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

% Continous model
delta_t = 0.25; % sampling time


Ac = [0  1  0          0          0          0;
       0  0  -K_2       0          0          0;
       0  0  0          1          0          0;
       0  0  -K_1*K_pp  -K_1*K_pd  0          0;
       0  0  0          0          0          1;
       0  0  0          0          -K_3*K_ep  -K_3*K_ed];

Bc = [0          0;
       0          0;
       0          0;
       K_1*K_pp   0;
       0          0;
       0          K_3*K_ep];

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

% Discrete time system model. x = [lambda r p p_dot]'
A1 = eye(mx) + delta_t*Ac;
B1 = Bc*delta_t;


q1 = 1;
q2 = 0;
```

```
q3 = 0;
q4 = 0;
q5 = 0;
q6 = 0;
Q = diag([q1, q2, q3, q4, q5, q6]);
R = diag([1 1]);

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

K = 1\R * B1'*P*((eye(mx)+B1*(1\R)*B1'*P)\A1);

% Initial values
x1_0 = pi;                                    % Lambda
x2_0 = 0;                                    % r
x3_0 = 0;                                    % p
x4_0 = 0;                                    % p_dot
x5_0 = 0;                                    % e
x6_0 = 0;                                    % e_dot
x0 = [x1_0 x2_0 x3_0 x4_0 x5_0 x6_0]';  % Initial values

% Time horizon and initialization
N  = 40;                                       % 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      = -Inf*ones(mu,1);                 % Lower bounds on control
uu      = Inf*ones(mu,1);                  % Upper bounds on control
ul(1)  = -pi/6;                       % First lower bound on control
uu(1)   = pi/6;                        % First 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(1);                                % Lower bound on state x3
xu(3)   = uu(1);                                % Upper bound on state x3
% xl(2)   = -pi/6;
% xl(6)  = -pi/6;
% xu(2)   = pi/6;
% xu(6)  = pi/6;

% 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
```

```
Q1 = gen_q(Q,R,N,M);                    % Generate Q, hint: gen_q
```

Generate system matrixes for linear model

```
Aeq = gen_aeq(A1,B1,N,mx,mu);           % Generate A, hint: gen_aeq
beq = Aeq*z0;                             % Generate b
beq(1:mx) = A1*x0;                      % Set first term
z0(1:mx) = x0;                          % Initial value for
 optimization
```

Solve QP problem with linear model

```
f = @(X) X'*Q1*X;
options = optimoptions('fmincon','Algorithm','sqp');
tic
z = fmincon(f,z0,[],[],Aeq,beq,vlb,vub, @constraint, options);
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
```

Extract control inputs and states

```
u1  = [z(N*mx+1:mu:end); z(N*mx+M*mu-1)];                % Control
 input from solution
u2  = [z(N*mx+2:mu:end); 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
x5 = [x0(5);z(5:mx:N*mx)];              % State x5 from solution
x6 = [x0(6);z(6:mx:N*mx)];              % State x6 from solution

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

u1   = [zero_padding; u1; zero_padding];
u2  = [zero_padding; u2; 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];
x5  = [zero_padding; x5; zero_padding];
```

```
x6  = [zero_padding; x6; zero_padding];

t = 0:delta_t:(N+2*num_variables)*delta_t;
x_ref = [t' x1 x2 x3 x4 x5 x6];
%save('x_ref.mat', 'x_ref')
u = [t' u1 u2];
```

Plotting

```
figure(2)
subplot(421)
stairs(t,u1),grid
ylabel('u1')
subplot(422)
stairs(t,u2),grid
ylabel('u2')
subplot(423)
plot(t,x1,'m',t,x1,'mo'),grid
ylabel('lambda')
subplot(424)
plot(t,x2,'m',t,x2','mo'),grid
ylabel('r')
subplot(425)
plot(t,x3,'m',t,x3,'mo'),grid
ylabel('p')
subplot(426)
plot(t,x4,'m',t,x4','mo'),grid
xlabel('tid (s)'),ylabel('pdot')
subplot(427)
plot(t,x5,'m',t,x5,'mo'),grid
ylabel('e')
subplot(428)
plot(t,x6,'m',t,x6','mo'),grid
xlabel('tid (s)'),ylabel('edot')
```

Published with MATLAB® R2020a