Q1

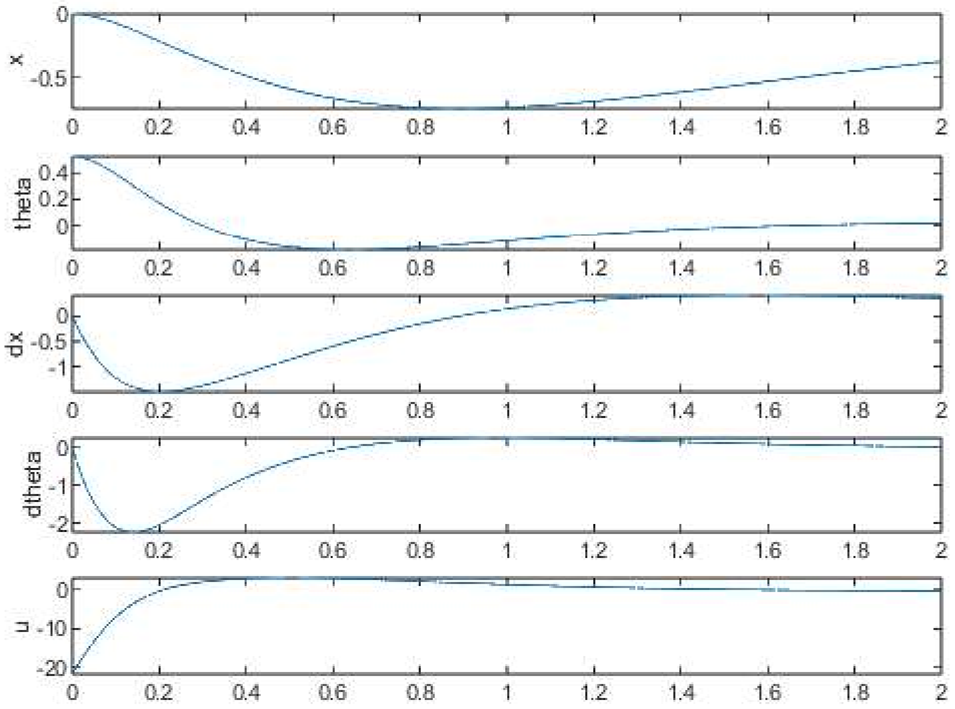
```
clear; clc; close all;

syms x dx ddx theta dtheta ddtheta u

p = [x;theta];
dp = [dx;dtheta];
M=1; m=0.2; L=0.3; g=9.81;

eqn1 = (M+m)*ddx + m*L*sin(theta)*(dtheta)^2 - m*L*cos(theta)*ddtheta == u;
eqn2 = m*L^2*ddtheta - m*L*cos(theta)*ddx - m*g*L*sin(theta) == 0;

S = solve([eqn1,eqn2], [ddx ddtheta]);
ddp = [S.ddx; S.ddtheta];
ddp_fun =  matlabFunction(ddp);

q = [p; dp];
dq = [dp; ddp];

qf = [0;0;0;0];
```

part a

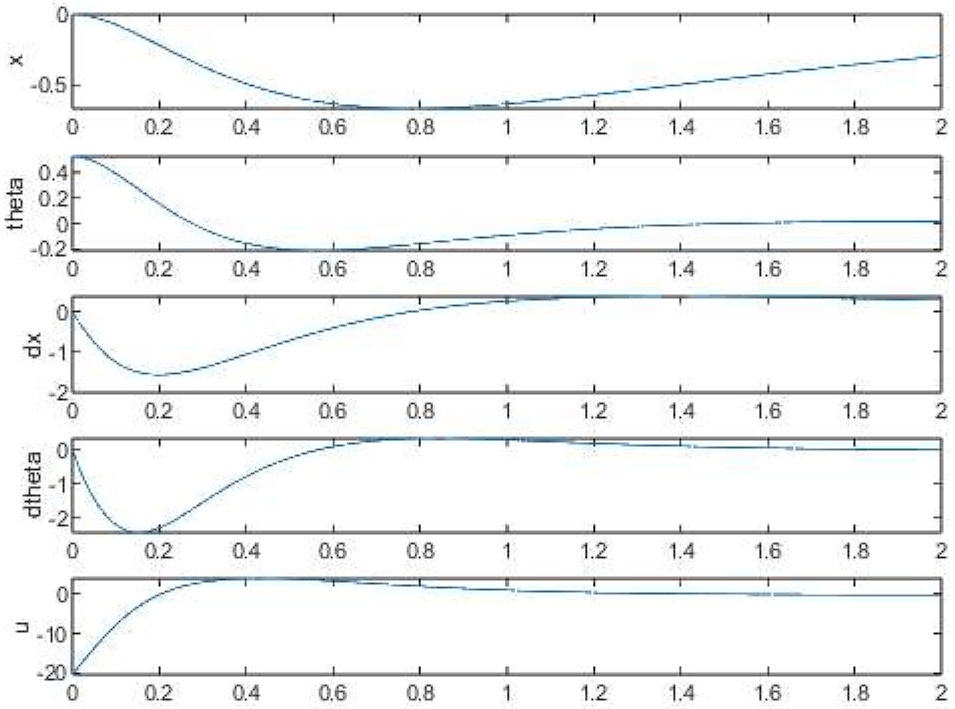
```
A = double(subs(jacobian(dq,q), q, qf));
B = double(subs(jacobian(dq,u), q, qf));

Q_i = diag([10 10 10 10]); R_i = 1;
Q_ii = diag([100 0.01 100 0.01]); R_ii = 10;
Q_iii = diag([100 0.01 100 0.01]); R_iii = 1;
[K_i,P_i,E_i] = lqr(A,B,Q_i,R_i);
[K_ii,P_ii,E_ii] = lqr(A,B,Q_ii,R_ii);
[K_iii,P_iii,E_iii] = lqr(A,B,Q_iii,R_iii);

x0=[0;pi/6;0;0]; % x0 is the intial state of the system
tspan= 0:0.0001:2; % simulation time
[t_i,q_i]=ode45(@(t_i,q_i) sys_dynamics(t_i,q_i,ddp_fun,K_i),tspan,x0);
[t_ii,q_ii]=ode45(@(t_ii,q_ii) sys_dynamics(t_ii,q_ii,ddp_fun,K_ii),tspan,x0);
[t_iii,q_iii]=ode45(@(t_iii,q_iii) sys_dynamics(t_iii,q_iii,ddp_fun,K_iii),tspan,x0);

u_i = -K_i*q_i';
u_ii = -K_ii*q_ii';
u_iii = -K_iii*q_iii';

figure(1)
subplot(5,1,1)
plot(t_i,q_i(:,1))
ylabel('x')
subplot(5,1,2)
plot(t_i,q_i(:,2))
ylabel('theta')
subplot(5,1,3)
```

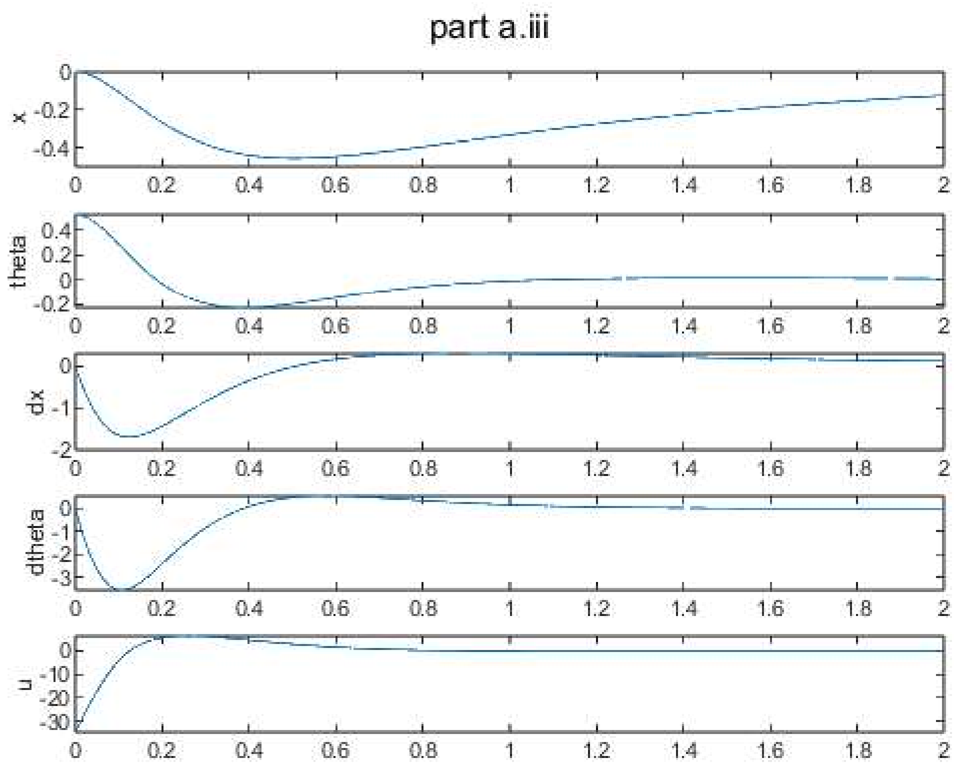
```
plot(t_i,q_i(:,3))
ylabel('dx')
subplot(5,1,4)
plot(t_i,q_i(:,4))
ylabel('dtheta')
subplot(5,1,5)
plot(t_i,u_i)
ylabel('u')
sgtitle("part a.i")

figure(2)
subplot(5,1,1)
plot(t_ii,q_ii(:,1))
ylabel('x')
subplot(5,1,2)
plot(t_ii,q_ii(:,2))
ylabel('theta')
subplot(5,1,3)
plot(t_ii,q_ii(:,3))
ylabel('dx')
subplot(5,1,4)
plot(t_ii,q_ii(:,4))
ylabel('dtheta')
subplot(5,1,5)
plot(t_ii,u_ii)
ylabel('u')
sgtitle("part a.ii")

figure(3)
subplot(5,1,1)
plot(t_iii,q_iii(:,1))
ylabel('x')
subplot(5,1,2)
plot(t_iii,q_iii(:,2))
ylabel('theta')
subplot(5,1,3)
plot(t_iii,q_iii(:,3))
ylabel('dx')
subplot(5,1,4)
plot(t_iii,q_iii(:,4))
ylabel('dtheta')
subplot(5,1,5)
plot(t_iii,u_iii)
ylabel('u')
sgtitle("part a.iii")
```

part a.i



part a.ii

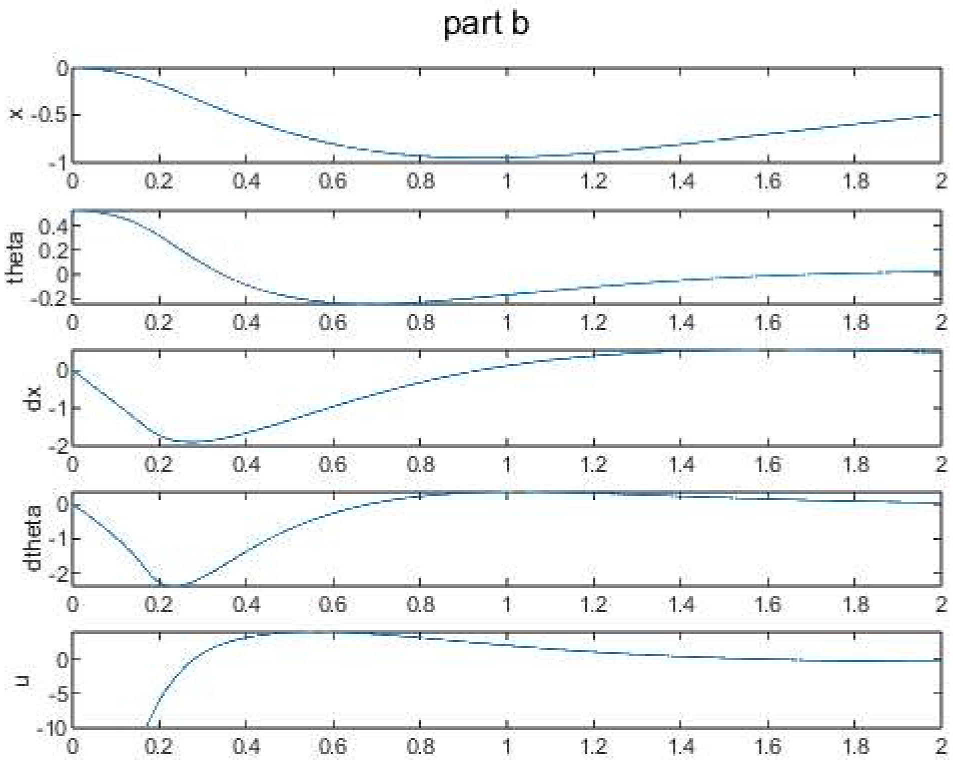




part b

```
x0=[0;pi/6;0;0]; % x0 is the intial state of the system
tspan= 0:0.0001:2; % simulation time
[t,q]=ode45(@(t,q) sys_dynamics_QP(t,q,ddp_fun,K_i),tspan,x0);

u_QP =zeros(1,length(t));
for i=1:length(t)
     u_QP(:,i)= QP_controller(q(i,:)',K_i);
end

figure(4)
subplot(5,1,1)
plot(t,q(:,1))
ylabel('x')
subplot(5,1,2)
plot(t,q(:,2))
ylabel('theta')
subplot(5,1,3)
plot(t,q(:,3))
ylabel('dx')
subplot(5,1,4)
plot(t,q(:,4))
ylabel('dtheta')
subplot(5,1,5)
plot(t,u_QP)
```

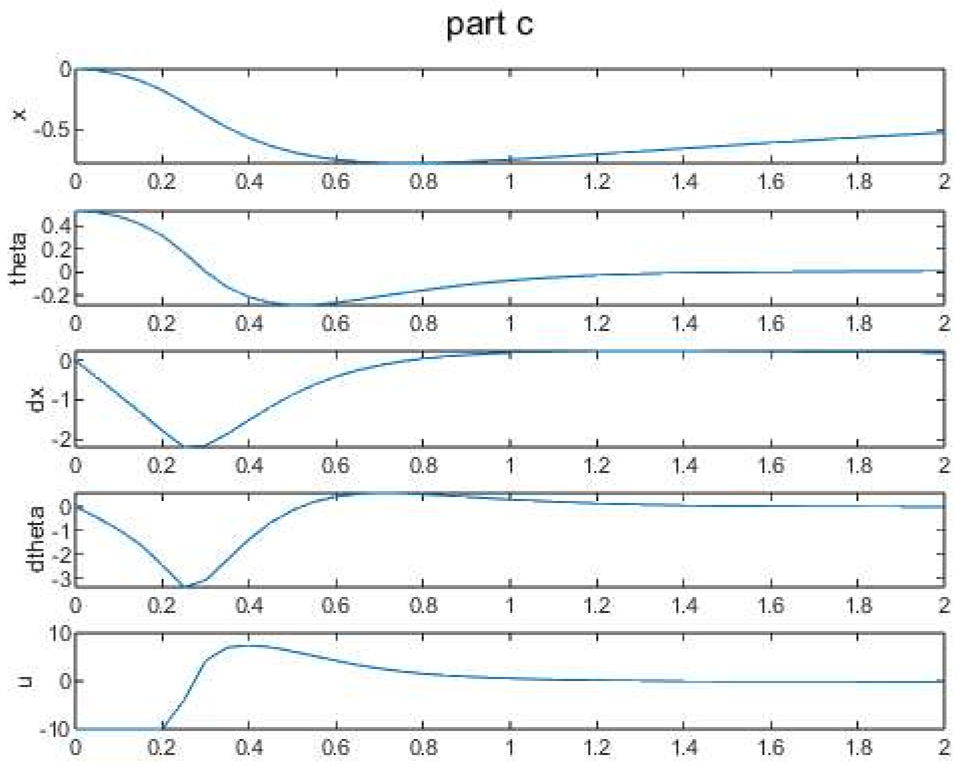
```
ylabel('u')
sgtitle("part b")
```



part c

```
x0=[0;pi/6;0;0]; % x0 is the intial state of the system
tspan= 0:0.05:2; % simulation time
dt = 0.05;
N=20;
Q = diag([100,0.01,100,0.01]);
R = 0.1;
[t,q]=ode45(@(t,q) sys_dynamics_MPC(t,q,ddp_fun,A,B,dt,N,Q,R),tspan,x0);

At = eye(4)+dt*A;
Bt = dt*B;

u_MPC =zeros(1,length(t));
for i=1:length(t)
     u_MPC(:,i)= MPC_controller(q(i,:)',At,Bt,N,Q,R);
end

figure(5)
subplot(5,1,1)
plot(t,q(:,1))
ylabel('x')
subplot(5,1,2)
```

```
plot(t,q(:,2))
ylabel('theta')
subplot(5,1,3)
plot(t,q(:,3))
ylabel('dx')
subplot(5,1,4)
plot(t,q(:,4))
ylabel('dtheta')
subplot(5,1,5)
plot(t,u_MPC)
ylabel('u')
sgtitle("part c")
```



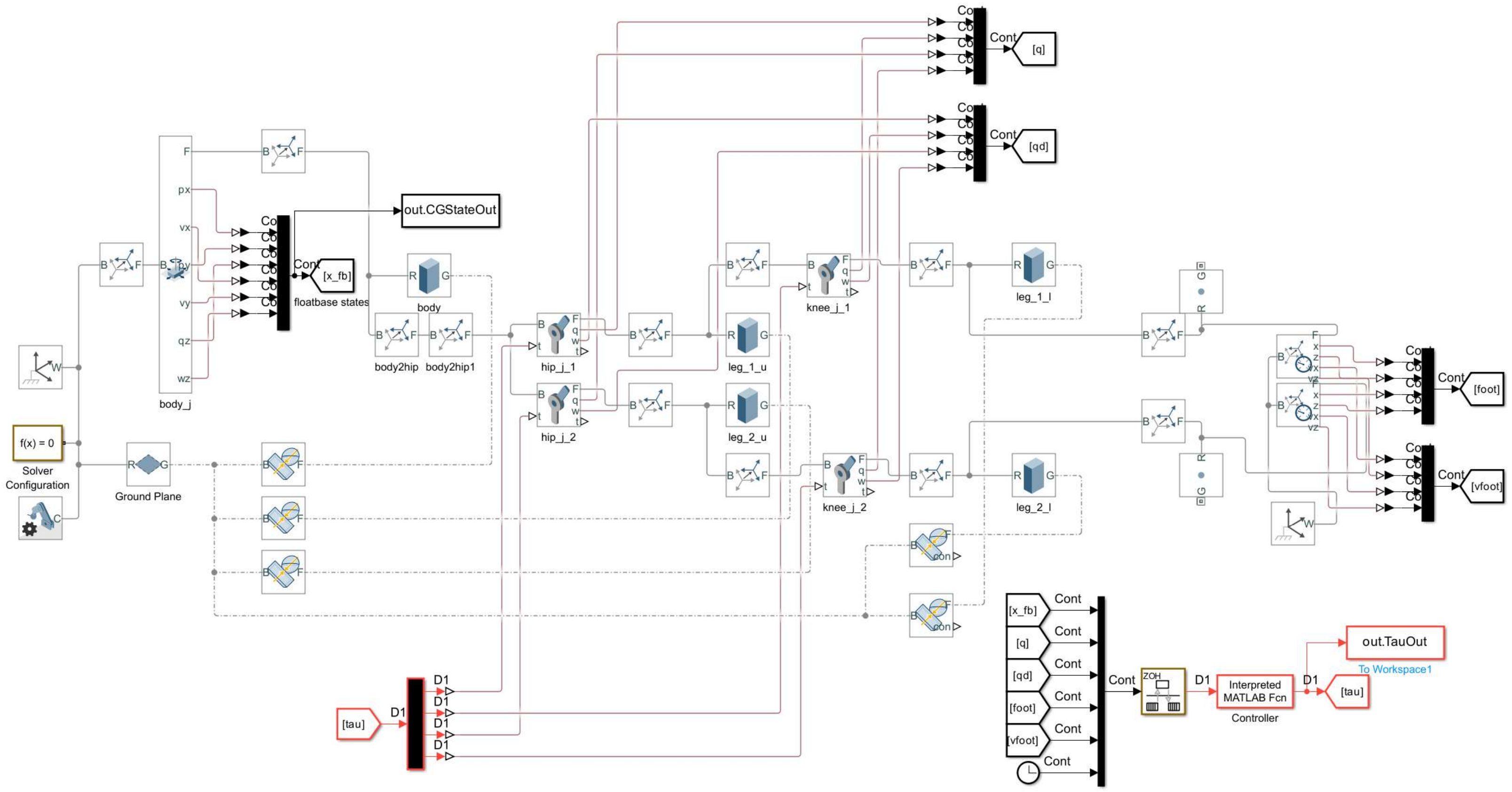
Functions

```
function dq=sys_dynamics(t,q,ddp_fun,K)
dp = q(3:4);
u = -K*q;
ddp = ddp_fun(q(4),q(2),u);
dq = [dp;ddp];
end

function dq=sys_dynamics_QP(t,q,ddp_fun,K)
dp = q(3:4);
u_star = QP_controller(q,K);
ddp = ddp_fun(q(4),q(2),u_star);
```

```
dq = [dp;ddp];
end

function dq=sys_dynamics_MPC(t,q,ddp_fun,A,B,dt,N,Q,R)
dp = q(3:4);
At = eye(4)+dt*A;
Bt = dt*B;
u_star = MPC_controller(q,At,Bt,N,Q,R);
ddp = ddp_fun(q(4),q(2),u_star);
dq = [dp;ddp];
end

function u = QP_controller(q,K)
u_LQR = -K*q;
H = 2;
f = -2*u_LQR;
A = [1;-1]; b =[10;10];
options = optimoptions('quadprog','Display','off');
u = quadprog(H,f,A,b,[],[],[],[],[],options);
end

function u = MPC_controller(q,At,Bt,N,Q,R)
state_num =4;
controller_num = 1;

H=[];
for i = 1:N
H = blkdiag(H,Q);
end

for i = 1:N
H = blkdiag(H,R);
end

f = zeros(N*(state_num+controller_num),1);

Aeq_bl1 = eye(state_num*N);
Aeq_bl2 =[];
for i = 1:N-1
Aeq_bl2 = blkdiag(Aeq_bl2,-At);
end

Aeq_bl2 = [zeros(state_num, state_num*N);
           Aeq_bl2, zeros(state_num*(N-1),state_num)];

Aeq_bl3 =[];
for i = 1:N
Aeq_bl3 = blkdiag(Aeq_bl3,-Bt);
end

Aeq = [Aeq_bl1+Aeq_bl2 Aeq_bl3];
Beq = [At*q; zeros((N-1)*state_num,1)];

Aineq_blk1 = [1 0 0 0;
```

```
             -1 0 0 0;
             0 1 0 0;
             0 -1 0 0];

Aineq_blk2 = [1;-1];

Bineq_blk1 = [0.8;0.8;pi/4;pi/4];
Bineq_blk2 = [10;10];

Aineq =[];
Bineq =[];
for i = 1:N
Aineq = blkdiag(Aineq,Aineq_blk1);
Bineq = [Bineq;Bineq_blk1];
end

for i = 1:N
Aineq = blkdiag(Aineq,Aineq_blk2);
Bineq = [Bineq;Bineq_blk2];
end

options = optimoptions('quadprog','Display','off');
X_star = quadprog(H,f,Aineq,Bineq,Aeq,Beq,[],[],[],options);
u = X_star(N*state_num+1);
end
```

Published with MATLAB® R2023b



body_j
F
px
vx
vy
qz
wz
floatbase states
[x_fb]
out.CGStateOut
body
body2hip
body2hip1
hip_j_1
hip_j_2
leg_1_u
leg_2_u
knee_j_1
knee_j_2
leg_1_l
leg_2_l
[q]
[qd]
[foot]
[vfoot]
Solver Configuration
f(x) = 0
Ground Plane
[tau]
D1
[x_fb]
[q]
[qd]
[foot]
[vfoot]
Cont
ZOH
Interpreted MATLAB Fcn
Controller
out.TauOut
To Workspace1
[tau]

Table of Contents

```
function tau = bipedControlWrapper(uin)

global f_ff
%state interpretation
x_fb = uin(1:6); %[x,y,theta,dx,dy,dtheta]
q = uin(7:10); %[hip1,knee1, hip2, knee2]
qd = uin(11:14);
foot = uin(15:22);
t = uin(end);
counter = ceil(uin(end)*1000);

%toggle print state information
% printStates(uin)

%parameter definition:
para.m = 8;
para.I = 0.0567;
para.g = 9.81;
para.mu = 0.5;

%control goal and parameters
ctrl.x_cmd = [0; 0.5; 0; 0;0;0];
ctrl.fmin = 1;
ctrl.fmax = 250;

%swing pd gains (unused for balancing)
ctrl.pd.kp = diag([200,300]);
ctrl.pd.kd = eye(2)*5;

%qp gains and weights
ctrl.qp.kp = diag([30, 30, 30]);
ctrl.qp.kd = diag([10, 10, 15]);
ctrl.qp.S = diag([1,1,1]);
ctrl.qp.alpha = 0.0001;

%mpc parameter, gains and weights
ctrl.mpc.dt = 0.04; % Sampling Rate
ctrl.mpc.h = 10; % Number of Time Horizons
ctrl.mpc.Q = diag([200 200 800 10 10 30 1]);
ctrl.mpc.R = eye(4)*0.0001;
ctrl.mpc.hyperSampleRate = 10;
ctrl.mpc.swingHeight = 0.1; % Unused for Balancing

% contact generator (Controls gait)
```

```
contact = ones(ctrl.mpc.h, 2); %(all 1s for Balancing)
%contact = getContactSequence(uin, ctrl) % stepping (NOTE: ONLY FOR MPC!)
```

force control (uncomment whichever controller you want)

QP controller f_ff = runQP(uin, para, ctrl); % this is QP controller MPC controller:

```
    if rem(counter, ctrl.mpc.dt*1000/ctrl.mpc.hyperSampleRate) == 0
        f_ff = runMPC_condense(uin, para, ctrl, contact); % this is MPC
controller
    end

% jacobian mapping (Force to Torque)
tau = lowLevelControl(f_ff, uin, ctrl, contact);

%Disable the controller entirely if we are off the ground (only for
%balancing)
if all(contact)
    if foot(2) > 0.01
        tau(1:2) = zeros(2,1);
    end
    if foot(4) > 0.01
        tau(3:4) = zeros(2,1);
    end
end

end
```

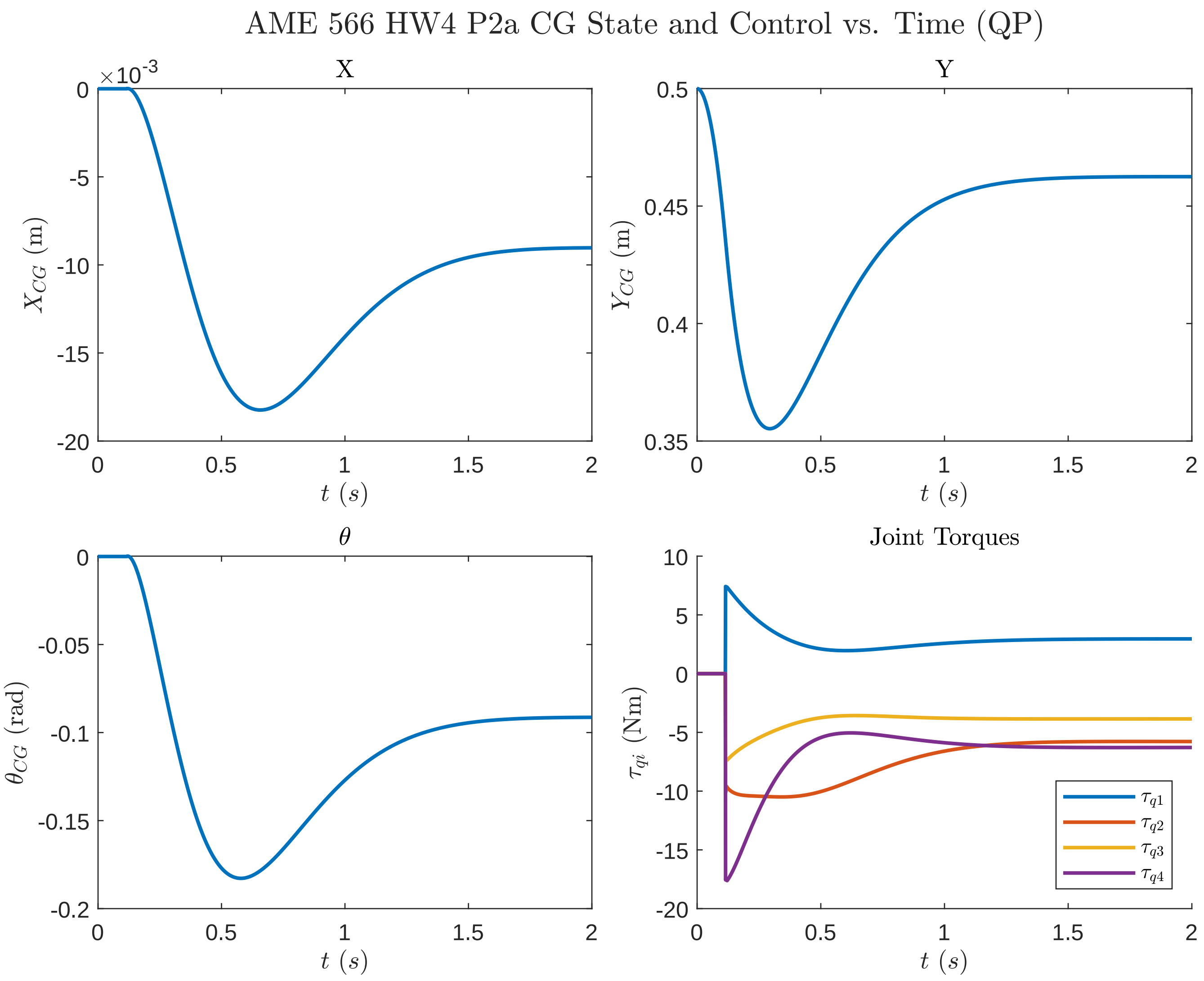
QP control (running 1khz)

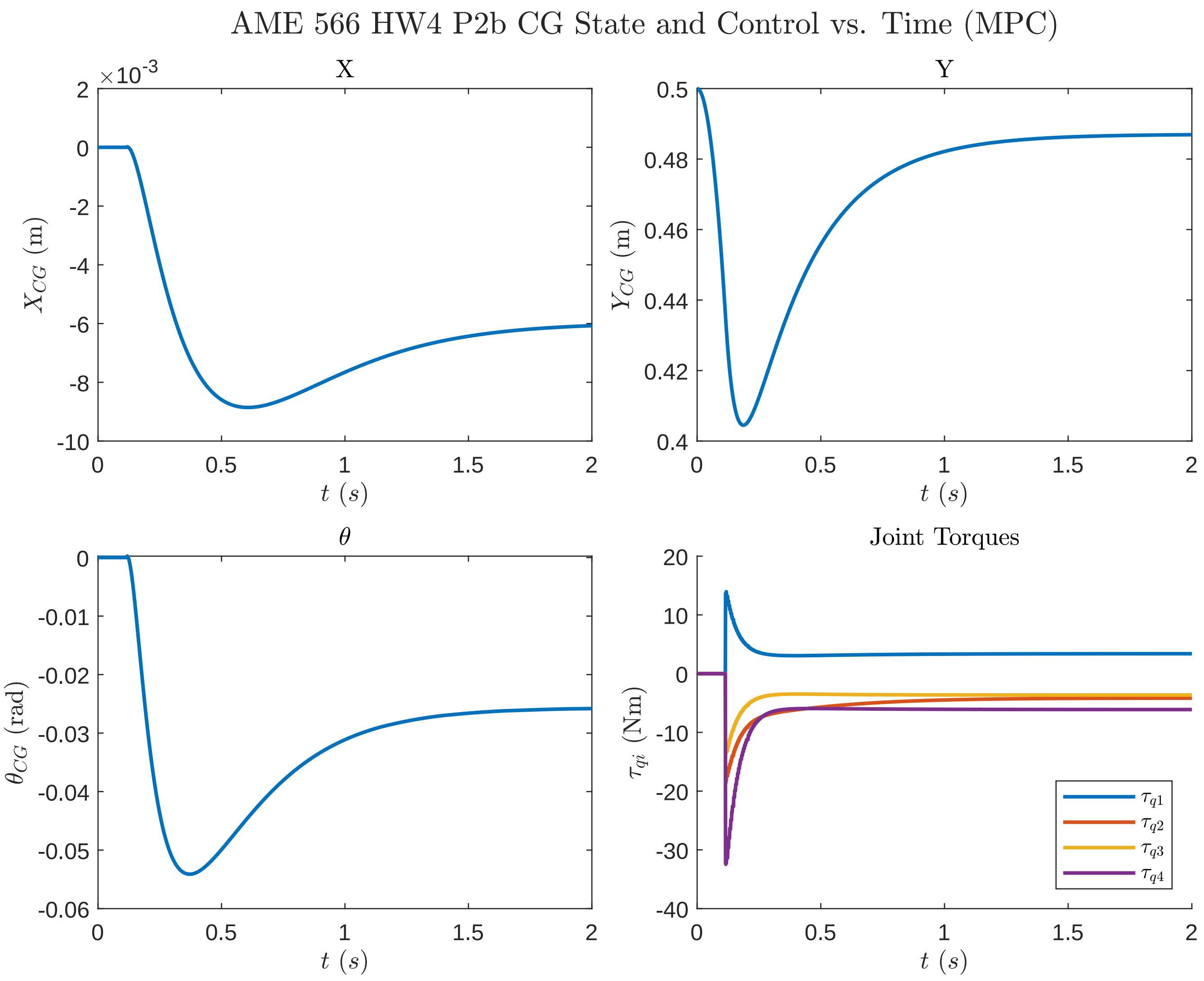
```
function f_ff = runQP(uin, para, ctrl)
    % Extract the States from uin
    x_fb = uin(1:6); %[x,y,theta,dx,dy,dtheta]
    q = uin(7:10); %[hip1,knee1, hip2, knee2]
    qd = uin(11:14);
    foot = uin(15:18);
    t = uin(end);
    % create the desired pd control law
    ddx_des = ctrl.qp.kp* (ctrl.x_cmd(1:3) - x_fb(1:3)) + ...
                  ctrl.qp.kd* (ctrl.x_cmd(4:6) - x_fb(4:6));
    %Calculate vectors from foot to CG
    r1 = foot(1:2) - x_fb(1:2);
    r2 = foot(3:4) - x_fb(1:2);
    % Build A, b Matrices
    A = [eye(2), eye(2);
         -r1(2), r1(1), -r2(2), r2(1)];
    bd = [para.m*(ddx_des(1:2) + [0;para.g]);
          para.I * ddx_des(3)];
    %construct qp problem
    H = 2 * (A'*ctrl.qp.S*A + eye(4)*ctrl.qp.alpha);
    % f = -2*A'*ctrl.qp.S*bd;
    f = -2*bd'*ctrl.qp.S*A;
    %friction constraint
    A_mu = kron(eye(2),[1 -para.mu; -1 -para.mu]);
    b_mu = zeros(4,1);
    %force saturation constraint
    A_f = kron(eye(2),[0 1; 0 -1]);
    b_f = repmat([ctrl.fmax; -ctrl.fmin],2,1);
    %concatenation:
    A = [A_mu; A_f];
    b = [b_mu; b_f];
    %solve:
    options = optimoptions('quadprog','Display', 'off');
    f_ff = quadprog(H,f,A,b,[],[],[],[],[], options);
end
```

MPC control

```
function f_ff = runMPC_condense(uin, para, ctrl, contact) % the condensed
version
    % Extract states from uin
    x_fb = uin(1:6); %[x,y,theta,dx,dy,dtheta]
    q = uin(7:10); %[hip1,knee1, hip2, knee2]
    qd = uin(11:14);
    foot = uin(15:18);
    t = uin(end);

    % MPC parameters
    dt = ctrl.mpc.dt;
    h = ctrl.mpc.h;
    numVar = 7;
    numCtrl = 4;
    % Pull reference trajectory (constant for this case)
    x_ref = getReferenceTrajectory(uin, para, ctrl);
    % MPC dynamics:
    A_hat=repmat({zeros(numVar,numCtrl)}, h, 1);
    B_hat=repmat({zeros(numVar,numVar)}, h, 1);
    for k = 1:h
        r1 = foot(1:2) - x_fb(1:2); % alternatively x_fb can be replaced
with x_ref for dynamic cases
        r2 = foot(3:4) - x_fb(1:2);
        %Linearized Dynamics
        Ac = [zeros(3) eye(3) zeros(3,1);
              zeros(3,numVar-1) [0 ; -para.g; 0];
              zeros(1,numVar)];
        Bc = [zeros(3, numCtrl);
              eye(2)/para.m,  eye(2)/para.m;
              -r1(2)/para.I, r1(1)/para.I, -r2(2)/para.I, r2(1)/para.I;
              zeros(1, numCtrl)];

        % Discrete-Time A and B matrices
        B_hat{k} = Bc * dt;
        A_hat{k} = eye(numVar) + Ac * dt;
    end
    % construct condensed MPC-QP problem
    y = reshape(x_ref,[numVar*h 1]);
    Aqp=repmat({zeros(numVar,numVar)},h,1);
    %Aqp
    Aqp{1}=A_hat{1};
    for i=2:h
        Aqp{i}=Aqp{i-1}*A_hat{i};
    end
    Aqp=cell2mat(Aqp);
    %Bqp
    Bqp=repmat({zeros(numVar,numCtrl)},h,h);
    for i=1:h
        Bqp{i,i}=B_hat{i};
        for j=1:h-1
            Bqp{i,j}=A_hat{i}^(i-j)*B_hat{j};
        end
    end
    for i=1:h-1
        for j=i+1:h
            Bqp{i,j}=zeros(numVar,numCtrl);
        end
    end
    Bqp=cell2mat(Bqp);
    %condensation:
    L = kron(eye(h),ctrl.mpc.Q);
    K = kron(eye(h),ctrl.mpc.R);
    H = 2*(Bqp'*L*Bqp+K);
    f = 2*Bqp'*L*(Aqp*[uin(1:numVar-1);1]-y);

    % 1.friction pyramid:
    A_mu = kron(eye(2),[1 -para.mu; -1 -para.mu]);
    A_mu_h = [kron(eye(h), A_mu) ];
    b_mu = repmat([0; 0; 0; 0], h,1);
    % 2.force limit:
    A_f = kron(eye(2),[0 1; 0 -1]);
    A_f_h = [kron(eye(h), A_f) ];
    b_f = [];
    for i = 1:h
        b_f = [b_f; [ctrl.fmax; -ctrl.fmin]*contact(i,1);
                    [ctrl.fmax; -ctrl.fmin]*contact(i,2)];
    end
    % concatenation:
    A = [A_mu_h; A_f_h];
    b = [b_mu; b_f];

    %solve:
    options = optimoptions('quadprog','Display', 'off');
    sol = quadprog(H,f,A,b,[],[],[],[],[],options);
    f_ff = sol(1:numCtrl);
end

function tau = lowLevelControl(f_ff, uin, ctrl, contact)
    % Do Force to Torque Mapping, and control swing phases (if walking)
    q = uin(7:10);
    Jc1 = getJacobian(q(1),q(2));
    Jc2 = getJacobian(q(3),q(4));
    R = theta2Rotm(uin(3));
    if contact(1,1) && contact(1,2)
        tau = [Jc1'*R'*-f_ff(1:2); Jc2'*R'*-f_ff(3:4)];
    elseif contact(1,1) == 0 && contact(1,2) == 1
        f_swing = swingLegControl(uin, uin(15:16), uin(19:20), ctrl);
        tau = [Jc1'*R'*f_swing; Jc2'*R'*-f_ff(3:4)];
    elseif contact(1,2) == 0 && contact(1,1) == 1
        f_swing = swingLegControl(uin, uin(17:18),uin(21:22), ctrl);
        tau = [Jc1'*R'*-f_ff(1:2); Jc2'*R'*f_swing];
    end

end

function x_ref = getReferenceTrajectory(uin, para, ctrl)
    %Generate our target trajectory for use in MPC
    dt = ctrl.mpc.dt;
    h = ctrl.mpc.h;
    x_ref = repmat([ctrl.x_cmd;1],1,h);
    % x_ref(:,1) = [uin(1:6);1];
    t = uin(end);
    for i = 1:3
        for k = 2:h
            if ctrl.x_cmd(i+3) ~= 0
                x_ref(i,k) = uin(i)+ctrl.x_cmd(i+3)*((k-1)*dt);
            else
                x_ref(i,k) = ctrl.x_cmd(i);
            end
        end
    end
end

function f_swing = swingLegControl(uin, foot, vfoot, ctrl)
    t = rem(uin(end), ctrl.mpc.dt*ctrl.mpc.h/2);
    foot_des_x = uin(1) + uin(4)*1/2*ctrl.mpc.h/2*ctrl.mpc.dt;
    foot_des_y = ctrl.mpc.swingHeight*sin(pi*t/(ctrl.mpc.dt*ctrl.mpc.h/2));
    f_swing = ctrl.pd.kp*([foot_des_x; foot_des_y] - foot) + ...
                ctrl.pd.kd*(0 - vfoot);
end

function J = getJacobian(q0,q1)
    %Generate the Jacobian for a foot based on joint angles
    l = 0.22;
    J = [l*cos(q0)+l*cos(q0+q1), l*cos(q0+q1);
         l*sin(q0)+l*sin(q0+q1), l*sin(q0+q1)];
end

function contact = getContactSequence(uin, ctrl)
    %Generate a contact schedule based on how far in the phase we are
    t = uin(end);
    contact = [1,1,1,1,1, 0,0,0,0,0;
                0,0,0,0,0, 1,1,1,1,1]';
    phase = floor(t/ctrl.mpc.dt);
    k = rem(phase,ctrl.mpc.h);
    if k == 1
        contact = [1,1,1,1, 0,0,0,0,0,1;
                   0,0,0,0, 1,1,1,1,1,0]';
    elseif k == 2
        contact = [1,1,1, 0,0,0,0,0, 1,1;
                   0,0,0, 1,1,1,1,1, 0,0]';
    elseif k == 3
        contact = [1,1, 0,0,0,0,0, 1,1,1;
                   0,0, 1,1,1,1,1, 0,0,0]';
    elseif k == 4
        contact = [1, 0,0,0,0,0, 1,1,1,1;
                   0, 1,1,1,1,1, 0,0,0,0]';
    elseif k == 5
        contact = [0,0,0,0,0, 1,1,1,1,1;
                    1,1,1,1,1, 0,0,0,0,0]';
    elseif k == 6
        contact = [0,0,0,0, 1,1,1,1,1, 0;
                    1,1,1,1, 0,0,0,0,0,1]';
    elseif k == 7
        contact = [0,0,0, 1,1,1,1,1, 0,0;
                    1,1,1, 0,0,0,0,0, 1,1]';
    elseif k == 8
        contact = [0,0, 1,1,1,1,1, 0,0,0;
                    1,1, 0,0,0,0,0, 1,1,1]';
    elseif k == 9
        contact = [0, 1,1,1,1,1, 0,0,0,0;
                    1, 0,0,0,0,0, 1,1,1,1]';
    end
end

function R2D = theta2Rotm(theta)
    R2D = [cos(theta), -sin(theta); sin(theta), cos(theta)];
end

function [] = printStates(uin)
    x_fb = uin(1:6)
    q = uin(7:10)
    qd = uin(11:14)
    foot = uin(15:18)
    t = uin(end)
end
```

Published with MATLAB® R2023b


AME 566 HW4 P2a CG State and Control vs. Time (QP)
X
$\times 10^{-3}$
X_{CG} (m)
t (s)
Y
Y_{CG} (m)
t (s)
θ
θ_{CG} (rad)
t (s)
Joint Torques
τ_{qi} (Nm)
t (s)
τ_{q1}
τ_{q2}
τ_{q3}
τ_{q4}


AME 566 HW4 P2b CG State and Control vs. Time (MPC)
X
$\times 10^{-3}$
X_{CG} (m)
Y
Y_{CG} (m)
θ
θ_{CG} (rad)
Joint Torques
τ_{qi} (Nm)
τ_{q1}
τ_{q2}
τ_{q3}
τ_{q4}
t (s)

Problem 2 Videos

QP Video:

https://drive.google.com/file/d/1rQNCiahqSB-hh892yap2okY7AZVkeIBa/view?usp=drive_link

MPC Video:

https://drive.google.com/file/d/1nTIQiEBZ_rZSnyyPdmmXEgAaFRw2AevL/view?usp=drive_link