Contents


```
clc
clear
close all
%#ok<*NASGU>
%#ok<*UNRCH>
```

Load the parameters of the system

```
run('load_suspension_params')

sample_time = 0.002; % seconds
sys = ss(Am,Bm,Cm,Dm);

% Create discrete time system
d_sys = c2d(sys, sample_time);
dA = d_sys.A;
dB = d_sys.B;
dC = d_sys.C;
dD = d_sys.D;
```

Settings

```
% Type of controller
load('ControllerIndices.mat')
ControllerIndex = controllers.MPC;

% Whether to inject noise
use_noise = 0;

% Calculate symbolic matrices
global use_sym
use_sym = 0;

% Toggle the controller on or off
open_loop = 0;
```

MPC Model

```
% Make the output equal to the state
dC = eye(4);
dD = zeros(4,1);
```

```matlab
% Weighting matrices
Q = eye(4);
Q(1,1) = 1;
Q(2,2) = 1;
Q(3,3) = 1;
Q(4,4) = 1;
R = 0.0005;

% Control and Prediction Horizon
Np = 100;
Nc = 50;
assert(Np >= Nc)

if use_sym
   syms A B C D real
end

% Create Ap and Bp Matrices
Ap = Zeros(size(dA, 1)*Np, size(dA, 2)*1 );
Bp = Zeros(size(dC, 1)*Np, size(dB, 2)*Nc);

% Populate the Ap Matrix
val = dC;            % Value to put into Ap
hA  = size(dA, 1);  % Height of the A matrix
for i = 1:Np
    start_idx = hA*(i-1) + 1;
    end_idx   = hA*i;

    val = val*dA;
    Ap(start_idx:end_idx, :) = val;
end

% Populate the Bp matrix
hB = size(dC,1);
wB = size(dB,2);
for row = 1:(size(Bp,1)/hB)
    row_start_idx = hB*(row-1) + 1;
    row_end_idx   = hB*row;

    row_entry = Zeros(hB, size(Bp,2));
    for col = 1:(size(Bp,2)/wB)
        % Find the chunk for this entry
        start_col_idx = wB*(col-1)+1;
        end_col_idx  = wB*col;

        % Find the power of A in this column for this row
        pow = row - col;

        % If the power at least 0, use the formula
        if (pow >= 0)
            row_entry(:, start_col_idx:end_col_idx) = dC*dA^pow*dB;

        % Otherwise, if it is exactly -1, place D
        elseif (pow == -1)
            row_entry(:, start_col_idx:end_col_idx) = dD;
        end
    end

    Bp(row_start_idx:row_end_idx, :) = row_entry;
end
```

```
% Create QNp
Qs = cell(1,Np);
[Qs{:}] = deal(Q);
QNp = blkdiag(Qs{:});

% Create RNc
Rs = cell(1,Nc);
[Rs{:}] = deal(R);
RNc = blkdiag(Rs{:});

% Calculate H2
H2 = Bp'*QNp*Bp + RNc;
H2_inv_T = inv(H2)';
```

Run the Simulation

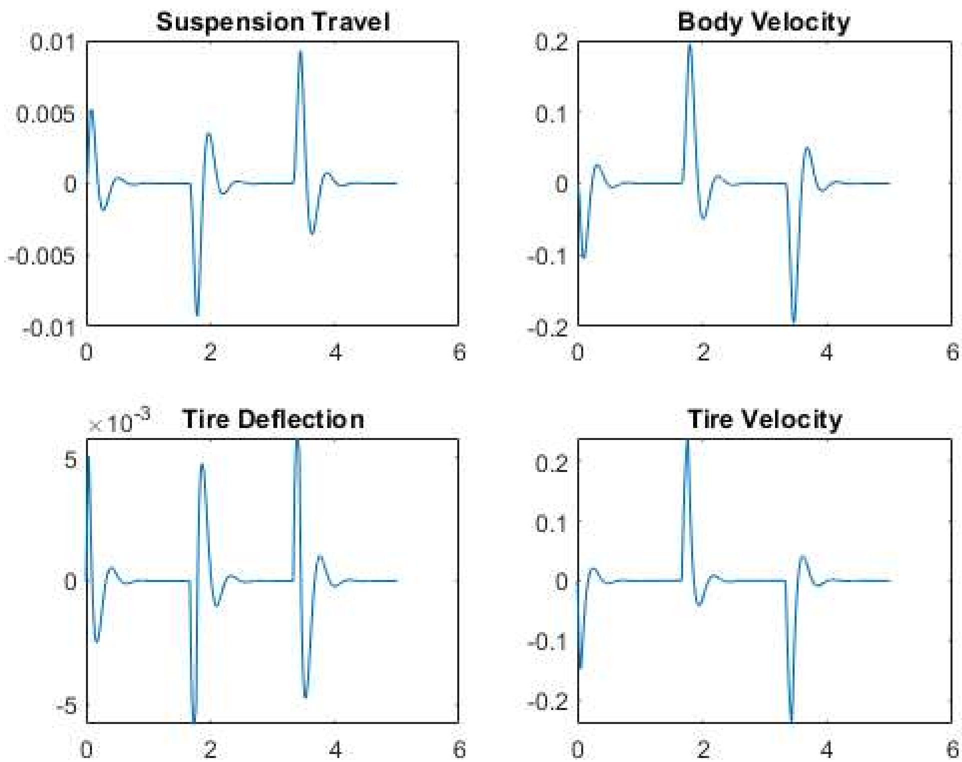
```
if open_loop
   ControllerIndex = 0;
end
sim('AdvancedControlSim.slx')
```

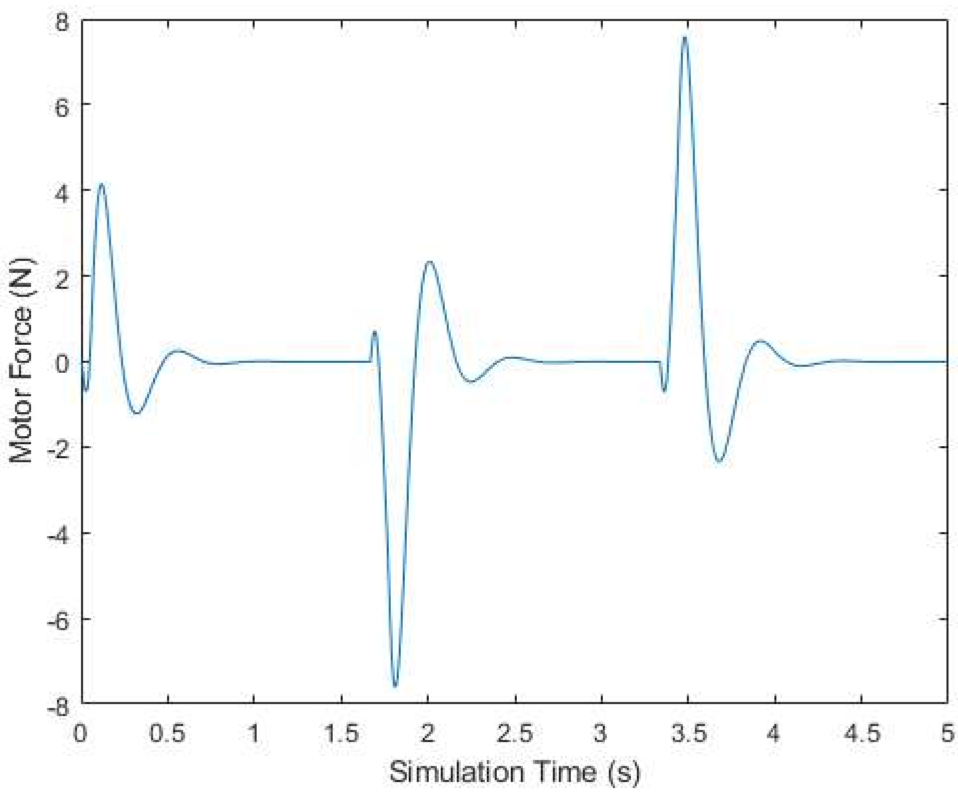
Plot the results

```
T = state.Time;
if use_noise
    x = noisy_state.Data;
else
    x = state.Data;
end
y = output.Data;
u = input.Data(:,2);

% Plot the 4 states
figure(1)
states = ["Suspension Travel", "Body Velocity", "Tire Deflection", "Tire Velocity"];
for i = 1:4
    subplot(2,2,i)
    plot(T, x(:,i))
    title(states(i));
end

% Plot the actuator force
figure(2)
plot(T, u)
ylabel("Motor Force (N)")
xlabel("Simulation Time (s)")
```



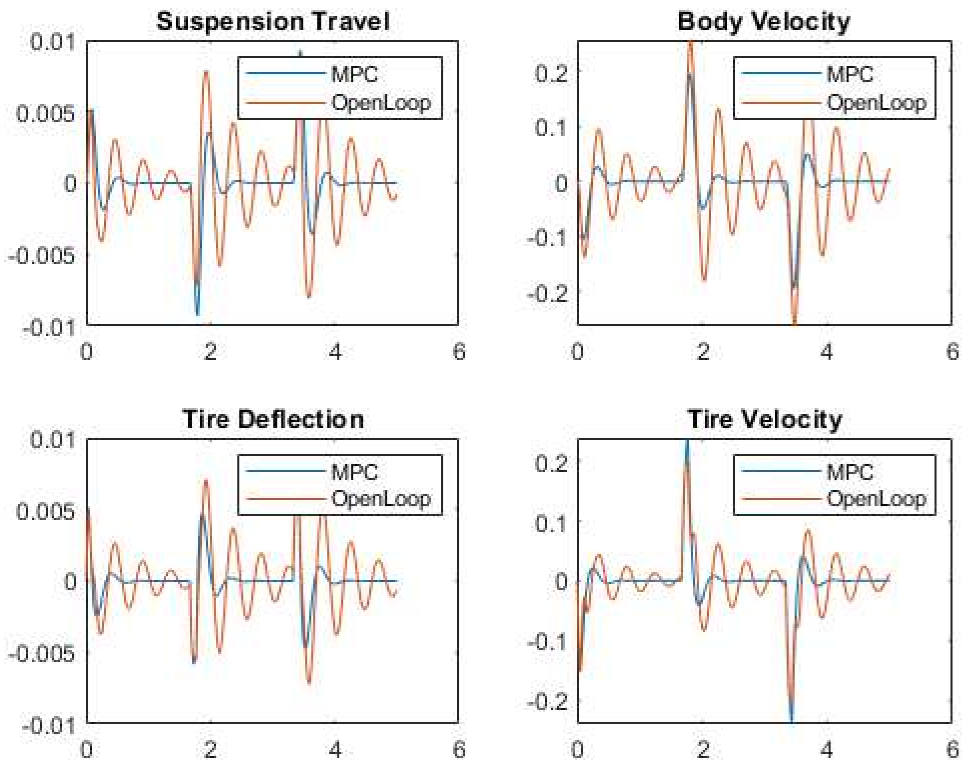


Compare to Open Loop

```
load("OpenLoopSimResults")

figure(1)
```

```
for i = 1:4
    subplot(2,2,i)
    hold on
    plot(OpenLoop.T, OpenLoop.x(:,i))
    legend("MPC", "OpenLoop")
end
```



Extra Functions

```
function y = Zeros(varargin)
    global use_sym
    y = zeros(varargin{:});
    if use_sym
       y = sym(y);
    end
end
```

Published with MATLAB® R2021a