Norwegian University of Science and Technology
Department for Engineering Cybernetics

Optimization and Control TTK4135 Spring 2021

Jørgen Rosager
Exercise 7

1 Choosing: $I = S$, $BR^{-1}B^\top P = UTV$, $R^{-1} = T \implies T^{-1} = R$, $VS^{-1}U = B^\top PB \implies U = B \wedge V = B^\top P$. These were chosen to match it with the expected result (6). By the matrix inversion lemme this gives us:

$$(I + BR^{-1}B^\top P)^{-1} = I - IB(R + B^\top PIB)^{-1}B^\top PI$$

Inserting this into (5) gives us:

$$\begin{aligned} P &= Q + A^\top P(I - IB(R + B^\top PIB)^{-1}B^\top PI)A \\ &= Q + A^\top PA - A^\top PB(R + B^\top PB)^{-1})B^\top PA \\ 0 &= Q + A^\top PA - A^\top PB(R + B^\top PB)^{-1})B^\top PA - P \end{aligned}$$

Which is (6).

2 **a)** The code used is shown below.

```
Q = 1/2 * [
        4  0
        0  4
 ];

A = [
        1  0.1
        -0.1  1-0.1
 ];

R = 1/2 * 1;

B = [
        0
        0.1
 ];

[K, S, e] = dlqr(A, B, Q, R, []);
disp(K);
disp(S);
disp(e);
```

Results in $K = \begin{bmatrix} 1.0373 \\ 1.6498 \end{bmatrix}$ and $e = \mathrm{eig}(A - BK) = 0.8675 \pm 0.0531i$

b) The code used is shown below, I used the poles suggested by the task.

```
Q = 1/2 * [
        4 0
        0 4
];

A = [
        1 0.1
        -0.1 1-0.1
];

R = 1/2 * 1;

B = [
        0
        0.1
];

C = [
        1
        0
];

p = [
        0.5 + 0.03i
        0.5 - 0.03i
];
K_f = place(A', C, p);

[K, S, e] = dlqr(A, B, Q, R, []);

x_0 = [5 1]';
x_0_e = [6 0]';

x = zeros(2,50);
x(:,1) = x_0;
x_e = zeros(2,50);
x_e(:,1) = x_0_e;
u = zeros(50);

for t=2:50
    u(t-1) = -K*x_e(:,t-1);
    y = C'*x(:,t-1);
    y_e = C'*x_e(:,t-1);
    x(:,t) = A*x(:,t-1) + B*u(t-1);
    x_e(:,t) = A*x_e(:,t-1) + B*u(t-1)+K_f'*(y-y_e);
end

figure(1);
```

```
subplot(2, 1, 1);
t = 0:49;
plot(t, x(1,:));
hold on;
plot(t, x_e(1,:));
hold off;
xlabel('t');
ylabel('x_1');
legend('$x_1(t)$', '$\hat{x}_1(t)$', 'Interpreter','latex
    ');
grid('on');

figure(1);
subplot(2, 1, 2);
t = 0:49;
plot(t, x(2,:));
hold on;
plot(t, x_e(2,:));
hold off;
xlabel('t');
ylabel('x_2');
legend('$x_2(t)$', '$\hat{x}_2(t)$', 'Interpreter','latex
    ');
grid('on');
```

This resulted in the graph shown in Figure 1.

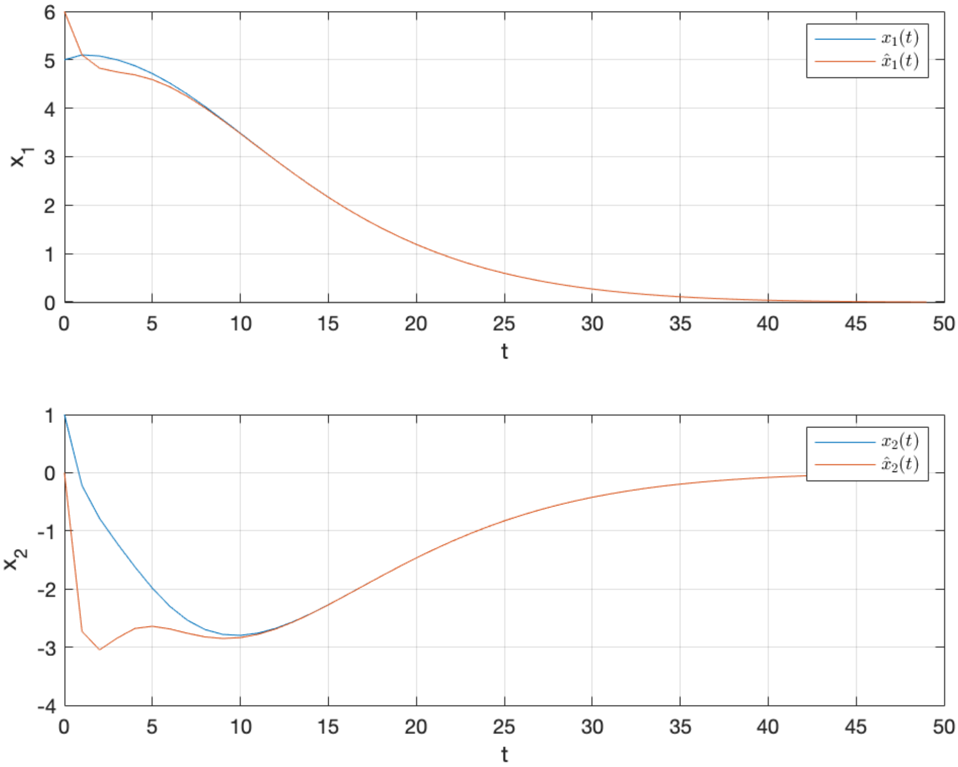

Figure 1: LQR with output feedback.

As we can see the estimated state converges pretty quickly towards the real state. The controller uses some time to converge towards the origin, but manages to do it within 50 time steps.

c) We have that:

$$BK = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \begin{bmatrix} 1.0373 & 1.6498 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0.10373 & 0.16498 \end{bmatrix}$$

$$A - BK = \begin{bmatrix} 1 & 0.1 \\ -0.1 & 0.9 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0.10373 & 0.16498 \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ -0.2037 & 0.7350 \end{bmatrix}$$

$$A - K_F C = \begin{bmatrix} 1 & 0.1 \\ -0.1 & 0.9 \end{bmatrix} - \begin{bmatrix} 0.9 \\ 1.509 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 \\ -1.609 & 0.9 \end{bmatrix}$$

$$\implies \Phi = \begin{bmatrix} 1 & 0.1 & 0 & 0 \\ -0.2037 & 0.7350 & 0.10373 & 0.16498 \\ 0 & 0 & 0.1 & 0.1 \\ 0 & 0 & -1.609 & 0.9 \end{bmatrix}$$

Using `eig` in Matlab to find the eigen values of Φ:

$$\lambda_1 = 0.8675 \pm 0.0530i$$
$$\lambda_2 = 0.5 \pm 0.03i$$

Which are the poles we got earlier.

3 **a)** The code used is shown below.

```
Qt = [
      4  0
      0  4
];

A = [
      1  0.1
      -0.1  1-0.1
];

B = [
      0
      0.1
];

C = [
      1
      0
];

x0 = [5  1]';
x0_e = [6  0]';

N = 10;
nx = 2;
nu = 1;
r = 1;

% Set G (objective function)
Q = kron(eye(N),Qt);
Rt = r;
R = kron(eye(N),Rt);
G = blkdiag(Q, R);


% Set equality constraints (Aeq and Beq)
Beq = zeros(N*nx, 1);

Aeq_1 = eye(N*nx);
Aeq_2 = kron(diag(ones(N-1,1),-1),-A);
Aeq_3 = kron(eye(N), -B);
Aeq = [Aeq_1 + Aeq_2, Aeq_3];

u_low = -4;
u_high = 4;

x_high = inf;
x_low = -inf;
```

```
lb = [x_low*ones(N*nx, 1); u_low*ones(N*nu, 1)];
ub = [x_high*ones(N*nx, 1); u_high*ones(N*nu, 1)];

p = [
      0.5 + 0.03i
      0.5 - 0.03i
];
K_f = place(A', C, p);


x = zeros(nx,N);
x(:,1) = x0;
x_e = zeros(nx,N);
x_e(:,1) = x0_e;

u = zeros(nu, 51);

for t = 1:50
     Beq(1:nx) = A*x_e(:,t);
     z = quadprog(G, [], [], [], Aeq, Beq, lb, ub);
     u(t) = z(N*nx+1); % get first element of u as actual control input
     disp(u(t));
     x(:,t+1) = A*x(:,t) + B*u(t); % state space update
     y_e = C'*x_e(:,t);
     y = C'*x(:,t);
     x_e(:,t+1) = A*x_e(:,t) + B*u(t) + K_f'*(y-y_e);
end

figure(1);
subplot(3, 1, 1);
t = 0:50;
plot(t, x(1,:));
hold on;
plot(t, x_e(1,:));
hold off;
xlabel('t');
ylabel('x_1');
legend('$x_1(t)$', '$\hat{x}_1(t)$', 'Interpreter','latex');
grid('on');

figure(1);
subplot(3, 1, 2);
plot(t, x(2,:));
hold on;
plot(t, x_e(2,:));
hold off;
```

```
xlabel('t');
ylabel('x_2');
legend('$x_2(t)$', '$\hat{x}_2(t)$', 'Interpreter','latex
    ');
grid('on');

disp(size(u));
disp(size(t));
figure(1);
subplot(3, 1, 3);
plot(t, u);
xlabel('t');
ylabel('u');
legend('$u(t)$', 'Interpreter','latex');
grid('on');
```

The results are given in Figure 2.

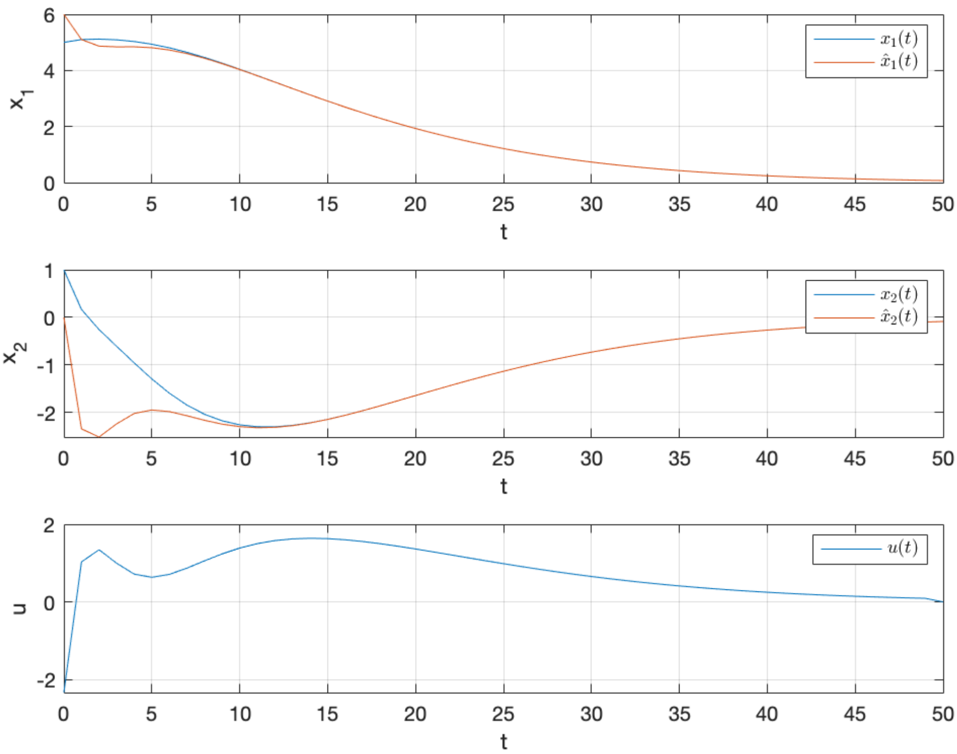

Figure 2: MPC with output feedback.

The code used is shown below.

```
Qt = [
    4 0
    0 4
];
```

```

A = [
        1  0.1
        -0.1  1-0.1
];

B = [
        0
        0.1
];

C = [
        1
        0
];

x0 = [5  1]';
x0_e = [6  0]';

N = 10;
nx = 2;
nu = 1;
r = 1;

% Set G (objective function)
Q = kron(eye(N),Qt);
Rt = r;
R = kron(eye(N),Rt);
G = blkdiag(Q, R);


% Set equality constraints (Aeq and Beq)
Beq = zeros(N*nx, 1);

Aeq_1 = eye(N*nx);
Aeq_2 = kron(diag(ones(N-1,1),-1),-A);
Aeq_3 = kron(eye(N), -B);
Aeq = [Aeq_1 + Aeq_2, Aeq_3];

u_low = -4;
u_high = 4;

x_high = inf;
x_low = -inf;

lb = [x_low*ones(N*nx, 1); u_low*ones(N*nu, 1)];
ub = [x_high*ones(N*nx, 1); u_high*ones(N*nu, 1)];

p = [
```

```
        0.5 + 0.03i
        0.5 - 0.03i
];
K_f = place(A', C, p);


x = zeros(nx,N);
x(:,1) = x0;
x_e = zeros(nx,N);
x_e(:,1) = x0_e;

u = zeros(nu, 51);

for t = 1:50
    Beq(1:nx) = A*x_e(:,t);
    z = quadprog(G, [], [], [], Aeq, Beq, lb, ub);
    u(t) = z(N*nx+1); % get first element of u as actual
        control input
    disp(u(t));
    x(:,t+1) = A*x(:,t) + B*u(t); % state space update
    y_e = C'*x_e(:,t);
    y = C'*x(:,t);
    x_e(:,t+1) = A*x_e(:,t) + B*u(t) + K_f'*(y-y_e);
end

figure(1);
subplot(3, 1, 1);
t = 0:50;
plot(t, x(1,:));
hold on;
plot(t, x_e(1,:));
hold off;
xlabel('t');
ylabel('x_1');
legend('$x_1(t)$', '$\hat{x}_1(t)$', 'Interpreter','latex');
grid('on');

figure(1);
subplot(3, 1, 2);
plot(t, x(2,:));
hold on;
plot(t, x_e(2,:));
hold off;
xlabel('t');
ylabel('x_2');
legend('$x_2(t)$', '$\hat{x}_2(t)$', 'Interpreter','latex');
grid('on');

disp(size(u));
```

```
disp(size(t));
figure(1);
subplot(3, 1, 3);
plot(t, u);
xlabel('t');
ylabel('u');
legend('$u(t)$', 'Interpreter','latex');
grid('on');
```

b) The results are given in Figure 4.

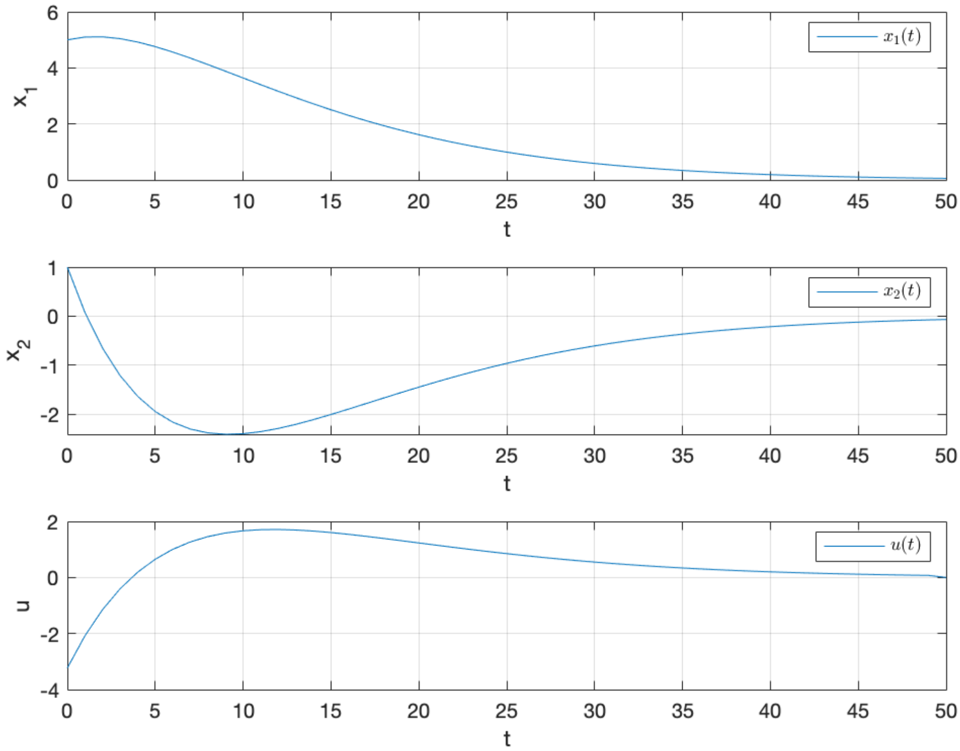

Figure 3: MPC with state feedback.

4 **a)** P was gotten in Problem 1:

$$P = \begin{bmatrix} 27.5170 & 7.2713 \\ 7.2713 & 10.2339 \end{bmatrix}$$

b) The code used is shown below.

```
Qt = [
        4  0
        0  4
];

```

```
A = [
      1  0.1
      -0.1  1-0.1
 ];

B = [
      0
      0.1
 ];

C = [
      1
      0
 ];
Rt = 1;

x0 = [5  1]';

N = 1;
nx = 2;
nu = 1;
r = 1;

% Set G (objective function)

[K, S, e] = dlqr(A, B, Qt/2, Rt/2, []);
disp(S);
Q = kron(eye(N),Qt);
Q(N*nx-1:N*nx, N*nx-1:N*nx) = S;
disp(Q);
Rt = r;
R = kron(eye(N),Rt);
G = blkdiag(Q, R);



% Set equality constraints (Aeq and Beq)
Beq = zeros(N*nx, 1);

Aeq_1 = eye(N*nx);
Aeq_2 = kron(diag(ones(N-1,1),-1),-A);
Aeq_3 = kron(eye(N), -B);
Aeq = [Aeq_1 + Aeq_2, Aeq_3];

u_low = -4;
u_high = 4;

x_high = inf;
x_low = -inf;
```

```

lb = [x_low*ones(N*nx, 1); u_low*ones(N*nu, 1)];
ub = [x_high*ones(N*nx, 1); u_high*ones(N*nu, 1)];

x = zeros(nx,N);
x(:,1) = x0;

u = zeros(nu, 51);

for t = 1:50
    Beq(1:nx) = A*x(:,t);
    z = quadprog(G, [], [], [], Aeq, Beq, lb, ub);
    u(t) = z(N*nx+1); % get first element of u as actual
        control input
    disp(u(t));
    x(:,t+1) = A*x(:,t) + B*u(t); % state space update
end

figure(1);
subplot(3, 1, 1);
t = 0:50;
plot(t, x(1,:));
xlabel('t');
ylabel('x_1');
legend('$x_1(t)$','Interpreter','latex');
grid('on');

figure(1);
subplot(3, 1, 2);
plot(t, x(2,:));
xlabel('t');
ylabel('x_2');
legend('$x_2(t)$','Interpreter','latex');
grid('on');

disp(size(u));
disp(size(t));
figure(1);
subplot(3, 1, 3);
plot(t, u);
xlabel('t');
ylabel('u');
legend('$u(t)$', 'Interpreter','latex');
grid('on');
```

The results are given in Figure **??**.

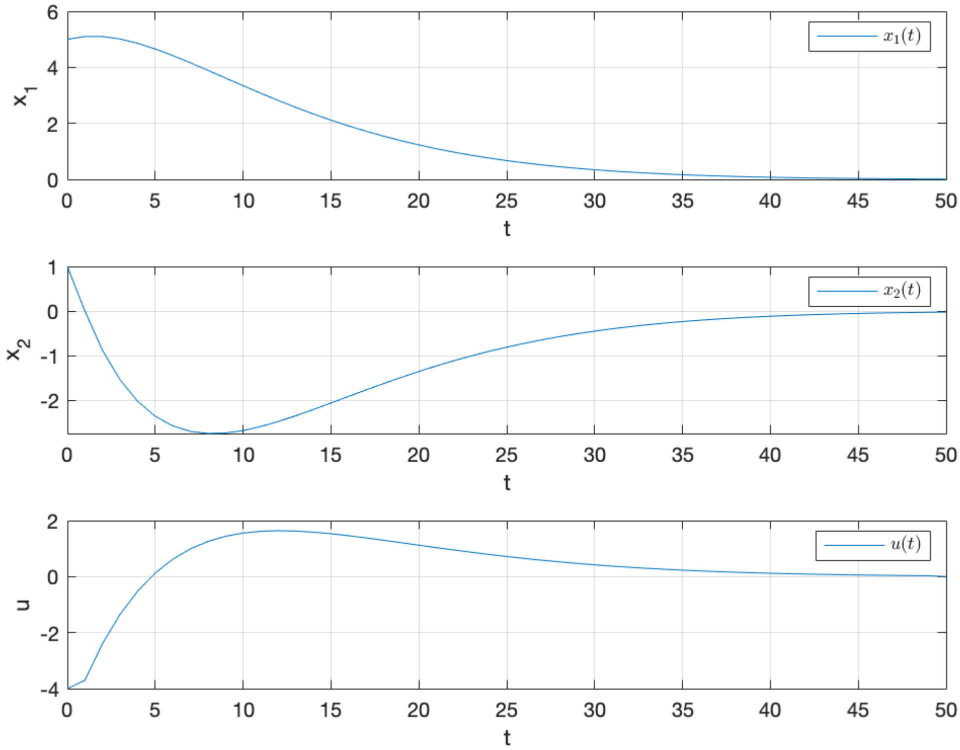

Figure 4: MPC with state feedback and infinite horizon.

We can see that this MPC converges slightly faster towards the origin than the MPC from Problem 3b. Changing N did not change the solution much, as far as my testing went (from values 1-30), the control input constraints where inactive except for the first few control inputs. N becomes more important for performance in realistic scenarios without state feedback and when there are clearer performance goals, like runtime or memory constraints.