# 模型预测控制

## Model Predictive Control

诸 兵，Bing Zhu
第七研究室，北京航空航天大学
The Seventh Research Division, Beihang University

2022 Spring

# Implementation in MATLAB

**Lecture 5**

- Coding MPC from zero in MATLAB
- MPC toolbox in MATLAB

# At first place, let's predict !!

- Suppose that, the plant is given by

$$x(k+1) = Ax(k) + Bu(k), \qquad\qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^p,$$

  where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^p$ are state and control input, respectively.

- The matrices $A$, $B$ are known with proper dimensions. $(A, B)$ is controllable/stabilizable.

- The state sequence can be predicted by

$$X(k) = Fx(k) + \Phi U(k), \qquad\qquad F = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \Phi = \begin{bmatrix} B & 0 & & \\ AB & B & 0 & \\ \vdots & & & \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix},$$

$$F = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix},$$

$$\Phi = \begin{bmatrix} B & 0 & & \\ AB & B & 0 & \\ \vdots & & & \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}.$$

- For example

$$A = \begin{bmatrix} 1.1 & 2 \\ 0 & 0.95 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.079 \end{bmatrix}$$

**DEMO**

# Weighting matrices and the terminal weight

- Infinite horizon cost function:

**DEMO**

$$J(k) = \sum_{i=1}^{\infty} \|x(i\,|\,k)\|_Q^2 + \|u(i-1\,|\,k)\|_R^2$$

$$= \|x(N\,|\,k)\|_P^2 + \|u(N-1\,|\,k)\|_R^2 + \sum_{i=1}^{N-1} \|x(i\,|\,k)\|_Q^2 + \|u(i-1\,|\,k)\|_R^2$$

$$= X^T(k)\mathcal{Q}X(k) + U(k)^T\mathcal{R}U(k)$$

$$\mathcal{Q} = \mathrm{diag}[Q, Q, \cdots, Q, P]$$

$$\mathcal{R} = \mathrm{diag}[R, R, \cdots, \cdots, R]$$

$$P - (A-BK)^T P (A-BK) = Q + K^T R K$$

```
%K for calculating  P
j = sqrt(-1);
p = [0.5+0.5*j, 0.5-0.5*j];
K = place(A, B, p);
```

# Weighting matrices and the terminal weight (cont'd)

- Solve the discrete-time Lyapunov equation to get the terminal weight

$$\mathcal{Q} = \mathrm{diag}[Q, Q, \cdots, Q, P] \qquad P - (A - BK)^T P(A - BK) = Q + K^T RK$$

```
%solve lyapunov equation
Ak = A-B*K;
Qk = Q+K'*R*K;
P = dlyap(Ak', Qk)
```

## Description

`X = dlyap(A,Q)` solves the discrete-time Lyapunov equation $AXA^T - X + Q = 0$,

where $A$ and $Q$ are $n$-by-$n$ matrices.

The solution $X$ is symmetric when $Q$ is symmetric, and positive definite when $Q$ is positive definite and $A$ has all its eigenvalues inside the unit disk.

**DEMO**

## For unconstrained MPC, simply calculate the linear feedback gain

$$u^*(k) = -\left[I_{p\times p}\ 0\ \cdots\ 0\right](\Phi^T \mathcal{Q}\Phi + \mathcal{R})^{-1}\Phi^T \mathcal{Q} F x(k) = -K_{mpc} x(k)$$

```
%Unconstrained MPC feedback gain
Kmpc4 = [1 0 0 0]*inv(Phi'*QQ*Phi+RR)*Phi'*QQ*F;
```

**DEMO**

# For constrained MPC, continue to construct the optimization

- Constrained optimization

```
%optimization
U = quadprog( Phi'*QQ*Phi+RR, x(:,i)'*F'*QQ*Phi, Ain, bin, [], [], lb, ub, u0);
```

$$U^*(k) = \arg\min_{U(k)} J(k)$$

$$= \arg\min_{U(k)} \left[ x^T(k)F^T\mathcal{Q}Fx(k) + 2x^T(k)F^T\mathcal{Q}\Phi U(k) + U^T(k)(\Phi^T\mathcal{Q}\Phi + \mathcal{R})U(k) \right],$$

s.t. $\quad Gx(i\,|\,k) + Hu(i\,|\,k) \le 1, \qquad \forall i = 0,1,\cdots,N-1.$

$$x(N\,|\,k) \in \mathcal{X}_f \subset \Omega\,.$$

`x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)` 从向量 x0 开始求解上述问题。如果不存在边界，请设置 `lb = []` 和 `ub = []`。一些 quadprog 算法会忽略 x0；请参阅 x0。

**注意**

x0 是 `'active-set'` 算法的必需参数。

**说明**

具有线性约束的二次目标函数的求解器。

quadprog 求由下式指定的问题的最小值

$$\min_x \frac{1}{2}x^THx + f^Tx \text{ such that } \begin{cases} A\cdot x \le b, \\ Aeq\cdot x = beq, \\ lb \le x \le ub. \end{cases}$$

H、A 和 Aeq 是矩阵，f、b、beq、lb、ub 和 x 是向量。

# Implementation of terminal constraints

- Terminal state:

$$x(N|k) = [0,\ 0,\ \cdots,\ I_{n\times n}]X(k)$$
$$= [0,\ 0,\ \cdots,\ I_{n\times n}]\big[Fx(k) + \Phi U(k)\big]$$

**DEMO**

- For example

$$x(N|k) \le \mathbf{1}$$
$$\Downarrow$$
$$[0,\ 0,\ \cdots,\ I_{n\times n}]\big[Fx(k) + \Phi U(k)\big] \le \mathbf{1}$$
$$\Downarrow$$
$$[A^{N-1}b, A^{N-2}b, \cdots, b]U(k) \le \mathbf{1} - A^N x(k)$$

```
%terminal inequality constraint
bin1 = [u1_terminal_upper; u2_terminal_upper]-A^Nc*x(:,i);
bin2 =- [u1_terminal_lower; u2_terminal_lower]+A^Nc*x(:,i);

bin = [bin1; bin2];
Ain = [Phi(2*Nc-1:2*Nc,:); -Phi(2*Nc-1:2*Nc,:)];
```

- Prediction: calculate $F$ and $\Phi$, such that $X(k) = Fx(k) + \Phi U(k)$
- Weighting matrices: calculate $P$
  - find $K$ such that $|\mathrm{eig}(A - BK)| < 0$
  - solve discrete-time Lyapunov equation to obtain $P$
- Constraints
  - state constraints and control constraints
  - terminal constraints

# What if the plant is nonlinear ??

$$[X^*(k), U^*(k)] = \arg \min_{X(k),U(k)} \left[ \|x(N+1\,|\,k)\|_P^2 + \|u(N\,|\,k)\|_R^2 + \sum_{i=1}^{N} \|x(i\,|\,k)\|_Q^2 + \|u(i-1\,|\,k)\|_R^2 \right]$$

$$\text{s.t.} \quad \begin{aligned} & Gx(i\,|\,k) + Hu(i\,|\,k) \leq 1, \qquad \forall i = 0,1,\cdots,N-1, \\ & x(0\,|\,k) = x(k), \\ & x(i+1\,|\,k) = f\big(x(i\,|\,k),\, u(i\,|\,k)\big), \qquad \forall i = 0,1,\cdots,N-1, \\ & x(N\,|\,k) \in \mathcal{X}_f \subset \Omega\,. \end{aligned}$$

- Both state sequence and control sequence are selected as decision variables satisfying
  - state and control constraints
  - initial equality constraint
  - dynamic equality constraint (state equation)
  - terminal constraints

# Let’s try a nonlinear example

$$x(k+1) = f(x(k), u(k))$$

$$f(x(k), u(k)) = \begin{bmatrix} 1.1x_1(k) + \sin(x_2(k)) \\ 0.12x_1^2(k) + x_2(k) + 0.079u(k) \end{bmatrix}$$

s.t. $-4 \leq u \leq 4$

```
%This is the nonlinear function for MPC demo

function xplus = nonlinear_func(x, u)

xplus = zeros(2,1);

xplus(1) = 1.1*x(1) +  sin(x(2));

xplus(2) =  0.12* x(1)^2 + x(2) + 0.079*u;


end
```

# Let's try a nonlinear example (cont'd)

$$J(k) = \sum_{i=1}^{N} \|x(i\,|\,k)\|_Q^2 + \|u(i-1\,|\,k)\|_R^2,$$

s.t. $-4 \le u \le 4 \qquad x(0\,|\,k) = x(k) \qquad x(N\,|\,k) = 0$

```
for jj = 1:N
c(dimu*jj) = abs(U(jj)) - 4;                    %Control constraints: -5<=u(i|k)<=5
end

%The vector ceq stores equality constraints
ceq(1:dimx) = X(1:dimx) - nonlinear_func(xcurrent, U(1: dimu)); %initial constraint

for i = 1: N-1
ceq(i*dimx+1: (i+1)*dimx) = X(i*dimx+1: (i+1)*dimx) -...
   nonlinear_func(X((i-1)*dimx+1: i*dimx), U((i-1)*dimu+1: i*dimu));   %dynamic constraint
end

%Terminal equality constraint
ceq(N*dimx+1: N*dimx+dimx) = X((N-1)*dimx+1: N*dimx);
```

```
%This is the objective function for MPC demo
function f = objfun( xx )

global N;

global dimx;
global dimu;
global dimy;


X = xx(1: N*dimx);                          %Predictive state series
U = xx(N*dimx+1: N*dimx+N*dimu);           %Predictive control series

f = 0;

%weighting matrix
Q = eye(dimx);
R = 0.1;

%Cost function
for i = 1: N
   f = f +(X((i-1)*dimx+1: i*dimx)'*Q*X((i-1)*dimx+1: i*dimx)) ...
      + (U((i-1)*dimu+1: i*dimu)'*R* U((i-1)*dimu+1: i*dimu));
end

end
```

# Let’s try a nonlinear example (cont’d)

```
%Run optimization
xi = fmincon(@objFun, xi, Aueq, bueq, Aeq, beq, lb, ub, @nonlinear_constraints, options);

%Receding horizon implementation.
%Only the first step of predicive control is implemented.
u(:,k) = xi(N*dimx+1: N*dimx + dimu);

%model
x(:, k+1) = nonlinear_func(x(:, k), u(:, k));
```

**DEMO**

**说明**

非线性规划求解器。

求以下问题的最小值：

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub, \end{cases}$$

b 和 beq 是向量，A 和 Aeq 是矩阵，c(x) 和 ceq(x) 是返回向量的函数，f(x) 是返回标量的函数。f(x)、c(x) 和 ceq(x) 可以是非线性函数。

# MPC toolbox in MATLAB

- Plant

  Continuous-time —> Discrete-time

```
%plant
Ts = 0.2;          %sampling time for discretization
A = [0 1; 1 1];
B = [0; 1];
C = [1 0];
D = 0;
plant = c2d(ss(A,B,C,D), Ts);
```

- Create an MPC object

```
%mpc controller based on prediction of plant
mpcObj = mpc(plant);
```

`mpcobj = mpc(plant,ts,P,M,W,MV,OV,DV)` specifies the following controller properties. If any of these values are omitted or empty, the default values apply.

- `P` sets the `PredictionHorizon` property.
- `M` sets the `ControlHorizon` property.
- `W` sets the `Weights` property.
- `MV` sets the `ManipulatedVariables` property.
- `OV` sets the `OutputVariables` property.
- `DV` sets the `DisturbanceVariables` property.

# MPC toolbox in MATLAB

```
>> mpcObj

MPC object (created on 08-Jun-2022 14:28:43):
---------------------------------------------
Sampling time:      0.2 (seconds)
Prediction Horizon: 5
Control Horizon:    5

Plant Model:
                                      --------------
      1  manipulated variable(s)   -->|  2 states  |
                                      |            |-->  1 measured output(s)
      0  measured disturbance(s)   -->|  1 inputs  |
                                      |            |-->  0 unmeasured output(s)
      0  unmeasured disturbance(s) -->|  1 outputs |
                                      --------------
Disturbance and Noise Models:
        Output disturbance model: default (type "getoutdist(mpcObj)" for details)
         Measurement noise model: default (unity gain after scaling)

Weights:
        ManipulatedVariables: 1
    ManipulatedVariablesRate: 0
             OutputVariables: 10
                         ECR: 100000

State Estimation:  Default Kalman Filter (type "getEstimator(mpcObj)" for details)

Constraints:
 -4 <= MV1(t+0) <= 4, MV1/rate is unconstrained, -Inf <= MO1(t+1) <= Inf
 -4 <= MV1(t+1) <= 4                             -Inf <= MO1(t+2) <= Inf
 -4 <= MV1(t+2) <= 4                             -Inf <= MO1(t+3) <= Inf
 -4 <= MV1(t+3) <= 4                             -Inf <= MO1(t+4) <= Inf
 -4 <= MV1(t+4) <= 4                                0 <= MO1(t+5) <= 0
>>
```

```
%control constraints
mpcObj.MV(1).Min = -4;
mpcObj.MV(1).Max = 4;

%control horizon
mpcObj.ControlHorizon = 5;
mpcObj.PredictionHorizon = mpcObj.ControlHorizon;

%Weights
mpcObj.Weights.ManipulatedVariables = 1;
mpcObj.Weights.ManipulatedVariablesRate = 0;
mpcObj.Weights.OutputVariables = 10;

%Terminal constraints
Y = struct('Weight',[],'Min',[-0],'Max',[0]);
U = struct('Weight',[],'Min',[],'Max',[]);
setterminal(mpcObj, Y, U);
```

- Parameters of MPC can be set manually.
  - Constraints, terminal constraints,
  - Control/predictive horizon, weights…

# MPC toolbox in MATLAB

- Run MPC using “mpcmove”

DEMO

```
%Ref
r = 0;

t = 0: Ts : 5;
N = length(t);
y = zeros(N,1);
xsys = zeros(N, 2);
u = zeros(N,1);

for i = 1:N
    % simulated plant and predictive model are identical
    y(i) = plant.C*x.Plant;

    xsys(i, :) = x.Plant;

    %Run mpc
    u(i) = mpcmove(mpcObj, x, y(i), r);
end
```

# Nonlinear MPC using MPC toolbox

## Nonlinear MPC

R2020b

As in traditional linear MPC, nonlinear MPC calculates control actions at each control interval using a combination of model-based prediction and constrained optimization. The key differences are:

- The prediction model can be nonlinear and include time-varying parameters.
- The equality and inequality constraints can be nonlinear.
- The scalar cost function to be minimized can be a nonquadratic (linear or nonlinear) function of the decision variables.

Using nonlinear MPC, you can:

- Simulate closed-loop control of nonlinear plants under nonlinear costs and constraints.
- Plan optimal trajectories by solving an open-loop constrained nonlinear optimization problem.

# Nonlinear MPC using MPC toolbox

$$x(k+1) = f(x(k), u(k))$$

$$f(x(k), u(k)) = \begin{bmatrix} 1.1x_1(k) + \sin(x_2(k)) \\ 0.12x_1^2(k) + x_2(k) + 0.079u(k) \end{bmatrix}$$

s.t. $-4 \le u \le 4$

```
%This is the nonlinear function for MPC demo

function xplus = nonlinear_func(x, u)

xplus = zeros(2,1);

xplus(1) = 1.1*x(1) +  sin(x(2));

xplus(2) =  0.12* x(1)^2 + x(2) + 0.079*u;


end
```

```
%Define a nonlinear mpc object
nx = 2;
ny = 2;
nu = 1;
nlobj = nlmpc(nx, ny, nu);

%State equation of the nonlinear plant
nlobj.Model.StateFcn = "nonlinear_func";
%It is not discretized from continuous-time model.
nlobj.Model.IsContinuousTime = false;
```

# Nonlinear MPC using toolbox

- Parameters of the nonlinear MPC can be set manually.
  - Constraints, terminal constraints,
  - Control/predictive horizon, weights…

```
%Control horizon and predictive horizon
nlobj.PredictionHorizon = 10;
nlobj.ControlHorizon = 10;

%Control constraints
nlobj.MV = struct('Min',{-4},'Max',{4});

%Weight of output/state
nlobj.Weights.OutputVariables = [1 1];

%Weight of control input
nlobj.Weights.ManipulatedVariables = [0.1];
```

```
%Terminal constraint
Optimization.CustomEqConFcn = "myEqConFunction";


%This is to specify the terminal equality constraint

function ceq = myEqConFunction(X,U,data,params)

p = data.PredictionHorizon;

ceq = [X(p+1,1) - 0;
      X(p+1,2) - 0;];

end
```

# Run the nonlinear MPC

```matlab
%Run
for i = 1:N
    % simulated plant and predictive model are identical

    %Run mpc
    u(i) = nlmpcmove(nlobj, x(:, i), lastMV);
    lastMV = u(i);

    %Run nonlinear plant
    x(:, i+1)  = nonlinear_func(x(:, i), u(i));

end
```

## Description

`mv = nlmpcmove(nlmpcobj,x,lastmv)` computes the optimal manipulated variable control action for the current time. To simulate closed-loop nonlinear MPC control, call `nlmpcmove` repeatedly.

**DEMO**

# Explicit MPC

- Explicit MPC avoids online computations in “Implicit” MPC.
- Explicit MPC is simply “switches” among affine functions of states.

$$U^*(k) = \begin{bmatrix} u^*(0 \mid k) \\ u^*(1 \mid k) \\ \vdots \\ u^*(N-1 \mid k) \end{bmatrix}$$

$$u^*(0 \mid k) = \begin{cases} F_1 x + g_1, & \text{if } H_1 x \le K_1, \\ F_2 x + g_2, & \text{if } H_2 x \le K_2, \\ \vdots & \\ F_M x + g_M, & \text{if } H_M x \le K_M. \end{cases}$$

# Explicit MPC

- Constrained optimization to be solved:

$$U^* = \arg\min_{U} \left[ \frac{1}{2} x^T Q x + 2x^T F^T U + \frac{1}{2} U^T Y U \right],$$
$$s.t. \quad GU \leq W + Sx$$

- Optimization by using active-set algorithm

For some $x_0 \in \mathscr{X}$

**Active** $\quad G_i U^*(x_0) = W_i + S_i x_0$

**Inactive** $\quad G_j U^*(x_0) \leq W_j + S_j x_0$

# Explicit MPC

- Optimization with Lagrangian multiplier

$$U^* = \arg\min_U \left[\frac{1}{2}x^TQx + 2x^TF^TU + \frac{1}{2}U^TYU + \lambda(\tilde{G}z - \tilde{W} - \tilde{S}x)\right],$$

$$\frac{\partial J}{\partial U} = 0 \quad \Rightarrow \quad QU + Fx + \tilde{G}^T\tilde{\lambda} = 0 \quad \Rightarrow \quad U = Q^{-1}(Fx + \tilde{G}^T\tilde{\lambda})$$

$$\frac{\partial J}{\partial \lambda} = 0 \quad \Rightarrow \quad \tilde{G}U - \tilde{W} - \tilde{S}x = 0$$

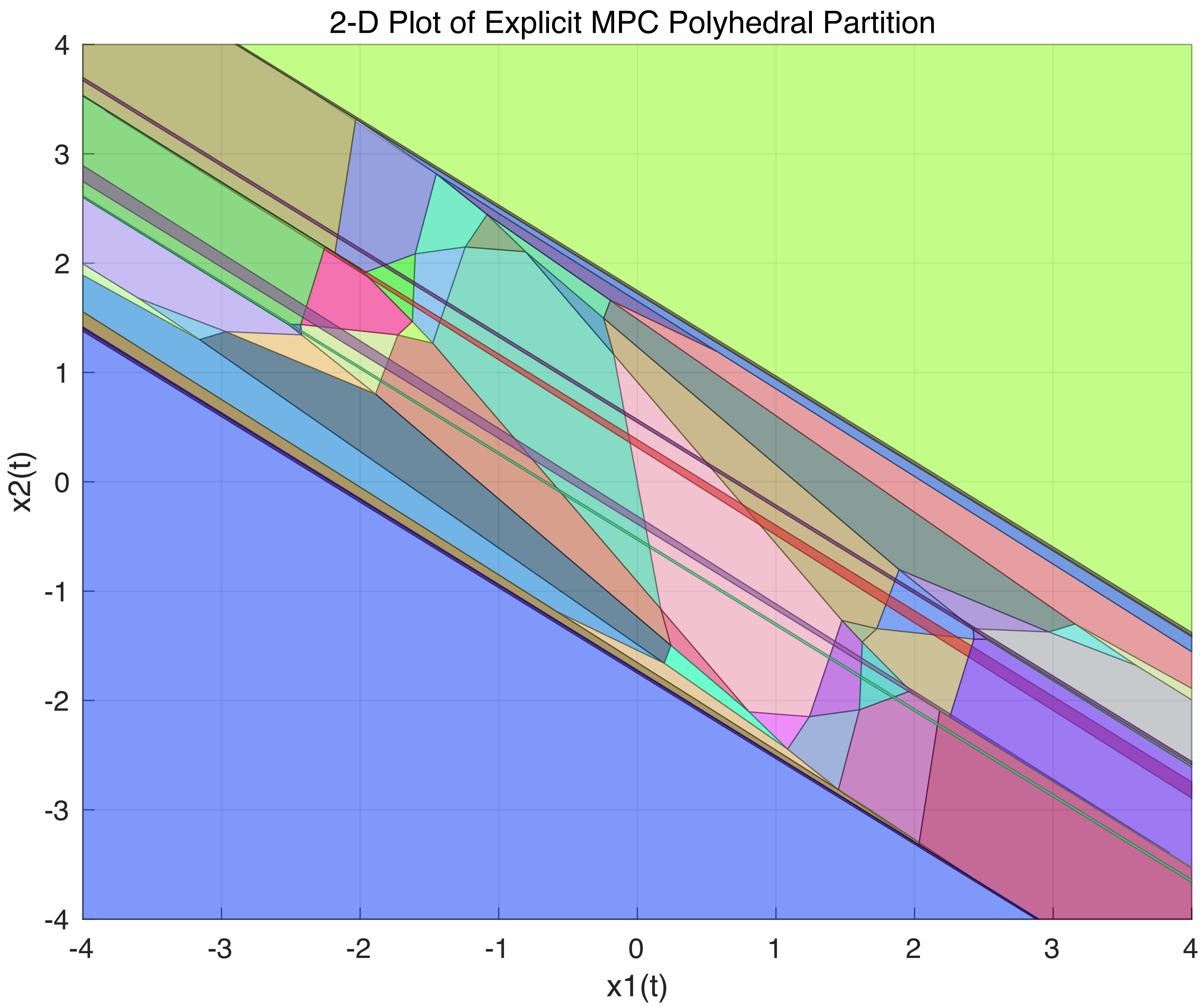
$$\Rightarrow \quad \tilde{\lambda}(x) = -(\tilde{G}Q^{-1}\tilde{G}^T)^{-1}(\tilde{W} + (\tilde{S} + \tilde{G}Q^{-1}F)x)$$

Affine function of $x$

$$\Rightarrow \quad U(x) = Q^{-1}\left[(\tilde{G}Q^{-1}\tilde{G}^T)^{-1}(\tilde{W} + (\tilde{S} + \tilde{G}Q^{-1}F)x) - Fx\right]$$

# Explicit MPC

$$u^*(0\,|\,k) = \begin{cases} F_1 x + g_1, & \text{if } H_1 x \le K_1, \\ F_2 x + g_2, & \text{if } H_2 x \le K_2, \\ \vdots & \\ F_M x + g_M, & \text{if } H_M x \le K_M\,. \end{cases}$$



# Explicit MPC

```matlab
% This file is to generate explicit MPC

range = generateExplicitRange(mpcObj);

%There are 3 states ???
%2 states, and 1 disturbance, as shown by  mpcstate(mpcObj)
range.State.Min = [-4;-4; -1];
range.State.Max = [4; 4; 1];

range.Reference.Min = -4;
range.Reference.Max = 4;

range.ManipulatedVariable.Min = -4;
range.ManipulatedVariable.Max = 4;


mpcobjExplicit = generateExplicitMPC(mpcObj, range);

mpcobjExplicit = simplify(mpcobjExplicit, 'exact');
display(mpcobjExplicit);
```

```matlab
for i = 1:N
    % simulated plant and predictive model are identical
    yExplicit(i) = plant.C*xExplicit.Plant;

     xsysExplicit(i, :) = xExplicit.Plant;

    %Run mpc
    uExplicit(i) = mpcmoveExplicit(mpcobjExplicit, xExplicit, yExplicit(i), r);

end
```

**DEMO**

# Summary

- Coding MPC in MATLAB
  - Linear MPC: constrained quadratic optimization
  - Nonlinear MPC: dynamic equality constraints
- MPC toolbox
  - Linear MPC
  - Nonlinear MPC: more freedom to include nonlinear cost and nonlinear constraints
  - Explicit MPC: avoid online computation, “switching” affine functions of states