Contents

Exercise 11, Tasks 11.6 - 11.12 (explicit MPC)

```
clear
close all
```

Task 11.6

Here we formulate an MPC problem for the system in Exercise 7.

```
% MPC data
A = [0.7115 -0.4345; 0.4345 0.8853];
B = [0.2173; 0.0573];
R = 1;
Q = 10*eye(2);
[~, P] = dlqr(A, B, Q, R);
umin = -5;
umax = 5;
xmin = [-2.8; 0];
xmax = [10; 10];
N = 5; % prediction horizon

% Prediction model
model = LTISystem('A', A, 'B', B); % x(k+1) = A*x(k) + B*u(k)
model.u.min = umin;
model.u.max = umax;
model.x.min = xmin;
model.x.max = xmax;
model.u.penalty = QuadFunction(R); % quadratic penalty u'*R*u
model.x.penalty = QuadFunction(Q); % quadratic penalty x'*Q*x
% the terminal penalty is optional and has to be enabled by the user
model.x.with('terminalPenalty');
model.x.terminalPenalty = QuadFunction(P);

% Construct the explicit solution
empc = EMPCController(model, N);
```

```
mpt_plcp: 43 regions
```

Task 11.7

Evaluate the explicit representation of the MPC feedback law for one particular initial condition

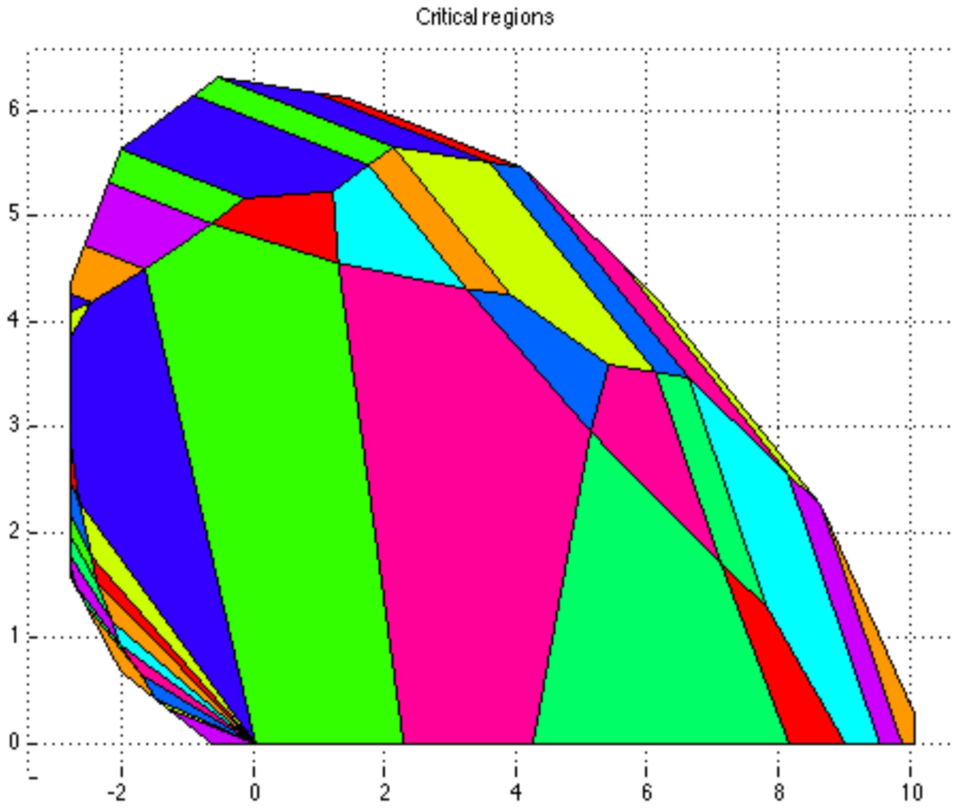
```
x0 = [0; 6];
% Obtain the optimal control action
uopt = empc.evaluate(x0)
% Obtain the full open-loop optimal control sequence
Uopt = empc.optimizer.feval(x0, 'primal')
```

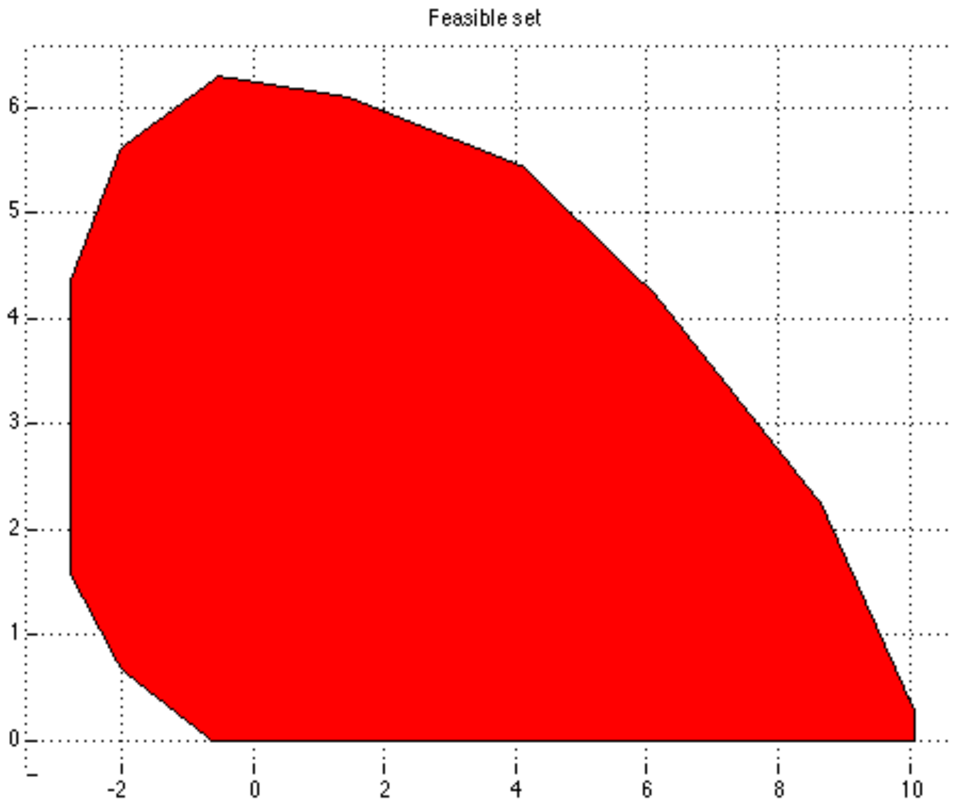
```
uopt =

    2.1305


Uopt =

    2.1305
    5.0000
    4.6117
    4.8905
    3.9839
```

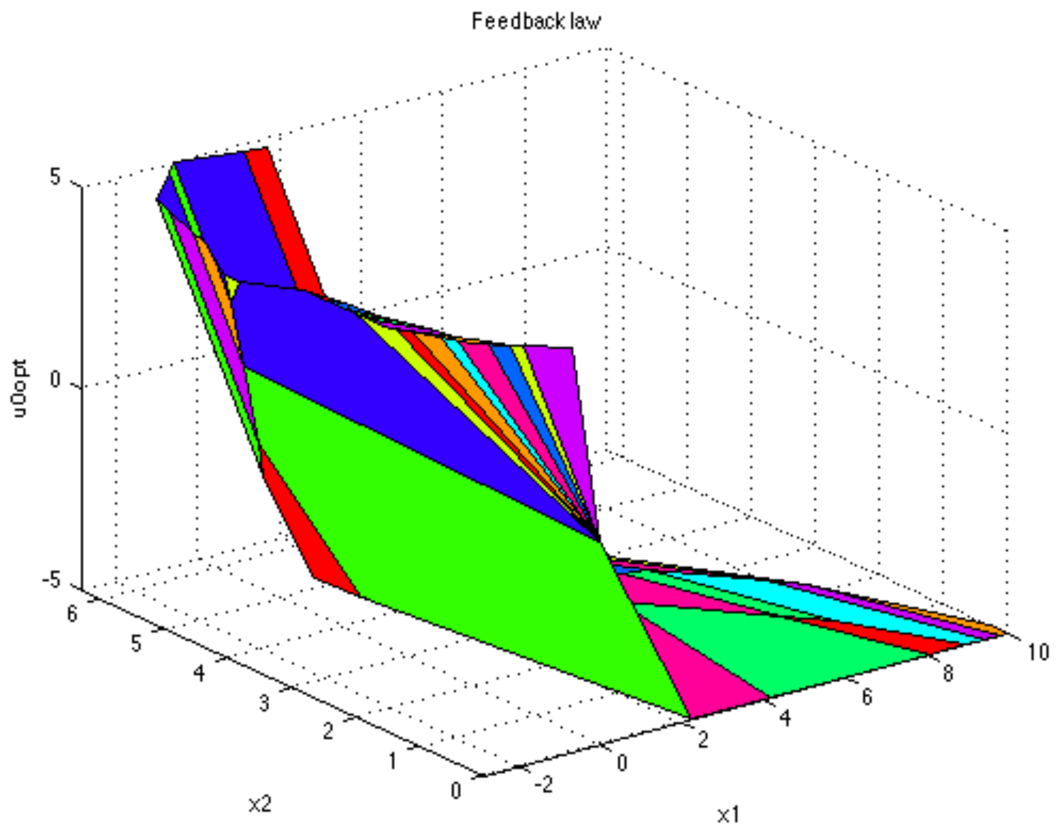
Task 11.8

Plot the critical regions, the feasible set and the piecewise affine feedback law

```
% Plot the critical regions
figure; empc.partition.plot(); title('Critical regions');
% Plot the feasible set
figure; empc.partition.Domain.plot(); title('Feasible set');
% Plot the feedback law
figure;
empc.feedback.fplot();
xlabel('x1'); ylabel('x2'); zlabel('u0opt');
title('Feedback law');
```


Critical regions
6
5
4
3
2
1
0
-2
0
2
4
6
8
10


Feasible set
6
5
4
3
2
1
0
-2
0
2
4
6
8
10

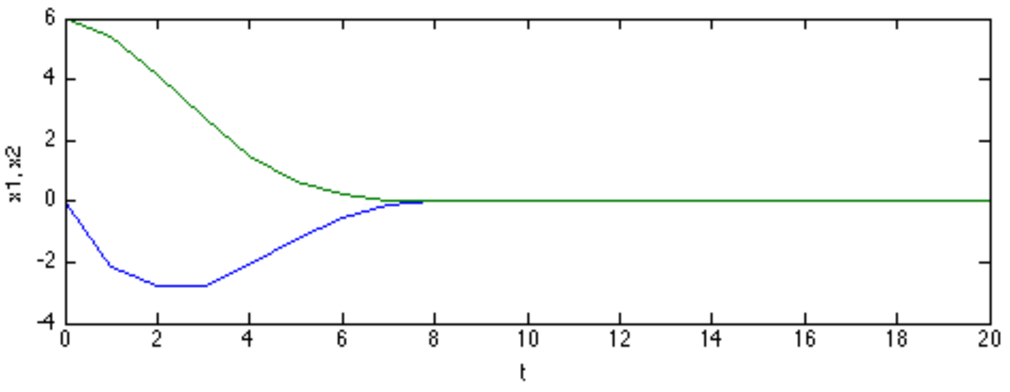


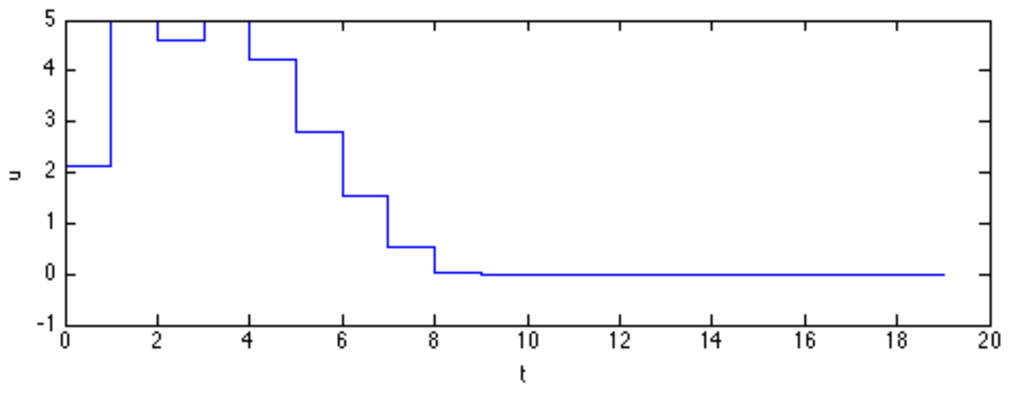
Task 11.9

Closed-loop simulation with the explicit MPC controller

```
loop = ClosedLoop(empc, model);
% Simulate the closed loop for 20 steps
x0 = [0; 6]; Nsim = 20;
data = loop.simulate(x0, Nsim)
figure
subplot(2, 1, 1); plot(0:Nsim, data.X); xlabel('t'); ylabel('x1, x2');
subplot(2, 1, 2); stairs(0:Nsim-1, data.U); xlabel('t'); ylabel('u');
```

```
data = 

       X: [2x21 double]
       U: [1x20 double]
       Y: [0x20 double]
    cost: [1x20 double]
```





Calculate a Lyapunov function to certify closed-loop stability

```
lyap = loop.toSystem.lyapunov('pwq'); % PWQ = piecewise quadratic
% Successful construction means the closed-loop system is stable for all
% feasible initial conditions
```

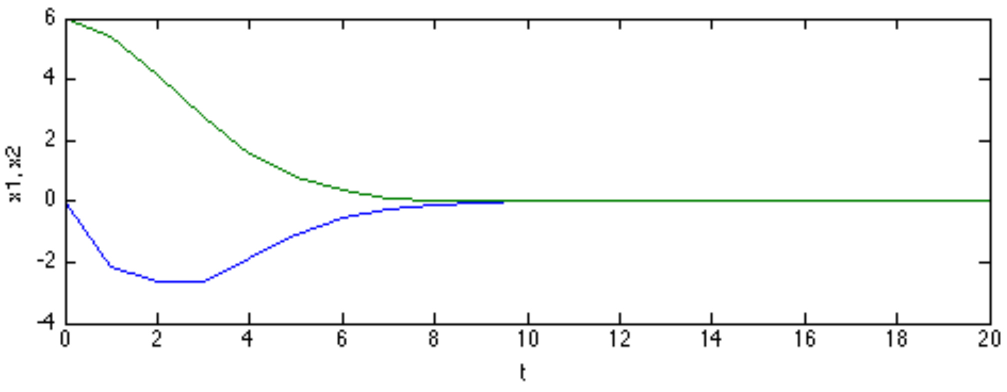
```
Computing the transition map...
progress: 19/43
progress: 37/43
...done (63 full-dimensional transitions)
Checking invariance...
progress: 40/43
...done
Formulating constraints...
...done
Solving...
SeDuMi 1.3 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, theta = 0.250, beta = 0.500
Put 30 free variables in a quadratic cone
eqs m = 1112, order n = 1672, dim = 2245, blocks = 108
nnz(A) = 8228 + 0, nnz(ADA) = 33618, nnz(L) = 18364
 it :     b*y       gap    delta  rate   t/tP*  t/tD*   feas cg cg  prec
  0 :            3.26E+03 0.000
  1 :   0.00E+00 1.57E+03 0.000 0.4808 0.9000 0.9000   1.00  1  1  1.1E+03
  2 :   0.00E+00 7.21E+02 0.000 0.4604 0.9000 0.9000   1.00  1  1  5.2E+02
  3 :   0.00E+00 3.14E+02 0.000 0.4353 0.9000 0.9000   1.00  1  1  2.2E+02
  4 :   0.00E+00 1.20E+02 0.000 0.3818 0.9000 0.9000   1.00  1  1  8.6E+01
  5 :   0.00E+00 7.78E+00 0.119 0.0649 0.9900 0.9900   1.00  1  1  5.6E+00
  6 :   0.00E+00 3.26E-02 0.000 0.0042 0.9990 0.9990   1.00  1  1  2.3E-02
  7 :   0.00E+00 1.00E-08 0.000 0.0000 1.0000 1.0000   1.00  1  1  7.5E-09
  8 :   0.00E+00 1.83E-09 0.000 0.1820 0.9000 0.9000   1.21  5  5  1.3E-09
  9 :   0.00E+00 3.49E-10 0.000 0.1912 0.9000 0.9000   2.23  5  5  1.3E-10

iter seconds digits       c*x               b*y
  9      0.6   10.8 -1.0739075253e-17  0.0000000000e+00
|Ax-b| =   1.1e-10, [Ay-c]_+ =   4.5E-15, |x|=  6.8e-01, |y|=  1.2e+02

Detailed timing (sec)
   Pre          IPM          Post
6.538E-03    5.827E-01    3.380E-03
Max-norms: ||b||=0, ||c|| = 1.000000e-06,
Cholesky |add|=3, |skip| = 0, ||L.L|| = 405.929.
...done

Feasible solution found
```

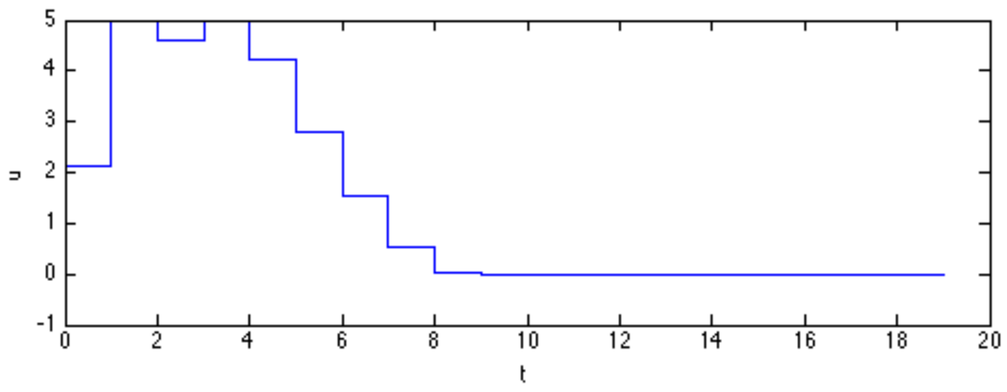
Task 11.10

Closed-loop simulation with a different simulation model

```
An = [0.6404 -0.4345; 0.4345 0.8853];
simmodel = LTISystem('A', An, 'B', B);
loop2 = ClosedLoop(empc, simmodel);
data2 = loop2.simulate(x0, Nsim)
figure
subplot(2, 1, 1); plot(0:Nsim, data2.X); xlabel('t'); ylabel('x1, x2');
subplot(2, 1, 2); stairs(0:Nsim-1, data.U); xlabel('t'); ylabel('u');
```

```
data2 =

       X: [2x21 double]
       U: [1x20 double]
       Y: [0x20 double]
    cost: [1x20 double]
```



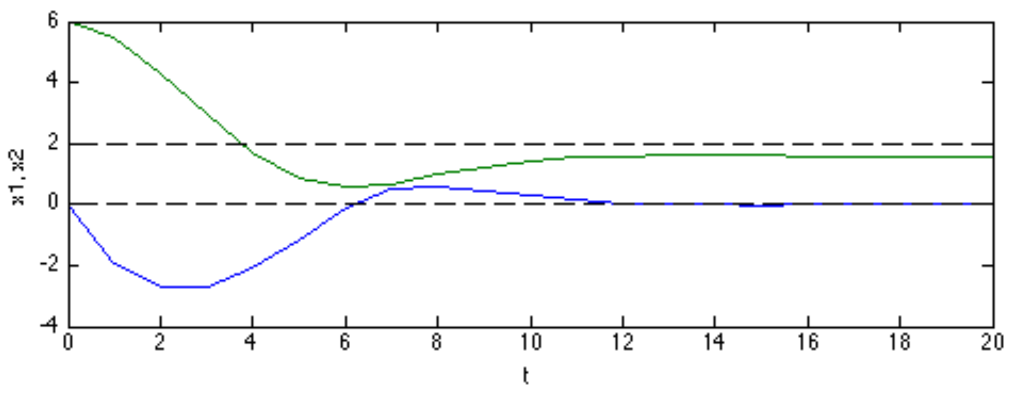


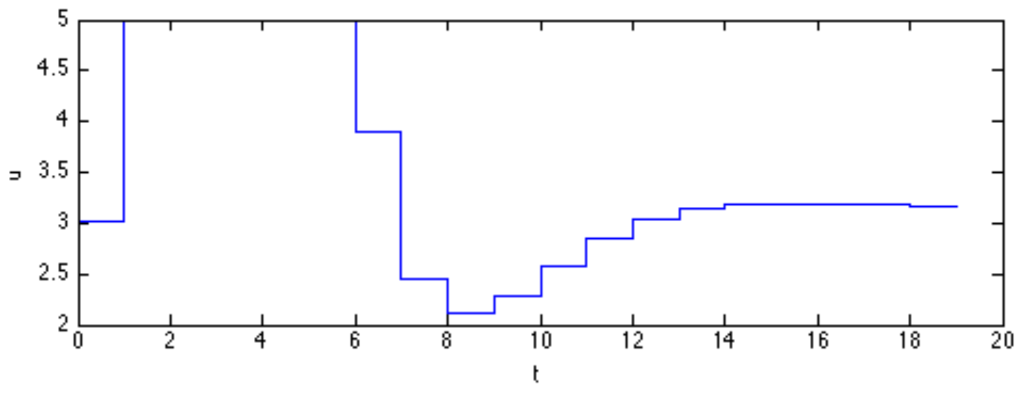
Task 11.11

Tracking of non-zero references

```
% State reference to be tracked
xref = [0; 2];
% Enable tracking of non-zero references
model.x.with('reference');
model.x.reference = xref;
% Create a new controller
empc = EMPCController(model, N);
% Simulate the closed loop
loop = ClosedLoop(empc, model);
data = loop.simulate(x0, Nsim)
% Notice a steady-state offset
figure
subplot(2, 1, 1);
plot(0:Nsim, data.X);
hold on
plot(0:Nsim, repmat(xref, 1, Nsim+1), 'k--');
xlabel('t'); ylabel('x1, x2');
subplot(2, 1, 2); stairs(0:Nsim-1, data.U); xlabel('t'); ylabel('u');
```

```
mpt_plcp: 31 regions

data =
       X: [2x21 double]
       U: [1x20 double]
       Y: [0x20 double]
    cost: [1x20 double]
```



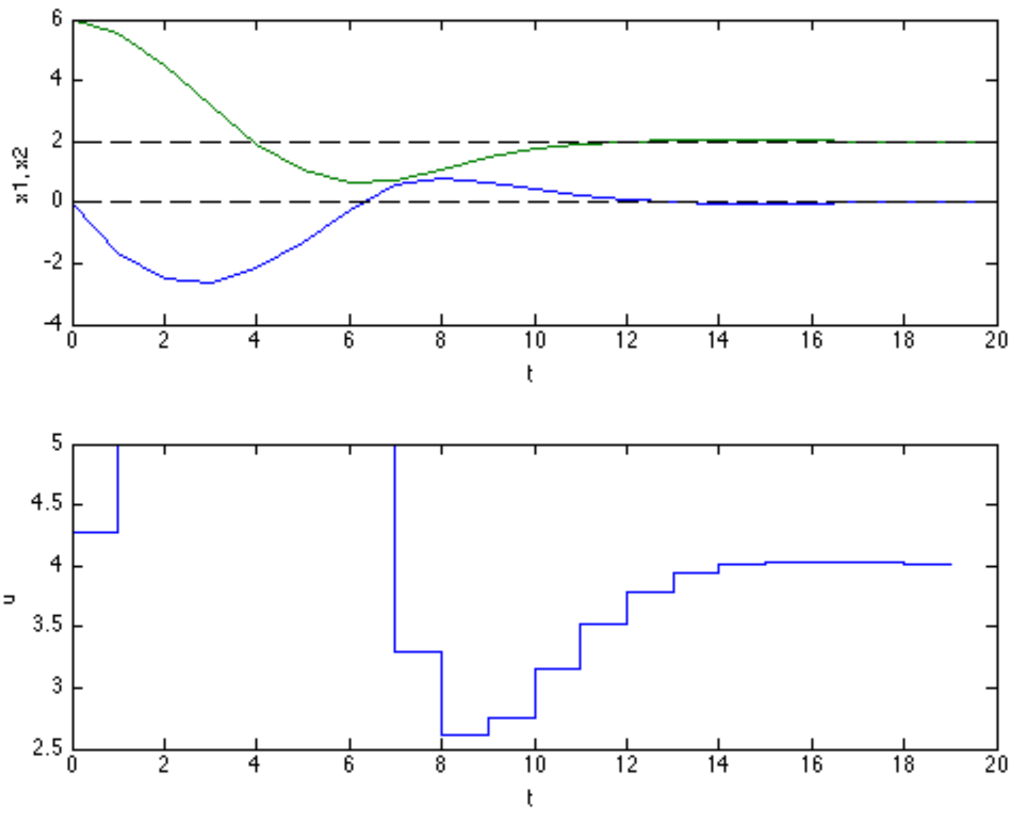


Task 11.12

Offset-free tracking of non-zero references

```
% Calculate the steady-state control input corresponding to xref
uref = B\(eye(2)-A)*xref;
% Enable tracking of non-zero references
model.u.with('reference');
model.u.reference = uref;
% Create a new controller
empc = EMPCController(model, N);
% Simulate the closed loop
loop = ClosedLoop(empc, model);
data = loop.simulate(x0, Nsim)
% Notice no steady-state offset
figure
subplot(2, 1, 1);
plot(0:Nsim, data.X);
hold on
plot(0:Nsim, repmat(xref, 1, Nsim+1), 'k--');
xlabel('t'); ylabel('x1, x2');
subplot(2, 1, 2); stairs(0:Nsim-1, data.U); xlabel('t'); ylabel('u');
```

```
mpt_plcp: 26 regions

data = 

        X: [2x21 double]
        U: [1x20 double]
        Y: [0x20 double]
    cost: [1x20 double]
```



Published with MATLAB® R2013a