Vol.2015-SLDM-170 No.37
Vol.2015-EMB-36 No.37
2015/3/7

モデル予測制御における非線形漸化式実行の並列化

山田 竜正[1]　枝廣 正人[1]

概要：近年，制御の分野では制御の高精度化が進んでいる．しかし，高精度な制御はシングルコアで処理するには演算負荷が大きい．そこでマルチ・メニーコアの使用が考えられる．高精度な制御として，モデル予測制御がある．この手法では，制御対象の振る舞いをある一定の範囲先まで計算を行う．この計算は制御対象のモデルを用いて行われ，漸化式となる．この漸化式が非線形の場合，漸化式計算は行列計算として表すことができず，計算負荷が大きい上，並列化が難しい．そこで本研究では，この時間かかる非線形漸化式や，それを含む制御の計算の並列実行を検討する．まず漸化式の計算を Simulink モデルにより記述し，その構造を元にタスク (関数) に分割された C コードと，そのタスクグラフを生成する．このタスクグラフに対し，CP/MISF 法を使用してスケジューリングを行う．16 コアのメニーコアシミュレータ上で評価した結果，CP/MISF 法を使用すると，非線形漸化式の計算を 16 コアで 10.5 倍高速化できることがわかった．また，Simulated Annealing 法と比較したところ，並列性能が良く，スケジューリング時間も 6 桁程度高速であり，CP/MISF 法が，非線形漸化式の並列実行に適していることを示した．

1. はじめに

近年，制御の分野では，制御の高度化・高精度化によって，制御性能の向上が試みられている．高精度な制御では演算負荷が大きい計算を高速で行う必要があるという問題がある．しかしシングルコアによる実行では，処理時間が大きくなりすぎてしまう．また消費電力の問題などにより，半導体回路の微細化やクロックの高速化が難しくなり，シングルコアの処理性能の向上も限界となっている．そこで，マルチ・メニーコアを使用することが考えられる．

高精度な制御として，モデル予測制御がある．モデル予測制御は，多変数制御や制御対象の入出力制約の扱いに対して利点があり，産業界で広く用いられている手法である．モデル予測制御は，制御対象のモデルを使用して，ある範囲先までの時間の振る舞いを最適化するような制御入力を求める．この最適な制御入力を決定するため，モデル予測制御では制御周期ごとに最適化問題を解く必要がある．本研究では，同時摂動最適化手法 [1] による最適化を考える．この最適化問題を解く際には，制御対象の状態をある範囲先まで予測・計算する必要がある．この計算は制御対象のモデルを用いて行われ，これは漸化式計算の形となる．この漸化式が非線形であるならば，ステップ間の依存関係により，計算を簡単化することが容易ではなく，実行時間が大きくなってしまう．

[1] 名古屋大学情報科学研究科
Sch. of Inf. Sci., Nagoya Univ, Japan

そこで本研究では，この実行時間が大きい非線形漸化式や，それを含む制御処理の並列実行による処理速度向上を検討する．

本研究では，漸化式の計算を Simulink[6] モデルにより記述する．その上で，モデルベース並列化ツール [2] を使用し，タスクグラフを生成する．本研究ではこのタスクグラフと，タスクの実行サイクル数を元にタスクのスケジューリングを考える．スケジューリング手法として，CP 法 [4] を改良した CP/MISF 法 [5] を使用する．CP 法は，最長路長を用いる手法である．CP/MISF 法との比較として，SA(Simulated Annealing) 法 [3] によるタスクスケジューリングも行う．

ルネサスエレクトロニクス社製の汎用メニーコア命令セットシミュレータを用いた結果，漸化式の計算は CP/MISF 法によるスケジューリングにより 5.9 倍の処理速度向上となった．同時摂動最適化の処理に対しては，SA 法によるスケジューリングでは 7.0 倍，CP/MISF 法では 10.5 倍の処理速度向上となった．CP/MISF 法は SA 法よりも 6 桁以上高速に最適化を実行でき，CP/MISF 法によるスケジューリングの方が容易に良いスケジュールを得られるという結果となった．

2. 漸化式の並列化手法

2.1 非線形漸化式から Simulink モデル

MATLAB/Simulink[6] は MathWorks 社により開発された，システムのモデリングやシミュレーションを行うこと

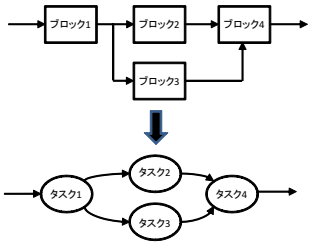


図 1　モデルからタスクグラフを生成

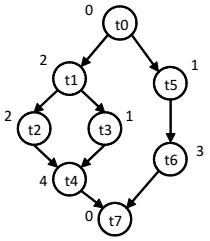


図 2　タスクグラフの例

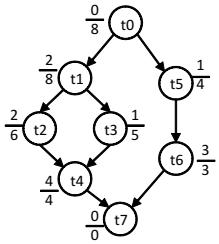


図 3　最長路長を追加した例

ができる環境である．Simulink では基本構成要素や機能をブロックで表し，それらを線で接続してブロック間の関係を表すブロック線図のようにシステムのモデルを表す．また Simulink では，SubSystem ブロックを用いることで，階層的なモデルを作成をすることができる．また，同じく MathWorks 社の Simulink Coder を利用することで Simulink モデルから C コードを自動生成することができる．

本研究ではこの Simulink により，対象とする計算の Simulink モデルを作成し，モデルベースでの並列化を検討することとした．

2.2　モデルベース並列化

本研究では，作成した Simulink モデルから，モデルベース並列化ツール [2] を用いることで，Simulink モデルの階層を展開し，その構造的な並列性を抽出，それを元に C コードの分割・並列化を行う (図 1)．並列化ツールは，Simulink モデルと，それから自動生成した C コードを入力として，C コードをタスク (関数) に分割したマルチタスク C コードと，タスク間の先行制約を表すタスクグラフを出力する．このとき，SubSystem ブロックに対し Atomic 化という設定を行うことで，階層を展開せずこのブロック単位で一つのタスクとすることができる．

2.3　タスクグラフとコア間通信

Simulink モデルの構造を元に，C コードの処理をタスク (関数) に分割すると，タスクを頂点，タスク間の先行制約を有向辺とした閉路のない有向タスクグラフとして表される．図 2 にタスクグラフの例を示す．頂点の中にタスクの番号を表し，外に添えている数字はそのタスクの実行時間である．タスクグラフのタスク実行のスケジュールを静的に決定して生成した並列 C コードは，各コアに割り当てられたタスク (関数) を順次実行していく形となる．実行の際，異なるコアに割り当てられたタスク間に先行制約を表す接続がある場合，同期・通信を行う必要がある．同期は共有メモリのフラグ変数を介して行われる．フラグ変数は同期を行う必要があるタスク一対に対して 1 つ宣言される．あるタスクの実行を行う前，共有メモリのフラグをポーリングで読み取り，依存関係のある全てタスクが完了したことを確認して実行を開始する．タスクの実行が完了した後，共有メモリのフラグに書き込み，完了を依存関係のあるタスクに通知する．

2.4　タスクスケジューリング手法

マルチ・メニーコアを使用して処理時間をできるだけ小さくするためには，これらのタスクのコア配置，実行順序をうまく決定する必要がある．本研究では処理能力が同一の限られた数のコアに対し，実行時間が異なり，かつ割り込みが入ることのないタスクを割り当てる問題を扱う．またスケジュールは静的に決定する．これは，車載制御のようなリアルタイムシステムでは一般的な方法である．この処理時間を最小とするタスクのスケジュールを求める問題は，NP 困難な組合せ最適化問題として知られている．このような最適解を求めることが難しいタスクスケジューリング問題に対し，様々なアルゴリズムが考案されている．

本研究では，非線形漸化式を元に生成されたタスクグラフのスケジューリング手法として，CP 法を改良した CP/MISF(Critical Path/Most Immediate Successors First) 法を使用する．CP 法 [4] は，タスクスケジューリング問題における代表的なヒューリスティックアルゴリズムである．CP 法は，まずタスクグラフ上の各タスクから終点となるタスクまでの実行時間に基づいた最長路長を計算し，その最長路長の値をそのままそれぞれのタスクの実行優先度とする手法である．最長路長が大きいタスクが，より優先度が高く設定される．優先度を決定した後，タスクの先行制約と実行時間に基づき，タスクグラフの始点から順に空いているコアに割り当てていく．割り当ての際，複数のタスクが割り当て可能となっていた場合は，より優先度の高いタスクから割り当てを行っていく．タスク t_i の最長路長 $l(t_i)$ は，以下の式を用いてタスクグラフの終点のタスクから順に求めることができる．

$$l(t_i) = \max_{t_j \in suc(t_i)} |t_i| + l(t_j)$$

始点タスクにおける最長路長が，そのタスクグラフ全体の実行時間の下限となる．コアの数が無制限ならば，この下限の実行時間で全タスクの実行の完了できる計算となる．図 2 のグラフの各タスクに，最長路長を追加したグラフを図 3 に示す．横棒の下の値が追加された最長路長であり，始点タスクにおける最長路長 8 がこのタスクグラフ全体の

実行時間の下限となる．

CP/MISF 法 [5] は，CP 法におけるタスクの優先度に拡張を加えたものであり，同じ優先度を持つタスク同士であるならば，直後に接続されたタスク数がより多いタスクに対して，より高い優先度を設定する手法である．

図 3 のタスクグラフのタスクを，コアが 2 個の場合において，CP 法によりスケジューリングを行っていくと，図 4 に示すようになる．なお，このタスクグラフでは，CP 法と CP/MISF 法では同じ結果となる．まず開始時点の図 4(a) では，始点タスクからの始まるタスク t_1，t_5 が実行可能となっている．このとき 2 個ともコアが空いており，タスク t_1，t_5 の最長路長はそれぞれ 8，5 であるため，最長路長がより大きいタスク t_1 から順にコア 1，次に最長路長が大きいタスク t_5 がコア 2 に割り当てられる．図 4(b) の時点では，タスク t_5 の実行が終了し，タスク t_6 が実行可能となる．このとき実行可能であるタスクはこのタスク t_6 のみであるので，タスク t_6 はすぐに空いているコア 2 に割り当てられる．

図 4(c) の時点では，タスク t_1 の実行が終了したことでタスク t_2，t_3 が実行可能となる．タスク t_2，t_3 の最長路長はそれぞれ 6，5 であるため，最長路長がより大きいタスク t_2 が空いているコア 1 へ割り当てられる．図 4(d) の時点では，タスク t_2，t_6 の実行が終了するが，新たに実行可能となるタスクは無いため，前の段階で割り当てられなかったタスク t_3 がコア 1 へ割り当てられる．図 4(e) の時点でも同様にタスク t_4 がコア 1 に割り当てられる．図 4(f) の時点では，実行可能なタスクは存在せず，実行中のコアも無いため，割り当てを完了する．

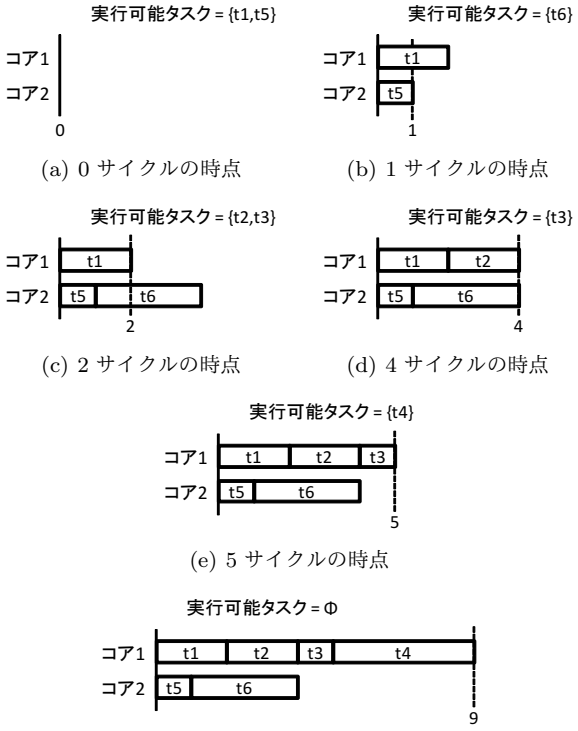
最後の時点のサイクル数 9 が，このスケジュールに対して算出された全体の実行時間となる．全タスクの実行時間の合計は 13 サイクルであるため，このスケジュールによる並列実行では 4 サイクル分だけ実行時間が小さくすることができる．

3. 非線形漸化式に対する提案手法の評価

対象とする非線形漸化式 10 ステップに対し，並列化手法を適用し，その並列実行結果を示す．ルネサスエレクトロニクス社の汎用メニーコア命令セットシミュレータを用いた．コア数は 16 コアであり，メモリ構成の設定を行うことができる (図 5)．本研究では，メモリはローカルメモリは使用せず，全てグローバルメモリとする．

本研究では，並列実行を行うモデル予測制御の対象とするモデルとして，エンジン過給圧制御モデルを使用する．この漸化式は 4 次元ベクトルであり，$f = f(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}), f_4(\boldsymbol{x}))$，$x = (x_1, x_2, x_3, x_4)$ である．変数の除算，冪乗等を含む非線形漸化式である．本研究では非線形漸化式の 10 ステップ分の計算の並列実行を検討する．対象漸化式の計算を Simulink モデル上にタスクに分割する．そしてタスク実行のスケジューリングを行い，シミュレータ上で実行結果を確認する．

粒度を合わせるため対象の漸化式を，各 x の計算のタスクに分割し，更に実行サイクル数のかかる冪乗の計算を分割した．この計算の分割を図 6 に示す．分割の結果，漸化



(a) 0 サイクルの時点　(b) 1 サイクルの時点

(c) 2 サイクルの時点　(d) 4 サイクルの時点

(e) 5 サイクルの時点

(f) 9 サイクルの時点

図 4 タスクのコア割り当ての過程

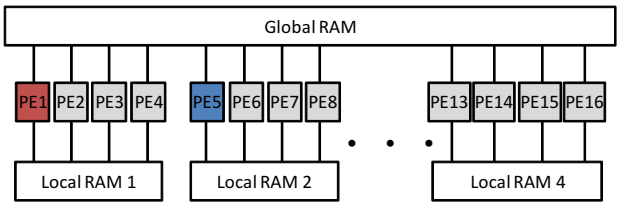


図 5 シミュレータのアーキテクチャ

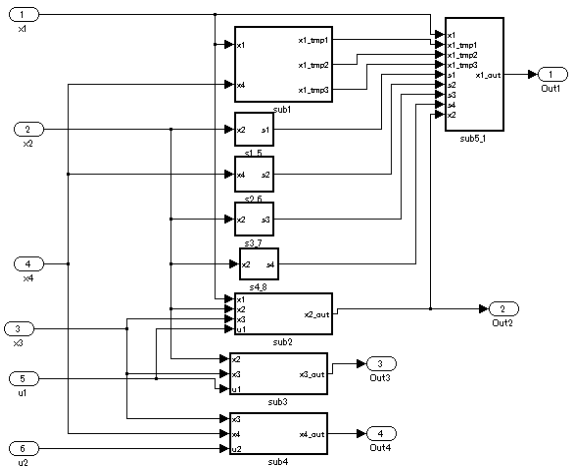


図 6 漸化式を 8 タスクに分割

Vol.2015-SLDM-170 No.37
Vol.2015-EMB-36 No.37
2015/3/7

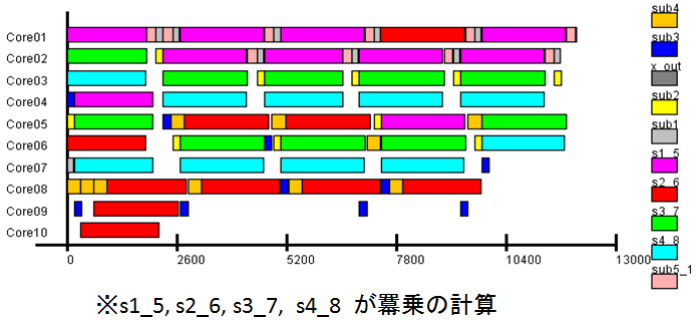


図 7　CP/MISF 法によるタスク割り当て

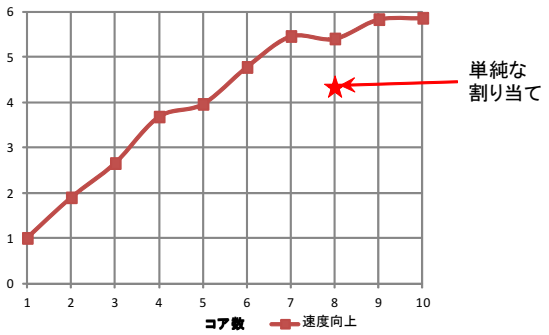


図 8　各コア数における速度向上

式の計算は 8 並列に分割される．このとき 8 並列のブロックに対して Atomic 化の設定を行い，各ブロックでそれぞれ一つのタスクとなるようにした．そして図 6 の Simulink モデルを，10 ステップ分にコピーし，漸化式 10 ステップ分の計算を行う Simulink モデルを作成する．その後，並列化ツールを使用しタスクグラフを生成する．

生成したタスクグラフに対して，図 6 の各ブロックに対してステップ間で同一のコアを単純に割り当てと，CP/MISF 法によるスケジューリングを行い結果を比較する．

CP/MISF 法によりスケジューリングを行うと，図 7 に示すようなスケジュールとなった．このガントチャートは，図 6 の Simulink モデルの 8 並列のブロックに対応して色分けされている．グラフ中の長い部分は冪乗の計算に対応している．同じ色の冪乗の計算が，ステップをまたいで並列に実行される計算となることが確認できる．またこのスケジュールにおいては，タスクは最大 10 並列で実行することができる計算となる．そこで，コア数を 1 から 10 まで制限を変化させ，同様に CP/MISF 法によりスケジューリングを行い，そのスケジュールに対する並列実行結果を取得した．図 8 に結果を示す．図 8 の結果によると，10 コアでの実行サイクル数は 15396 サイクルであった．これは 1 コアでの実行と比較すると，5.9 倍の処理速度向上となる．8 コアでの実行結果に関しても 5.3 倍の速度向上となり，単純な割り当てによる 4.3 倍の速度向上と比較して，処理速度の向上が確認できた．またコア数が 6，7 個のあたりまで，コア数を多くするに従って効率よく実行サイクル数を削減できることが確認できた．

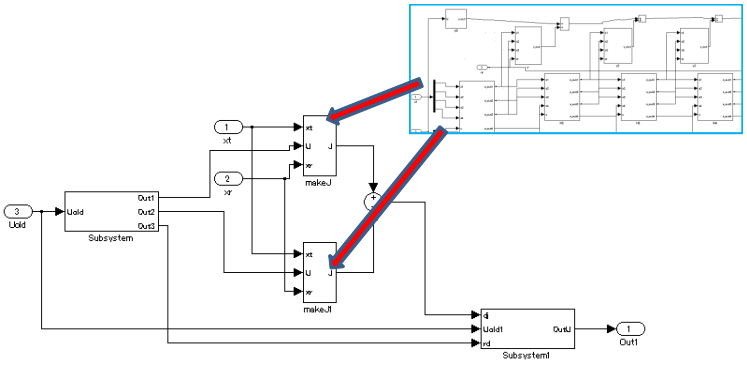


図 9　制御入力の更新 1 回の Simulink モデルの作成

以上の結果から，漸化式のステップをまたいで並列実行を行うことができるようにスケジューリングすることで，効果的に並列実行することができる可能性があることが確認できた．

4. 制御入力の最適化処理に対する提案手法の評価

本章では，モデル予測制御における同時摂動最適化手法 [1] の最適化処理の主要部に対して，提案並列化手法を適用する．この処理の中に，非線形漸化式の計算が含まれている．漸化式の計算は 10 ステップ先まで行う．スケジュールを決定し，並列化した制御量の最適化処理を，前章と同じメニーコアシミュレータ上で実行し並列化の結果を確認する．また，SA 法によりコア間通信を考慮して決定したスケジュールによる並列化との実行結果の比較を行う．

本研究では，モデル予測制御の対象となるモデルとして，前章と同様のエンジン過給圧制御モデルを使用する．対象漸化式を含む演算を Simulink モデルで作成し，スケジューリングを行い，作成した生成 C コードをシミュレータ上で実行，結果を確認する．

初めに並列実行対象となる，モデル予測制御における制御量の最適化処理の対象部分の Simulink モデルを作成する．このとき，モデルのブロックは Atomic 化の設定は行わない．そのためタスクはブロックを展開した細かい粒度のものとなる．まず漸化式の 10 ステップの計算を行うモデルを同様に作成する．次に，同時摂動最適化の制御入力の更新部分の Simulink モデルを作成する (図 9)．この Simulink モデルでは，漸化式の計算が 2 並列の形となる．これは同時摂動最適化の 2 個の評価関数の計算を行う部分に対応する．

以上のように作成した Simulink モデルから，同様に並列化ツールにより処理を全て関数に分割されたマルチタスク C コードとタスクグラフを生成し，スケジュールを決定する．生成したタスクグラフのスケジューリングを CP/MISF 法により行う．CP/MISF 法により 1 から 16 コ

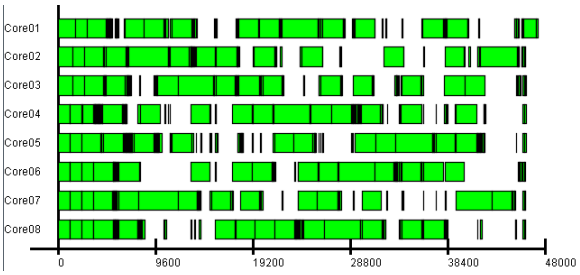


図 10　CP/MISF 法による 8 コアでのスケジューリング

アまでのスケジューリングを行い，並列化の結果を確認する．

次に比較となる SA 法によるスケジューリングの決定法について以下に述べる．

(1) 全タスクにコア 1 を割り当てるスケジュールを初期解とする．

(2) タスクのコア割り当てを別のコアに割り当てに変えた場合の評価関数の変化を求める．

(3) 評価関数の変化と現時点の温度に応じて，このタスクのコア割り当ての遷移を採択するか棄却するかを決定する．

- 評価関数の値が小さくなれば採択する．
- 評価関数の値が大きくなれば，その変化量と現時点の温度に応じた確率で採択する．

(4) 以上の 2，3 をタスクグラフのタスク数とコア数の設定に応じて一定数繰り返す．

(5) 一定数繰り返した後，温度を下げて 2〜4 を繰り返し，ある温度まで下がったら終了する．このとき，これまでに評価関数の値を最小としたスケジュールを最終的な解とする．

評価関数に関しては，まず SA 法のため，コア割り当てが予め決定されたタスクグラフに対して CP/MISF 法を適用して各コアのタスク実行順序を決定する．その後，決定したタスク実行順序に対して，コア間通信コストを加えた実行サイクル数を計算し，それを評価関数の値とする．

例として図 10 に CP/MISF 法による，コア数を 8 とし，コア間通信を考慮しないタスクの実行スケジュールを示す．タスクの色は全て同じで表される．CP/MISF 法と SA 法それぞれで求めたスケジュールによる並列 C コードを実行すると，図 11 に示すような結果となった．図 11 の結果のコア数 0 は，並列化ツールによるマルチタスク化を行わない，Simulink モデルから自動生成した C コードを直接実行した結果を表す．また，図 12 に，CP/MISF 法によるスケジュールのコア間通信を考慮しない実行時間の計算結果と，実行結果の比較を示す．

この逐次 C コードの実行サイクル数より，マルチタスクを 1 コアで実行したときのほうがサイクル数が大きくなっており，コードを関数へ分割したことによりオーバーヘッドがかかっていることが確認できる．16 コアの場合では

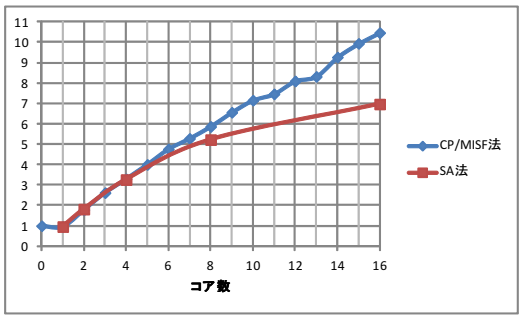


図 11　各コア数での速度向上

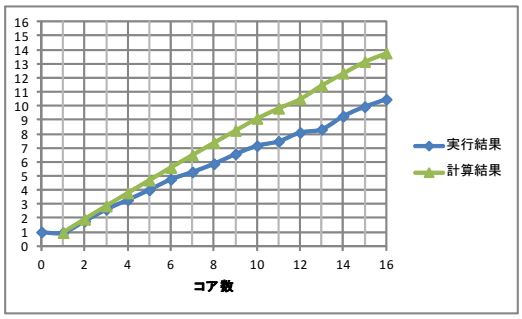


図 12　CP/MISF 法によるスケジュールの実行時間の計算結果と実行結果

CP/MISF 法は 10.5 倍，SA 法は 7.0 倍の速度向上であった．結果より，今回並列化を行った非線形漸化式を含む処理では，コア数を増やすことで，効率は落ちていくものの実行速度を向上させることができることが確認できた．しかし，図 12 に示されるように，コア数が大きくなるに連れ，CP/MISF 法によるスケジューリング時の計算で求められた実行サイクル数との差が大きくなるという結果となり，コア間の通信・同期コストの影響が大きいことが確認できる．

また CP/MISF 法は 1 パスで解を生成するのに対し，SA 法によるスケジューリングは 10 万回以上繰り返す．よって今回の非線形漸化式のような逐次的な演算では，良い解を求めるのに大きな計算時間がかかる SA 法よりも，CP/MISF 法によるスケジューリングのほうが簡単に良い解を求めることができる可能性があることが確認できた．

5. まとめ

本研究では，モデル予測制御における最適化問題を解くために必要となる非線形漸化式の計算と，非線形漸化式の計算を含む，最適化問題の解法である同時摂動最適化の処理の一部の並列実行を行った．並列化の方法として，モデルベース並列化ツールにより処理をタスクに分割し，そこからタスクスケジューリングを行った．スケジューリング手法として，CP/MISF 法を使用した．また比較として SA 法によるスケジューリングの最適化を行った．並列化した C コードは 16 コアのマルチコアシミュレータ上で並列実行した．制御入力の最適化の処理の並列実行に対して，コ

ア間通信を考慮しない CP/MISF 法でも，コア間通信を考慮し実行に多大な時間がかかる SA 法と比較して，簡単に良い結果が得られることが確認できた．

本研究では，コア間通信は SA 法によるタスクスケジューリング時の段階にのみ考慮した．しかしこれでは SA 法において，コア間通信がコストが大きくなるようなスケジュールも評価する必要がある．また CP/MISF 法によるスケジューリングでも，コア間通信のコストが大きくなり，計算上の最長路長よりも大きく実行サイクル数がかかってしまうようなスケジュールが導かれる可能性がある．そこで，タスクスケジューリングを行う前に，実行サイクル数が小さく余分に通信コストがかかってしまうタスクをマージし，タスク粒度調整を行うことで，より効率の良いスケジュールを求めやすくなる可能性があると考えられる．

謝辞 本研究を進めるにあたり，制御対象のモデルを提供して下さったトヨタ自動車株式会社，加古純一氏，佐多宏太氏，仲田勇人氏，そしてモデル予測制御における同時摂動最適化アルゴリズムを並列化対象として提供して下さった京都大学情報学研究科　東俊一准教授，田中洋輔氏に深く感謝致します．また，マルチコアシミュレータを提供して下さったルネサスエレクトロニクス株式会社とメニーコア OS を提供して下さったイーソル株式会社，並列化ツールを提供して下さった日本電気株式会社に心より感謝致します．

参考文献

[1] Y. Tanaka, S. Azuma, T. Sugie: Simultaneous perturbation stochastic approximation with norm-limited update vector: convergence analysis and application to system identification with partially known model structure, submitted to Automatica (2014)
[2] 久村孝寛，枝廣正人，中村祐一，石浦菜岐佐，武内良典，今井正治．Simulink モデルにもとづいた並列 C コード生成．電子情報通信学会技術研究報告，Vol．110，No．473，pp．303-308，2011．
[3] Sadiq M.Sait,Habib Youssef. 組合わせ最適化アルゴリズムの最新手法基礎から工学応用まで (白石洋一 訳). 丸善株式会社.(2002)
[4] Coffman, E.G. : Computer and Job-shop Scheduling Theory, John Willey & Sons (1976).
[5] Kasahara,H. and Narita, S. : Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing, IEEE Trans. Comput.,Vol.C-33,No.11,pp.1023-1029 (1988).
[6] http://jp.mathworks.com/products/simulink/