docs

MPC-MATLAB-Examples Documentation

Author: youkoutaku

Repository: youkoutaku/MPC-MATLAB-Examples

Date: 2025-12-05

System Model

Target System: Double Integrator

Continuous-time model:

$$\begin{aligned} \dot{x}_1 &= x_2 \quad (\text{position derivative} = \text{velocity}) \\ \dot{x}_2 &= u \quad (\text{velocity derivative} = \text{acceleration}) \end{aligned}$$

State-space representation:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Discretization (sampling time ts = 0.05~0.1 seconds):

```
sys_d = c2d(ss(A, B, C, D), ts);
Ad = sys_d.A;  % Discrete system matrix
Bd = sys_d.B;  % Discrete input matrix
```

File Structure

```
MPC-MATLAB-Examples/
├── README.md                          # Documentation (Japanese)
├── README_EN.md                          # Documentation (English)
```

```
├── docs/
│   ├── docs.pdf                    # Detailed Documentation (English)
│   ├── docs_jp.pdf                    # Detailed Documentation (Japanese)
│   ├── docs.md                     # Detailed Documentation (English)
│   └── docs_jp.md                  # Detailed Documentation (Japanese)
├── LICENSE                         # MIT License
├── lib/                            # Helper functions & libraries
│   ├── QP_Transform.m             # QP standard form conversion
│   └── Augmented_System_Input.m   # Augmented system (legacy)
├── test/                           # Test scripts
│   ├── Check_Constraints.m        # Constraint verification tool
│   └── Check_QP_Transform.m       # QP transformation test
├── MPC_trajectory_main.m           # Trajectory tracking MPC (recommended)
├── MPC_con_main.m                  # Constrained MPC
├── MPCvsLQR_QP_main.m              # MPC vs LQR comparison
└── MPC_Toolbox.m                   # Toolbox verification
```

Main Simulation Files

MPC_trajectory_main.m (Recommended)

Purpose: Basic MPC implementation for time-varying reference trajectory tracking

Theory:

- Prediction horizon:

$$N_p = 30$$

- Cost function:

$$J = \sum_{i=1}^{N_p-1} \|x(k+i) - x_r(k+i)\|_Q^2 + \sum_{i=1}^{N_p} \|\Delta u(k+i)\|_R^2 + \|x(k+N_p) - x_r(k+N_p)\|_{Q_f}^2$$

- Reference trajectory: Sinusoidal

$$x_r = \begin{bmatrix} 5\sin(0.5t) + 10 \\ 2.5\cos(0.5t) \end{bmatrix}$$

- QP formulation:

$$\min_{\Delta U} \; 0.5 \cdot \Delta U^T H \Delta U + f^T \Delta U$$

Usage:

```
run('MPC_trajectory_main.m')
```

MPC_con_main.m (Constrained MPC)

Purpose: MPC with input, state, and rate constraints

Constraints:

- Input: $u \in [-5, 5]$ m/s²
- Velocity: $v \in [-10, 10]$ m/s
- Position: $x \in [0, 20]$ m
- Rate: $\Delta u \in [-3, 3]$ m/s³

Usage:

```
run('MPC_con_main.m')
run('test/Check_Constraints.m')  % Verify constraints
```

MPCvsLQR_QP_main.m (Performance Comparison)

Purpose: Compare MPC and LQR control performance

Theory:

- MPC: Forward optimization with prediction
- LQR: Infinite horizon optimal regulator $u = -K(x - x_r)$
- Fair comparison with identical weight matrices

Usage:

```
run('MPCvsLQR_QP_main.m')
```

Helper Functions

QP_Transform.m

Purpose: Convert MPC cost function to QP standard form

Input:

$$A, B, Q, R, Q_f, N_p$$

Output:

$$A_p, B_p, Q_p, R_p, F, H$$

Prediction model:

$$X = A_p \cdot x(k) + B_p \cdot \Delta U$$

Usage:

```
addpath('lib')
[Sx, Su, Qp, Rp, F, H] = QP_Transform(A, B, Q, R, Qf, Np);
```

Check_Constraints.m

Location: `test/Check_Constraints.m`
Purpose: Detailed verification of constraint satisfaction

Features:

- Check constraint violations at all time steps (tolerance 1e-6)
- Statistics on input, velocity, position, and rate constraints
- Count constraint activations
- Generate 3 verification plots

Usage:

```
run('test/Check_Constraints.m')
```

Usage Guide

Basic Execution

Important: Add the lib folder to the path first:

```
addpath('lib')  % Add helper functions
```

1. **Trajectory Tracking MPC:**

```
run('MPC_trajectory_main.m')
```

2. **Constrained MPC:**

```
run('MPC_con_main.m')
run('test/Check_Constraints.m')  % Verify constraints
```

3. **Performance Comparison:**

```
run('MPCvsLQR_QP_main.m')  % MPC vs LQR
run('MPC_Toolbox.m')       % Manual vs Toolbox
```

Parameter Tuning

Adjustable parameters in each file:

```
% Prediction horizon
Np = 30;  % Short → faster computation, Long → better performance

% Weight matrices
Q_mpc = [1 0; 0 0.2];   % State tracking weight (larger → better tracking)
R_mpc = 10;             % Control input weight (larger → smoother input)

% Sampling time
ts = 0.05;  % Smaller → better accuracy, Larger → lower computation

% Constraints (MPC_con_main.m only)
```

```
u_min = -5; u_max = 5;          % Input constraint
x2_min = -10; x2_max = 10;      % Velocity constraint
x1_min = 0; x1_max = 20;        % Position constraint
du_min = -3; du_max = 3;        % Rate constraint
```

Theoretical Background

MPC Fundamentals

Prediction Model:

$$x(k+1) = A \cdot x(k) + B \cdot u(k)$$

Cost Function:

$$J = \sum_{i=1}^{N_p-1} \|x(k+i) - x_r(k+i)\|_Q^2 + \sum_{i=1}^{N_p} \|\Delta u(k+i)\|_R^2 + \|x(k+N_p) - x_r(k+N_p)\|_{Q_f}^2$$

Optimization:

- Unconstrained: Analytical solution

$$\Delta U^* = -H^{-1} f$$

- Constrained: Numerical solution via `quadprog()`

Constrained MPC Formulation

Integrated QP Problem:

$$\begin{aligned} \min_{\Delta U} \quad & 0.5 \cdot \Delta U^T H \Delta U + f^T \Delta U \\ \text{s.t.} \quad & A_{\text{ineq}} \Delta U \leq b_{\text{ineq}} \end{aligned}$$

MPC vs LQR Comparison

Property	MPC	LQR
Optimization Horizon	Finite (N_p)	Infinite
Constraint Handling	Direct	Difficult (saturation)

Property	MPC	LQR
Computational Load	High (online QP)	Low (pre-computed gain)
Prediction	Future reference	Current error only
Optimality	Local optimal	Global optimal (linear)

Important Notes

1. Discrete vs Continuous Time Systems

Always discretize using `c2d()` before simulation:

Correct:

```
sys_d = c2d(ss(A_c, B_c, C, D), ts);
A = sys_d.A;  % Use this
B = sys_d.B;  % Use this
```

Incorrect (avoid):

```
x(k+1) = x(k) + ts * (A * x(k) + B * u(k));  % Low accuracy
```

2. Initial State and Constraint Compatibility

The initial state must satisfy constraints:

```
% Constraint
x2_min = -3; x2_max = 3;  % Velocity constraint

% Good initial state
x0 = [10; 2];  % Velocity 2 m/s (within constraint)

% Bad initial state
x0 = [10; 5];  % Velocity 5 m/s (violates constraint!) → QP infeasible
```

3. Prediction Horizon Selection

Short horizon:

- ✓ Fast computation
- ✓ Suitable for real-time control
- ✗ Slightly worse performance

Long horizon:

- ✓ Better control performance
- ✓ Anticipates and avoids constraints
- ✗ Increased computation time $O(N_p^3)$