Faculty of Engineering and Applied Science
University of Regina
ENIN-833
Assignment #3 (out: February 24th, due: March 5th)

Full Name: Behnam Moradi

Student ID: 200 433 555

[100 marks] Problem #1: Figure 1 shows a schematic of an inverted pendulum on a cart. The cart is driven by a DC motor, attached to the wheels, whose specs are provided as an attachment.

(1) Derive the state-space model of the system, including motor's dynamics.
(2) Design an MPC controller for keeping the pendulum in an upward configuration, and tracking a position setpoint on the cart, simultaneously (i.e., an inverted pendulum control problem). Set constraints on the pendulum angle and cart position as: $|\theta| \ll 30\ degrees, and\ |X| \ll 0.5\ meters$
(3) Repeat (2) when adding a full-state observer. Design proper observer gains and explain your design process.

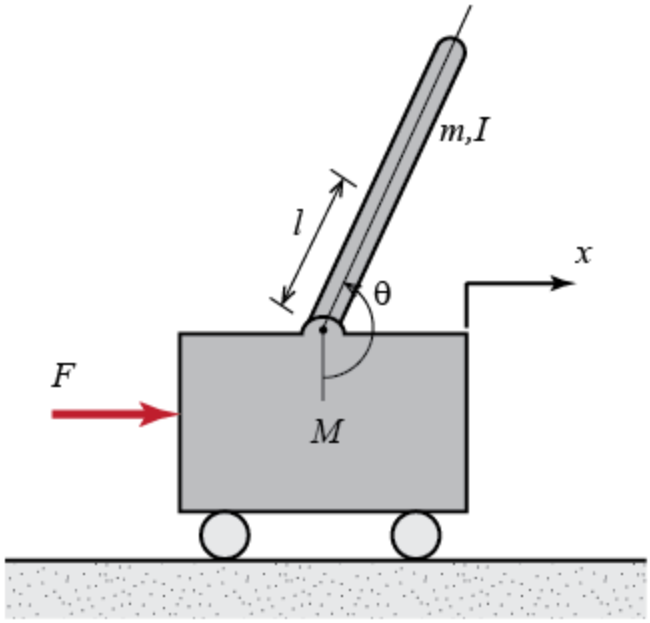


Figure 1: Pendulum on a cart – a cart-pole control problem.

Physical parameters of the system:

(M)	mass of the cart	0.5 kg
(m)	mass of the pendulum	0.2 kg
(b)	coefficient of friction for cart	0.1 N/m/sec
(l)	length to pendulum center of mass	0.3 m
(I)	mass moment of inertia of the pendulum	0.006 kg.m^2
(r)	radius of the wheels	0.10 m

Motor specs:

PittmanExpress™

GM9413-2

Lo-Cog® DC Gearmotor

Assembly Data	Symbol	Units	Value	
Reference Voltage	E	V	12	
No-Load Speed	S_{NL}	rpm (rad/s)	142	(14.9)
Continuous Torque (Max.)[1]	T_C	oz-in (N-m)	45	(3.2E-01)
Peak Torque (Stall)[2]	T_{PK}	oz-in (N-m)	109	(7.7E-01)
Weight	W_M	oz (g)	15.2	(432)
Motor Data				
Torque Constant	K_T	oz-in/A (N-m/A)	5.60	(3.95E-02)
Back-EMF Constant	K_E	V/krpm (V/rad/s)	4.14	(3.95E-02)
Resistance	R_T	õe	8.33	
Inductance	L	mH	6.17	
No-Load Current	I_{NL}	A	0.10	
Peak Current (Stall)[2]	I_P	A	1.44	
Motor Constant	K_M	oz-in/ÕW (N-m/ÕW)	1.94	(1.37E-02)
Friction Torque	T_F	oz-in (N-m)	0.50	(3.5E-03)
Rotor Inertia	J_M	oz-in-s^2 (kg-m^2)	3.9E-04	(2.8E-06)
Electrical Time Constant	t_E	ms	0.74	
Mechanical Time Constant	t_M	ms	14.7	
Viscous Damping	D	oz-in/krpm (N-m-s)	0.011	(7.6E-07)
Damping Constant	K_D	oz-in/krpm (N-m-s)	2.8	(1.9E-04)
Maximum Winding Temperature	$\urcorner_{MAX}$	ºF (ºC)	311	(155)
Thermal Impedance	R_{TH}	ºF/watt (ºC/watt)	66.4	(19.1)
Thermal Time Constant	t_{TH}	min	11.1	
Gearbox Data				
Reduction Ratio			19.7	
Efficiency			0.73	
Maximum Allowable Torque		oz-in (N-m)	175	(1.24)
Encoder Data				

1 - Specified at max. winding temperature at 25ºC ambient without heat sink. 2 - Theoretical values supplied for reference only.

Included Features

2-Pole Stator
Ceramic Magnets
Heavy-Guage Steel Housing
7-Slot Armature
Silicon Steel Laminations
Stainless Steel Shaft
Copper-Graphite Brushes
Diamond Turned Commutator
Motor Sleeve Bearings
Output Sleeve Bearing
Standard Gears

Customization Options

Alternate Winding
Sleeve or Ball Bearings
Modified Output Shaft
Custom Cable Assembly
Special Brushes
EMI/RFI Suppression
Alternate Gear Material
Special Lubricant
Optional Encoder
Fail-Safe Brake

1 Model Predictive Controller: Design and Implementation

Basically, the main aim of this section is to design and implement an MPC controller to keep an inverted pendulum in upward position, while the whole system is subjected to physical constraint as follows:

- The cart is allowed to operate in |x| < 0.5;
- The Pendulum is allowed to oscillate in |theta| <pi/6;
- The maximum allowed DC motor voltage is 12 volts: Vdc <= 12;

1.1 Dynamic Model of the System

The main aim of this sub-section is to extract the dynamic response of the system, a DC motor included. So basically, the input of the system will change from applied force to the applied DC voltage.

According to the systems components, we have three governing equations: The first and second one are related to Cart-pendulum system. And the third one is related to the dynamic of the DC motor.

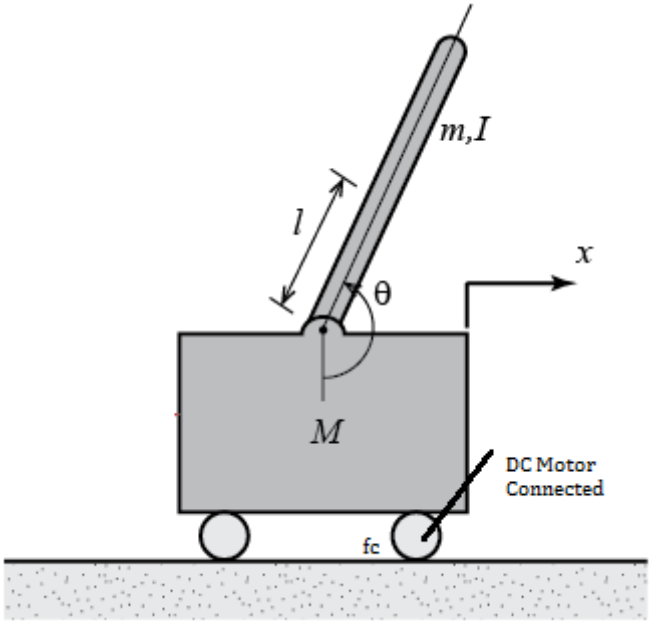


Figure 1: Cart-Pendulum system, DC motor included

$$\left.\begin{aligned} &\sum F = M\ddot{x} \\ &F - C_x - b\dot{x} = M\ddot{x} \\ &M\ddot{x} - ml(\sin(\theta)\dot{\theta}^2 + \cos(\theta)\ddot{\theta}) = C_x \end{aligned}\right\rangle F - (m+M)\ddot{x} - b\dot{x} - ml(\sin(\theta)\dot{\theta}^2 - \cos(\theta)\ddot{\theta}) = 0$$

$$\left.\begin{array}{l} \sum F_y = 0 \\ ml\ddot{\theta} + m\ddot{x}\cos(\theta) - C_y\sin(\theta) - C_x\cos(\theta) + mg\sin(\theta) = 0 \\ \sum M_G = I\ddot{\theta} \\ I\ddot{\theta} + C_y l\sin(\theta) + C_x l\cos(\theta) = 0 \end{array}\right\rangle (ml^2 + I)\ddot{\theta} + ml(g\sin(\theta) + \ddot{x}\cos(\theta)) = 0$$

In order to extract the dynamic model of the DC motor, it is required to follow the following schematic:

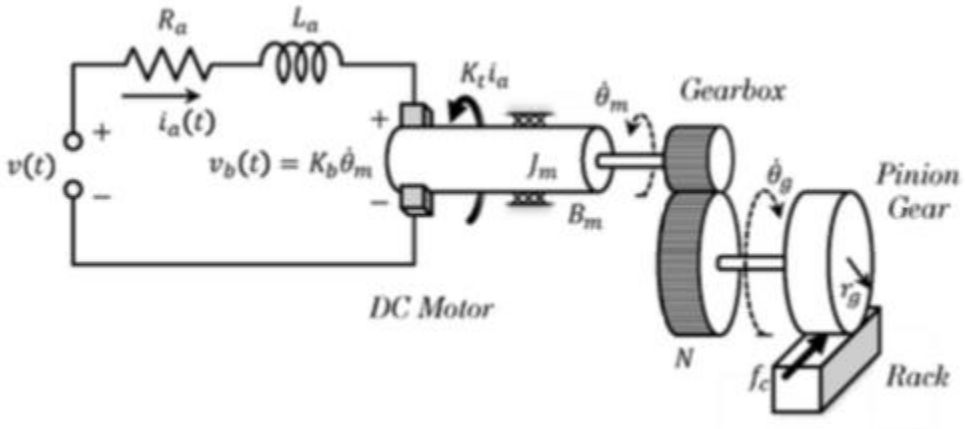


Figure 2: DC Motor and an attached spur gear box.
www.chegg.com/

By using Kirchhof's law and on the circuit diagram, we get to the following equation;

$$R_a i_a(t) + L_a\frac{di_a(t)}{dt} + V_{BMEF}(t) = V_A(t)$$

Here, the constant R and L represent the armature resistance and inductance, and the variable i(t) and v(t) represent the motor current draw and the applied voltage respectively.

The voltage fed back into the circuit, as a result of motion of wires in a magnetic field is called the back electromotive force and it can be determined from the following equation:

$$V_{BMEF} = NK_B\dot{\theta}_g$$

The sum of moments about the rotor of the DC motor gives the second equation:

$$J_m N\ddot{\theta}_g + B_m N\dot{\theta}_g = \tau_m + \tau_L$$

Here, J_m represents the rotor inertia, B_m represents the damping acting on the rotor, T_m and T_L represents the torques on the motor due to the magnetic field and external load applied to the rotor. The equation for this torques can extracted as follows:

$$\tau_m = K_t i_a(t)$$

$$\tau_L = -f_c\frac{r_g}{N}$$

We assume that there is no slippage associated with the wheel movement. Hence, we will end up to the following equation:

$$\left.\begin{aligned} & J_m N\ \ddot{\theta}_g + B_m N\ \dot{\theta}_g - K_t i_a(t) = -f_c \frac{r_g}{N} \\ & i_a(t) = \frac{v(t)}{R_a} \end{aligned}\right\} \Rightarrow J_m N\ \ddot{\theta}_g + B_m N\ \dot{\theta}_g - K_t \frac{v(t)}{R_a} = -f_c \frac{r_g}{N}$$

So far, the governing equation for the DC motor is extracted and a logical connection has established between the applied force and the DC Input Voltage. Know, it is required to connect the DC motor equation to the main mathematical model of the system. The process is as follows:

$$\left.\begin{aligned} & x = r_g \theta \\ & J_m N\ \ddot{\theta}_g + B_m N\ \dot{\theta}_g - K_t i_a(t) = -f_c \frac{r_g}{N} \end{aligned}\right\} \Rightarrow -J_m N^2 \frac{\ddot{x}}{r_g^2} - B_m N^2 \frac{\dot{x}}{r_g^2} + N\ \frac{K_t i_a(t)}{r_g} = f_c$$

As a result, the mathematical model of the system was extracted as follows:

$$\begin{cases} F - (m+M)\ddot{x} - b\dot{x} - ml(\sin(\theta)\dot{\theta}^2 - \cos(\theta)\ddot{\theta}) = 0 \\ (ml^2 + I)\ddot{\theta} + ml(g\sin(\theta) + \ddot{x}\cos(\theta)) = 0 \\ -J_m N^2 \frac{\ddot{x}}{r_g^2} - B_m N^2 \frac{\dot{x}}{r_g^2} + N\ \frac{K_t i_a(t)}{r_g} = F \end{cases}$$

$$\left.\begin{aligned} & M = 0.5kg \\ & m = 0.2kg \\ & l = 0.3m \\ & b = 0.1N\ /\ m\ /\ \sec \\ & r_g = 0.1m \\ & I = 0.006kg.m^2 \end{aligned}\right\} and \left\{\begin{aligned} & J_m = 2.8e-6kg\ /\ m^2 \\ & B_m = 7.6e-7Nm\sec \\ & N = 19.7 \\ & K_t = 3.95e-2Nm\ /\ A \\ & R = 8.3\Omega \end{aligned}\right\}$$

The state space equation of the system is as follows:

$$\left.\begin{aligned} & x_1 = x \\ & x_2 = \dot{x} \\ & x_3 = \theta \\ & x_4 = \dot{\theta} \end{aligned}\right\} \Rightarrow \begin{bmatrix} \dot{x}_2 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ \dfrac{-0.1431x_2 + 2.095\sin(x_3)\cos(x_3) + 0.0856\sin(x_3)x_4^2 + 1.3373V}{(1 - 0.2143(\cos(x_3))^2} \\ x_4 \\ \dfrac{-24.5\sin(x_3) + 0.3579x_2\cos(x_3) - 0.2146\sin(x_3)\cos(x_3)x_4^2 - 3.32\cos(x_3)V}{(1-0.2143(\cos(x_3))^2} \end{bmatrix}$$

1.2 Model Predictive Control Algorithm

In this section, the model predictive control algorithm was extracted. The Euler discretization method was used to extract the discretized model of the system as follows:

$$\dot{x} = f_c(x(t), u(t)) \xrightarrow{Euler\_Method} x(k+1) = f(x(k), u(k))$$

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} + \Delta T \times \begin{bmatrix} x_2 \\ \dfrac{-0.1431x_2(k) + 2.095\sin(x_3(k))\cos(x_3(k)) + 0.0856\sin(x_3(k))x_4^2(k) + 1.3373F(k)}{(1 - 0.2143(\cos(x_3(k)))^2} \\ x_4 \\ \dfrac{-24.5\sin(x_3(k)) + 0.3579x_2(k)\cos(x_3(k)) - 0.2146\sin(x_3(k))\cos(x_3(k))x_4^2(k) - 3.32\cos(x_3(k))F(k)}{(1-0.2143(\cos(x_3(k)))^2} \end{bmatrix}$$

The running cost for this system is as follows:

$$\ell(x,u) = \| x_u - x_{ref} \|_Q^2 + \| u - u_{ref} \|_R^2$$

The objective function for our optimal control problem is as follows:

$$\begin{aligned} &\underset{\mathbf{u}\ \text{admissible}}{\text{minimize}}\ J_N(\mathbf{x}_0, \mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_\mathbf{u}(k), \mathbf{u}(k)) \\ &\text{subject to}: \mathbf{x}_\mathbf{u}(k+1) = \mathbf{f}(\mathbf{x}_\mathbf{u}(k), \mathbf{u}(k)), \\ &\qquad\qquad \mathbf{x}_\mathbf{u}(0) = \mathbf{x}_0, \\ &\qquad\qquad \mathbf{u}(k) \in U, \ \forall k \in [0, N-1] \\ &\qquad\qquad \mathbf{x}_\mathbf{u}(k) \in X, \ \forall k \in [0, N] \end{aligned}$$

To solve the objective function, we are going to use CasADi library in MATLAB. This library has some useful and strong application:

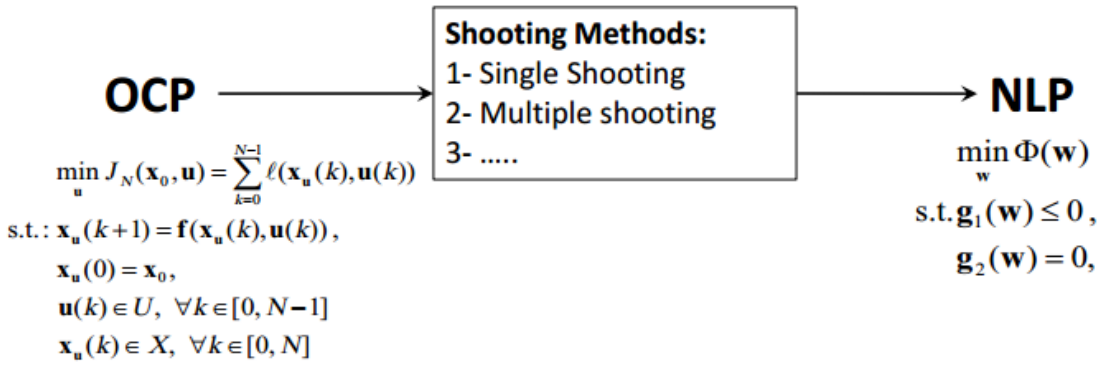
- Has a general scope of Numerical optimization.
- In particular, it facilitates the solution of NLP's
- Facilitates, not actually solves the NLP's
- Solver\plugins" can be added post-installation.
- Free & open-source (LGPL), also for commercial use.
- Project started in December 2009, now at version 3.4.5 (August, 2018)
- 4 standard problems can be handled by CasADi
  1) QP's (Quadratic programs)
  2) NLP's (Nonlinear programs)
  3) Root finding problems
  4) Initial-value problems in ODE/DAE

Basically, we use CasADi to transform our optimal control problem to a non-linear programming problem. To do so, CasADi uses single shooting method.

Finally, our objective function will be a non-linear equation to minimize. This objective function could be subjected to input and state constraints. The number of prediction horizon in N = 14 and dT = 0.1 for every iteration.

$$\begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \end{bmatrix} = \begin{bmatrix} -0.25 \\ 0 \\ pi - \frac{pi}{20} \\ 0 \end{bmatrix} \Rightarrow Initial\_Condition$$

$$\begin{bmatrix} x_1(s) \\ x_2(s) \\ x_3(s) \\ x_4(s) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ pi \\ 0 \end{bmatrix} \Rightarrow The\_REFRENCE\_POINT$$



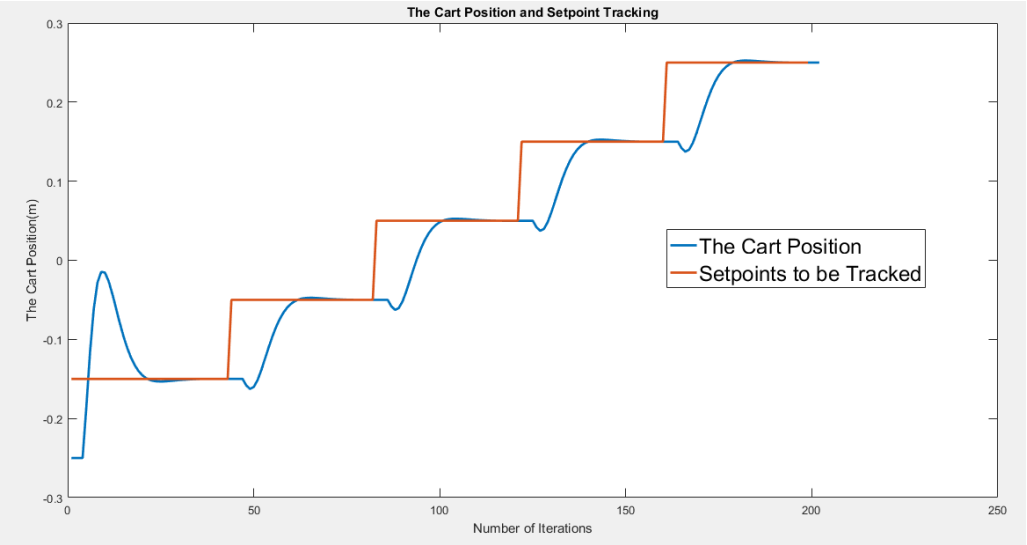
1.2.1 MATLAB Coding

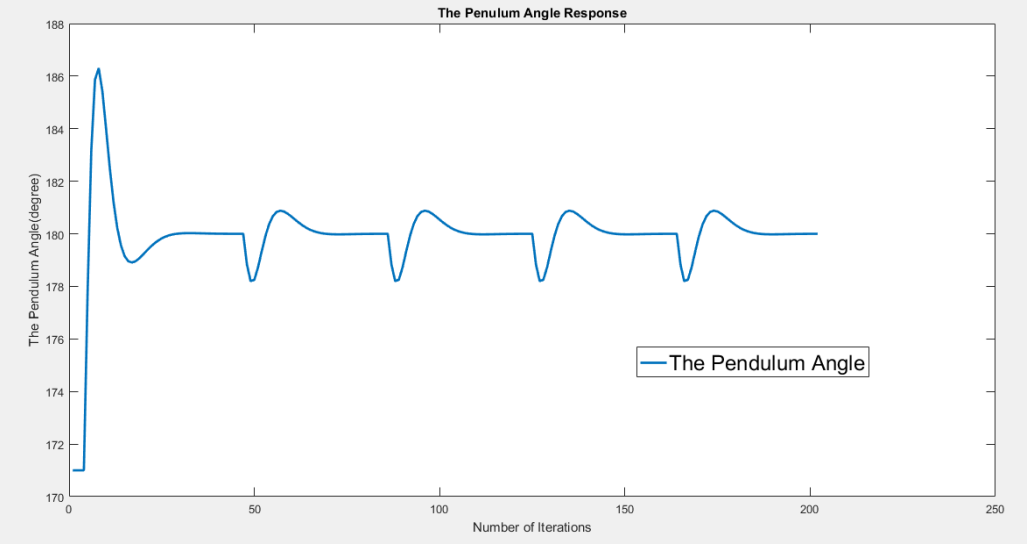
In this section, the MPC approach for the motorized Cart-inverted pendulum is implemente using CasADi library in MATLAB.

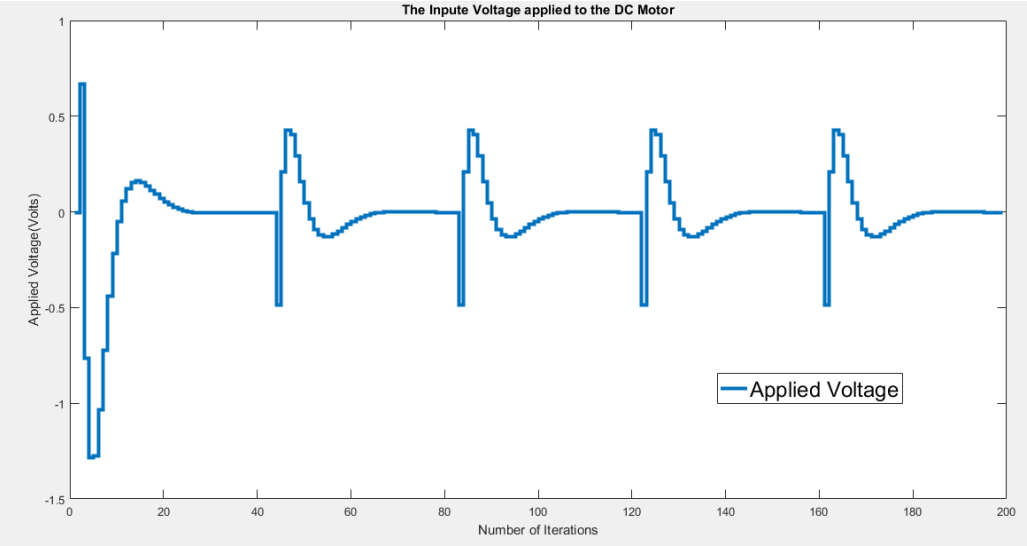
```
%% Inverted Pendulum Problem
% Behnam Moradi
% Student ID: 200 433 555
% Camputer Aided Process Engineering
clc; clear all; close all;
% CasADi v3.4.5
%% Symbol Definition
import casadi.* % Importing Casadi Library
T = 0.1; % Sampling time [s]
N = 14; % Prediction horizon
F_max = 12; F_min = -F_max; % Saturation Constraints
x1 = SX.sym('x1');  % State X1
x2 = SX.sym('x2');  % State X2
x3 = SX.sym('x3');  % State X3
x4 = SX.sym('x4');  % State X4
states = [x1;x2;x3;x4]; % States Vestor
n_states = length(states);
F = SX.sym('F'); % Input Variavble or Decision Variable
controls = [F]; % Input Vectors
n_controls = length(controls);
rhs = [x2;(-0.1431*x2 + 2.095*sin(x3)*cos(x3) + 0.0856*sin(x3)*x4^2 +1.3373*F)/(1 -
0.2143*(cos(x3))^2); x4; (-24.5*sin(x3) + 0.3579*x2*cos(x3) -
0.2146*sin(x3)*cos(x3)*x4^2 - 3.32*cos(x3)*F)/(1-0.2143*(cos(x3))^2)];
f = Function('f',{states,controls},{rhs}); % nonlinear mapping function f(x,u)
U = SX.sym('U',n_controls,N); % Decision variables (controls)
P = SX.sym('P',n_states + n_states); % parameters (which include the initial and the
reference state of the robot)
% P = SX.sym('P',N*(n_states + n_controls));
X = SX.sym('X',n_states,(N+1)); % A Matrix that represents the states over the
optimization problem.

%% compute solution symbolically
X(:,1) = P(1:4); % initial state
for k = 1:N
    st = X(:,k);
    con = U(:,k);
    f_value  = f(st,con);
    st_next  = st+ (T*f_value);
    X(:,k+1) = st_next;
end
% this function to get the optimal trajectory knowing the optimal solution
ff=Function('ff',{U,P},{X});
obj = 0; % Objective function
g = [];  % constraints vector
Q = zeros(4,4);
Q(1,1) = 100;
Q(2,2) = 0;
Q(3,3) = 50;
Q(4,4) = 0;% weighing matrices (states)
R = 1; % weighing matrices (controls)
```

```
%% Compute Objective
for k=1:N
    st = X(:,k);
    con = U(:,k);
    obj = obj+(st-P(5:8))'*Q*(st-P(5:8)) + con'*R*con; % calculate obj
%     obj = obj+(st-P(5*(k-1)+1:5*k-1))'*Q*(st-P(5*(k-1)+1:5*k-1)) + (con -
P(5*k))'*R*(con - P(5*k));
end

%% Compute Constraints
for k = 1:N+1
    g = [g ; X(1,k)];   %state x1
    g = [g ; X(2,k)];   %state x3
    g = [g ; X(3,k)];   %state x1
    g = [g ; X(4,k)];   %state x3

end

% make the decision variables one column vector
OPT_variables = reshape(U,N,1);
nlp_prob = struct('f', obj, 'x', OPT_variables, 'g', g, 'p', P);
opts = struct;
opts.ipopt.max_iter = 100;
opts.ipopt.print_level =0;%0,3
opts.print_time = 0;
opts.ipopt.acceptable_tol =1e-8;
opts.ipopt.acceptable_obj_change_tol = 1e-6;
solver = nlpsol('solver', 'ipopt', nlp_prob,opts);
args = struct;
args.lbg(1:4:4*(N+1),1) = -0.5;  % lower bound of the states x1
args.ubg(1:4:4*(N+1),1) = 0.5;   % upper bound of the states x1
args.lbg(2:4:4*(N+1),1) = -inf;  % lower bound of the states x2
args.ubg(2:4:4*(N+1),1) = inf;   % upper bound of the states x2
args.lbg(3:4:4*(N+1),1) = -pi;   % lower bound of the states x3
args.ubg(3:4:4*(N+1),1) = 2*pi;  % upper bound of the states x3
args.lbg(4:4:4*(N+1),1) = -inf;  % lower bound of the states x4
args.ubg(4:4:4*(N+1),1) = inf;   % upper bound of the states x4
% input constraints
args.lbx(1:1:N,1) = F_min;
args.ubx(1:1:N,1) = F_max;
%% THE SIMULATION LOOP SHOULD START FROM HERE
%-------------------------------------------
t0 = 0;
x0 = [-0.25 ; 0 ;pi-pi/20; 0]; % initial condition.
xs = [0 ; 0 ; pi; 0]; % Reference posture.
xref1 = -0.25;
xs(1) = xref1;
xx(:,1) = x0; % xx contains the history of states
t(1) = t0;
u0 = zeros(N,1);  % two control inputs
sim_tim = 20; % Maximum simulation time
```

```
%% Start MPC
mpciter = 0;
xx1 = [];
u_cl=[];
main_loop = tic;
xref1 = -0.25;
uref = 0;
j = 1;
for i = 1:5
    xs(1) = xs(1) + 0.1;
while norm((x0 - xs),3) > 1e-4
    args.p = [x0;xs]; % set the values of the parameters vector
    args.x0 = reshape(u0',N,1); % initial value of the optimization variables
    sol = solver('x0', args.x0, 'lbx', args.lbx, 'ubx', args.ubx,...
                 'lbg', args.lbg, 'ubg', args.ubg,'p',args.p);

    u = reshape(full(sol.x)',1,N)';
    ff_value = ff(u',args.p); % compute OPTIMAL solution TRAJECTORY
    xx1(:,1:4,mpciter+1)= full(ff_value)';
    u_cl= [u_cl ; u(1,:)];
    t(mpciter+1) = t0;
    [t0, x0, u0] = shift(T, t0, x0, u,f); % get the initialization of the next
optimization step
    xx(:,mpciter+4) = x0;
    mpciter
    mpciter = mpciter + 1;
    setP(mpciter) = xs(1);
    itr(mpciter) = mpciter;
end;
end
main_loop_time = toc(main_loop);
ss_error = norm((x0-xs),3)
average_mpc_time = main_loop_time/(mpciter+1);
u_cl(1) = 0;
xx(1,2) = xx(1,1);
xx(1,3) = xx(1,4);
xx(3,2) = xx(3,1);
xx(3,3) = xx(3,4);

figure;
plot(xx(1,:), 'LineWidth',2);
hold on
plot(itr,setP, 'LineWidth',2);
title('The Cart Position and Setpoint Tracking');
legend('The Cart Position','Setpoints to be Tracked');
xlabel('Number of Iterations');
ylabel('The Cart Position(m)');

figure;
plot(xx(3,:)*180/pi, 'LineWidth',2);
title('The Penulum Angle Response');
legend('The Pendulum Angle');
xlabel('Number of Iterations');
ylabel('The Pendulum Angle(degree)');

figure;
stairs(u_cl, 'Linewidth', 3);
title('The Inpute Voltage applied to the DC Motor');
legend('Applied Voltage');
xlabel('Number of Iterations');
ylabel('Applied Voltage(Volts)');
```


The Cart Position and Setpoint Tracking
The Cart Position(m)
Number of Iterations
The Cart Position
Setpoints to be Tracked


The Penulum Angle Response
The Pendulum Angle(degree)
Number of Iterations
The Pendulum Angle


The Inpute Voltage applied to the DC Motor
Applied Voltage(Volts)
Number of Iterations
Applied Voltage

2 Full State Observer Design

Note: Not Completed!

March 10, 2020