

# MPC-MATLAB-Examples ドキュメント

著者: youkoutaku

リポジトリ: [youkoutaku/MPC-MATLAB-Examples](https://github.com/youkoutaku/MPC-MATLAB-Examples)

日付: 2025年12月5日

---

## システムモデル

対象システム：二重積分系

連続時間モデル：

$$\begin{aligned}\dot{x}_1 &= x_2 && \text{(位置の微分 = 速度)} \\ \dot{x}_2 &= u && \text{(速度の微分 = 加速度)}\end{aligned}$$

状態空間表現：

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

離散化（サンプリング時間  $ts = 0.05\sim0.1$ 秒）：

```
sys_d = c2d(ss(A, B, C, D), ts);
Ad = sys_d.A; % 離散化システム行列
Bd = sys_d.B; % 離散化入力行列
```

---

## ファイル構成

```
MPC-MATLAB-Examples/
├── README.md                      # 本ドキュメント
├── LICENSE                         # MITライセンス
└── lib/
    ├── QP_Transform.m               # QP標準形変換関数
    └── Augmented_System_Input.m   # 拡大系構築（レガシー）
```

```

└── test/
    ├── Check_Constraints.m          # テストスクリプト
    ├── Check_QP_Transform.m        # 制約検証ツール
    └── MPC_trajectory_main.m       # QP変換テスト
    └── MPC_main.m                  # 軌道追従MPC (推奨)
    └── MPCvsLQR_QP_main.m         # 制約付きMPC
    └── MPC_Toolbox.m              # MPC vs LQR比較

```

## 1. メインシミュレーションファイル

### MPC\_trajectory\_main.m 推奨

目的： 時変参照軌道追従のための基本的なMPC実装

理論：

- 予測区間

$$N_p = 30$$

- コスト関数：

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

- 参照軌道：正弦波

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

- QP定式化：

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

使用方法：

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

## MPC\_con\_main.m 制約付きMPC

目的：入力・状態・変化率制約を考慮したMPC

理論：

- 制約条件：
  - 入力制約： $u \in [-5, 5] \text{ m/s}^2$
  - 速度制約： $v \in [-10, 10] \text{ m/s}$
  - 位置制約： $x \in [0, 20] \text{ m}$
  - 変化率制約： $\Delta u \in [-3, 3] \text{ m/s}^3$
- 制約付きQP：

$$\min_{\Delta U} J \quad \text{s.t.} \quad A_{\text{ineq}} \Delta U \leq b_{\text{ineq}}$$

- 制約処理：累積和行列  $T_{\text{sum}}$  による入力制約、選択行列  $C_x$  による状態制約

使用方法：

```
run('MPC_con_main.m')
```

検証ツール：

```
run('test/Check_Constraints.m') % 制約充足性の詳細検証
```

---

## MPCvsLQR\_QP\_main.m 性能比較

目的：MPCとLQRの制御性能比較

理論：

- MPC：予測制御による前向き最適化
- LQR：無限区間最適レギュレータ  $u = -K(x - x_r)$
- 同一重み行列で公平比較
- 入力制約： $\pm 5 \text{ m/s}^2$  飽和処理

使用方法：

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

## MPC\_Toolbox.m ツールボックス検証

目的： MATLABツールボックスとの実装検証

理論：

- MATLAB MPC Toolbox の mpc() オブジェクト使用
- 重み設定、予測区間、制約を手動実装と同一に設定
- 手動QP実装との結果一致を確認

使用方法：

```
run('MPC_Toolbox.m')  
% 注意：MATLAB MPC Toolboxが必要
```

## 2. 補助関数・ツール

### QP\_Transform.m QP変換

目的： MPCコスト関数をQP標準形に変換

理論：

入力：

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

出力：

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

予測モデル：

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

コスト関数変換：

$$\begin{aligned} J &= (A_p x + B_p \Delta U - X_r)^T Q_p (A_p x + B_p \Delta U - X_r) + \Delta U^T R_p \Delta U \\ &= 0.5 \cdot \Delta U^T H \Delta U + f^T \Delta U + \text{const} \end{aligned}$$

ここで：

- 状態予測行列:

$$A_p = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix}$$

- $B_p$  = 下三角構造の制御予測行列
- $Q_p = \text{diag}(Q, Q, \dots, Q_f)$
- $R_p = \text{diag}(R, R, \dots, R)$
- $F = B_p^T Q_p A_p$
- $H = B_p^T Q_p B_p + R_p$

使用方法：

```
% libフォルダにパスを追加  
addpath('lib')  
[Sx, Su, Qp, Rp, F, H] = QP_Transform(A, B, Q, R, Qf, Np);
```

---

### Check\_Constraints.m 制約検証ツール

場所： test/Check\_Constraints.m

目的： 制約付きMPCの制約充足性を詳細検証

機能：

- ・全ステップでの制約違反チェック（許容誤差 1e-6）
- ・入力、速度、位置、変化率の制約充足統計
- ・制約到達回数のカウント
- ・3つの検証プロット生成

使用方法：

```
cd test  
run('Check_Constraints.m')  
% または  
run('test/Check_Constraints.m')
```

出力：

- ・制約違反の詳細レポート（ステップ番号、値、違反量）
- ・制約統計サマリー
- ・検証用3プロット（入力・速度・変化率 vs 制約）

---

### Augmented\_System\_Input.m レガシー

場所： lib/Augmented\_System\_Input.m 目的： 入力を含む拡大系の構築（目標値のダイナミクスまで考慮する）

注意： 現在の実装では使用していません。参考用に保存。

---

### Check\_QP\_Transform.m 単体テスト

場所： test/Check\_QP\_Transform.m 目的： QP\_Transform関数の単体テスト

---

使用方法

基本的な実行手順

重要： 最初にlibフォルダをパスに追加してください：

```
addpath('lib') % 補助関数をパスに追加
```

## 1. 軌道追従MPC :

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

## 2. 制約付きMPC :

```
run('MPC_con_main.m')
run('test/Check_Constraints.m') % 制約検証
```

## 3. 性能比較 :

```
run('MPCvsLQR_QP_main.m') % MPC vs LQR
run('MPC_Toolbox.m') % 手動実装 vs ツールボックス
```

## パラメータ調整

各ファイルで調整可能なパラメータ :

```
% 予測区間
Np = 30; % 短い → 計算速度向上、長い → 制御性能向上

% 重み行列
Q_mpc = [1 0; 0 0.2]; % 状態追従重み (大 → 追従精度向上)
R_mpc = 10; % 制御入力重み (大 → 滑らかな入力)

% サンプリング時間
ts = 0.05; % 小さい → 精度向上、大きい → 計算負荷低減

% 制約 (MPC_con_main.m のみ)
u_min = -5; u_max = 5; % 入力制約
x2_min = -10; x2_max = 10; % 速度制約
x1_min = 0; x1_max = 20; % 位置制約
du_min = -3; du_max = 3; % 变化率制約
```

---

## 理論背景

## 1. MPC の基本原理

予測モデル：

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

予測区間での状態：

$$\begin{aligned} X &= \begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+N_p) \end{bmatrix} \\ &= S_x \cdot x(k) + S_u \cdot \Delta U + I_u \cdot u(k-1) \end{aligned}$$

コスト関数：

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

最小化：

$$\min_{\Delta U} J(\Delta U)$$

最適化：

- 二次計画法 (QP) による解法
- 制約なし：解析解  $\Delta U^* = -H^{-1}f$
- 制約付き：`quadprog()` による数値解

---

## 2. 制約付きMPCの定式化

制約の種類：

1. 変化率制約：  $\Delta u_{\min} \leq \Delta u(k+i) \leq \Delta u_{\max}$

$$A_{du} = \begin{bmatrix} I \\ -I \end{bmatrix}, \quad b_{du} = \begin{bmatrix} \Delta u_{\max} \cdot \mathbf{1} \\ -\Delta u_{\min} \cdot \mathbf{1} \end{bmatrix}$$

2. 入力制約 :  $u_{\min} \leq u(k-1) + \sum \Delta u \leq u_{\max}$

$$A_u = \begin{bmatrix} T_{\text{sum}} \\ -T_{\text{sum}} \end{bmatrix}, \quad b_u = \begin{bmatrix} u_{\max} \cdot \mathbf{1} - u(k-1) \\ \dots \end{bmatrix}$$

$T_{\text{sum}} = \text{tril}(\mathbf{1}_{N_p \times N_p})$  (累積和行列)

3. 状態制約 :  $x_{\min} \leq x(k+i) \leq x_{\max}$

$$A_x = \begin{bmatrix} C_x S_u \\ -C_x S_u \end{bmatrix}$$

$C_x$  = 状態選択行列 (位置・速度を個別に選択)

統合QP問題 :

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

ここで :

$$A_{\text{ineq}} = \begin{bmatrix} A_{du} \\ A_u \\ A_x \end{bmatrix} \quad (\text{全制約を統合})$$


---

### 3. MPCとLQRの比較

特性	MPC	LQR
最適化区間	有限区間 ( $N_p$ )	無限区間
制約対応	直接対応可能	困難 (飽和処理)
計算負荷	高 (オンラインQP)	低 (事前計算ゲイン)
予測能力	将来の参照を考慮	現在の誤差のみ
最適性	局所最適	大域最適 (線形系)

使い分け :

- 制約が重要 → MPC
- 高速制御が必要 → LQR

- 複雑な軌道追従 → MPC
- 

## 重要な注意事項

### 1. 離散時間システムと連続時間システム

離散化の重要性：

連続時間システムを直接使用すると誤差が発生します。必ず `c2d()` で離散化してください。

正しい実装：

```
% システム定義 (連続時間)
A_c = [0 1; 0 0];
B_c = [0; 1];

% 離散化
sys_d = c2d(ss(A_c, B_c, C, D), ts);
A = sys_d.A; % これを使用
B = sys_d.B; % これを使用

% シミュレーション
for k = 1:N
    u(k) = ... % MPC or LQR
    x(k+1) = A * x(k) + B * u(k);
end
```

誤った実装 (避けるべき)：

```
% 前進差分近似 (精度が低い)
x(k+1) = x(k) + ts * (A * x(k) + B * u(k));
```

### 2. 最適ゲインの計算

推奨：離散系で直接計算

```

sys_d = c2d(sys_c, ts);
[Sx, Su, Qp, Rp, F, H] = QP_Transform(sys_d.A, sys_d.B, Q, R, Qf, Np);
[K, S, e] = dlqr(sys_d.A, sys_d.B, Q, R); % LQR

```

### 避けるべき：連続系で計算後に変換

```

% 連続系LQR → 離散化は非推奨
[K_c, S_c, e_c] = lqr(A_c, B_c, Q, R);
% K_c を離散系に変換する処理が複雑で誤差の原因

```

### 3. 初期状態と制約の整合性

制約付きMPCでは、初期状態が制約を満たす必要があります：

```

% 制約
x2_min = -3; x2_max = 3; % 速度制約

% 良い初期状態
x0 = [10; 2]; % 速度 2 m/s (制約内)

% 悪い初期状態
x0 = [10; 5]; % 速度 5 m/s (制約違反！) → QP不可解になる

```

### 4. 予測区間の選択

短い予測区間：

- ✓ 計算が速い
- ✓ リアルタイム制御に適する
- ✗ 制御性能がやや低下

長い予測区間：

- ✓ 制御性能が向上
- ✓ 制約を先読みして回避

- $\times$  計算時間が増加  $O(N_p^3)$
- 

## 5. quadprog収束失敗への対処

`exitflag` ≠ 1 の場合の処理：

```
if exitflag ~= 1
    warning('QP failed at step %d', k);

    % 対処法1：重み行列調整
    % 初期誤差が大きい場合に、終端コストの重みと入力重みを上げる

    % 対処法2：制約を緩和
    % 状態制約の範囲を広げる

    % 対処法3：予測区間を増やす/減らす
    % Np を調整して再試行
end
```