Table of Contents

Prepare

```
clc
clear
close all
```

Constants

```
g = 9.81; % m/s^2
m = 2050; % kg
I = 3344; % kg-m^2
% mu = .3;  % Slipery
rho = Inf;  % m
W_line = 3; % m
x_dot_ref = 125/9; % m/s

% Geometry
a = 2;
b = 2;
c = 1.8;

C_fu = (4200-(-4200))/(4-(-4));   % ? +
C_rL = (3800-(-3800))/(5-(-5)*8);   % ? +
C_ru = (2000-(-2000))/(2-(-2))*8;   % ?
```

Variables

```
T_sampling  = 0.1;
T_duration  = 20;
N_horizon   = 30;
R_rho       = .01;
x_0  = [0 0 0 0 0]';
umin = [-pi/3 1]';
umax = [pi/3 1]';
xmin = [];
xmax = [];
```

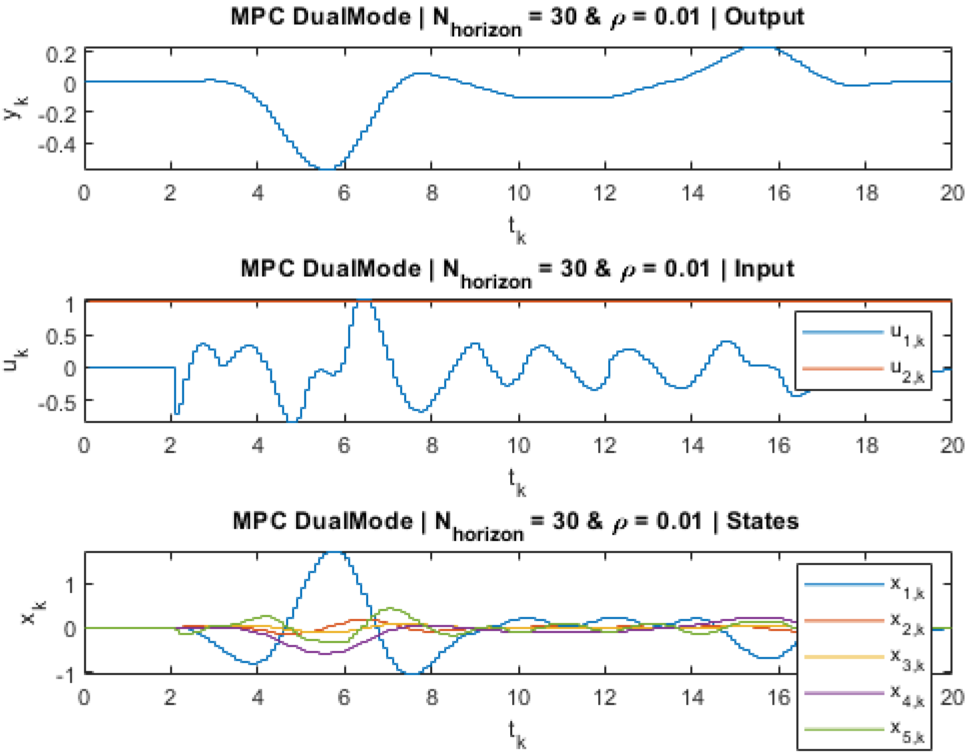
```
f_x = [
       0 0 0 1  0;      % e_y
       0 0 0 -1 0;
       0 0 0 0  1;      % delta
       0 0 0 0 -1;
       1/x_dot_ref a/x_dot_ref 0 0 -1;  % alpha_f,lim
       -1/x_dot_ref -a/x_dot_ref 0 0 1;
       1/x_dot_ref -b/x_dot_ref 0 0 0;  % alpha_r,lim
       -1/x_dot_ref b/x_dot_ref 0 0 0;
       ];
f_u = [];
f_u_eq = [
    0 1
    ];
v_x = [
       W_line/2; % e_y
       W_line/2;
       pi/3;     % delta
       pi/3;
       pi/6;  % alpha_f_lim
       pi/6;  % alpha_f_lim
       pi/6;  % alpha_r_lim
       pi/6;  % alpha_r_lim
       ];
v_u = [];
v_u_eq = [
    1
    ];
obstacle1.duration  = [5 6];
obstacle1.width     = [1 -0.5];
obstacle2.duration  = [15 16];
obstacle2.width     = [.2 -0.5];
obstacle3.duration  = [10 12];
obstacle3.width     = [.2 -0.1];
Obstacles = [obstacle1 obstacle2 obstacle3];
[f_x_TV, v_x_TV, TV_x] = CreateObstacle(Obstacles, T_sampling);

showPlots = true;
Psi_r = 1/rho;
```

State Space Modeling

```
States = ["y_dot", "Phi_dot", "e_Phi", "e_y", "delta"];
A_c = [ (C_fu+C_ru)/(x_dot_ref*m) , (-x_dot_ref + (C_fu*a-b*C_ru)/
(x_dot_ref*m)), 0, 0, -C_fu/m ;
      (a*C_fu-b*C_rL)/(I*x_dot_ref), (a*a*C_fu+b*b*C_rL)/
(I*x_dot_ref), 0, 0, -a*C_fu/I  ;
      0, 1, 0, 0, 0; % Constant Term
      1, 0, x_dot_ref, 0, 0;
      0, 0, 0, 0, 0 ];
B_c = [0 0 0 0 1; 0 0 -x_dot_ref*Psi_r 0 0]';
C_c = [0 0 0 1 0];
D_c = [0 0];
```

Discritize

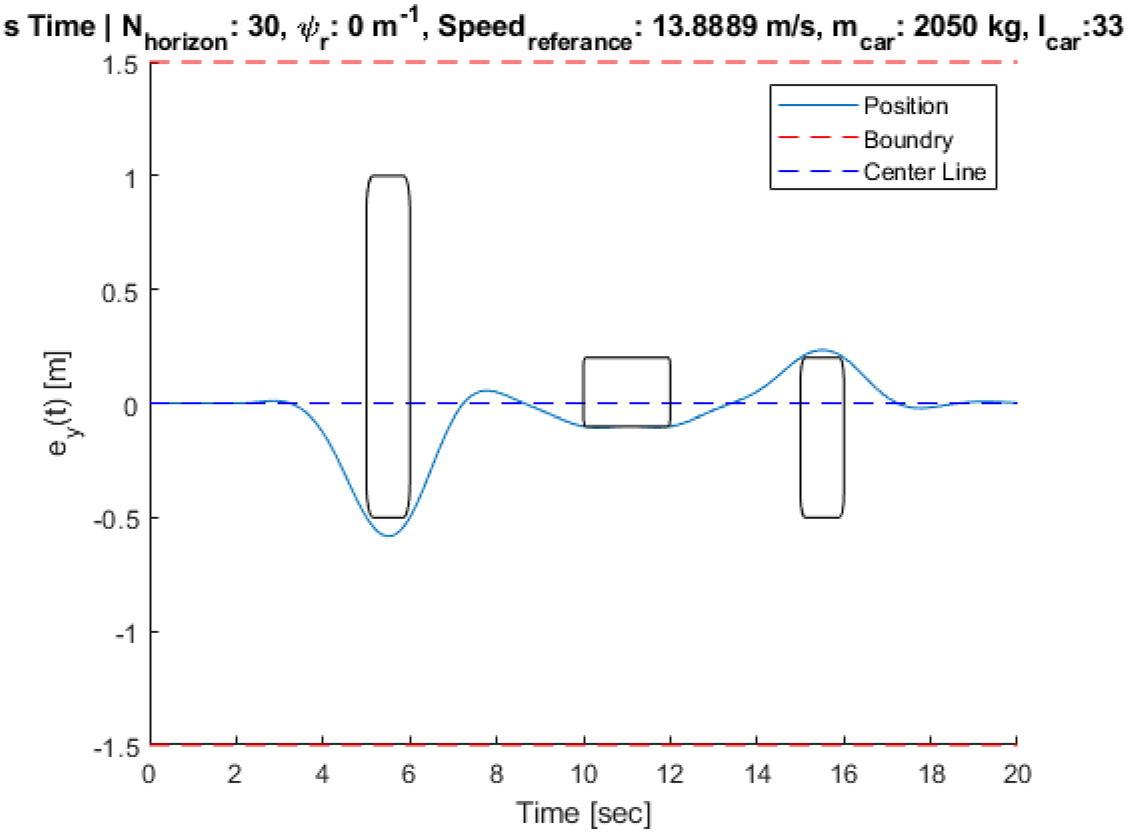
```
[A, B, C, D] = DiscritizeStateSpace(A_c, B_c, C_c, D_c, T_sampling);

[X, U, Y] = MPC_DualMode(A, B, C, D, N_horizon, T_sampling,
 T_duration, R_rho, x_0, umin, umax, xmin, xmax, f_x, f_u, f_u_eq,
 f_x_TV, v_x_TV, TV_x, v_x, v_u, v_u_eq, showPlots);
```



Plot Road

```
figure;
hold on
time_vec = 0:T_sampling:T_duration;
plot(time_vec, Y, "DisplayName", "Position")
plot(time_vec,
 (W_line/2)*ones(size(time_vec)), "r--", "DisplayName", "Boundry")
plot(time_vec, -
(W_line/2)*ones(size(time_vec)), "r--", "HandleVisibility", "off")
plot(time_vec, zeros(size(time_vec)), "b--", "DisplayName", "Center
 Line")
plotObstacles(Obstacles);
legend
title("e_y(t) vs Time | N_{horizon}: "+N_horizon+", \psi_r: "+Psi_r
+" m^{-1}, Speed_{referance}: "+x_dot_ref+" m/s, m_{car}: "+m+" kg,
 I_{car}:" +I+" kg-m^2")
```

```
xlabel("Time [sec]")
ylabel("e_y(t) [m]")
```



MPC_DualMode

```
function [X, U, Y] = MPC_DualMode(A, B, C, D, N_horizon, T_sampling,
 T_duration, rho, x_0, umin, umax, xmin, xmax, f_x, f_u, f_u_eq,
 f_x_TV, v_x_TV, TV_x, v_x, v_u, v_u_eq, showPlots)
    warning off all
    N_step = floor(T_duration / T_sampling);
    [n, m] = size(B);
    % m : # of inputs
    % n : # of states

    Q    = C'*C;
    R    = rho * eye(length(D));
    G    = zeros(n*(N_horizon), m*N_horizon);
    temp = [B zeros(n, m*(N_horizon-1))];
    for i = 1:N_horizon
        G((i-1)*n+1:i*n, :) = temp;
        temp = [A^(i)*B temp(:, 1:m*(N_horizon-1))];
    end
    G   = [zeros(n, m*N_horizon);G];
    clear temp
```

```
H = zeros(n*(N_horizon+1), n);
for i = 1:N_horizon+1
    H((i-1)*n+1:i*n, :) = A^(i-1);
end

Q_temp  = Q;
for i = 1:N_horizon-1
    Q_temp = blkdiag(Q_temp, Q);
end
[P_inf, ~, ~] = idare(A,B,Q,R,[],[] );
K_inf = (R+B'*P_inf*B)\(B'*P_inf*A);

P = dlyap( (A-B*K_inf)',Q+K_inf'*R*K_inf);
Q_f = P;

Q_bar   = blkdiag(Q_temp, Q_f);
clear Q_temp

R_temp  = R;
for i = 1:N_horizon-2
    R_temp = blkdiag(R_temp, R);
end
R_bar   = blkdiag(R_temp, R);
clear R_temp

M = G' * Q_bar * G + R_bar;

X = zeros(n, N_step+1);
X(:, 1) = x_0;
U = zeros(m, N_step);

Y(:, 1) = C*x_0;

if ~isempty(umin)
    % umin = [u1_min u2_min ... u_m_min]'
    U_min = ones(m,  N_horizon) .* umin;
    U_min = reshape(U_min, [], 1);
else
    U_min = [];
end

if ~isempty(umax)
    % umax = [u1_max u2_max ... u_m_max]'
    U_max = ones(m,  N_horizon) .* umax;
    U_max = reshape(U_max, [], 1);
else
    U_max = [];
end

F = [];
if ~isempty(xmin)
    % xmin = [x1_min x2_min ... x_n_min]'
    X_min_temp = ones(n,  N_horizon+1) .* xmin;
    X_min_temp = reshape(X_min_temp, [], 1);
    F = [F;-G];
end

if ~isempty(xmax)
    % xmax = [x1_max x2_max ... x_n_max]'
    X_max_temp = ones(n,  N_horizon+1) .* xmax;
    X_max_temp = reshape(X_max_temp, [], 1);
    F = [F;G];
end

if ~isempty(v_x)

    f_x_temp = [];
    for f_i = 1:N_horizon+1
        f_x_temp = blkdiag(f_x_temp, f_x');
    end

    v_x_temp = ones(1, N_horizon+1) .* v_x;
    v_x_temp = reshape(v_x_temp, [], 1);

    F     = [F; (f_x_temp' * G)];
end

if ~isempty(v_u)

    f_u_temp = [];
    for f_i = 1:N_horizon
        f_u_temp = blkdiag(f_u_temp, f_u');
    end

    v_u_temp = ones(1, N_horizon) .* v_u;
    V_u = reshape(v_u_temp, [], 1);

    F     = [F; f_u_temp'];
else
    V_u = [];
end

F_eq = [];
if ~isempty(v_u_eq)

    f_u_eq_temp = [];
    for f_i = 1:N_horizon
        f_u_eq_temp = blkdiag(f_u_eq_temp, f_u_eq');
    end

    v_u_eq_temp = ones(1, N_horizon) .* v_u_eq;
    V_u_eq = reshape(v_u_eq_temp, [], 1);

    F_eq     = [F_eq; f_u_eq_temp'];
else
    V_u_eq = [];
end
```

```
   V_eq = [V_u_eq];

   for i = 1:N_step
       %% Constraints
       if ~isempty(xmin)
           % xmin = [x1_min x2_min ... x_n_min]'
           X_min = - X_min_temp + H * X(:, i);
       else
           X_min = [];
       end

       if ~isempty(xmax)
           % xmax = [x1_max x2_max ... x_n_max]'
           X_max = X_max_temp - H * X(:, i);
       else
           X_max = [];
       end

       if ~isempty(v_x)
           V_x   = v_x_temp - f_x_temp' * H * X(:, i);
       else
           V_x = [];
       end

       if ~isempty(TV_x)
           F_step = [];
           V_x_step = [];
           for TV_cond = 1:length(TV_x)
               matches = ismember(i:i+N_horizon-1, TV_x{TV_cond});
               f_x_TV_temp = [];
               f_x_TV_temp = blkdiag(f_x_TV_temp, zeros(n,1));
               for match = matches
                   if (match)
                       vec = f_x_TV(TV_cond, :)';
                   else
                       vec = zeros(n,1);
                   end
                   f_x_TV_temp = blkdiag(f_x_TV_temp, vec);
               end
               F_step      = [F_step; (f_x_TV_temp' * G)];
               v_x_TV_temp = [0 matches] .* v_x_TV(TV_cond);
               v_x_TV_temp = reshape(v_x_TV_temp, [], 1);
               V_x_step    = [V_x_step;v_x_TV_temp - f_x_TV_temp' * H
* X(:, i)];
           end
       else
           F_step = F;
           V_x_step = V_x;
       end
       F_step      = [F; F_step];


       V = [X_min; X_max; V_x; V_u; V_x_step];
```

```
        Alpha = ( X(:, i)' * H' * Q_bar * G )';
        U_sol = quadprog(M, Alpha', F_step, V, F_eq, V_eq, U_min,
 U_max, [], optimset("Display", "off"));
        U(:, i) = U_sol(1:m);

        X(:, i+1) = A * X(:, i) + B * U(:, i);
        Y(:, i+1) = C * X(:, i+1) + D * U(:, 1);

    end

    warning on

    if (showPlots)
        figure;
        subplot(3, 1, 1)
        stairs(0:T_sampling:T_duration, Y, 'DisplayName', 'Output')
        title("MPC DualMode | N_{horizon} = "+N_horizon+" & \rho =
 "+rho+" | Output")
        xlabel("t_k")
        ylabel("y_k")
        subplot(3, 1, 2)
        stairs(T_sampling:T_sampling:T_duration, U')
        title("MPC DualMode | N_{horizon} = "+N_horizon+" & \rho =
 "+rho+" | Input")
        xlabel("t_k")
        ylabel("u_k")
        legend(arrayfun(@(mode) sprintf('u_{%d,k}', mode),
 1:m, 'UniformOutput', false));
        subplot(3, 1, 3)
        stairs(0:T_sampling:T_duration, X')
        title("MPC DualMode | N_{horizon} = "+N_horizon+" & \rho =
 "+rho+" | States")
        xlabel("t_k")
        ylabel("x_k")
        legend(arrayfun(@(mode) sprintf('x_{%d,k}', mode),
 1:n, 'UniformOutput', false));
    end
end
```

DiscritizeStateSpace

```
function [A, B, C, D] = DiscritizeStateSpace(A_c, B_c, C_c, D_c,
 T_sampling)
    A = expm(A_c*T_sampling);
    B_func = @(x) expm(A_c*x);
    B = integral(B_func, 0, T_sampling, 'ArrayValued', true)*B_c;
    C = C_c;
    D = D_c;
end
```

CreateObstacle

```
function [f_x_TV, v_x_TV, TV_x] = CreateObstacle(Obstacles,
 T_sampling)
    f_x_TV  = [];
    v_x_TV  = [];
    TV_x    = cell(size(Obstacles));
    for obs = 1:length(Obstacles)
        obstacle = Obstacles(obs);
        if mean(obstacle.width) < 0
            f_x_TV = [f_x_TV;0 0 0 -1 0]; % e_y
            v_x_TV = [v_x_TV;-max(obstacle.width)];
        else
            f_x_TV = [f_x_TV;0 0 0 1 0]; % e_y
            v_x_TV = [v_x_TV;min(obstacle.width)];
        end
        TV_x{obs}   = obstacle.duration(1)/
T_sampling:obstacle.duration(2)/T_sampling;
    end

end
```

plotObstacles

```
function plotObstacles(Obstacles)
    for obstacle = Obstacles
        rectangle('Position', [min(obstacle.duration)
 min(obstacle.width) abs(obstacle.duration(2)-obstacle.duration(1))
 abs(obstacle.width(2)-obstacle.width(1))], 'Curvature', 0.3)
    end
end
```

Published with MATLAB® R2020b