Report MPC Programming Excercise

Samuel Nyffenegger and Sebastian Ratz

# General Comments

The figures in this report are the figures produced by your script, just combined into subplots. The curves are the same.

# Nonlinear model and linearization

## 1. Structure of Ac and Bc

Ac and Bc are the partial derivatives of system dynamics with respect to the states and inputs in order to linearize the system dynamics.

F = (kF kF kF kF)\*(u1 u2 u3 u4)’ (1)

xB’’ = uB’ = 1/m(F\*sin(beta)) (2)

yB’’ = vB’ = 1/m(-F\*sin(alpha)) (3)

zB’’ = wB’ = 1/m(cos(alpha)\*cos(beta)\*F - m\*g) (4)

Ac(4,8) = d(2)/d(beta) = F/m\*cos(beta) = g\*cos(beta) = beta small ~ g = 9.81

Ac(5,7) = d(3)/d(alpha) = F/m\*cos(alpha) = -g\*cos(alpha) = alpha small ~ g = -9.81

Bc(6,1) = d(4)/d(u1) = 1/m\*cos(alpha)\*cos(beta)\*dF/du1 = cos(alpha)\*cos(beta)/m\*kF = alpha & beta small ~ kF/m = 3.5

Bc(6,2) … Bc(6,4) analog

Bc(10:13,1:4) = jac(M/I) where M = [M\_alpha M\_beta M\_gamma]’

# First MPC Controller

## 2. Tuning Parameters

Q = diag([0.5,10,10,0.5,0.5,0.5,0.5])

R = 0.1\*eye(n\_inputs)

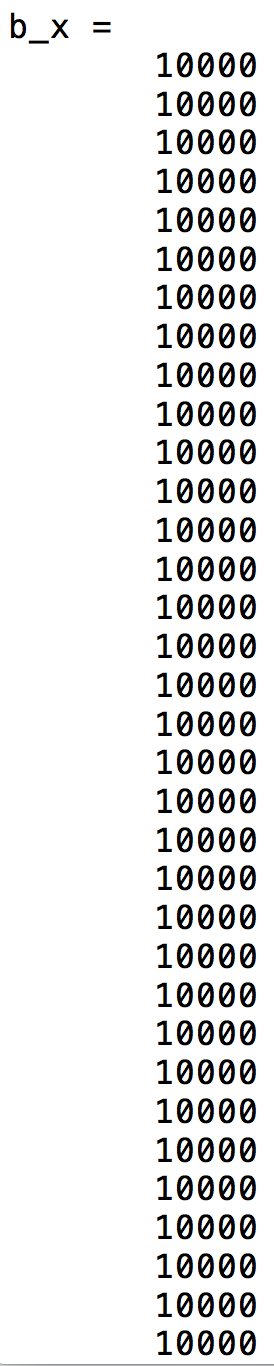
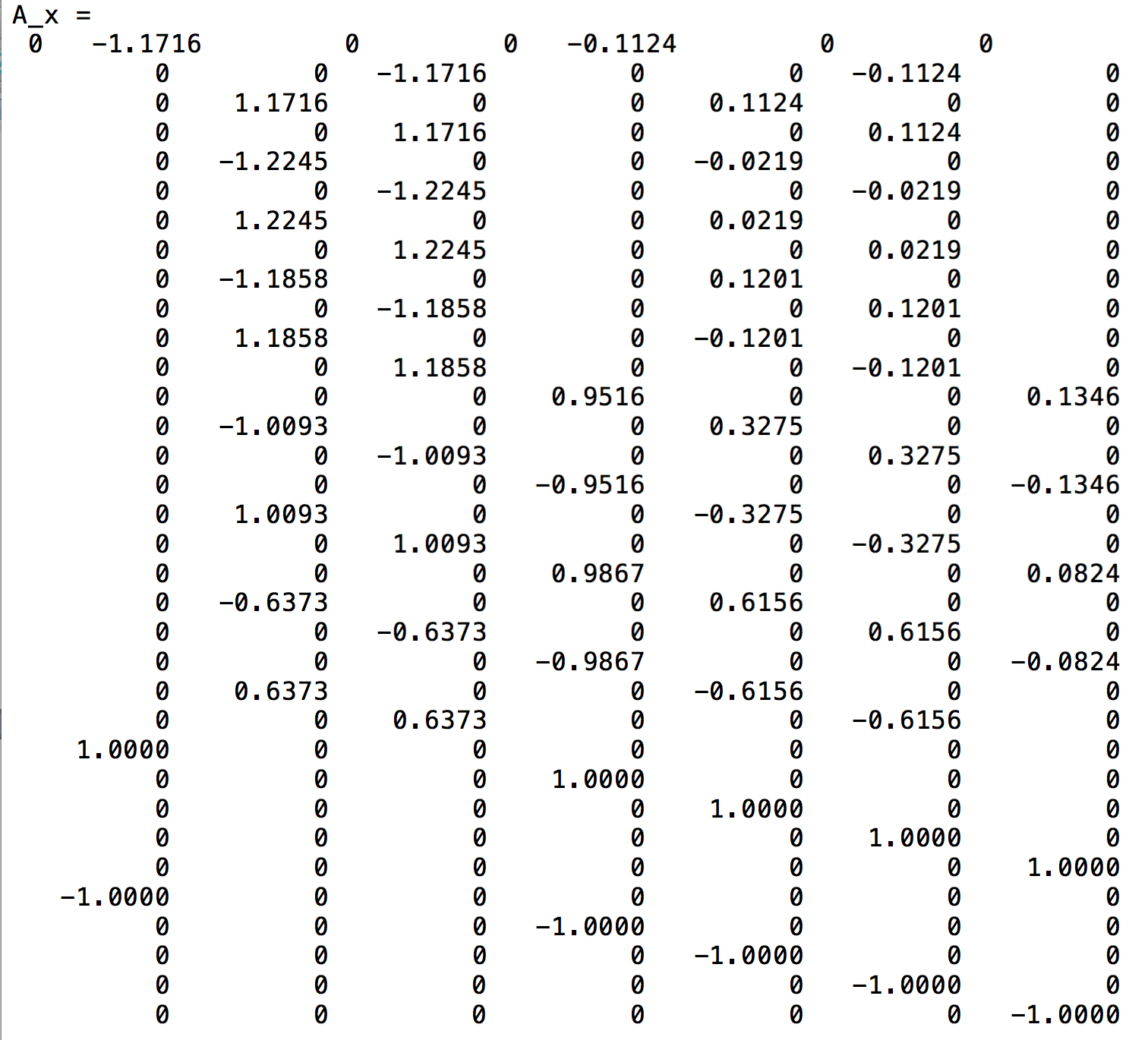
Q was chosen diagonal for simplicity. The parameters in Q were chosen such that the system output fulfills the specified requirements on z’ and alpha, beta. The cost on the states x2 and x3 was chosen more expensive in order to ensure that they are controlled to zero quickly.

R was chosen diagonal as well, with relatively cheap input since there were no restrictions on power consumption or strain on the motors.

The terminal set and terminal cost were chosen corresponding to a LQR controlled system, due to stability guarantees and the relatively simple implementation.

P was thus chosen as the solution of the DARE and the terminal set as the maximum positive invariant set of the autonomous system x(k+1) = (A+BK)x(k).

A\_x and b\_x form the constraints of the polyhedral terminal set A\_x \* x <= b\_x:



## 3. Plots

See next page

## 4. Steady state (xr, ur) as function of r

See matlab code section II.

constraints\_mpc = [];

%target constraints

constraints\_mpc = constraints\_mpc + [C\*xr == r];

constraints\_mpc = constraints\_mpc + [sys.A\*xr+sys.B\*ur == xr];

%delta shifting

constraints\_mpc = constraints\_mpc + [delta\_x(:,1) == xk-xr];

constraints\_mpc = constraints\_mpc + [delta\_u(:,1) == uk-ur];

for i = 2:N

%system constraints

constraints\_mpc = constraints\_mpc + [delta\_x(:,i) == sys.A\*delta\_x(:,i-1)+sys.B\*delta\_u(:,i-1)];

%state constraints

constraints\_mpc = constraints\_mpc + [-z\_dot\_max-xr(1) <= delta\_x(1,i) <= z\_dot\_max-xr(1)];

constraints\_mpc = constraints\_mpc + [-alpha\_beta\_max-xr(2:3) <= delta\_x(2:3,i) <= alpha\_beta\_max-xr(2:3)];

constraints\_mpc = constraints\_mpc + [-alpha\_beta\_dot\_max-xr(5:6) <= delta\_x(5:6,i) <= alpha\_beta\_dot\_max-xr(5:6)];

constraints\_mpc = constraints\_mpc + [-gamma\_dot\_max-xr(7) <= delta\_x(7,i) <= gamma\_dot\_max-xr(7)];

%input constraints

constraints\_mpc = constraints\_mpc + [u\_min-ur <= delta\_u(1:4,i) <= u\_max-ur ];

end

constraints\_mpc = constraints\_mpc + [u\_min-ur <= delta\_u(1:4,1) <= u\_max-ur ];

## 5. Plots constant reference

See this page + 1 (next page)

## 6. Plots of varying reference signal

See this page + 2

# First simulation of nonlinear model

## 7. Plots

See this page + 3 & 4

# Offset free MPC

## 8. L Matrix

L = [Lx; Ld] = [eye(7); eye(7)]

Due to the independence of the disturbances, Lx and Ld were chosen diagonal. Their influence on the states are thus estimated directly and without bias or scaling.

## 9. Plots of constant reference

See next two pages

## 10. Plots of varying reference

See this page + 3 & 4

# Simulation of the nonlinear model

## 11. Plots of step signal

See next 3 pages

## 12. Plots of hexagon reference

See this page + 4 to 6

## 13. Plots infinity shaped reference

See this page + 7 to 9

# Slew rate constraints

## 14. Plots and delta value

Plot see the next two pages.

Delta = 0.19 was chosen in order to avoid performance deterioration or infeasibility problems.

# Soft constraints

## 15. Cost function

The cost function for the soft constraints was chosen as:

delta\_x = sdpvar(n\_states, N,'full');

delta\_u = sdpvar(n\_inputs, N,'full');

delta = 0.1;

v = 1\*[1;1;1;1]';

s = 0.1\*[1;1;1;1]';

objective\_mpc = 0;

for i = 1:N

objective\_mpc = objective\_mpc + delta\_x(:,i)' \* Q \* delta\_x(:,i) + delta\_u(:,i)' \* R \* delta\_u(:,i);

objective\_mpc = objective\_mpc + v\*epsilon(:,i) + s\*epsilon(:,i).^2;

end

## 16. Plots of constant reference

See this page + 3 to 4

# Forces Pro

## 17. Comparison of running times:

**MPC reference tracking (task 5):**

Forces: 0.0081s

Quadprog: 0.0256s

**Offset free MPC (task 9):**

Forces: 0.0082s

Quadprog: 0.0198s