Vol.2013-ARC-206 No.11
2013/7/31

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

川上 哲志[1]　岩永 明人[2]　井上 弘士[3]　大塚 敏之[4]

概要：本稿では，入力値予測に基づくメニーコアプロセッサによる投機実行の性能モデルを提案する．提案する性能モデルは，従来の与えられた問題をスレッドに分割し空間方向に展開することで並列に処理することに対し，将来の処理に必要な入力値を予測することで時間的に並列処理する理論モデルである．本稿では，性能モデルを基にメニーコア投機実行の応用として，モデル予測制御（MPC：Model Predictive Control）のリアルタイム処理の実現可能性性を検討する．アーム型振上げ振り子制御システムの例題では，投機実行に必要なプロセッサコア数を算出し，8 コアでリアルタイム処理が可能であることを確認した．

1. はじめに

計算機性能の飛躍的な発展に伴い，多様な制御システムへの応用が進展している．制御システムは，航空機，自動車，船舶などの動力・操舵制御をはじめ，発電機やプラントの製造機器などに適用され，社会基盤に多大な恩恵を与えている．次世代の制御システムにおいては，大規模複雑な非線形システムを実時間で制御できる計算機環境が求められている．

制御器として搭載されるマイクロプロセッサの処理は，一般的にセンサを介した入力値に基づき，制御対象が仕様通りに振舞うための出力値を一定周期で計算する．出力値はアクチュエータを介して制御対象に伝達され，制御対象は出力値に基づく動作を行う．今後の制御システムの実現に向けては，複雑な計算に対して入力値が与えられた時刻から次の入力が到着するまでに処理が完了するリアルタイム制約を満足するための計算機環境が求められている．

一方で，マイクロプロセッサの性能は動作周波数向上が鈍化した 2000 年初頭以降，1 つのチップに複数のプロセッサコアを搭載したマルチコアプロセッサが主流となった．また，微細加工技術の進歩に伴い，例えば 64〜128 個のコアを搭載したメニーコアプロセッサの実用化も現実のものとなりつつある．そしてその応用は，ハイエンド・サーバやスーパーコンピュータといった高性能計算システムのみならず，近年では組込みシステムへも拡大している [1, 2]．一般的にオンチップ並列処理においては，並列性の度合いが性能向上に対して大きな影響を与える．しかしながら，制御システムでは処理を開始するための入力値が時系列で与えられる．そのため，制御アプリケーションの実行においては，様々な処理の間にデータ依存関係が存在する場合が殆どであり，その結果として逐次処理が支配的となる．したがって，従来のスレッドおよびデータレベル並列性を活用する空間方向の並列化が本質的に困難であり，メニーコアの潜在能力を十分に引き出すことができないといった課題が生じている．

本稿では，メニーコアプロセッサの新たな有効性を創出するために，新しい並列処理の実行方式及びその性能モデルを提案する．まず，制御システムが特定の処理を周期的に実行する点に着目し, 入力値予測に基づき投機実行を実施する時間並列処理方式を提案する．これにより，制御アプリケーションにおける厳しいリアルタイム制約を満たすための性能向上を実現する．次に，提案方式による性能向上の可能性を理論的に示すための性能モデルを考案する．そして，Amdahl のスピードアップ式 [3] に基づく従来型の空間並列処理性能との違いを論じる．さらに，具体的な応用例として，モデル予測制御 (MPC:Model Predictive Control) [4] を対象にし，提案する性能モデルの妥当性検証，ならびに，実時間最適制御の実現可能性を示す．

本稿は，次のように構成される．2 節では，制御システムにおける並列性に限界が存在する問題点について述べ

[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
Dept. of Advanced Information Technology, Graduate School of Information Science and Electrical Engineering, Kyushu Univ.
[2] ボッシュ株式会社シャシーコントロールシステム事業部
Chassis Systems Control Division, Bosch Ltd.
[3] 九州大学大学院システム情報科学研究院情報知能工学部門
Dept. of Advanced Information Technology, Kyushu Univ.
[4] 京都大学大学院情報学研究科システム科学専攻
Dept. of Systems Science, Kyoto Univ.

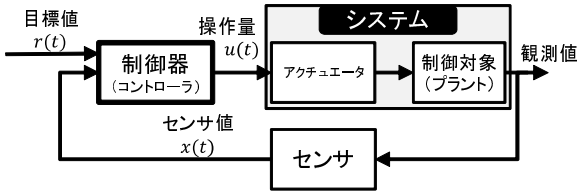

図 1 閉ループ制御のブロック図

る．3 節では，既存の並列化手法である空間方向への並列化と提案する投機実行による時間的な並列化について述べる．4 節では，既存の並列処理における性能モデルである Amdahl の法則に対して，時間的な並列処理による性能モデルを提案する．5 節では，MPC の概要と数値計算における負荷分析を行う．6 節では，MPC によるアーム型振り上げ振り子制御を用いて，リアルタイム制約と制御モデルの特性を分析し，投機実行のリアルタイム処理の実現可能性を評価する．最後に 7 節では，本稿の結論を述べる．

2. 制御システムにおける並列性の壁問題

本稿では，図 1 に示す閉ループ制御システムを対象とする．閉ループ制御システムは，制御器，制御対象，センサ，アクチュエータから構成される．制御器への入力値は，制御対象を目的の状態に制御するための目標値 $r(t)$ と制御対象の状態を示すセンサ値 $x(t)$ である．制御器の出力値は制御対象を操作するための操作量 $u(t)$ であり，アクチュエータを介して制御対象へ伝達される．制御器の主な目的は，目標値 $r(t)$ とセンサ値 $x(t)$ の差を最小化する操作量 $u(t)$ を決定することである．

次に制御システムにおける並列性の壁問題について議論する．制御器に搭載されるマイクロプロセッサの処理内容は，目標値 $r(t)$ とセンサ値 $x(t)$ を入力とした関数 $f(r,x)$ を周期的に計算することである．制御器は，入力が与えられてから次の入力が到達するまでの間に $f(r,x)$ の計算を完了させなければならない．各周期で実行すべき関数 $f_i(r,y)$（i は自然数）は相互に依存関係が存在する．すなわち，$f_i(r,y)$ の結果がアクチュエータに入力され，その結果としてシステムが状態を変え，それに伴うセンサ値が $f_{i+1}(r,y)$ の入力となる．したがって，異なる周期で実行すべき $f(r,y)$ をスレッドとし並列に実行することができない．また，多くの制御システムでは比較的小さなサイズのデータが時系列に連続入力されるため，$f_i(r,y)$ の処理においてデータレベル並列性を活用することによる性能向上も期待できない．

3. 並列処理の実行方式

本節では，解くべき問題をスレッドに分割し空間方向に展開する従来の並列化手法と，投機実行により時間的に並列処理する提案手法の実行方式について述べる．

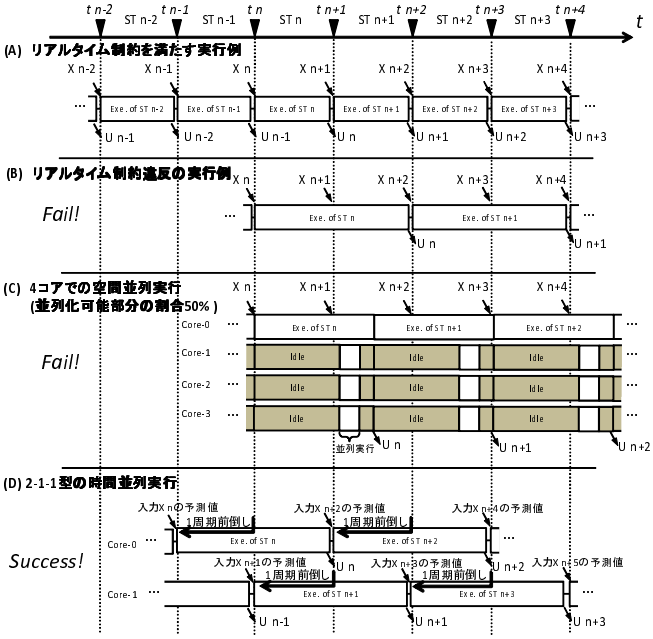

図 2 各並列処理の実行方式, (C):空間並列処理, (D):時間並列処理

3.1 空間並列処理

リアルタイムシステムにおいて，各時刻の入力に対する処理は次時刻の入力が到着するまでに実行を完了させなければならない．リアルタイム制約を満たしている実行例を図 2(A) に示す．X_n は，時間 t_n に与えられるセンサ値を示しており，ST_n，U_n はそれぞれ X_n に関連付けられたサンプリング周期の間隔，操作量を表す．図 2(B) では，各時刻の処理におよそサンプリング周期の 2 倍ほどの実行時間を要するので，リアルタイム制約を満たせていない．

従来の並列処理では，与えられたプログラムを複数の部分問題に分割することで，複数のプロセッサコアで処理することが一般的である．本稿では，従来の並列処理を**空間並列処理**，与えられたプログラムの実行単位を**プロセス**，分割した部分問題は**スレッド**の単位で実行するとする．空間並列処理の実行方式を表したものを図 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．空間並列処理は，主にグラフィックス処理をはじめとするマルチメディアの分野，科学技術計算の分野において有効な処理方式である．しかしながら, 第 2 節で説明したように，制御系の処理においては逐次処理部分が支配的である．そのため, 図 2(C) で示す様に，並列化可能部分の割合を 50%と仮定した場合でもリアルタイム制約を満たすことができない．

3.2 投機実行による時間並列処理

従来の並列処理に対して本稿では，投機実行による時間的な並列処理を新たに提案する．一般的に，最適制御システムをはじめとする制御システムの処理ではサイズが小さ

い入力データが時系列で入力され，各時系列におけるデータ入力後の処理は逐次的な処理が支配的であるという特徴を持つ．提案する並列処理では，特定のプロセスが時系列で実行されることに着目し，各プロセスを複数のプロセッサコアで投機的に実行することで，実時間処理を実現することを狙う．本稿では，提案する並列処理を**時間並列処理**と表現する．時間並列処理の実行方式を表したものを図 2(D) に示す．時間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.
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のかを示す.

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，見かけ上の 2 倍の性能向上を達成している．実際の実行時間は従来の 1 コア実行時と変わらないため，次時刻の処理を別のコアでパイプライン的に処理することにより，図 2(D) はリアルタイム制約を満たしている．

より厳しいリアルタイム制約を満たす必要がある場合は，図 2(D) よりもさらに前倒しで投機実行を行う（z を大きくする）ことで性能向上を達成できる．しかしながら，投機の度合いが大きくなるにしたがい入力値の予測精度は低下することが予想される．したがって，入力予測値の候補を複数用意する（y を大きくする）ことにより入力値の予測精度向上を狙う．

4. 並列処理の性能モデル

本節では，各並列処理における性能モデルに関する議論を行うために，時間的な並列処理の性能モデルを新たに導入する．

4.1 Amdahl の法則

与えられた問題をスレッドに分割し空間方向に展開する並列化手法の場合の性能向上について示した法則として Amdahl の法則 [3] がある．Amdahl の法則は逐次的に実行した問題に対して，問題分割し空間方向に並列化することで理論的に達成可能な性能向上比をモデル化したものである．Amdahl の法則による性能向上率 $S_{amdahl}(N)$ は，

$$S_{amdahl}(N) = \frac{1}{(1-P) + \frac{P}{N}} \qquad (1)$$

と表される．ただし，N はプロセッサコア数であり，P は与えられた問題に対して並列化が可能な割合である．並列化可能な割合 P に対する $S_{amdahl}(N)$ を図 3 に示す．図 3 より，空間的な並列処理による性能向上比は問題の並列可能な割合 P に大きく依存し，P によって性能向上比が抑制されることがわかる．したがって，本稿で対象とする最適制御においては，逐次的な処理が支配的であるため，空間

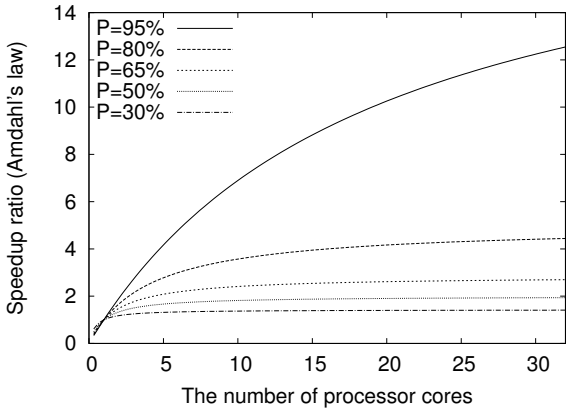

図 3 Amdahl の法則による性能向上

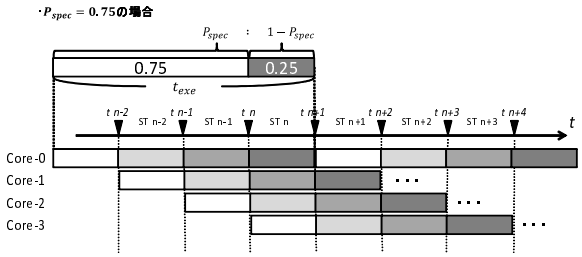

図 4 時間並列処理の実行時間

並列化の手法が効果的な性能向上を実現できないと言える．

4.2 時間並列処理の性能モデル

提案手法である時間並列の性能モデルを構築するために，投機実行した際の見かけ上の実行時間，すなわち前倒し実行時間 t_{spec} を導入する．ある処理の実行時間を t_{exe}，処理時間全体に対する前倒し実行時間の割合を P_{spec} とすると，

$$t_{spec} = t_{exe}(1 - P_{spec}), (0 \leq P_{spec} < 1) \qquad (2)$$

$$Speedup_{spec} = \frac{t_{exe}}{t_{exe}(1 - P_{spec})} = \frac{1}{1 - P_{spec}} \qquad (3)$$

となる．$P_{spec} = 0$ は，投機的な実行を行わないことを意味し，$P_{spec} = 1$ は全体の処理時間だけ前倒し実行を行い，見かけ上の実行時間が 0 になることを表す．図 4 に $P_{spec} = 0.75$ の場合の例を示す．

次に，入力がサンプリング周期ごとに到達し，処理を次入力が到達するまでに完了させなければならないリアルタイム処理系を想定する．第 3.2 節で述べたように，時間並列処理では実際の実行時間は従来の 1 コア実行時と変わらないため，パイプライン処理の様な実行モデルを考える必要がある．リアルタイム処理系において，実行時間の短縮は目的ではなく制約であるため，実行時間はサンプリング周期以下になれば良いという前提のもとで，t_{spec} = サンプリング周期と仮定する．この際，式 (3) はパイプラインの段数を表す．例えば，図 4 においてパイプラインの段数は 4 段であり，かつ，式 (3) より $1/(1 - 0.75) = 4$ となるこ

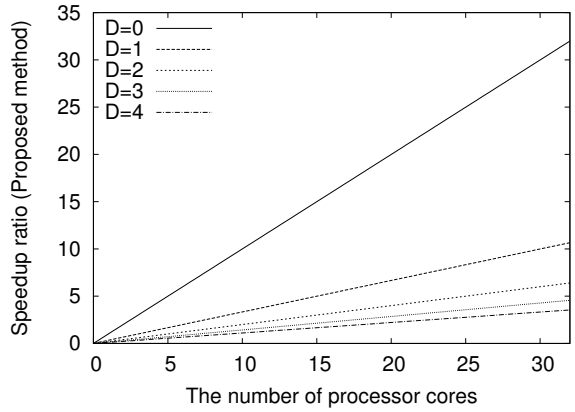

図 5 時間並列処理による性能向上（D 定数）

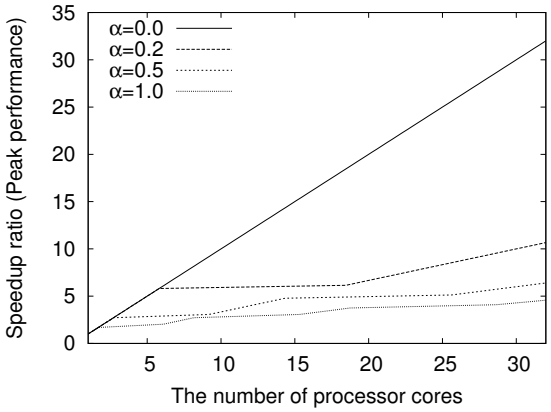

図 6 時間並列処理による性能向上（$D(z) = \lfloor \alpha * z \rfloor$）

とが分かる．

本稿では，プログラムの特性を表す指標として，投機実行で使用した予測値と当該プロセス実行時の入力値との差を表す**バラつき：D（Dispersion）**を新たに導入する．例えば，バラつき D が 0 の場合は事前の予測値と入力値が完全に一致し，D が 1 の場合は予測値に対して入力値が ± 1 の範囲で差が発生する特性を持つプログラムであることを意味する．すなわち，D が 1 増加するごとに入力値候補が 2 つ増える．バラつき D を用いて，性能向上率 $S_{temporal}(N)$ に対するプロセッサコア数 N の関係式は，

$$S_{temporal}(N) = \frac{1}{\frac{2D+1}{N}} = \frac{N}{2D+1}, (D \in \mathbb{N}) \qquad (4)$$

と表すことができ，式 (4) を時間並列の基本性能モデルとする．時間並列の基本性能モデルによる性能向上比を図 5 に示す．図 5 より，時間並列処理による性能向上比は予測値に対する入力値のバラつき D に依存していることが分かる．

また，3.2 節で述べたように投機の度合いにより，入力値の予測精度は変化する．そこで，バラつき D が投機の度合い z に対して線形に増加する性能モデルの性能向上比を図 6 に示す．性能モデル式は，基本式 (4) において，$D(z) = \lfloor \alpha * z \rfloor$ と仮定したものである．図 6 の D の初期値はいずれも 0 であるが，より性能向上を達成するために投機の度合いをあげることで，D の増加を引き起こし，結果として性能向上比が抑制されている．

5. MPC（Model Predictive Control）

本節では，入力値予測に基づく投機実行の有効性を示すために，最適制御手法の 1 つであるモデル予測制御の概要を述べる．次に，MPC の数値解法における並列性抽出の困難性について議論する．

5.1 MPC の概要

MPC とは，制御対象の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．従来の制御手法では，観測値に対する操作量をルックアップテーブルによる表引きで行う MAP 制御や，操作量を観測値の偏差・積分・微分の 3 要素によって決定する PID 制御が代表的である．しかしながら，既存の制御手法ではシステムの振舞いが非線形で制御則が複雑なシステムへの適用が困難であるという問題点がある．

MPC では，制御対象の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及び定常状態の振る舞いを正確に制御することが可能である．MPC はサンプリング周期毎に有限区間の最適制御問題を解くことで操作量を決定する．MPC は一般的に非線形システムを対象とし，非線形システムは，

$$\dot{x}(t) = f(x(t), u(t), t) \qquad (5)$$

と表される．ただし，$x(t) \in \mathbb{R}^n$ は状態ベクトルで，$u(t) \in \mathbb{R}^m$ は操作量のベクトルである．式 (5) に対して，最適制御問題は以下のように定式化される．

minimize

$\phi(x^*(t+T), t+T) + \int_t^{t+T} L(x^*(\tau), u^*(\tau), \tau) d\tau$

subject to

- $\dot{x}^*(\tau) = f(x^*(\tau), u^*(\tau), \tau)$
- $x^*(t) = x(t)$
- $x_{min} \leq x^*(\tau) \leq x_{max}$
- $u_{min} \leq u^*(\tau) \leq u_{max}$

ただし，$\tau (t \leq \tau \leq t+T)$ は変数，$x^*(\tau), u^*(\tau)$ は変関数である．目的関数は，終点コスト ϕ と区間コスト L によって構成され，ある時刻 t における状態 $x(t)$ が与えられたとき，時刻 $t+T$ と評価区間 $[t, t+T]$ において評価関数を最小化することを表している．制約式中の $x^*(\tau)$ 及び $u^*(\tau)$ は，状態値と操作量である．

現在の時刻を t_1 としたときの MPC の例を図 7 に示す．時刻 t_1 の時の状態値 $x(t_1)$ を与え，最適制御問題を解くことで操作量 $u^*(t)$ を得る．実際の操作量は，初期値のみを用いて $u(t_1) = u^*(t_1)$ を与える．以降，MPC はサンプリング周期毎に最適制御問題を解き，解の初期値を操作量として与える．

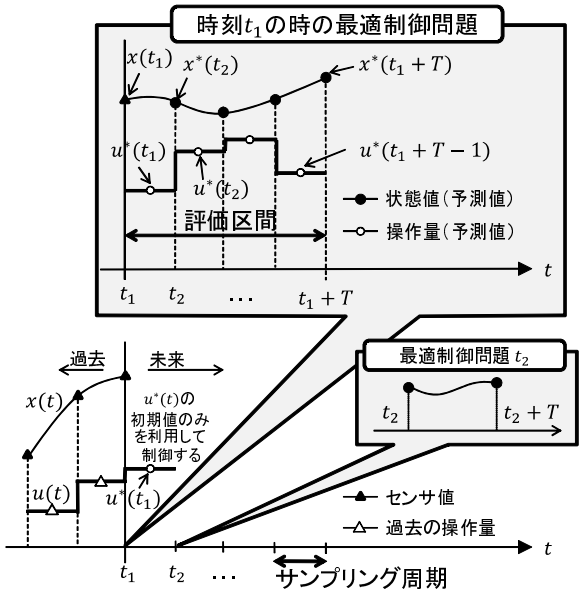

図 7 時刻 t_1 におけるモデル予測制御

5.2 MPC の数値計算と処理の依存関係

MPC の最適制御問題は，一般的に 2 次計画（QP:Quadratic Programming）問題に帰着する．LQ 問題の時間計算量は入力 N に対して $O\left(N^3\right)$ であり [5]，制御対象の複雑化に伴い実時間処理の困難性が高まる．

MPC を数値計算する際に既存のスレッド／データレベルの並列性抽出の困難性について議論する．各時刻で解く最適制御問題を以下の様に分割する．

$$t = \tau_0 < \tau_1 < ... < \tau_n = t + T \tag{6}$$

評価区間 T は，n 個の部分区間 $[\tau_i, \tau_{i+1}], 0 \leq i \leq n-1$ に分けられる．各々の部分区間において，以下の常微分方程式 (ODE: Ordinary Differential Equation) を解くことで，軌跡 $x_i(t)$ を算出する．

$$\dot{x_i}(t) = f(x_i(t), u_i(t), t) \quad \forall t \in [\tau_i, \tau_{i+1}] \tag{7}$$

$$x_i(\tau_i) = s_i \tag{8}$$

ここで，s_i は $[\tau_i, \tau_{i+1}]$ に対する初期値である．全ての $x_i(\tau_i)$ は s_i を初期値として用いることで s_{i+1} を求める．したがって，各ステップは前のステップと依存関係が存在する．6 節の評価実験で用いたアーム型振り上げ振り子制御プログラムでは，上記の逐次処理に要する実行時間は全体の実行時間の 70%に及んだ．

6. 評価実験

本節では，アーム型振り上げ振り子制御を例に実時間制約と投機実行によるバラつきの変動を調査し，性能モデルを用いて提案手法の有効性を検証する．

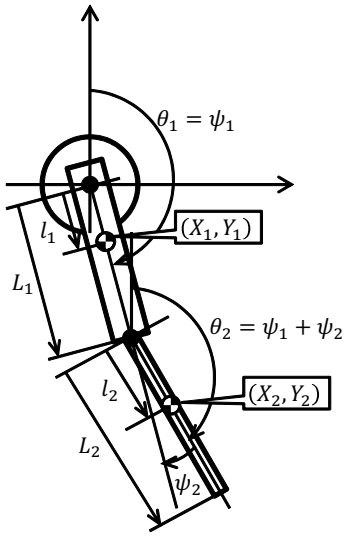

図 8 アーム型振り上げ振り子 (ψ_i:相対角，θ_i:絶対角)

6.1 制御システムの概要と実験環境

提案する投機実行による時間並列処理と性能モデルを検証するために，MPC プログラムに対して実時間制約，及び投機実行の度合いに対するバラつき D の変動を調査した．本稿では，大塚ら [6] が作成したアーム型振り上げ振り子のシミュレーションプログラムを使用した．本稿では，状態方程式と最適制御問題の概要について述べる．定式化に関する詳細は [6] を参照されたい．シミュレーションのために用いた計算機の Linux kernel のバージョンは 2.6.32，CPU は Intel Xeon5670 2.93GHz である．

アーム型振り上げ振り子の模式図を表したものを図 8 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．まず，アーム型振り上げ振り子の状態方程式を定式化する．各リンクの重心位置は，

$$\left\{ \begin{array}{l} X_1 = l_1 \sin\theta_1 \\ Y_1 = l_1 \cos\theta_1 \end{array} \right., \left\{ \begin{array}{l} X_2 = L_1 \sin\theta_1 + l_2 \sin\theta_2 \\ Y_2 = L_1 \cos\theta_1 + l_2 \cos\theta_2 \end{array} \right. \tag{9}$$

であり，振り子全体の運動エネルギー W，位置エネルギー U，及び損失エネルギー D はそれぞれ，

$$\left\{ \begin{array}{l} W = \sum_{i=1}^{2} \left\{ \frac{1}{2} m_i \left(\dot{X_i}^2 + \dot{Y_i}^2 \right) + \frac{1}{2} J_i \dot{\theta_i}^2 \right\} \\ U = \sum_{i=1}^{2} m_i g Y_i, D = \sum_{i=1}^{2} \frac{1}{2} \mu_i \dot{\psi_i}^2 \end{array} \right. \tag{10}$$

と表される．ただし，m_i：各リンクの質量，J_i：各リンクの重心まわりの慣性モーメント，g：重力加速度，μ_i：各リンクの粘性摩擦係数である．運動方程式を基に MPC の最適制御問題に定式化すると評価関数 J は，

$$J = \frac{1}{2} x^T(t+T) S_f x(t+T) + \int_t^{t+T} \left(\frac{1}{2} x^T(\tau) Q_x(\tau) + \frac{r_1}{2} u^2(\tau) - r_2 v(\tau) \right) d\tau \tag{11}$$

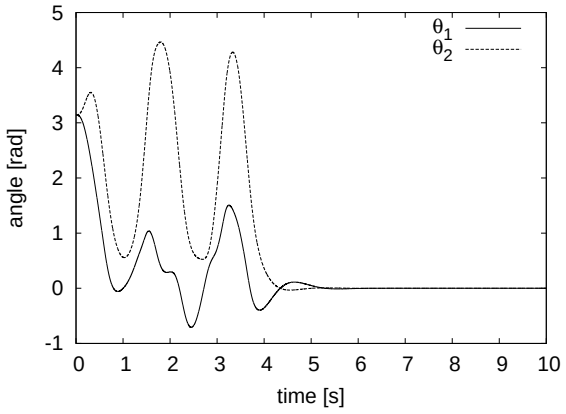

図 9 シミュレーション結果

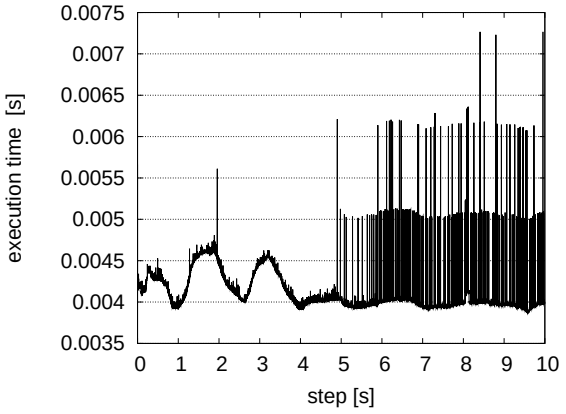

図 10 各サンプリングの実行時間

と表される．ここで，$x = \left[\theta_1\ \theta_2\ \dot{\theta_1}\ \dot{\theta_2}\right]^T$，各リンクの絶対角を根元から順に θ_1, θ_2．S_f, Q は準正定行列，r_1, r_2 は正の実数である．評価区間の長さ T は $0.5[s]$，サンプリング間隔は $1[ms]$ とする．

次に制約条件は，

$$u^2 + v^2 - {u^2}_{max} = 0 \tag{12}$$

と表される．ここで，u は制御入力であるモータへの入力電圧であり，u の大きさは $|u| \leq {u^2}_{max}$ という拘束が課されるとする．最大入力電圧 u_{max} は 2.5[V] とする．さらに，v は制約式のために新たに導入した仮想的な入力である．

シミュレーションの結果を図 9 に示す．図 9 では，モータによって制御された各アームの絶対角を時系列で示している．今回の制御では，振り子を数回振ることで最終的な鉛直上向きに静止する振る舞いを示す結果となっている．

6.2 MPC における投機実行のための入力値予測手法

投機実行のための入力値の予測には，現在の時刻から最も新しい最適制御問題の解を用いる手法を使用する．5.1 節で述べた通り，MPC では各周期においてある有限区間（評価区間）のシステムの状態値を予測している．したがって，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，制御対象の状態値が急激に変化しないという仮説の下，過去の最適制御問題の解を活用する．

さらに，入力値候補として予測値の近傍を考慮した複数の値にすることで予測精度を向上させる．本稿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と，予測値の近傍を含めた入力値候補集合 $X_{t_{n+1}}$ は，

$$X_{t_{n+1}} = \{{x^*}_{LRC}(t_n) \pm Rd | d, D \in \mathbb{N}^0, 0 \leq d \leq D\} \tag{13}$$

と表される．ただし，${x^*}_{LRC}(t_n)$，R，及び D は，最も新しい最適制御問題の解，丸めの単位，及びバラつきである．

入力値の予測精度について議論するために，予測値とセンサー値の一致を示す hit を以下の様に定義する．

$$hit(t, R) = \begin{cases} 1 & (Round(x_{\mathrm{diff}}(t), R) = 0) \\ 0 & (otherwise) \end{cases} \tag{14}$$

$$x_{\mathrm{diff}}(t) = \min_{x_{pred}(t) \in X_t} |x_{pred}(t) - x_{obser}(t)| \tag{15}$$

ここで $x_{\mathrm{diff}}(t)$ は時刻 t_1 における観測値と予測値の差の絶対値の中で最小のものである．さらに，式 (14) を用いて，hit_rate は，

$$hit\_rate(M, R) = \frac{\sum_{t=0}^{M} hit(t, R)}{M} \tag{16}$$

となる．ここで M は総サンプリング数（シミュレーション時間/サンプリング時間）である．

6.3 リアルタイム制約と制御モデル特性分析

振り子制御のシミュレーション結果をもとに，リアルタイム制御を満たすための必要性能向上比，及び制御モデル特性である性能向上比に対するバラつきを調査する．まず，リアルタイム制御を実現するために必要な性能向上比を算出するために，各ステップの最適制御に要した時間を計測した．各ステップにおける最適制御の実行時間を図 10 に示す．図 10 より，最大の実行時間は 7.266[ms] であり，リアルタイム制約を満たすためには 8 倍の性能向上比が必要である．

次に，投機実行の前倒しの数に対する始点状態値の予測と観測値のバラつきを調査するために，各リンクの観測値と最も新しい最適制御の解から生成した始点状態値の差の絶対値を計測した．投機実行の前倒しの数は，プログラムの性能向上比と対応することを意味する．計測の結果を図 11 に示す．投機の度合いが大きくなるに連れて予測の精度が低下していることを確認できる．さらに，図 11 の拡大図を図 12 に示す．丸めの単位が 0.001[rad] であると仮定すると，リアルタイム制御に必要な速度向上が 8 倍とするとバラつき D は 4 である．

6.4 空間並列と時間並列の比較

空間並列処理によって，振り子制御プログラムで 8 倍の

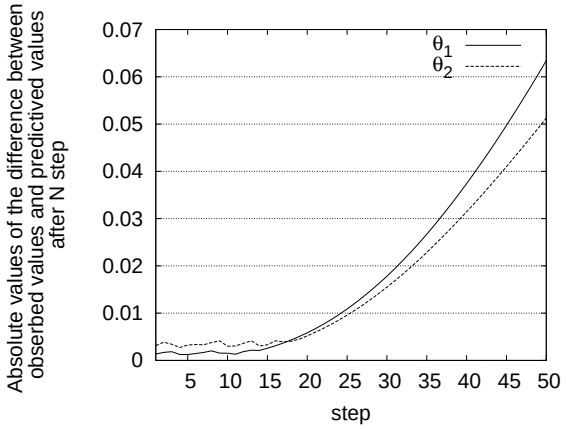

図 11 予測値と観測値の差の絶対値

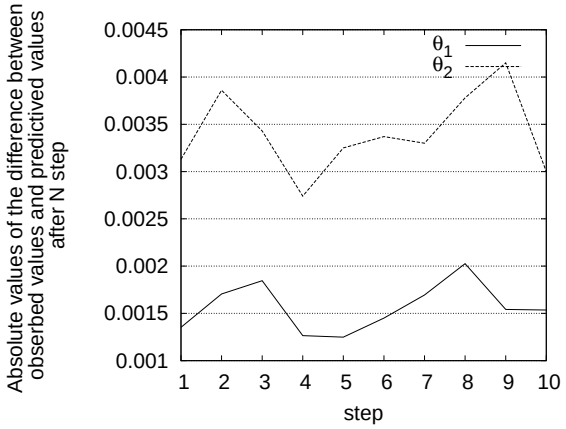

図 12 予測値と観測値の差の絶対値 (拡大)

性能向上を達成することはできない．5.2 節で述べたように並列化可能部分の割合 $P = 0.3$ とすると，ハードウェア資源（コア数）が無限大だとしても，式 (1) より

$$\lim_{N\to\infty} S_{amdahl}(N) = \lim_{N\to\infty} \frac{1}{(1-0.3)+\frac{0.3}{N}} \simeq 1.43$$

となり，1.4 倍程度の性能向上しか達成できないことが分かる．

一方，時間並列によるコア数と性能向上の関係を図 11 と式 (4) を用いることにより算出する．図 13 より，8 倍の性能向上を達成するためには，それぞれ $R = 0.1$，$R = 0.01$，及び $R = 0.001$ において 8 コア，8 コア，及び 56 コア必要であることが確認できる．したがって，MPC プログラムにおいて本提案手法は非常に有効な並列化手法であるということが明らかになった．

7. おわりに

本稿では，メニーコアプロセッサの新たな可能性を創出するために，時間軸に着目した新たな並列処理の実行方式およびその性能をモデル化した．まず，並列処理の実行方式では，特定の処理を周期的に計算する制御むけプログラムにおいて入力値予測に基づいた投機実行による時間並列を提案した．次に，投機実行による時間並列の性能向上を

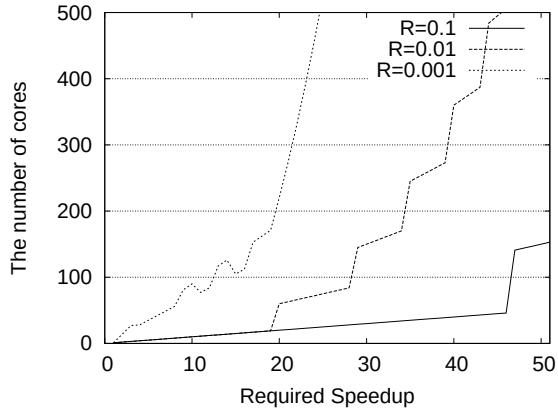

図 13 時間並列処理によるコア数と性能向上の相関

理論的に示すための性能モデルを提案した．提案した性能モデルと既存の並列処理の性能モデルを比較し，逐次処理の割合が大きいプログラムに対する提案手法の有効性を定量的に示した．さらに，MPC によって制御されたアーム型振り上げ振り子システムにおいて 8 コアでリアルタイム制御が可能であることも確認した．今後は，本稿で提案した手法をメニーコアプロセサへ実装し，性能評価を行う．さらに，異なる制御システムでの評価も行う予定である．

謝辞 日頃から御討論頂いております九州大学村上・井上研究室並びにシステム LSI 研究センターの諸氏に感謝いたします．

参考文献

[1] Xu, H., Tanabe, J., Usui, H., Hosoda, S., Sano, T., Yamamoto, K., Kodaka, T., Nonogaki, N., Ozaki, N. and Miyamori, T.: A low power many-core SoC with two 32-core clusters connected by tree based NoC for multimedia applications, *Symposium on VLSI Circuits (VLSIC)*, pp. 150 –151 (online), DOI: 10.1109/VLSIC.2012.6243834 (2012).

[2] Bell, S., Edwards, B., Amann, J., Conlin, R., Joyce, K., Leung, V., MacKay, J., Reif, M., Bao, L., Brown, J., Mattina, M., Miao, C.-C., Ramey, C., Wentzlaff, D., Anderson, W., Berger, E., Fairbanks, N., Khan, D., Montenegro, F., Stickney, J. and Zook, J.: TILE64 - Processor: A 64-Core SoC with Mesh Interconnect, *Proc. IEEE International Solid-State Circuits Conference*, pp. 88 –598 (online), DOI: 10.1109/ISSCC.2008.4523070 (2008).

[3] Amdahl, G. M.: Validity of the single processor approach to achieving large scale computing capabilities, *Proceedings of the April 18-20, 1967, spring joint computer conference*, AFIPS '67 (Spring), New York, NY, USA, ACM, pp. 483–485 (online), DOI: 10.1145/1465482.1465560 (1967).

[4] 大塚敏之：非線形最適制御入門，コロナ社 (2011).

[5] Wang, Y. and Boyd, S.: Fast Model Predictive Control Using Online Optimization, *IEEE Trans. Control Systems Technology*, Vol. 18, No. 2, pp. 267 –278 (online), DOI: 10.1109/TCST.2009.2017934 (2010).

[6] 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).