情報処理学会論文誌　コンピューティングシステム　Vol.5　No.3　1–10 (May 2012)

ブロック化赤–黒順序付け法に基づく並列マルチグリッドポアソンソルバ

河合 直聡[1,a)]　岩下 武史[2,3]　中島 浩[2]

受付日 2011年10月7日, 採録日 2011年12月28日

概要：本論文では，3 次元ポアソン方程式の差分解析を対象としたマルチグリッド法の並列化について述べる．マルチグリッド法の並列化に際してしばしば問題となるスムージング部について，ブロック化赤–黒順序付け法によるガウス–ザイデルスムーザの並列化を行う．さらに，同手法の改良法として，スムージングと制約・補間演算をキャッシュブロッキングする実装方式を導入する．また，本論文では，ブロック化赤–黒順序付け法において赤および黒ブロック内のガウス–ザイデル演算を複数回行う乗法シュワルツスムーザを新たに提案する．4 個のクワッドコア AMD Opteron プロセッサを備える共有メモリマルチプロセッサシステム上での数値実験により提案手法を評価した結果，既存手法である重み付きヤコビ法とガウス–ザイデル法のハイブリッド手法，および赤–黒順序付け法に基づく手法に対してそれぞれ 2.88 倍，2.22 倍の高速化を実現した．

キーワード：3 次元ポアソン方程式，マルチグリッド法，ブロック化赤–黒順序付け法，シュワルツスムーザ

Parallel Multigrid Poisson Solver Based on Block Red-black Ordering

Masatoshi Kawai[1,a)]　Takeshi Iwashita[2,3]　Hiroshi Nakashima[2]

Received: October 7, 2011, Accepted: December 28, 2011

Abstract: This paper describes parallelized multigrid solver for finite difference analysis of three dimensional Poisson equation. We introduce block red-black ordering to parallelize Gauss-Seidel smoother, which is often a bottleneck in parallelizaion of multigrid methods. Next, we introduce a new cache-blocking implementation to combine smoothing and restriction or prolongation in a block. Finally, we propose a new multiplicative Schwarz smoother, in which multiple Gauss-Seidel iterations are performed in each block in red-black ordered block. Numerical tests on a shared memory multi-processor system comprising 4 quad-core AMD Opteron processors examine the proposed method, to show that the proposed method attains 2.22 and 2.88 times as high performance as the hybridization of Jacobi and Gauss-Seidel smoothers and red-black Gauss-Seidel smoother, respectively.

Keywords: 3D-Poisson equation, multigrid method, block red-black ordering, Schwarz smoother

1. はじめに

近年，計算科学や数値シミュレーション分野において，複数の物理現象を同時に扱うマルチフィジックスシミュレーションに関する研究が進んでいる．たとえば，プラズマシミュレーションでは，粒子間の相互作用と解析空間内の電磁場の双方を同時に解析する必要性がある．このようなマルチフィジックスシミュレーションでは，ある特定の物理現象に関する計算が解析全体の並列処理を阻害したり，計算コストの大部分を占めたりする場合がある．そのようなマルチフィジックスシミュレーションにおいて高速化が要請される計算核の 1 つとしてポアソン方程式の求解

1　京都大学情報学研究科
Graduate School of Informatics, Kyoto University, Kyoto 606–8501, Japan
2　京都大学学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University, Kyoto 606–8501, Japan
3　科学技術振興機構戦略的創造研究推進事業
JST.CREST, Chiyoda, Tokyo 102–0075, Japan
a)　kawai@sys.i.kyoto-u.ac.jp

（ポアソンソルバ）があげられる．ポアソン方程式は楕円型偏微分方程式の 1 つであり，マルチフィジックスシミュレーションにおいては解析空間全体におけるある物理量の平衡状態を模擬するために解かれることが多い．このことから，同方程式の求解は，複数の独立した部分領域内の計算として分割することが本質的に困難であり，陽解法等の並列化が容易な他の物理現象との連成解析において計算時間の点でしばしば問題となる．

上記の背景の下，本研究ではマルチフィジックスシミュレーションで解かれることが多い，均一媒質内の 3 次元ポアソン方程式の求解法について検討する．本論文では大規模問題での有用性が高いことで知られるマルチグリッド法 [1] による差分解析を対象とし，スレッド並列処理による高速化について述べる．

マルチグリッド法は与えられた解析格子に対して，それよりも粗い格子を複数用意し，これらの格子群を活用することで効率的に誤差を修正し，高速な求解を可能とする方法である．同手法の主な手順は制約演算・補間演算・スムージングよりなるが，このうち制約・補間演算は格子点ごとに独立な計算であるため領域分割による並列処理が可能である．一方，スムージングについてはその並列化が問題となる場合がある．

ポアソン方程式の差分解析から生ずる連立一次方程式を対象とした場合，ガウス–ザイデル法をスムーザ（スムージングに使用する手順）として利用することが一般的で，性能も良好である．しかし，同スムーザは逐次的な手法であるため，そのまま並列化することができない．そこで従来，重み付きヤコビ法とガウス–ザイデル法を併用したハイブリッドなスムーザ [2] や赤–黒順序付け法に基づいて同スムーザを並列化した方法 [3] が用いられてきた．特に後者は，ハイブリッドなスムーザに見られる並列化にともなう収束性の劣化を生じず，マルチグリッド法によるポアソンソルバとしては標準的な並列化スムーザといえる．しかし，同手法の標準的な実装方式であり，マルチフィジックスシミュレーションとの親和性もよいストライドアクセスによる実装を用いた場合，キャッシュ内のデータ再利用性が低下するという問題がある．そこで，本論文では，より高速な 3 次元ポアソンソルバの確立を目的として，新たな並列化スムージング手法について論じる．

本研究では，ガウス–ザイデルスムーザの並列化手法として，IC/ILU 分解前処理において提案されたブロック化赤–黒順序付け法 [4], [5] を利用することを考える．同手法は解析格子を複数のブロックに分割し，これらのブロックに対して赤–黒順序付け法を適用する方法である．同手法では，各ブロック内における配列要素へのアクセスが連続的となるという利点がある．次に，ブロック化赤–黒順序付けガウス–ザイデルスムーザの特徴を利用した制約・補間演算とスムージングをキャッシュブロッキングする実装法を提示する．さらに，ブロック化赤–黒順序付けガウス–ザイデルスムーザの各ブロック内におけるガウス–ザイデル演算を複数回行う新しい並列化スムーザを提案する．本スムーザは通常のガウス–ザイデルスムーザと異なり，乗法シュワルツスムーザの一種と見なされる．複数のマルチコアプロセッサによる SMP ノードを使用した数値実験による各並列化スムーザの評価に基づき，提案するスムーザにより従来の赤–黒順序付けに基づいた方法と比べて 2 倍以上の速度向上を実現できることを示す．

2.　3 次元ポアソン方程式の差分解析

本論文で対象とするのは 3 次元ポアソン方程式である．本式は空間的スカラ関数 $\phi(x,y,z)$ および $\rho(x,y,z)$ を用いて次のように表される．

$$\nabla^2\phi = -\rho \text{ on } \Omega \tag{1}$$

ここで，$\rho(x,y,z)$ は既知であるものとし，$\phi(x,y,z)$ が未知関数となる．また，解析領域 Ω の境界 $\partial\Omega$ 上では，ディレクレ，ノイマン，周期のいずれかの境界条件が与えられる．本論文では解析領域として直方体形状の領域を考え，式 (1) を 7 点差分公式による有限差分法により解く．解析領域を x，y，z の各方向について，均等に nx，ny，nz 分割し，各格子点上に未知変数を配置する．この場合，式 (1) は，格子点座標 (i,j,k) の格子点について，

$$\begin{aligned}&\frac{\phi_{i+1,j,k}+\phi_{i-1,j,k}-2\phi_{i,j,k}}{(\Delta x)^2}+\\&\frac{\phi_{i,j+1,k}+\phi_{i,j-1,k}-2\phi_{i,j,k}}{(\Delta y)^2}+\\&\frac{\phi_{i,j,k+1}+\phi_{i,j,k-1}-2\phi_{i,j,k}}{(\Delta z)^2}=\rho_{i,j,k}\end{aligned} \tag{2}$$

のように近似される．ここで，Δx，Δy，Δz はそれぞれ x 方向，y 方向，z 方向の格子間隔を表し，$\phi_{i,j,k}$ および $\rho_{i,j,k}$ はそれぞれ格子点 (i,j,k) 上の離散化された ϕ および ρ の値を表すものとする．

式 (2) を全格子点について連立し，境界条件を適用することで，全格子点数を N として，解くべき N 元連立一次方程式

$$\boldsymbol{Au} = \boldsymbol{f} \tag{3}$$

を得る．ただし，ベクトル $\boldsymbol{u}$ および $\boldsymbol{f}$ は $\phi_{i,j,k}$ および $\rho_{i,j,k}$ を格子点の順序付けに基づいて並べたベクトルである．ここで本論文では，格子点の順序付けとして辞書式順序付けを用いることで，ベクトル $\boldsymbol{u}$ および $\boldsymbol{f}$ の値を保持する配列を，格子空間に対応した 3 次元配列とすることができるようにする．このような実装手法はポアソン方程式以外の基礎方程式により記述される現象との連成解析，すなわちマルチフィジックスシミュレーションにおいて，ϕ および ρ を複数の解法プログラム間でやりとりするのに都合がよ

く，広く用いられている．また，係数行列 $\boldsymbol{A}$ に関しては，式 (2) で与えられる均質媒質問題の場合，係数行列に関するデータをメモリ上に確保する必要はない．しかし，本論文で用いるプログラムでは，拡散係数が空間的関数として与えられる場合への対応も考慮し，格子点と自身を含む周辺の 7 格子点間の関係性をそれぞれ配列として保持し，合計で 7 つの 3 次元配列を用いて係数行列に関する情報を保持するものとする．

3. 並列マルチグリッド法

本論文では連立一次方程式 (3) をマルチグリッド法により解く．マルチグリッド法はポアソン方程式の差分解析から導出される連立一次方程式に対して，反復（サイクル）数が問題サイズに依存しないことが知られており [6]，大規模問題に対する有用性が高い方法である．本章では，マルチグリッド法とその代表的な既存並列化手法について述べる．

3.1 マルチグリッド法

ポアソン方程式の差分近似により導出される連立一次方程式 (3) に対してガウス–ザイデル（以下 GS）法や重み付きヤコビ法を適用した場合，解析格子間隔を基準とした波長が短い誤差成分は速やかに収束するが，長波長の誤差成分の収束には多数の反復を要することが知られている．そこでマルチグリッド法では，元の解析格子 Ω^h に対して，粗い解析格子 Ω^H を別途用意し，Ω^h 上で収束しにくい誤差成分を Ω^H 上で効率的に除去することを行う．以下に具体的なマルチグリッド法の手順について述べる．

ここでは，2 つの解析格子 Ω^h，Ω^H によるレベル数 2 の場合のマルチグリッド法について述べる．式 (3) に対して，GS 法等を用いて近似解 $\tilde{\boldsymbol{u}}$ を得たとする．この操作は（プリ）スムージングと呼ばれる．ここで誤差ベクトル $\boldsymbol{e} = \boldsymbol{u} - \tilde{\boldsymbol{u}}$ と残差ベクトル $\boldsymbol{r} = \boldsymbol{f} - \boldsymbol{A}\tilde{\boldsymbol{u}}$ を用いて誤差方程式

$$\boldsymbol{A}\boldsymbol{e} = \boldsymbol{r} \tag{4}$$

を得ることができる．式 (4) を解くことは式 (3) を解くことと同義であるため，マルチグリッド法では Ω^h に対して格子数の少ない解析格子 Ω^H を利用し，粗い格子上で誤差方程式を解くことを行う．Ω^H 上の誤差方程式の右辺ベクトル $\boldsymbol{r}^H$ は制約演算子 $\boldsymbol{I}_h^H$ を用いて

$$\boldsymbol{r}^H = \boldsymbol{I}_h^H \boldsymbol{r} \tag{5}$$

として求め，Ω^H 上で式 (1) を離散化することで係数行列 $\boldsymbol{A}^H$ を得る．Ω^H 上の誤差（修正）ベクトルを $\boldsymbol{e}^H$ と表すと，解くべき Ω^H 上の誤差方程式は

$$\boldsymbol{A}^H \boldsymbol{e}^H = \boldsymbol{r}^H \tag{6}$$

となる．式 (6) を解くことにより得られる，$\boldsymbol{e}^H$ の近似値 $\tilde{\boldsymbol{e}}^H$ と補間演算子 $\boldsymbol{I}_H^h$ を用いて

$$\tilde{\boldsymbol{u}} \leftarrow \tilde{\boldsymbol{u}} + \boldsymbol{I}_H^h \tilde{\boldsymbol{e}}^H \tag{7}$$

のように近似解 $\tilde{\boldsymbol{u}}$ の修正を行う．式 (5)–(7) の一連の操作はコースグリッドコレクションと呼ばれる．コースグリッドコレクションの終了後，さらに（ポスト）スムージングを行い近似解 $\tilde{\boldsymbol{u}}$ を更新する．プリスムージング，コースグリッドコレクション，ポストスムージングの一連の手順がレベル数を 2 とした場合のマルチグリッド法であり，この手順を繰り返すことにより求解を行う．

$$\begin{aligned}
&\mathit{Smoothing\ on}\ \boldsymbol{A}^h\boldsymbol{u}^h = \boldsymbol{f}^h \\
&\boldsymbol{f}^{2h} = \boldsymbol{I}_h^{2h}\left(\boldsymbol{f}^h - \boldsymbol{A}^h\tilde{\boldsymbol{u}}^h\right) \\
&\qquad \mathit{Smoothing\ on}\ \boldsymbol{A}^{2h}\tilde{\boldsymbol{u}}^{2h} = \boldsymbol{f}^{2h} \\
&\qquad\qquad \vdots \\
&\qquad\qquad \mathit{Solve}\ \boldsymbol{A}^{Lh}\boldsymbol{u}^{Lh} = \boldsymbol{f}^{Lh} \\
&\qquad\qquad \vdots \\
&\qquad \mathit{Smoothing\ on}\ \boldsymbol{A}^{2h}\boldsymbol{u}^{2h} = \boldsymbol{f}^{2h} \\
&\tilde{\boldsymbol{u}}^h \leftarrow \tilde{\boldsymbol{u}}^h + \boldsymbol{I}_{2h}^h\boldsymbol{u}^{2h} \\
&\mathit{Smoothing\ on}\ \boldsymbol{A}^h\boldsymbol{u}^h = \boldsymbol{f}^h
\end{aligned}$$

図 1　V サイクルマルチグリッド法のアルゴリズム
Fig. 1　Algorithm of V-cycle multigrid method.

実用的な解析では，2 以上のレベルの解析格子を用いた求解が行われる．この場合，式 (6) に対してレベル数 2 のマルチグリッド法を再帰的に適用することにより，解法手順が導出される．本手法によるマルチグリッド法は解析格子の推移形態から V-サイクルマルチグリッド法と呼ばれ，本論文でも同手法を用いる．**図 1** にレベル数 L の V-サイクルマルチグリッド法の手順を示す．図 1 において，各行列，ベクトルの右上添え字 lh は l レベル番目の解析格子上で定義されていることを示し，連立一次方程式 $\boldsymbol{A}^h\boldsymbol{u}^h = \boldsymbol{f}^h$ は式 (3) と同一である．本論文では，レベル l の解析格子に対してレベル $l+1$ の解析格子は各辺の格子数を 1/2 として構成する．これらの階層的なグリッド上で，制約演算子，補間演算子として，文献 [3] の p.11 に記載の Full-Weighting 法，Trilinear 法をそれぞれ用いる．また，本論文で対象とする問題では，最密の解析格子において，$\mathrm{mod}(nx, 2^L) = 0$，$\mathrm{mod}(ny, 2^L) = 0$，$\mathrm{mod}(nz, 2^L) = 0$ が満たされるものとする．

3.2 マルチグリッド法の並列化

3.1 節で述べたように，マルチグリッド法の手順はスムージングと制約・補間演算からなる．このうち制約・補間演算は格子点ごとに独立であるため，解析空間を分割することにより容易に並列化することができる．一方，スムーザはその種別により並列化が容易でない場合がある．たとえば，スムーザとして重み付きヤコビ法を用いた場合，計算は格子点ごとに独立となり，並列化は容易である．対して，

GS 法を用いた場合，格子点間にデータ依存性が発生するため，そのまま並列化することはできない．しかし，GS スムーザは重み付きヤコビスムーザと比較して収束性の点で優位であることが広く知られており，その並列化は重要である．そこで，本論文では GS スムーザの並列化について検討を行う．

3.3　並列化 GS スムーザ

本節では GS スムーザの既存並列化手法について述べる．

3.3.1　ハイブリッドスムーザ

GS 法の並列化において，解析領域を分割し，各部分領域をスレッドに割り当てる方式を用いた場合，データ依存性が生ずる格子点は各部分領域の周辺部に位置する．そこで，部分領域の内部の格子点に関しては通常の GS 法の手順を適用し，領域境界上の節点については，重み付きヤコビ法を用いることにより，データ依存性を回避する方法が用いられている．本手法はハイブリッドスムーザ（以下 Hybrid）と呼ばれており，BoomerAMG [2] にも採用されている．本手法は概念が平易で，実装も容易であるが，重み付きヤコビ法の導入による収束性の劣化が問題となる．一般に，並列数（部分領域数）の増加にともなって，収束性が悪化する傾向がみられ，逐次 GS スムーザと比べて収束に要する反復回数が 2 倍程度に大きく悪化する場合がある．

3.3.2　赤–黒順序付け GS スムーザ

2 章で述べたように，ポアソン方程式を 7 点差分法により離散化した場合，ある格子点は隣接する格子点とのみデータ依存関係を持つ．そこで，隣接する 2 つの格子点を 2 色（赤および黒）で交互に色分けし，並列化を可能にする赤–黒順序付け GS スムーザ（以下 RB-GS）がある．本手法では，同色の格子点間に接続関係がないため，GS 法の並列化を行うことができる．同手法では収束性が並列数（コア数）に依存せず，また式 (1) のような均一媒質問題では逐次 GS 法と比べて同程度の収束性が得られるため，並列化スムーザとして幅広く利用されている．2 章に述べたように各配列が 3 次元構造を有する場合，**図 2** に示されるように各配列を 1 つ飛ばしにストライドアクセスすることで RB-GS を実装することができる．本実装手法は簡便であり，またマルチグリッド法における制約・補間演算との親和性も高いため，標準的な RB-GS の実装法といえる．しかし，同手法ではストライドメモリアクセスのためにキャッシュデータの再利用性が低下し，逐次 GS スムーザと比べて 1 回のスムージングに要する計算時間が長くなるという問題点がある．

```
  real(8) u(nx,ny,nz), f(nx,ny,nz)
  real(8) x_way, y_way, z_way, dgn
  integer i, j, k, f_start, i_start

  x_way = (nx-1)**2
  y_way = (ny-1)**2
  z_way = (nz-1)**2
  dgn = -0.5d0 / (x_way + y_way + z_way)

!Red-Black Gauss-Seidel method
!when f_start is 1, calculate red point
!when f_start is 2, calculate black point
  do f_start = 1, 2
     i_start = f_start
     do k = 1, nz
        do j = 1, ny
           do i = i_start, nx, 2
              u(i, j, k) = ( f(i, j, k) &
  - x_way * (u(i+1, j, k) + u(i-1, j, k)) &
  - y_way * (u(i, j+1, k) + u(i, j-1, k)) &
  - z_way * (u(i, j, k+1) + u(i, j, k-1)) &
                            ) * dgn
           enddo
           i_start = 3 - i_start
        enddo
     enddo
  enddo
```

図 2　赤–黒順序付け GS スムーザのソースコード
Fig. 2　Source code of red-black Gauss-Seidel smoother.

4.　ブロック化赤–黒順序付け法による並列化スムーザ

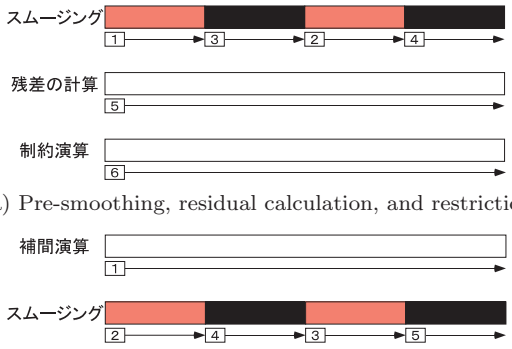
4.1　ブロック化赤–黒順序付け GS スムーザ

上記で述べたように，赤–黒順序付けに基づく GS スムーザの並列化は収束性の観点から優れているが，ストライドアクセスによる計算速度の低下が問題となる．そこで，本論文では IC/ILU 分解前処理の並列化手法として提案されたブロック化赤–黒順序付け法を GS スムーザの並列化手法として活用することを考える．同手法は，対象とする解析格子をまずブロック状に分割し，そのブロック群を赤，黒の 2 色で塗り分ける手法である．本手法を GS スムーザに適用した場合，ブロック内は逐次的な GS 法を適用するが，同色のブロック間では計算依存性がないため各色内でブロックごとの並列化が可能である．

ブロック化赤–黒順序付け GS スムーザ（以下では BRB-GS と表記する）は，ブロックサイズを拡大するに従い，その手順は逐次 GS スムーザの手順に近づく．したがって，十分に大きなブロックサイズを用いることにより逐次 GS スムーザと同程度の収束性が期待できる．また，ブロックサイズを 1 とすれば，赤–黒順序付け GS 法とスムージングの手順は一致する．BRB-GS では各ブロック内の処理は逐次 GS 法であり，各配列の要素アクセスは連続的である．したがって，BRB-GS 法の利用により，収束性を維持しつつ，RB-GS の問題点であるストライドアクセスを回避することが可能となる．また，BRB-GS の実装は，各配列の 3 次元構造を維持したまま，各ブロックの範囲を指定するだけで容易に行うことができる．**図 3** に

Loop parallelization Loop $n = 1, N_r$
　n 番目の赤ブロックに対する GS スムージング
End loop
Thread synchronization
Loop parallelization Loop $n = 1, N_b$
　n 番目の黒ブロックに対する GS スムージング
End loop

図 3　BRB-GS の手順
Fig. 3　Procedure of BRB-GS smoother.



(a) Pre-smoothing, residual calculation, and restriction
(b) Prolongation and post-smoothing

図 4　キャッシュブロッキングを行わない場合の計算順序
Fig. 4　Calculation order of normal BRB-GS smoother.

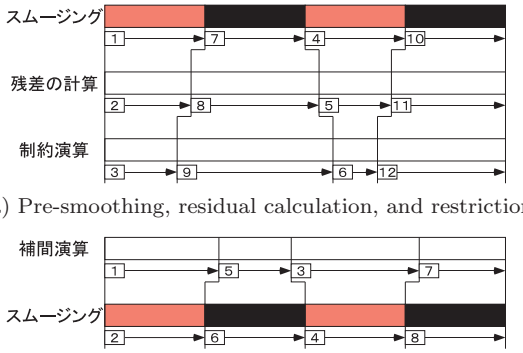
BRB-GS による並列化スムージングの手順を示す．ただし，図中の N_r および N_b はそれぞれ赤ブロック数，黒ブロック数を表すものとする．

4.2　改良型ブロック化赤–黒順序付け GS スムーザ

本節では，BRB-GS において残差計算，および制約・補間演算とスムージングをキャッシュブロッキングする実装手法と，BRB-GS の各ブロック内で複数回の GS 法を行う新たなスムーザについて述べる．

4.2.1　制約・補間演算とスムージングのキャッシュブロッキング

一般に，マルチグリッド法の実装では，スムージングと制約・補間演算はプログラム内で分離されていることが多い（たとえば，別の関数や別のループとして実装されるのが一般的である）．本実装方式によると，BRB-GS では図 4 のような順序で各格子点（配列要素）にアクセスすることになる（図中の例は各色のブロック数が 2 の場合を表す）．すなわち，スムージングや制約・補間演算および残差計算はいずれも全解析格子点を走査する計算となる．そこで，文献 [7] では，逐次 GS スムーザを用いたマルチグリッドソルバにおいて格子空間をブロック分割し，ブロック内でスムージング計算後，ただちに残差計算・制約演算を行うキャッシュブロッキング手法が提案されている．同様の性能改善手法は並列化スムーザでも利用可能であり，特に BRB-GS 法ではスムージングの計算がすでにブロック化されているため，スムージング計算と残差計算/制約演算および補間演算とのキャッシュブロッキングを簡単に導入することができる．



(a) Pre-smoothing, residual calculation, and restriction
(b) Prolongation and post-smoothing

図 5　キャッシュブロッキングを行った場合の計算順序
Fig. 5　Calculation order in the proposed cache-blocking technique.

Loop parallelization Loop $n = 1, N_r$
　(i) n 番目の赤ブロックに対する逐次 GS スムージング
　(ii) 部分領域 $(rxs_n + 1 : rxe_n - 1,$ $rys_n + 1 : rye_n - 1,\ rzs_n + 1 : rze_n - 1)$ に対する残差計算および制約演算
Thread synchronization
Loop parallelization Loop $n = 1, N_b$
　(i) n 番目の黒ブロックに対する逐次 GS スムージング
　(ii) 部分領域 $(rxs_n - 1 : rxe_n + 1,$ $rys_n - 1 : rye_n + 1,\ rzs_n - 1 : rze_n + 1)$ に対する残差計算および制約演算
End loop

図 6　BRB-GS と制約演算のキャッシュブロッキング実装
Fig. 6　Cache-blocking implementation of BRB-GS and restriction.

すなわち，BRB-GS 法におけるブロックサイズをキャッシュメモリの容量を考慮して適切に定め，図 5 のように，残差計算，および制約・補間演算をブロック内（ただし，ブロック境界上の格子点を除く）でスムージング処理と連続して行えばよい．ブロック境界上の格子点については，たとえば，赤ブロックに属する格子点に関するプリスムージング後に行う残差計算において，隣接する黒ブロック内の格子点に関するスムージング後のデータが必要となるため，スムージング処理と残差計算・制約演算をブロッキングすることはできない．一方，各ブロック内の境界部を除いた格子点群については，上記で述べたキャッシュブロッキングが可能である．

図 6 に，本論文で対象とする 3 次元問題における，スムージングと残差計算・制約演算のキャッシュブロッキング手法の手順を示す．図中において，$(rxs_n : rxe_n,\ rys_n : rye_n, rzs_n : rze_n)$ は n 番目の赤ブロックを示す部分配列

```
Loop parallelization Loop n = 1, N_r
  (i) n 番目の赤ブロックに対する逐次 GS
      スムージングを α 回行う
  (ii) 部分領域 (rxs_n + 1 : rxe_n − 1,
      rys_n + 1 : rye_n − 1, rzs_n + 1 : rze_n − 1)
      に対する残差計算および制約演算
End loop
Thread synchronization
Loop parallelization Loop n = 1, N_b
  (i) n 番目の黒ブロックに対する逐次 GS
      スムージングを α 回行う
  (ii) 部分領域 (rxs_n − 1 : rxe_n + 1,
      rys_n − 1 : rye_n + 1, rzs_n − 1 : rze_n + 1)
      に対する残差計算および制約演算
End loop
```

図 7　mBRB-GS の実行手順
Fig. 7　Procedure of mBRB-GS(α).

で，$(bxs_n : bxe_n,\ bys_n : bye_n, bzs_n : bze_n)$ は n 番目の黒ブロックを示す部分配列である．まず，赤ブロック内の格子点については，ブロック内のスムージングを行った後，当該ブロックの境界上の格子点を除いた 1 格子点分内側のブロック内部領域に関する残差の計算を行い，同領域内で可能な範囲での制約演算を行う．次に，黒ブロック内の計算では，スムージングの終了後，赤ブロック内で残差計算や制約演算が完了していない格子点を含めた計算を行うために，ブロックを 1 格子点分広げた領域での残差計算・制約演算を行う．なお，補間演算とスムージングのキャッシュブロッキングは，黒ブロックの境界上の格子点に関する補間演算を赤ブロックの格子点に関する計算時に行うことで実装することができる．

4.2.2　改良型ブロック化赤–黒順序付け GS スムーザ

本項では，BRB-GS の改良版として，改良型ブロック化赤–黒順序付け GS スムーザと呼ぶ手法を提案する．以下において，同手法を mBRB-GS と表記するものとする．mBRB-GS はブロック化赤–黒順序付け法に基づくが，4.2.1 項で述べた BRB-GS とは異なる手順を持つ乗法シュワルツスムーザの一種である．mBRB-GS(α) の手順は**図 7** のように与えられる（ただし，4.2.1 項で述べたキャッシュブロッキングを含めた実装を示している）．

図 7 に示すように，mBRB-GS(α) では，BRB-GS において，各赤および黒ブロック内の逐次 GS 法の回数を α 回とする．mBRB-GS(α) では α の値を 2 以上とするが，1 とした場合はキャッシュブロッキングを適用した BRB-GS と同一の手順となる．以下に mBRB-GS において期待される性能向上について考察する．

一般にマルチグリッド法のスムーザとして定常反復法を用いる場合，1 スムージングあたりの反復数 β を増加させることにより，ソルバ全体の収束性を改善することが可能となる [6]．しかし，一般に，解析格子はキャッシュメモリの容量と比べて大きく，β を増加させた場合，スムージングに要する計算時間はほぼ β に比例して増加する．スムージングに要する計算コストはマルチグリッドソルバの計算コストの大部分を占めるため，この場合ソルバ全体の収束性がおよそ β 倍改善されないと全体としての性能向上は望めない．したがって，マルチグリッド型ソルバでは，1 スムージングあたりの定常反復法の反復数を 1 として実行する場合が多い．これらの既存研究の成果をふまえた場合，4.2.1 項で述べた BRB-GS の 1 スムージングあたりの回数を増加させてもほとんど性能向上は期待できない．

一方，mBRB-GS ではブロック単位で逐次 GS 法を複数回行う．この場合，ブロック内での 1 回目の逐次ガウス–ザイデル法を実行した後，スムージングに必要な配列要素はキャッシュメモリ上に存在するため，2 回目以降の逐次 GS 法の実行にはメモリアクセスは発生せず，1 回目と比べて短時間で実行できると期待される．すなわち，1 回の BRB-GS に要する計算時間を t_s とした場合，mBRB-GS(α) に要する計算時間は $t_s + (\alpha - 1)\tilde{t}_s$ と書け，$\tilde{t}_s \ll t_s$ となることが期待される．その結果，BRB-GS に対する mBRB-GS(α) の収束性の改善率が α に満たない場合でもソルバ全体の速度向上を期待することができる．

5.　数値実験

本論文で対象とする 3 次元ポアソン方程式のテスト問題として，1 辺の長さが 1 m の立方体の解析領域中心に，半径 3.1 cm の球体を設置し，球体の内部のみに $\rho = 1$ を与えた問題を用いた．領域境界にはディレクレ境界条件を与えた．V サイクルマルチグリッド法による求解において，相対残差の無限大ノルムが 10^{-7} 以下になった時点で収束と判定した．解析空間を 512^3 分割し，マルチグリッド法のレベル数は 9 とした．プログラム言語には Fortran を使用し，スレッド並列化は OpenMP により行った．数値実験環境として，京都大学学術情報メディアセンターの Fujitsu HX600，1 ノードを用いた．本ノードは 4 個のクワッドコア AMD Opteron プロセッサ（2.3 GHz）から構成され，32 GB の主記憶を備えている．また，並列実行時の各プロセッサ（ソケット）に対するスレッド割当てに関しては，ソケットあたりのスレッド数が最小となるように設定した．

5.1　ブロック化赤–黒順序付け GS スムーザの性能評価

マルチグリッド法の並列化に際して，ハイブリッドなスムーザ（Hybrid），赤–黒順序付け GS スムーザ（RB-GS），およびブロック化赤–黒順序付け GS スムーザ（BRB-GS）をそれぞれ用いた場合について，スレッド並列時（1，2，4，8，16 スレッド）の性能評価を行った．なお，BRB-GS については，予備的な実験結果に基づいて，本プログラムの各配列においてメモリ上で連続となる x 方向にはブロッ

表 1　BRB-GS における各スレッド数でのブロック数
Table 1　Number of blocks for each thread when using BRB-GS.

Number of threads	x-way	y-way	z-way
1	1	1	2
2	1	2	2
4	1	2	4
8	1	4	4
16	1	4	8

表 2　各スレッド数における各スムーザを用いた場合の反復回数
Table 2　Number of itration using each smoother.

	Number of threads				
	1	2	4	8	16
Hybrid	11	21	21	21	21
RB-GS	10	10	10	10	10
BRB-GS	11	11	11	11	11

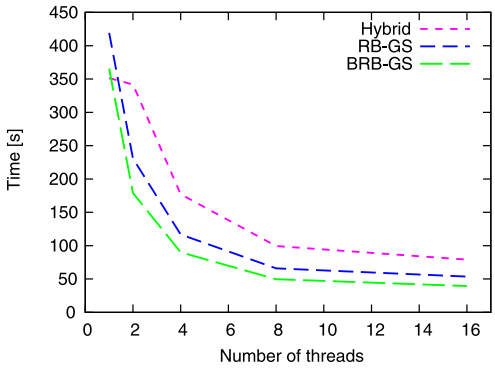


図 8　各スムーザの計算時間の比較
Fig. 8　Comparison of parallel smooothers in computational time.

ク数が 1 となるようにブロックサイズを設定した．また，y 方向，z 方向のブロック数については，BRB-GS の並列度は 1 色あたりのブロック数で与えられるため，実行スレッド数と 1 色あたりのブロック数が等しくなるようにブロックサイズを設定した．**表 1** に各方向のブロック数を示す．

表 2，**図 8** に各スムーザを用いて並列化した場合のマルチグリッド法の反復回数（V サイクル数），計算時間をそれぞれ示す．Hybrid でスレッド数を 1 とした場合の結果が逐次 GS スムーザによる結果を表す．Hybrid を用いた場合，複数スレッドによる実行時において，一部の格子点を重み付きヤコビスムーザにより処理するため収束性が逐次 GS スムーザと比べて約 2 倍程度悪化し，良好な並列性能が得られない．

次に，RB-GS を用いた場合，収束性は逐次 GS スムーザと同程度かむしろ本テスト問題では優位であり，Hybrid よりも高い並列性能を得ている．次に，BRB-GS を用いた場合，逐次 GS 法と同等の収束性を有しており，収束性の点では良好であるため，Hybrid と比較して大幅に性能が向上している．また RB-GS と比較すると，収束性は同程度であるが，1 反復あたりの計算時間が短縮されるため，かなり大きな性能向上が達成されている．具体的には 16 スレッド実行時に，BRB-GS は Hybrid に対して約 2.00 倍，RB-GS に対して 1.36 倍の高速化を達成しており，本論文で導入した BRB-GS の有効性が確認された．

5.2　改良型ブロック化赤–黒順序付け GS スムーザの性能評価

次に，4.2 節に述べた BRB-GS においてスムージングと残差計算，および制約・補間演算をキャッシュブロッキングする実装，およびブロック内の GS 演算を複数回行う改良型ブロック化赤–黒順序付け GS スムーザ（mBRB-GS）の性能評価を行う．ただし，mBRB-GS の性能評価では，BRB-GS の場合と異なり，各ブロックのサイズはキャッシュメモリの容量を考慮して，各グリッドレベルごとに定めた．たとえば，最密グリッド上では，x，y，z 方向のブロック分割数をそれぞれ，1，128，128 とした．

BRB-GS に対してキャッシュブロッキングを適用した実装（以下 cBRB-GS）では，16 スレッド実行時での BRB-GS の計算時間が 39.47 秒であるのに対して 32.49 秒に短縮され 17.7%の性能向上が得られた．

次に，mBRB-GS 法による性能改善について述べる．mBRB-GS(2) において 16 スレッド実行時における 1 回目と 2 回目のスムージングに要する計算時間を最密格子上で計測した．その結果，1 回目のスムージングに 0.67 秒を要しているのに対して，2 回目では 0.11 秒と約 1/6 の計算時間で済んでいる．したがって，mBRB-GS(2) を用いた場合には BRB-GS と比較した収束性の改善率が 2 未満であってもソルバ全体の計算時間が短縮される可能性がある．そこで，プリスムージングとポストスムージングにそれぞれ mBRB-GS(p_r) と mBRB-GS(p_o) を用いた場合の 16 スレッド実行時の計算時間および V サイクル数（反復回数）を計測し，その結果を**表 3** に示す．本表に見られるように，mBRB-GS を利用することで BRB-GS と比べて性能向上が可能な p_r と p_o の組合せが複数存在した．mBRB-GS を用いた場合，16 スレッド実行時において最も良い結果を得たのは，$p_r = 4$ と $p_o = 3$ の場合で，V サイクル数 6 で収束し，計算時間は 24.10 秒であった．

次に，逐次 GS スムーザを用いた場合の計算時間を基準とした各手法の台数効果を**図 9** に示す．なお，図中の mBRB-GS の結果は，1〜8 スレッド実行時には $(p_r, p_o) = (1, 2)$ とし，16 スレッド実行時には $(p_r, p_o) = (4, 3)$ とした場合を表す．同図において，mBRB-GS 以外の手法では 8 スレッド以降の速度向上に劣化が見られる．これは，ポアソン方程式の 7 点差分解析では，格子点あたりの計算量に対してメモリからのロード/ストア量が多く，実効演算性能が実効メモリバンド幅に律速されるためと考えられる．本実験で用いた計算機では，1 クアッドコアプロセッサあ

表 3　mBRB-GS スムーザによるソルバの計算時間，反復回数（計算時間/反復回数と表記）

Table 3　Calculation time and number of iteration of the multigrid solver based on mBRB-GS (calculation-time/number of iteration).

p_r \ p_o	1	2	3	4	5
1	32.40/11	28.18/9	27.00/8	25.61/7	27.55/7
2	28.03/9	26.28/8	24.72/7	26.66/7	24.55/6
3	26.75/8	24.60/7	26.34/7	24.24/6	25.97/6
4	28.85/8	26.45/7	24.10/6	25.79/6	27.54/6
5	31.08/8	28.42/7	25.78/6	27.48/6	29.20/6
6	33.38/8	30.36/7	27.58/6	29.14/6	30.94/6
7	35.74/8	32.42/7	29.29/6	30.98/6	32.71/6
8	38.05/8	34.44/7	31.05/6	32.68/6	28.67/5
9	40.55/8	36.55/7	32.91/6	34.57/6	30.18/5
10	42.95/8	38.84/7	34.72/6	36.26/6	31.75/5

p_r \ p_o	6	7	8	9	10
1	29.62/7	31.69/7	33.83/7	35.70/7	37.82/7
2	26.34/6	28.04/6	29.86/6	31.65/6	33.40/6
3	27.66/6	29.47/6	31.14/6	33.09/6	34.81/6
4	29.26/6	30.96/6	27.38/5	28.77/5	30.31/5
5	25.76/5	27.21/5	28.80/5	30.14/5	31.73/5
6	27.18/5	28.71/5	30.22/5	31.65/5	33.13/5
7	28.71/5	30.19/5	31.68/5	33.15/5	34.62/5
8	30.17/5	31.66/5	33.18/5	34.54/5	36.13/5
9	31.71/5	33.07/5	34.58/5	36.09/5	37.54/5
10	33.15/5	34.64/5	36.16/5	37.60/5	39.09/5

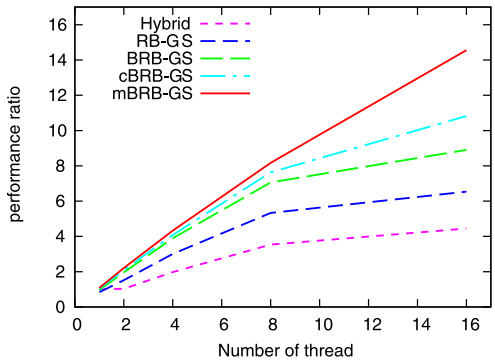


図 9　逐次 GS スムーザによるソルバと比較した各並列化ソルバに基づくソルバの並列台数効果

Fig. 9　Speedup of parallel multigrid solver with various smoothers compared to a solver with sequential GS smoother.

たりのメモリチャネル数が 2 となっているため，スレッド数が 8 から 16 に増加しても実効メモリバンド幅はあまり改善されない．

一方，mBRB-GS の場合には，ブロック内の GS 反復数の増加による収束性の改善による性能改善が可能となる．すなわち，8 スレッドから 16 スレッドへスレッド数を増加させた場合，p_r および p_o の値を増加させることで性能改善を図ることができる．本現象について以下に簡単に説明する．8 スレッド実行時について，最密格子上で mBRB-GS(2) における 1 回目と 2 回目のスムージングに要する計算時間を計測した結果，それぞれ 1.22 秒，0.49 秒であった．この両者の比はおよそ 2.5 : 1 であり，本節冒頭で示した 16 スレッドでの比率 6 : 1 に比べると，かなり小さくなっている．すなわち，16 スレッド実行時では 8 スレッド実行時と比べて 1 コア（スレッド）あたりの実効メモリバンド幅が減少するため，キャッシュ上での計算を行う 2 回目以降のスムージングが 8 スレッドでの BRB-GS に比べて相対的に高速になる．結果として，16 スレッド実行時では反復数増加にともなう収束性の良化の程度が 8 スレッド実行時と比べて低い場合でもソルバ全体の性能向上が得られることとなり，実際に数値実験結果に見られるように，反復数 p_r および p_o を増加させることにより性能向上を行うことが可能となる．このように，mBRB-GS 法はコアあたりの実効メモリバンド幅が低下する状況において有用性が高い手法であるといえる．

以上をまとめると，すべてのスレッド数に対して mBRB-GS が最も高速であり，以下性能が高い順に cBRB-GS，BRB-GS，RB-GS，Hybrid となることが明らかになった．mBRB-GS の 16 スレッドでの性能は Hybrid に対して 2.88 倍，RB-GS に対して 2.22 倍であり，いずれも大きな性能向上となっている．また，逐次スムーザと比べると，16 スレッドで 14.6 倍という理想に近い性能向上が得られた．

6.　関連研究

本章では関連する既存研究を取り上げ，本論文で提案した手法との差異について記述する．

本論文では，GS スムーザの既存並列化手法として Hybrid や RB-GS を提示したが，すでに引用した文献 [2], [3] 以外にも文献 [8], [9], [10], [11] 等にこれらのスムーザに関する報告がある．また，文献 [12] では Hybrid の改良法と見なすことのできる，別の並列化手法を提案している．具体的には，Hybrid において重み付きヤコビ法の部分を多色順序付け GS 法に変更した方法を提示している．各プロセッサの内部領域（ブロック）内では逐次 GS 法が用いられるため，本論文で述べた BRB-GS スムーザと関連性があると考えられる．しかし，当該の手法は，非構造メッシュを対象としたプロセス並列時の性能評価において，Hybrid 同様に収束性が並列数の増加に対して低下，変動しており，その点が性能上問題となる場合があると考えられる．

また，マルチグリッドソルバにおけるキャッシュメモリを考慮した実装法に関する論文として，文献 [7] や [13] があげられる．前者の文献では GS スムーザにおいて，複数回のスムージングや残差の計算および制約・補間演算のキャッシュブロッキングを行っている．しかし，当該文献は逐次スムーザを対象としており，並列化スムーザにおけるキャッシュ最適化を扱っている本論文とその点で内容が異なる．また，後者の文献では，2 次元ポアソン方程式に対するマルチグリッドソルバにおいて RB-GS スムーザを採用し，本スムーザにおいてキャッシュの利用性を高める手法を提案している．論文中で Windshield Wiper と呼ばれる手法は，L1 キャッシュを効率的に利用することを可能と

する．当該文献と本論文との差異は，使用しているスムーザの違い，および本論文で提案するキャッシュ最適化手法がスムージングだけでなく，その他の計算部分とのキャッシュブロッキングを行っている点にある．また，当該文献では，2 次元問題を対象としており，同文献内に 3 次元問題への拡張が容易でないことが記述されているのに対し，本論文では 3 次元問題を対象としている点も異なっている．

7. まとめ

本研究では，3 次元ポアソン方程式の差分解析を対象としたマルチグリッド法の並列化について検討を行った．まず，ガウス–ザイデル（GS）スムーザの並列化手法としてブロック化赤–黒順序付け法を導入し，性能評価を行った．その結果，既存手法である重み付きヤコビ法と GS 法を併用したハイブリッドなスムーザ（Hybrid），および赤–黒順序付け法に基づく並列化 GS スムーザ（RB-GS）との比較において，16 スレッドで Hybrid に対して 2.00 倍，RB-GS に対して 1.36 倍の高速化が達成された．

次に，ブロック化赤–黒順序付け法に基づく並列 GS スムーザ（BRB-GS）においてスムージングと制約・補間演算をキャッシュブロッキングする実装方式を導入し，BRB-GS 法においてブロック内の GS 反復を複数回行う並列化スムーザを新たに提案した．本手法は乗法シュワルツスムーザの一種である．提案手法により，16 スレッド実行時において，Hybrid に対して 2.88 倍，RB-GS に対して 2.22 倍の高速化を達成した．また，提案手法では，ブロック内の GS 反復数の増加によりメモリアクセスを抑制しながらソルバの収束性を改善することが可能なため，コア数を増加させても実効メモリバンド幅が拡大しない場合でも並列台数効果を得ることができ，逐次 GS スムーザによるソルバと比較して 16 スレッド実行で 14.6 倍の速度向上を実現した．

今後の課題として，ノイマン条件や周期境界条件等のディレクレ条件以外の境界条件を持つポアソン方程式の境界値問題や移流拡散方程式等の他の偏微分方程式問題における提案手法の性能評価や，大規模 SMP 環境を用いたより多くのコア数（スレッド数）での実験，プロセス並列計算環境への対応等があげられる．

参考文献

[1] Wsseling, P.: *An Introduction to Multigrid Methods*, John Wiley & Sons Ltd., Philadelphia (2004). Corrected Reprint, R.T. Edwards, Inc.

[2] Henson, V.E. and Yang, U.M.: BoomerAMG: A parallel algebraic multigrid solver and preconditioner, *Applied Numerical Mathematics*, Vol.41, pp.155–177 (2002).

[3] Thoman, P.: *Multigrid Methods on GPUs*, VDM (2008).

[4] Iwashita, T. and Shimasaki, M.: Block red-black ordering: A new ordering strategy for parallelization of ICCG method, *Int. J. Parallel Prog.*, Vol.31, pp.55–75 (2003).

[5] Iwashita, T., Nakanishi, Y. and Shimasaki, M.: Comparison criteria for parallel orderings in ILU preconditioning, *SIAM J. Sci. Comput.*, Vol.26, pp.1234–1260 (2005).

[6] Briggs, W.L., Henson, V.E. and McCormick, S.F.: *A Multigrid Tutorial Second Edition*, SIAM, Philadelphia, PA (2000).

[7] Douglas, C.C., Throme, D.T., Hu, J., Ray, J. and Tuminaro, R.S.: Cache aware multigrid on adaptively refined meshes, *ECCOMAS* (2004).

[8] Hülsemann, F., Kowarschik, M., Mohr, M. and Rüde, U.: Parallel Geometric Multigrid, *Numerical Solution of Partial Differential Equations on Parallel Computers*, Vol.51, No.2, pp.165–208 (2006).

[9] Yavneh, I.: On Red-Black SOR Smoothing in Multigrid, *SIAM Journal on Scientific Computing*, Vol.17, No.1, pp.180–200 (1996).

[10] Adams, F., Brezina, M., Hu, J. and Tuminaro, R.: Parallel multigrid smoothing: Polynomial versus Gauss-Seidel, *Journal of Computational Physics*, Vol.188, No.2, pp.593–610 (2003).

[11] Thole, C.A. and Trottenberg, U.: A short note on standard parallel multigrid algorithms for 3D problems, *Applied Mathematics and Computation*, Vol.27, No.2, pp.101–115 (1988).

[12] Adams, M.F.: A distributed memory unstructured gauss-seidel algorithm for multigrid smoothers, *Proc. 2001 ACM/IEEE Conference on Supercomputing*, *Supercomputing'01*, p.4, ACM (2001).

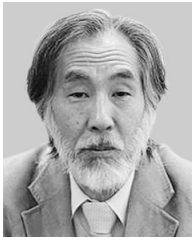
[13] Kowarschik, M., Rüde, U., Weiß, C. and Karl, W.: Cache-aware multigrid methods for solving poisson's equation in two dimensions, *Springer*, Vol.64, No.4, pp.381–399 (1999).

河合 直聡　（学生会員）

2011 年京都大学大学院情報学研究科システム科学専攻修士課程修了．現在，同専攻博士課程に在籍．高性能計算，線形反復法の研究に従事．IEEE，電気学会各学生会員．

岩下 武史　（正会員）

1998 年京都大学大学院工学研究科電気工学専攻博士課程修了．京都大学リサーチアソシエイト（日本学術振興会未来開拓学術研究推進事業 PD），同大学助手を経て，2003 年より同大学学術情報メディアセンター助教授（2007 年職名変更により同准教授），現在に至る．高性能計算，線形反復法，電磁界解析，並列処理に関する研究に従事．京都大学博士（工学）．IEEE，SIAM，電気学会，日本 AEM 学会，日本計算工学会，日本応用数理学会各会員．



中島 浩　（正会員）

1956 年生．1981 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年三菱電機（株）入社．推論マシンの研究開発に従事．1992 年より京都大学工学部助教授．1997 年より豊橋技術科学大学教授．2006 年より京都大学教授．並列計算機のアーキテクチャ，プログラミング言語の実装方式に関する研究に従事．工学博士．1988 年元岡賞，1993 年度坂井記念特別賞受賞．IEEE-CS，ACM，ALP，TUG 各会員．