Vol.2016-ARC-218 No.9
Vol.2016-SLDM174 No.9
2016/1/19

# モデル予測制御を対象としたメニーコアプロセッサ向け投機実行法の制御性能評価

藤井 卓[1,a)]　小野　貴継[2]　井上　弘士[2]　大塚　敏之[3]

**概要**：本稿では，これまでに提案したモデル予測制御（MPC: Model Predictive Control）の投機実行方式を対象とした制御可能性の評価を行う．提案方式では，MPC の主要処理となる最適制御問題の実行において入力値を予測し，それに基づく投機実行を行う．この入力値予測の精度が低い場合，本来の目的であるシステムの制御が収束しない，または，制御が完了するまでの所要時間が長くなる可能性がある．評価の結果，高い演算精度が求められる実行期間が存在する場合には，提案方式の適用により制御可能性が損なわれることが明らかになった．

## 1. はじめに

近年，自動車やロボットなど様々なシステムを制御する手法として，モデル予測制御 (MPC:Model Predictive Control) が注目されている．MPC では，システムの挙動をあらわす動的モデルを利用して将来のシステム状態を予測し，それに基づきアクチュエータの操作量を決定する．多数の状態変数を扱うことができ，非線形な挙動を示すシステムに対しても適用できるといった利点を有する．しかしながら，MPC の主要処理である最適制御問題の計算は演算負荷が極めて高く，定められたサンプリング周期毎に実行を完了するリアルタイム性の保証が課題となっている．

我々は，これまでに MPC の高速化を目的とした新しいメニーコア活用法を提案した [1], [3], [4]．本実行法では，MPC が最適制御問題を周期的に計算する点に着目し，最適制御問題を 1 個のプロセッサコアで投機的に実行する．そして，多数のプロセッサコアで各最適制御問題を並列実行し，MPC の実行時間短縮を狙う．投機実行においては，MPC がシステムの将来の状態値を予測する点に着目し，この状態予測値を最適制御問題を解く際の入力として用いる．このような入力値予測により，理論上は最適制御問題の実行時間をほぼゼロに短縮できる．しかしながらその反面，入力値の予測誤差が大きい場合には制御の高速性と安定性（以降，制御性能と呼ぶ）に悪影響を与える可能性がある．これまでの研究では，提案方式の性能モデリングを行い，MPC アプリケーションの実行により採取したトレースデータに基づく有効性評価を実施した．しかしながら，実行中に実施される動的な状態値予測の精度が制御性能に与える影響は明らかになっていない．

そこで，本稿では提案するモデル予測制御向けメニーコア実行法のプロトタイプを実装し，MPC アプリケーションを対象とした制御可能性評価を行う．まず，最適制御問題の投機実行に必要となる予測入力値の管理法を検討する．そして，汎用プロセッサを対象としたソフトウェア実装を行い，MPC アプリケーションを実行することで，動的な入力値予測が制御性能に与える影響を解析する．なお，本研究では制御性能の評価を目的とするため，メニーコアを想定した複数最適制御問題の並列実行は想定しない（つまり，逐次実行において入力予測値を用いた投機実行を行った場合の制御性能評価となる）．

以降，第 2 節ではモデル予測制御の概要を述べる．第 3 節では投機実行法を説明し，第 4 節で実装の詳細を示す．第 5 節では MPC 応用を対象とした制御性能評価を行い，最後に第 6 節で本稿をまとめる．

## 2. モデル予測制御

モデル予測制御（Model Predictive Control: MPC）とは，制御対象の動的モデルを利用してシステムの状態値を予測し，操作量を求める制御手法である [5]．制御対象の現在の状態を示す観測値は，センサを介してサンプリング周期毎にコントローラに入力される．コントローラは，それぞれの入力に対応する出力（操作量）を求める計算（操作量決定のための計算）を次の入力が到達するまでに完了させなければならない．次入力が到着するまでにこの計算

[1] 九州大学大学院システム情報科学府情報知能工学専攻
[2] 九州大学大学院システム情報科学研究院
[3] 京都大学大学院情報学研究科システム科学専攻
[a)] taku.fujii@cpc.ait.kyushu-u.ac.jp

Vol.2016-ARC-218 No.9
Vol.2016-SLDM174 No.9
2016/1/19

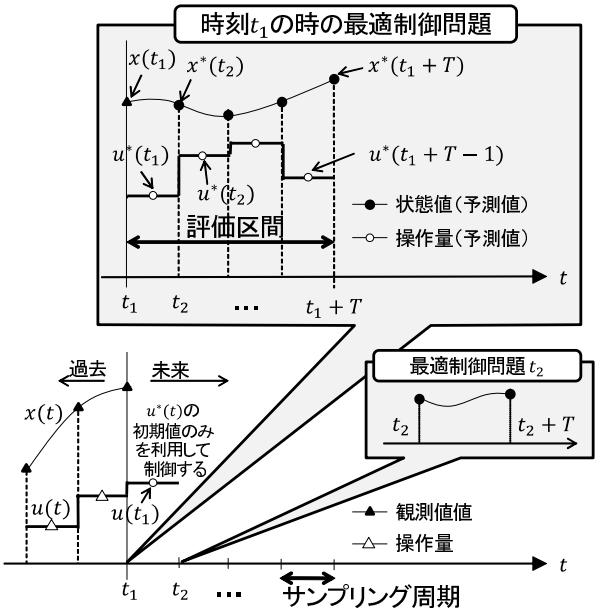


図 1　時刻 $t_1$ におけるモデル予測制御

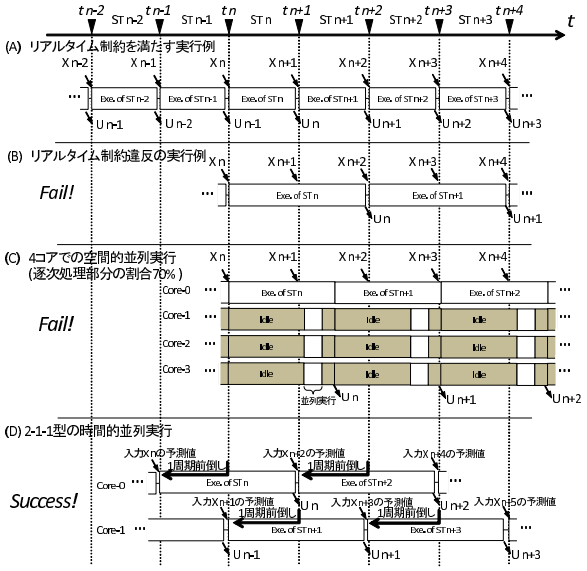


図 2　従来手法と提案手法の処理

が終わらなかった場合はデッドライン違反となる．ここで操作量決定のための計算とは，最適化問題（以下，サンプリング周期毎に解くべき処理の内容を**最適制御問題**と呼称する）を解くことである．制御対象の挙動を示す動的モデルを正確に定式化することで，制御システムの過渡および定常状態の振る舞いを正確に制御することが可能となる．MPC の特出すべき特徴は，所望の区間の最適な操作量を決定するために，その区間以上の有限時間未来（評価区間: $T$）まで最適な操作量を求めることである．

**図 1** に時刻 $t_1$ におけるモデル予測制御の例を示す．横軸は，時間軸であり時刻 $t_1$ までの過去の観測値が $x(t)$，操作量が $u(t)$ の軌跡で示されている．時刻 $t_1$ 以降の将来の操作量を求めるための処理が各サンプリング周期毎に吹き出しで示している最適制御問題である．コントローラは，観測値 $x(t_1)$ を入力として受け取り，対応する出力 $u(t_1)$ を計算するために最適制御問題を解き始める．この処理により，評価区間内において制御目標を達成するために最適な状態値 $x^*(t_2), x^*(t_3), ..x^*(t_1+T)$ を予測し，一連の操作量 $u^*(t_1), u^*(t_2), ..u^*(t_1+T-1)$ を算出する．ここでの状態値 $x^*(t)$ と操作量 $u^*(t)$ はあくまで予測値であり，実際のシステムの観測値 $x(t)$ と操作量 $u(t)$ とは異なることに注意されたい．処理が終了すると予測した操作量の初期値 $u^*(t_1)$ のみを実際の操作量 $u(t_1)$ として出力し，制御対象はこの操作量に従い制御される．時刻 $t_2$ も，同じ長さの評価区間（$[t_2, t_2+T]$）内の状態値と操作量が最適計算され，その初期値 $u^*(t_2)$ のみが操作量として出力される．以下，同様の処理が繰り返される．

## 3. メニーコア投機実行法

### 3.1 入力予測に基づく最適制御問題の投機的実行

これまでに我々は，MPC アプリケーションにおける最適制御問題を複数のプロセッサコアで投機的に実行する**時間的並列実行方式**を提案した [1], [3], [4]．これは，本来操作量を求めるために算出される状態の予測値を最適制御問題単位の投機実行の入力予測値として利用するものである．

リアルタイムシステムでの MPC の最適制御問題は，**図 2**(A) に示すように次時刻の入力が到達するまでに完了させなければならない．ここで $x_n$ は時刻 $t_n$ にコントローラへ入力される観測値を示す．$ST_n$ は，時刻 $t_n$ からのサンプリング周期幅の区間を示す識別子であり，$x_n$ を入力として算出された操作量を $u_n$ で表す．また，$x_n$ を入力として受け取り $t_{n+1}$ までに計算を完了させなければならない処理を最適制御問題 $ST_n$（Exe. of $ST_n$）と呼ぶ．図 2(A) は，サンプリング周期毎に到達する入力に対し，次入力が到達するまでにシングルコア実行を完了させた場合であり，リアルタイム性を保証できる例である．しかしながら，現実的にはシングルコア性能が全く不十分であり，図 2(B) で示すように，リアルタイム制約を満たすことができない．この例では，各入力に対する最適制御問題の計算時間が，サンプリング周期の約 2 倍を要している．

従来の一般的な並列処理では，与えられたプログラムを複数の独立した部分問題に分割し，これらを複数コアで同時実行する．本稿では，このような並列実行方式を**空間的並列処理**と呼ぶ．また，実行の単位に関して，最適制御問題を**プロセス**，これを分割した部分問題を**スレッド**とする．図 2(B) の処理の 70%が逐次処理部分であると仮定した場合，4 コアでスレッドレベル並列化を施した実行例を

図 2(C) に示す．空間並列処理では，1 つのプロセス中で複数のスレッドを生成し，各スレッドに対してプロセッサコアを割当てることにより並列処理を実現する．空間並列処理によっても多少の性能向上は達成できるが，逐次処理部分の実行時には，多数のプロセッサコアがアイドル状態となり，効率的な性能改善は見込めない．

これに対し，我々が提案している MPC 向けメニーコア実行法は，図 2(D) に示すように並列化されていない最適制御問題をパイプライン的にオーバラップ実行する（すなわち，スレッドによる空間並列化は行わない）．本稿ではこれを**時間的並列処理**と呼ぶ．時間的並列処理の特徴を以下に示す 3 種類の数字の組を用いて，x-y-z 型と表現する．

x: 処理の実行に要する全プロセッサコア数.
y: 各周期で行う投機実行のための入力予測値の候補数. 詳細は第 3.2 節で説明する．
z: 投機の度合い. 真の入力値が到着する時刻より何周期前倒しで投機実行を行うかを示す．

図 2(D) は，2-1-1 型の時間並列実行を表す．図 2(B) と比較すると，各周期で行う処理は 1 周期前倒して実行され，見かけ上 2 倍の性能向上を達成している．たとえば，コア 0 において本来時刻 $t_n$ に計算を開始する処理（最適制御問題 $ST_n$）は，投機実行により時刻 $t_{n-1}$ から計算を開始する．時刻 $t_{n-1}$ の時点では，観測値 $x_n$ をセンサーより得ることが不可能なため，$x_n$ の予測値を利用して計算を開始する．同様に，コア 1 は $x_{n+1}$ の値を時刻 $t_n$ の時点で予測し，最適制御問題 $ST_{n+1}$ の投機実行を行う．

MPC は，サンプリング周期より大幅に大きな時間間隔にわたって，制御対象の状態を予測する．これは，最適制御問題 $ST_n$，$ST_{n+1}$，$ST_{n+2}$ のどの処理においても $x_{n+3}$ を独立に予測していることを意味する．ここで，投機実行に利用する入力予測に関しては，最も新しく生成された値を用いるべきである．たとえば，図 2(D) の最適制御問題 $ST_{n+3}$ の入力 $x_{n+3}$ は，もし時刻 $t_{n+2}$ までに最適制御問題 $ST_{n+2}$ の計算が終了していれば，その予測値を選択する．そうでない場合は，最適制御問題 $ST_{n+1}$ の値が最も新しい予測値として採用される．実際の実行時間（つまり，各最適制御問題の実行におけるレイテンシ）はシングルコア実行（図 2(B)）と変わらないが，投機実行により見かけ上の実行時間（実際の状態値がセンサーに与えられてから処理が完了するまでの時間）を短縮することができ，図 2(D) の例では 2 倍近い性能向上を達成している．

### 3.2 予測精度の向上手法

MPC は，制御対処の数式モデルを用いて状態値を予測するため，入力値を高精度で予測可能であるが，投機の度合いが大きくなるにしたがい予測精度は低下することが予想される．そこで，入力予測値の候補を複数用意する（$y$ を増加する）ことにより入力値の予測精度を向上する．具体的には，MPC による予測値付近に真値が存在すると仮定し，最も新しい MPC の予測値を基準として等間隔近傍の値も候補とする．

本研究で対象とする制御システムは，観測値，操作量が離散値であるデジタル制御系であるとする．入力予測値 $x(t_n)$ とその近傍も含めた入力値候補の集合 $X_{t_n}$ を定式化すると，

$$X_{t_n} = \{x^*{}_{LRC}(t_n) \pm Rd | d, D \in \mathbb{N}^0, 0 \leq d \leq D\} \quad (1)$$

と表される．ただし，$x^*{}_{LRC}(t_n)$，$R$，および $D$ は，最も新しい最適制御問題の解，丸めの単位，および状態値の予測値と観測値のズレを示す**ばらつき**である．ばらつきは，システムの状態の観測値と予測値の差の絶対値の丸め値に対する指標である．たとえば $R = 10^{-2}$ のとき，観測値と予測値の差の絶対値の丸め値が 0，0.01，0.02 とすると，ばらつきはそれぞれ，$D = 0$，$D = 1$，$D = 2$ となる．すなわち，式 (1) において $R * D$ は予測値の近傍範囲を表す．

ここで，予測値と観測値が一致することを**予測ヒット**と定義する（不一致の場合を予測ミスと呼ぶ）．予測ヒット $hit$ の条件は，

$$hit(t, R) = \begin{cases} 1 & (Round(x_{\mathrm{diff\_min}}(t), R) = 0) \\ 0 & (otherwise) \end{cases} \quad (2)$$

$$x_{\mathrm{diff\_min}}(t) = \min_{x_{pred}(t) \in X_t} |x_{pred}(t) - x_{obser}(t)| \quad (3)$$

と表される．ただし，$R$，$x_{\mathrm{diff\_min}}(t)$ は，離散化による丸めの単位，時刻 $t$ における観測値と予測値の差の絶対値が最も小さいものである．

### 3.3 入力値予測ミス時の対応策

サンプリング周期が十分に小さい制御システムが対象の場合には，MPC における高い精度での状態値予測が期待できる．しかしながら，あらゆる制御対象，制御環境，センサー値を想定すると予測ミスは必ず発生する．仮に，正確に予測ができたとしても外乱の影響などを考慮すると，100%正確な制御を保証することはできない．予測ミスへの対処として最も簡単な方法は，計算した結果を無視することである．コントローラの本来の目的は操作量を決定することにあるため，予測値による計算結果を無視した上で何らかの操作量を与えればよい. 操作量の決定方法は，前時刻の操作量を引き続き利用することが考えられる．この手法は，仮想的に制御の一部分でサンプリング周期が延びることと等しい．別の手法としては，入力値に対する出力値を事前に決定しテーブルで保存しておく MAP 制御とのハイブリッド方式が考えられる．MAP 制御は予測ミスした際の例外処理用として利用できる．

Vol.2016-ARC-218 No.9
Vol.2016-SLDM174 No.9
2016/1/19

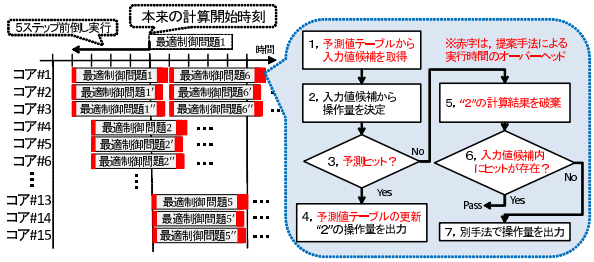


図 3　提案方式の実行フロー

| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | ··· | $t_n$ |
|---|---|---|---|---|---|---|---|
| $x_1$ | ${x_1}^*(t_1)$ | ${x_1}^*(t_2)$ | ${x_1}^*(t_3)$ | ${x_1}^*(t_4)$ | ${x_1}^*(t_5)$ | ··· | ${x_1}^*(t_n)$ |
| $x_2$ | ${x_2}^*(t_1)$ | ${x_2}^*(t_2)$ | ${x_2}^*(t_3)$ | ${x_2}^*(t_4)$ | ${x_2}^*(t_5)$ | ··· | ${x_2}^*(t_n)$ |
| $x_3$ | ${x_3}^*(t_1)$ | ${x_3}^*(t_2)$ | ${x_3}^*(t_3)$ | ${x_3}^*(t_4)$ | ${x_3}^*(t_5)$ | ··· | ${x_3}^*(t_n)$ |
| $x_4$ | ${x_4}^*(t_1)$ | ${x_4}^*(t_2)$ | ${x_4}^*(t_3)$ | ${x_4}^*(t_4)$ | ${x_4}^*(t_5)$ | ··· | ${x_4}^*(t_n)$ |

図 4　予測テーブルの構成

## 4. 実装

### 4.1 実装方針

本研究では，提案方式の制御可能性評価を目的とする．そこで，各最適制御問題の実行に関して入力値予測に基づく投機実行を実装し，複数最適制御問題の並列投機実行サポートは今後の課題とした．64 個以上のコアを搭載した組込み向けメニーコア・プロセッサの使用は容易でないため，本実装では，Intel Xeon プロセッサを用いることを想定する．提案手法を実装するために，pthread によるスレッド並列処理を利用した．提案手法での投機実行では入力予測値と観測値の誤差が生じる [4]．そこで，誤差が存在する予測値を入力として計算された操作量によって適切な制御が可能であるかを検証するため，非線形バネとアーム型振り上げ振り子に対し提案手法を実装し，整定時間を計測する．本実装では以下の前提条件を設ける．

(1)　予測ヒット判定は，最適制御問題の処理終了後に行う．
(2)　入力予測値の中で，観測値に最も近い値を予測ヒットとして扱う．
(3)　各入力に対応する最適制御問題の計算開始時刻は全て同じと仮定する．

### 4.2 実装内容

**図 3** の左側は 15-3-5 型の提案実行の例であり，その際の各最適制御問題の実行フローを右側に示す．各最適制御問題は，予測テーブルから入力予測値を取得し処理を開始する．ここで，**予測テーブル**とは，**図 4** に示すように，過去の最適制御問題の計算結果を利用可能とするためのものであり，制御対象の全ての状態値の各時刻における予測値を保持する．最適制御問題の計算が完了した後，予測ヒットか否かを判定し，ヒットした場合にはその計算結果を対応する予測テーブルに書き込み，さらに操作量を出力する．予測テーブルへは随時書き込みが行われるため，予測テーブル内の値は常に最新となる．また，読み込みも入力予測値が必要な際に随時行うため排他制御は必要ない．予測ミスした際には，最適制御問題の計算結果を破棄する（同じ入力に対応する予測値候補の中に予測ヒットが存在する場合は何も行わない）．同じ入力の予測値候補が全て予測ミスした場合，第 3.3 節で示した手法により操作量を決定出力する．本研究では入力値候補の中で真値に最も近い値を予測ヒットと判定する場合を想定するため，入力候補が全て予測ミスすることは起こりえない（図 3 のフロー 7 に到達しない）．なお，図 3 のフローにおける赤字部分は提案手法の実装にともなう計算オーバーヘッドを表す．

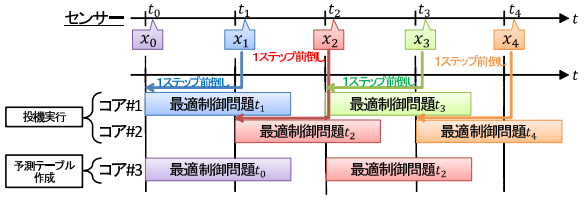


図 5　予測テーブル作成用コア

入力予測値候補に最も新しい MPC の予測値を利用する場合，予測値を入力として処理された最適制御問題の計算結果が次の入力予測値の候補として算出される．そのため，これらの値の差（すなわち誤差）が拡大する可能性がある．そこで，定期的にセンサーから得られた真の観測値を入力として状態値を予測するプロセスを作成する．**図 5** に，予測テーブル作成用プロセスを追加した 2-1-1 型の時間並列実行の例を示す．仮に，予測テーブル作成用プロセスが無い場合，最適制御問題 $t_3$（図 5 中緑色で示す）は最適制御問題 $t_1$（図 5 中青色で示す）によって予測された状態値を入力として受け取り，最適制御問題 $t_4$（図 5 中橙色で示す）は最適制御問題 $t_2$（図 5 中赤色で示す）によって予測された状態値を入力予測値候補とする．この際，最適制御問題 $t_1$，ならびに最適制御問題 $t_2$ も予測値を入力として受け取り実行しているため，予測値と観測値の誤差が伝搬/拡大する．そこで，予測テーブル作成用プロセスを追加することで常にセンサーから得られた真の観測値に基づく最適制御問題の予測値を受け取ることができる．たとえば，最適制御問題 $t_3$ と最適制御問題 $t_4$ の入力予測値は最適制御問題 $t_0$ によって生成される．

## 5. 制御可能性の評価

本節では，提案する入力値予測に基づく投機実行法が制御性能に与える影響を評価する．なお，本研究では制御性能の評価を目的とするため，メニーコアを想定した複数最適制御問題の並列実行は想定しない．すなわち，逐次実行

Vol.2016-ARC-218 No.9
Vol.2016-SLDM174 No.9
2016/1/19

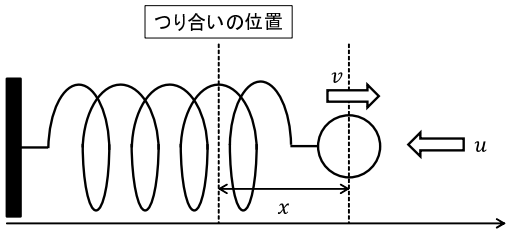


図 6　非線形バネ

において入力予測値を用いた投機実行を行った場合と等価になるが，制御性能の観点からは大きく一般性を失うことはない．

### 5.1　ベンチマークプログラム

#### 5.1.1　非線形バネ

非線形バネ制御シミュレーションプログラムは，自由振動する錘のついたバネに外力を加えることによりつり合いの位置にて停止させるものである．入力拘束を含む非線形バネモデルの詳細は文献 [2] を参照されたい．本プログラムでは，状態値を $x = [\dot{x}, \dot{v}]^T$ とし，釣り合いの位置から錘までの距離を $x$ とする．非線形バネ制御は，初期値が $x = 2$ とし，目標値 $x = 0$（つり合いの位置）に到達するよう錘に外力を与え制御する．評価区間の長さは 1 秒，サンプリング周期は $10^{-2}$ 秒，シミュレーション時間は 20 秒である．非線形バネの模式を図 6 に示す．釣り合いの位置から右側に錘がある場合は $x > 0$，左側に重りがある場合は $x < 0$ とする．錘が正の位置 $(x = 2)$ で静止している状態を初期状態とし，錘がつり合いの位置 $(x = 0)$ で静止している状態を目標状態とする．

#### 5.1.2　アーム型振り上げ振り子

アーム型振り上げ振り子制御シミュレーションプログラムは，アーム型に接続された 2 本のアームを鉛直上向きで静止させるものである．内側のアームのみ DC モータで制御されている．なお，本プログラムに関する詳細は文献 [5] を参照されたい．本プログラムでは，状態値を $x = [\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}]^T$ とし，アーム部と振り子部の絶対角をそれぞれ $\theta_1$ ならびに $\theta_2$ とする．振り子制御は，初期値が $\theta_1 = \theta_2 = \pi$ で，目標値 $\theta_1 = \theta_2 = 0$ に到達するように根元アームの操作量を制御する．根元アームの操作量の決定には，それぞれの絶対角 $\theta_1, \theta_2$ を使用する．評価区間の長さは 0.5 秒，サンプリング周期は $10^{-3}$ 秒，シミュレーション時間は 10 秒である．アーム型振り上げ振り子の模式を図 7 に示す．2 つのアームが鉛直下方向 $(\theta_1 = \theta_2 = \pi)$ で静止している状態を初期状態とし，両アームが鉛直上方向 $(\theta_1 = \theta_2 = 0)$ で静止している状態を目標状態とする．

### 5.2　実験結果：静定時間

本節では，非線形バネ制御ならびに振り子制御の整定時

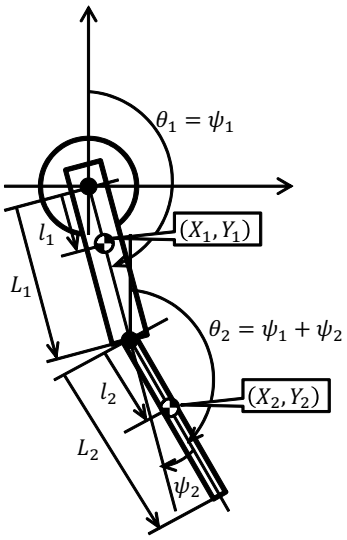


図 7　アーム型振り上げ振り子

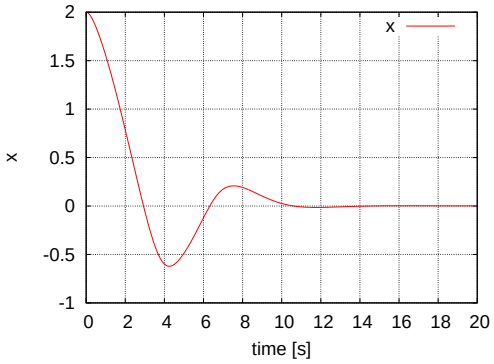


図 8　非線形バネ制御の結果

間について考察を行う．ここで，整定時間を「時刻 $t$ から時刻 $\infty$ において，初期観測値に対する目標値の誤差が ±5% 以内に収まる時刻 $t$ の最小値」と定義する．

図 8 に評価区間分割数（1 つの評価区間において処理される最適制御問題の数に相当）100 における提案手法実装時の非線形バネ制御結果を示す．ここでは，逐次実行に対して要求される性能向上は 2 倍，実行冗長度は 1，実行前倒し度は 2，使用するコア数は 3（最適制御問題処理用に 2 個，予測テーブル作成用に 1 個）とする．実験の結果，図 8 に示すように，9.04 秒でシステム状態を目標へと収束できていることが分かる．

次に，振り子制御に関する評価結果を解析する．ここでは，逐次実行に対して要求される性能向上は 10 倍，実行冗長度は 1，実行前倒し度は 9，使用するコア数は 11（最適制御問題処理用に 10 個，予測テーブル作成用に 1 個）とする．なお，評価区間分割数は 500 である．本ベンチマークを提案方式で実行した場合，振り子制御を安定させることができなかった．その原因を解析したところ，3 秒〜3.7 秒の区間において 0.000001 の誤差も許容することができないことが分かった．このように，極めて高精度な演算が

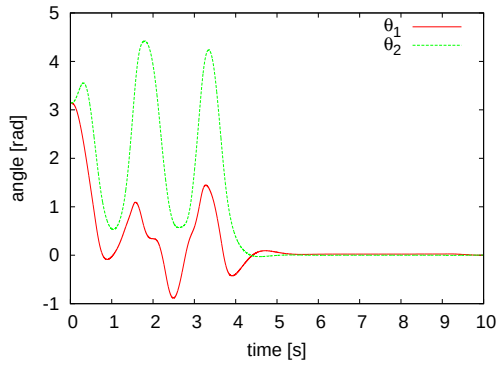


図 9　振り子の制御の結果

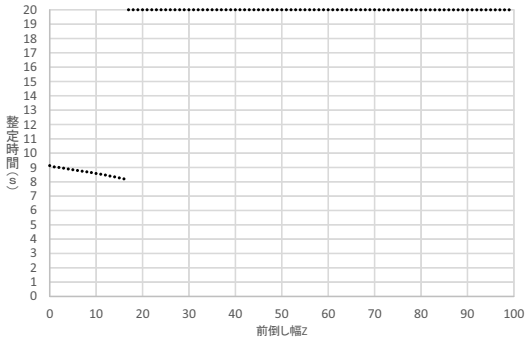


図 10　前倒し幅 z に対する整定時間

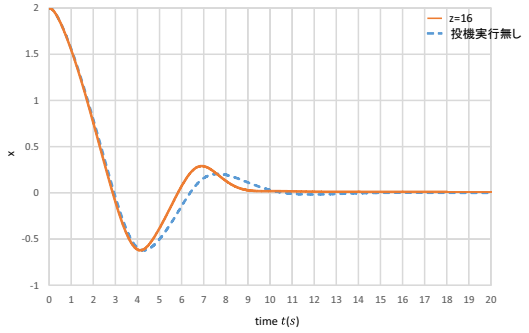


図 11　z=16 と投機実行を実装していない場合の制御結果

要求される状況においては，提案方式の適用は困難である．次に，このような特殊な状況の影響を排除するため，3 秒〜3.7 秒の区間のみ逐次実行とし，それ以外は提案する投機実行を適用し評価を行った．その結果を実験結果を図 **9** に示す．この実験結果より，振り子は安定して制御できていることが確認できる．

### 5.3　実験結果：投機の度合いの影響

本節では，非線形バネ制御を対象とし，投機の度合い z が制御性能に与える影響を考察する．投機の度合い z を 0 から 100 まで増加させた時の整定時間の測定結果を図 **10** に示す．ここで，非線形バネのシミュレーション時間は 20 秒，提案投機実行モデルは x-1-z 型である．また，図中の結果において，整定時間が 20 秒となってる実行では，制御対象がシミュレーション終了時に目標値（$x = 0$）に収束しなかったこと（制御に失敗したこと）を意味する．実験結果より，投機の度合い z を増加させると整定時間が短くなっており，z=16 の時に最小（8.20 秒）となった．また，z を 17 以上に設定した場合には制御対象が目標値に収束しないことが分かる．本来，z が小さいほど観測値と予測値の誤差が小さくなり，制御対象を早く収束させることができると考えられる．しかしながら，実験結果はこの予想に反しており原因を調査中である．

また，整定時間が一番短い場合（z=16）と投機実行を実装していない場合の制御結果を図 **11** に示す．投機実行を実装していない場合シミュレーション終了時に制御対象の状態値が $x = -0.000097$，z=16 の場合では $x = 0.011098$ となった．測定結果より，投機実行による予測値と観測値の誤差が制御性能に悪影響を及ぼしたと考えられる．

## 6.　おわりに

本稿では，これまでに我々が提案したモデル予測制御向け投機実行法に関するシステム制御可能性を評価した．非線形バネ制御アプリケーションに関しては，逐次実行に対して 2 倍の性能要求が求められる際，入力予測値を用いた場合においても整定時間を損なうことなく制御可能であることが分かった．一方，2 種の状態値を有するより複雑なアーム型振り上げ振り子制御アプリケーションでは，ごく短期間での精度低下が原因により制御可能性を失う結果となった．今後，このような高い精度が求められる場合の対策を検討しなければならない．また，本実装では実現できていない複数最適制御問題の並列実行をサポートし，実行性能ならびに制御性能を総合的に評価する予定である．

**謝辞**　本研究を進めるにあたり，活発な議論とご協力を頂いた九州大学井上研究室の皆様に心より感謝の意を表します．なお，本研究は，一部文部科学省科学研究費補助金 26540022 の助成による．

### 参考文献

[1] Satoshi, K., Akihito, I. and Koji, I.: Many-core Acceleration for Model Predictive Control Systems, *Proceedings of the First International Workshop on Many-core Embedded Systems*, pp. 17–24 (2013).
[2] 嘉納秀明：システムの最適理論と最適化，コロナ社 (1987).
[3] 川上哲志，岩永明人，井上弘士：メニーコアプロセッサにおける実時間モデル予測制御のための投機実行法，先進的計算基盤システムシンポジウム論文集，Vol. 2013, pp. 78–82 (2013).
[4] 川上哲志，岩永明人，井上弘士，大塚敏之：モデル予測制御のためのメニーコア投機実行の性能モデリング，研究報告計算機アーキテクチャ（ARC），Vol. 2013, No. 11, pp. 1–7 (2013).
[5] 大塚敏之：モデル予測制御 (発展編, <特集>初学者のための図解でわかる制御工学 II)，システム/制御/情報: システム制御情報学会誌，Vol. 56, No. 6, pp. 310–312 (2012).