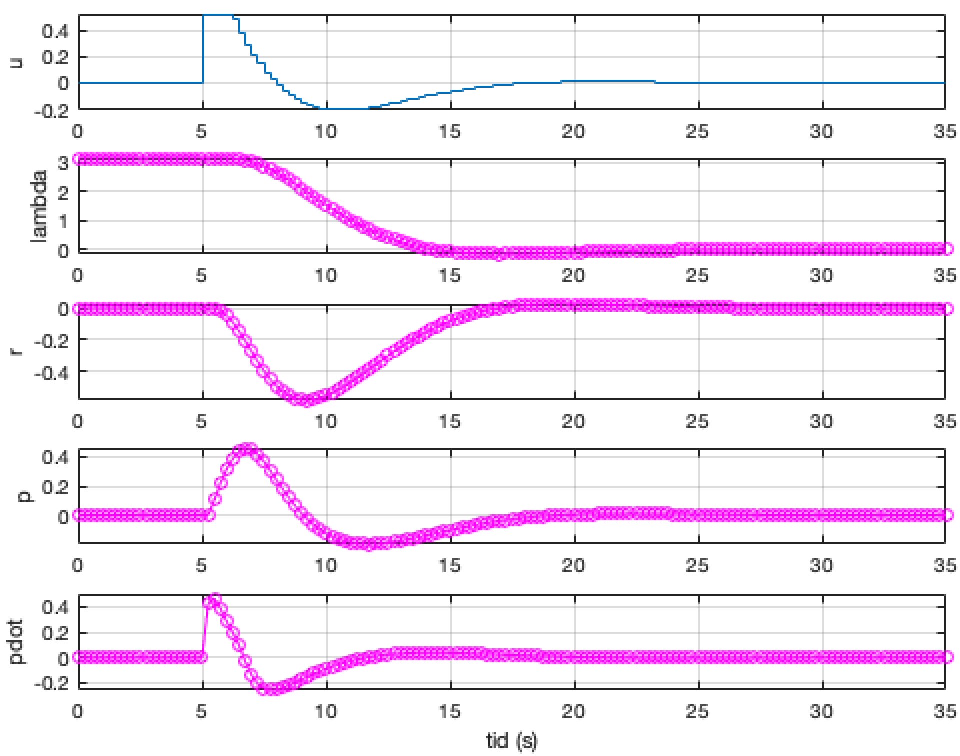
Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

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/delta_t;
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];
```

Plotting

```
t = 0:delta_t:delta_t*(length(u)-1);


%u = [t' u];
x_ref = [t' x1 x2 x3 x4];
save('x_ref.mat', 'x_ref')

figure(2)
subplot(511)
stairs(t,u),grid
ylabel('u')
subplot(512)
plot(t,x1,'m',t,x1,'mo'),grid
ylabel('lambda')
subplot(513)
plot(t,x2,'m',t,x2','mo'),grid
ylabel('r')
subplot(514)
plot(t,x3,'m',t,x3,'mo'),grid
ylabel('p')
subplot(515)
plot(t,x4,'m',t,x4','mo'),grid
xlabel('tid (s)'),ylabel('pdot')
```



Published with MATLAB® R2020a