```
clc;
clear;
% model parameters
S.m1 = 1;
S.m2 = 1;
S.l1 = .5;
S.l2 = .5;
S.lc1 = .25;
S.lc2 = .25;
S.I1 = S.m1*S.l1/12;
S.I2 = S.m2*S.l2/12;
S.g = 9.8;

% cost function parameters
S.Q = .0* diag([5, 5, 1, 1]);
S.R = .1 * diag([1, 5]);
S.Qf = diag([5, 5, 1, 1]);

S.Qs = sqrt(S.Q);
S.Rs = sqrt(S.R);
S.Qfs = sqrt(S.Qf);

% time horizon and number of segments
tf = 4;
N = 128;
S.h = tf/N; %time-step

% initial state

% start from the origin with some initial velicity
x0 = [0; 0; 1; 0];
S.x0 = x0;

% controls
us = zeros(2, N);

% states
xs = zeros(4, N+1);
xs(:,1) = x0;

for k=1:N
    xs(:, k+1) = arm_f(k, xs(:,k), us(:,k), S);
end

us = lsqnonlin(@(us)arm_cost(us, S), us);
%update trajectory:
xs = sys_traj(x0,us,S);

y = arm_cost(us,S);
J = (1/2)*(y'*y);
disp(['cost : ' num2str(J)])
subplot(1,2,1);
```

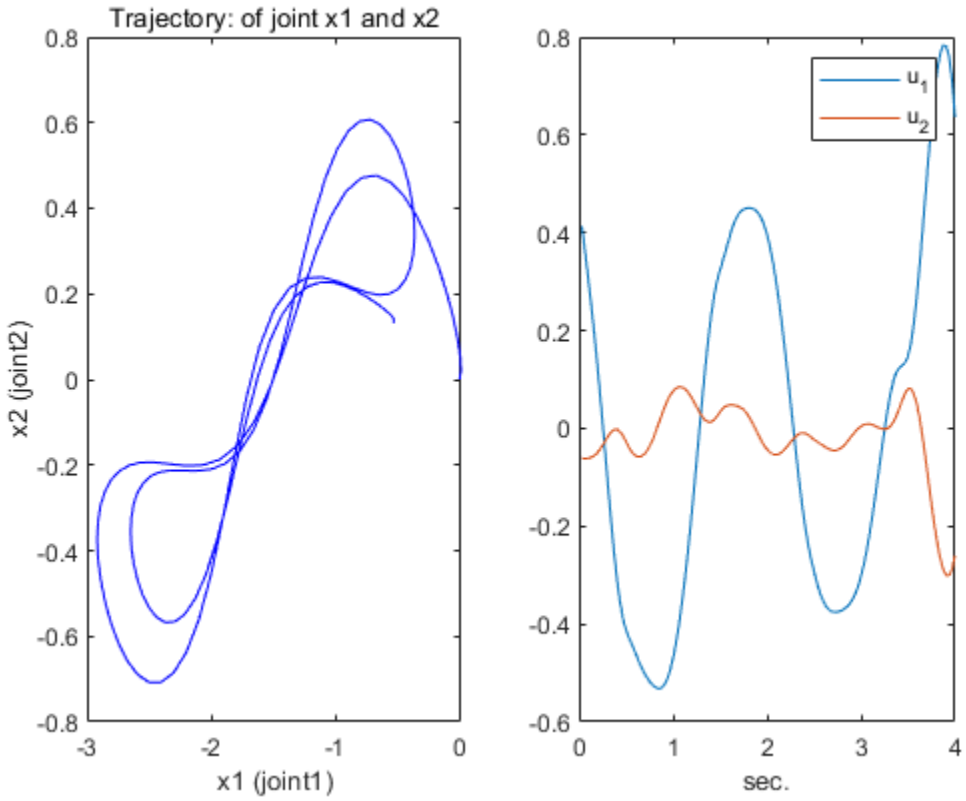
```
plot(xs(1,:), xs(2,:), '-b')
xlabel('x1 (joint1)');
ylabel('x2 (joint2)');
title(sprintf('Trajectory: of joint x1 and x2'));
subplot(1,2,2)
plot(S.h:S.h:tf, us(1,:),S.h:S.h:tf, us(2,:));
xlabel('sec.')
legend('u_1','u_2')

function xs = sys_traj(x0, us, S)
    N = size(us, 2);
    xs(:,1) = x0;
    for k=1:N
        xs(:, k+1) = arm_f(k, xs(:,k), us(:,k), S);
    end
end

function x = arm_cost(us, S)

    us = reshape(us, 2, 128);
    xs = sys_traj(S.x0, us, S);
    N=size(us,2);

    x=[];
    for k=1:N
        x = [x; S.Rs*us(:,k)];
    end

    for k=2:N
        x = [x; S.Qs*xs(:,k)];
    end
    x = [x; S.Qfs*xs(:,N+1)];
end


function [x, A, B] = arm_f(k, x, u, S)
% S.h : time-step
q = x(1:2);
v = x(3:4);
c1 = cos(q(1));
c2 = cos(q(2));
s2 = sin(q(2));
c12 = cos(q(1) + q(2));
% coriolis matrix
C = -S.m2*S.l1*S.lc2*s2*[v(2), v(1) + v(2);
    -v(1), 0] + diag([.2;.2]);

% mass elements
m11 = S.m1*S.lc1^2 + S.m2*(S.l1^2 + S.lc2^2 +2*S.l1*S.lc2*c2) + S.I1 + S.I2;

m12 = S.m2*(S.lc2^2 + S.l1*S.lc2*c2) + S.I2;

m22 = S.m2*S.lc2^2 + S.I2;
```

```
% mass matrix
M = [m11, m12;
    m12, m22];
% gravity vector
fg = [(S.m1*S.lc1 + S.m2*S.l1)*S.g*c1 + S.m2*S.lc2*S.g*c12;
          S.m2*S.lc2*S.g*c12];
% acceleration
a = M\(u - C*v - fg);
v = v + S.h*a;
x = [q + S.h*v; v];
% leave empty to use finite difference approximation
A= [];
B= [];
end
```

Local minimum possible.

lsqnonlin stopped because the final change in the sum of squares relative to its initial value is less than the value of the function tolerance.

cost : 1.9528



Published with MATLAB® R2021b