ツリー型データ構造に基づく適合直交格子法 —非等方格子分割への拡張—

An Adaptive Cartesian Mesh Flow Solver Based on the Tree-data with Anisotropic Mesh Refinement

小川 隆申, 成蹊大学, 〒 180-8633 武蔵野市吉祥寺北町 3-3-1，E-mail: ogawa@me.seikei.ac.jp
Takanobu OGAWA, Seikei Univ., 3-3-1 Kichijoji-kitamachi, Musashino-shi, Tokyo, JAPAN

Abstract A flow solver is developed for the adaptive Cartesian mesh with anisotropic cell subdivision which is very effective in capturing a boundary layer. The solver is based on the tree data structure which is suitable for adaptively subdivided mesh. It is shown that the algorithms developed for the tree data based flow solver can be straightforwardly extended to anisotropic mesh refinement.

1 はじめに

計算手法の進歩と計算機の演算能力の向上によって，CFD は非常に複雑な形状の流れ場も対象とすることができるようになりつつある．しかし，計算の前処理となる計算格子の生成には依然膨大な時間と労力が必要となっており，実用的な問題を解く上での大きな障害となっている．計算機の処理能力向上により数値解析そのものに要する計算時間は短くなってゆくのに対し，人手によるところが大きい格子生成作業は自動化されない限り所用時間が短縮される見込みが小さい．

こういった背景から近年，適合直交格子による流体解析手法が注目を集め，主に実用的な流れ場を中心に利用されている [1],[2],[3],[4]．格子を局所的に再分割してゆくこの手法によれば，物体の内外判定によって格子生成ができるため格子生成を自動化できる可能性が大きい．

適合直交格子を用いた流体解析におけるデータ管理方法の一つにツリー型データがある [5]．著者はこれまでにツリー型データ構造に基づく流体解析プログラムが，単純なアルゴリズムによるループから実現できること，流束計算のための隣接セルを簡単な演算により探索できること，構造格子向けに開発された陰解法が容易に適用できること，データ構造を追加することなく Multigrid 法を実現できることなどを示し，ツリー型データが構造格子の規則性と非構造格子の柔軟性を併せ持ち，流体解析のためのデータ構造として様々な優れた特徴があることを指摘した [6]．

これらのアルゴリズムは格子が各方向に等方的に分割されることを前提としていた．しかし，必要以上に格子数を増加させないためには非等方的な分割 [7],[8],[9] が有効である．そこで，本研究ではこれまでに提案したアルゴリズムを拡張し，非等方分割に対応したツリー型データ構造に基づく流体解析手法を開発する．そして，従来の等方分割のためのアルゴリズムがほぼそのまま非等方分割へ拡張できることを示す．

2 計算アルゴリズム

2.1 支配方程式とその離散化

流体解析手法の例として本研究では疑似圧縮性法 [10] を用いるが，ここで示すアルゴリズムは他の解析手法についても適用可能である．支配方程式はカーテシアン座標系における疑似圧縮パラメータを含んだ 3 次元ナビエ・ストークス方程式である．

$$\frac{\partial}{\partial t}\int_\Omega \mathbf{Q}dv + \int_{\partial\Omega}(\mathbf{F}-\mathbf{F}_v)\cdot d\mathbf{s} = 0, \tag{1}$$

ここで，

$$\mathbf{Q} = \begin{bmatrix} p \\ \mathbf{u} \end{bmatrix}, \mathbf{F} = \begin{bmatrix} \beta\mathbf{u} \\ \mathbf{u}\otimes\mathbf{u} + p\mathbf{I} \end{bmatrix}, \mathbf{F}_v = \begin{bmatrix} 0 \\ \tau \end{bmatrix}. \tag{2}$$

これを図 1 に示すようにセル中心有限体積法により離散化し，次式に基づいて計算を行う．

$$\frac{\partial}{\partial t}(QV)_i + \sum_l (F - F_v)\cdot s_l = 0\,. \tag{3}$$

（和はセル i を取り囲むセル境界面について行う．）

セル境界での数値流束は Flux Difference Splitting[11] により求めた．なお，現時点では物理量はセル内で一様であるとして一次精度のみの計算とした．

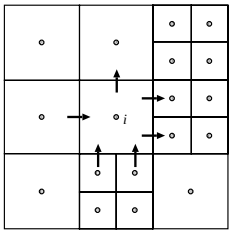

図 1: The cell-centered FVM

2.2 適合直交格子法

適合直交格子法では流れ場を直方体（以降，セルと呼ぶ）に分割し，必要なところで局所的かつ再帰的にセルを再分割してゆく．等方的にセルを分割する場合，図 2 のように x,y,z 全方向にセルを分割し，元のセルは体積の等しい 8 つのセルになる．

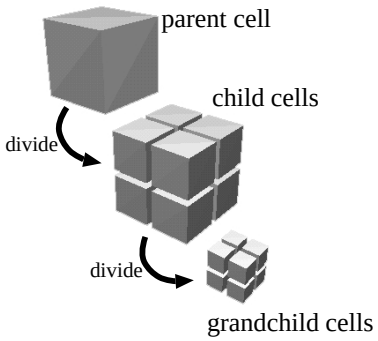

図 2: Cell subdivision in the adaptive Cartesian mesh method

こういった等方的なセル分割はしばしば不必要にセル数を増加させてしまう．特に境界層を捉える場合，図 3 のように必然的に流れ方向にもセルが分割されてしまい効率的でない．そのため，図 3 右図のように壁方向にのみセルを分割する非等方分割への拡張が重要となる．

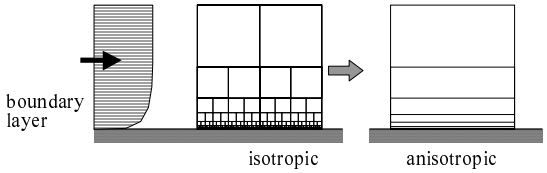

図 3: The adaptive cartesian method applied to a boundary layer

セルを非等方に分割する場合の分割形態を図 4 に示す．x,y,z 各方向のうちいずれか 1 方向に分割した場合は 2 つのセルが，2 方向に分割する場合は 4 つのセルが生成される．

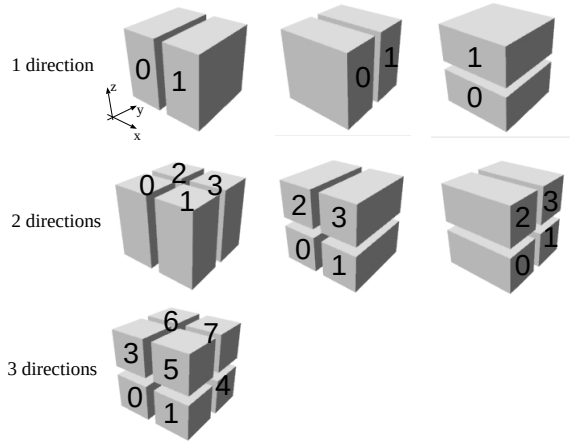

図 4: Anisotropic cell subdivision

2.3 ツリー型データ構造

適合直交格子における再帰的かつ階層的に分割される格子を取り扱うには，ネスティングした構造格子あるいは非構造格子などを利用することも考えられるが，ツリー型データによって管理するのが最も自然であろう．

図 5 にツリー型データ構造の一例を示す．セルの従属関係を親子関係になぞらえ，分割されるセルを「親セル」(parent cell)，分割されたセルを元のセルの「子セル」(child cell)，セルの世代を「レベル」(level)，そして，最上位のセルを「root セル」，末端のセルを「leaf セル」と呼ぶことにする．それぞれセルは子セルと親セルへのポインタを持ち，ツリーを辿ることによって任意のセルから任意のセルへ到達することができる．ツリー型データ構造を構成するセルのうち，ツリーの末端にある leaf セルが同じ場所の中で最も小さいサイズのセルとなる．ここでは，leaf セルだけを実際の流体解析に利用する．また，解適合などにおいてセルを再分割したり，削除したりする場合も leaf セルだけが対象となる．

等方的にセルを分割する場合には必ず $2^3=8$ つのセルに分割されるため，このようなツリー構造は”Octree” と呼ばれる．非等方分割の場合はセルを分割する方向が n 方向であるとすると 2^n-tree となり，等方分割はこれを $n=3$ に限定した特殊な場合となる．

2.4 基本となる計算アルゴリズム

実際に計算を行うためには全ての leaf セルを辿る必要がある．これはツリー型データの場合，関数再帰呼び出しを用いた次の非常に単純なアルゴリズムによって実現できる [6]．

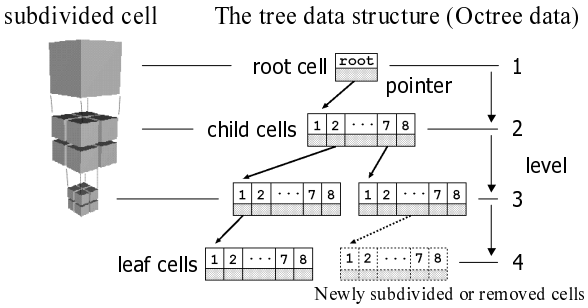

図 5: The tree data structure

```
sub(cell pointer){
  if (the cell has children)
    for each child cell
      sub(child cell pointer);
  else
    procedures on the cell;
  endif
  return
}
```

図 6: Pseudo program code to visit all the leaf cells

非等方セル分割の場合は，図 6 中の子セルについてのループ (`for each child cell`) において，図 4 中に記した番号順に子セルを呼び出せばよい．このプログラムを実行すると，例えば図 7 左図のセル分布の場合，同図中の右図に示した順序でセルを処理することになる．

この順序は空間充填曲線 (Space-filling curve) の一つである Morton ordering あるいは Z-ordering に相当する．この他にも Peano-Hilbert ordering といった ordering もあるが，後述するように陰解法の適用を考慮してここでは Z-ordering を利用する．こういった空間充填曲線による ordering は一般的にデータの局所性が高まるためスカラー型計算機に適していると考えられるが，どれほどの効果があるかまだ検証はしていない．なお，図 6 の仮想コードで用いていた関数再帰呼び出しは一般にオーバーヘッドが大きいため，実際のプログラムでは ordering のために計算初期だけに実行する．

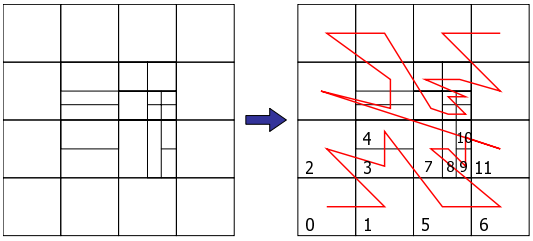

図 7: Z-ordering path to visit all the leaf cells

2.5 隣接セル探索

流束を計算するためには対象とするセルに隣接するセルを見つける必要がある．ツリー型データでは親子関係を辿ってゆき隣接セルを探索する方法が一般的である．この時，セルに「系譜」(pedigree) とよぶ key を割り当てることによって，隣接セルの探索を容易に行うことができる [6]．ここで,「系譜」とは root セルから対象とするセルへ至る間にどのセルを経由したかを表す情報である．より具体的には図 8 に示すように，セルの親セル内での位置を各方向に 0 か 1 によって表し（これを child number と呼ぶ），root セルから対象とするセルに至る各レベルで得ら

れた child number を各方向それぞれ一つの整数変数に格納した値である．こうして得られた値は対象とするセルが root セル内のどの位置にあるかという位置情報を表しており，したがって，隣接セルの位置情報とツリーデータ構造上の位置も図 8 下図に示すように代数演算によって簡単に求めることができる．

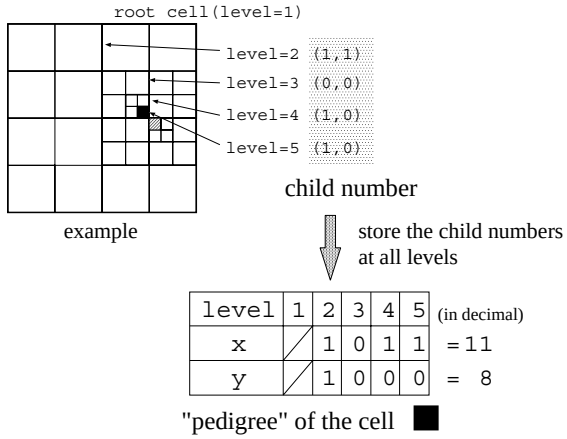

図 8: Definition of the “pedigree”

上述の求め方からわかるように「系譜」は x,y,z 各方向について独立に求められる．したがって，セルを非等方的に分割した場合でも同じ方法で「系譜」を定義することができ，それによってセルの位置情報を得たり，ツリーを辿って隣接セルを探索することができる．例として，図 9 のようなセル分布において定義される「系譜」を示す．

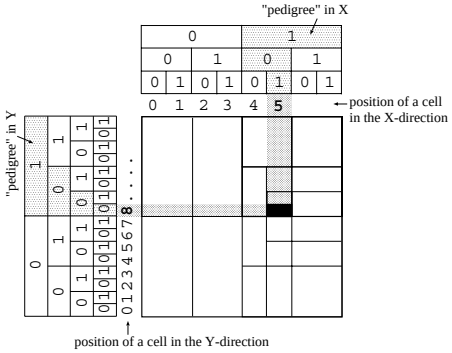

図 9: Meaning of the “pedigree,” It indicates the cell position.

しかし，非等方分割の場合は同じセルの分割数でもセルのレベルが増加してしまう場合がある．例として，あるセルを 8 つに等方分割する場合と非等方分割する場合を図 10 に比較する．8 つのセルに分割するまでに等方分割では 1 レベルであるのに対し，この例では非等方セル分割では 3 レベル必要となり，ツリーを辿ることによる隣接セル探索にそれだけ時間がかかることになる．特に，本報告では一次精度の計算であるため，すぐ隣のセルの情報しか利用していないが，空間高次精度化のためにより多くの隣接セルの情報を利用することになると，隣接セル探索に要する時間の比率が大きくなることが懸念される．そのため，ここでは x,y,z 各軸の正負それぞれの全 6 方向の隣接セルを記憶しておく方法を取った．ただし，隣接セルに子セルがある場合はその親セルを記憶し，子セルは親セルから探索することにより必要となるメモリを少なくした．

2.6 陰解法

セル再分割を繰り返すとセルサイズが小さくなり時間刻み幅が制限されるため，陰解法が重要となる．ここでは，代表的な

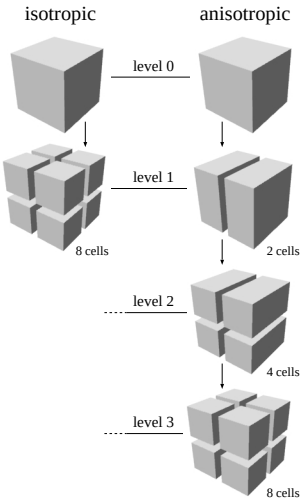

図 10: Level difference between the isotropic and the anisotropic cell subdivision

手法である LU-SGS 法 [12] を例にツリー型データ構造へ陰解法を実装する方法について述べる．

LU-SGS 法は次式のように前進，後退二つの Gauss-Seidel スィープによって構成される．

$$\begin{cases} Forward: & \Delta Q_i^* = D^{-1}(RHS^n - \sum_l^{lower} s_l A_k^+ \Delta Q_k)\,, \\ Backward: & \Delta Q_i = \Delta Q_i^* + D^{-1} \sum_l^{upper} s_l A_j^- \Delta Q_j\,. \end{cases} \quad (4)$$

ここで，上式中の *upper* および *lower* はセルの上方，または下方にあるセルを指す．すなわち，前進スィープでは自分より下方のセルを，後退スィープでは上方のセルの値を利用する．

これらのスィープは非等方セル分割の場合も等方セル分割の場合と [6] 同様に，分割形態に応じて図 4 の図中に記した番号順に Z-ordering を行うことによって実現できる．後退スィープでは前進スィープと反対の経路を辿ればよい．具体的な例を図 11 に示す．この図からあるセルの lower セルと upper セルがそれぞれ前進，後退スィープで事前にスィープされることがわかる．

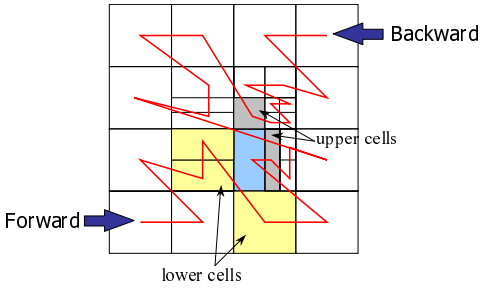

図 11: Z-ordering for the LU-SGS scheme

2.7 所要メモリ

適合直交格子法では物体適合格子と異なり，物体境界でセルを細してゆくことによって形状適合性を高めているため，流体解析プログラムにはセル当たりに必要となるメモリを極力減らすことが求められる．そこで，前節までに述べたアルゴリズムで 1 セル当たりどれくらいメモリが必要となるかを次表にまとめてみる．ただし，これは空間一次精度の場合についてであり，今後，高次精度化や演算速度のチューニングなどによって変化する可能性もある．

項目	word
物理量 (p, u, v, w, $\cdots$)	$4+\alpha$
RHS (Δp, Δu, Δv, Δw, $\cdots$)	$4+\alpha$
親セルへのポインタ	1
子セルへのポインタ	8
隣接セルへのポインタ	6
「系譜」(x, y, z 各方向)	3
セルレベル (x, y, z 各方向)	約 1
データ管理用情報 (セルの材質，分割形態など)	約 2
合計	$(8+\alpha)+21$

表 1: Memory consumption per cell

データの中には 1 byte 単位で管理している情報もあるため，それらは 4 byte で 1 word として換算した.「物理量」と「RHS」を除いた表の下段がツリー型データとして必要となるメモリで，その合計は約 21 word となる．従来の直交格子法では 1 格子点あたりに必要となるメモリはこれよりも更に少ないが，ツリー型データによる適合直交格子法では任意の場所に任意の解像度を得ることができるので，問題によっては適合直交格子の方が全体のメモリ量が少なくなる可能性もある．

3 計算結果

非等方分割の計算例として図 12 に二つに並んだ直方体まわりの流れ場を示す.

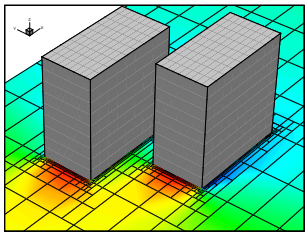

図 12: Flow around two blocks

4 まとめ

ツリー型データ構造に基づく適合直交格子による流体解析手法を非等方セル分割に拡張した．そして，等方分割のために開発されたアルゴリズムがほぼそのまま非等方セル分割の場合にも利用できることを示した．本報告では試みていないものの，Multigrid 法も等方セル分割の場合と同様に非等方セル分割時でも容易に実装できるものと考えられる．

今後，様々な流れ場について結果の検証や演算効率の評価を行うとともに，高次精度化と並列化などについても取り組む予定である．特に，高次精度化に関しては，ステンシルの取り方や隣接セルのさらに隣のセルへのデータアクセスなど検討すべき点が多い．

参考文献

[1] 佐藤ら.「エンジン房内流れ解析システムの開発」. 自動車技術会学術講演会前刷, Vol. 956, No. 9540156, 1995.

[2] 小野ら.「3 次元流れ解析によるラジエータ通過風量の予測」. 自動車技術会学術講演会前刷, Vol. 912, No. 912223, 1991.

[3] Welterlen, T.J. and Karman, S.L. “Rapid Assessment of F-16 Store Trajectories Uing Unstructured CFD”. *AIAA paper*, No. 95-0354, 1995.

[4] Melton, J.E., Berger, M.J., Aftosmis, M.J. and Wong, M.D. “3D Applications of a Cartesian Grid Euler Method”. *AIAA paper*, No. 95-0853, 1995.

[5] Zeeuw, D.D. and Powell, K.G. “An Adaptively Refined Cartesian Mesh Solver for the Euler Equations”. *J. Comp. Phys.*, No. 104, pp. 56–68, 1993.

[6] Ogawa, T. “An Efficient Numerical Algorithm for the Tree-data Based Flow Solver”. *Computational Fluid Dynamics 2000*, pp. 337–342, 2000.

[7] Wang, Z.J., Chen, R.F., N. Hariharan, A.J. Przekwas and D. Grove. “A 2^N Tree Based Automated Viscous Cartesian Grid Methodology for Feature Capturing”. *AIAA paper*, No. 99-3300, 1999.

[8] Berger, M. and Aftosmis, M. “Aspects (and Aspect Ratios) of Cartesian Mesh Methods”. *Proceedings of the 16th ICNMFD*, pp. 1–12, 1998.

[9] Lahur, P.R. and Nakamura, Y. “Anisotropic Cartesian Grid Adaptation”. *Trans. Japan Soc. Aero. Space Sci.*, Vol. 44, No. 143, pp. 31–39, 2001.

[10] Chorin, A.J. “A Numerical Method for Solving Incompressible Viscous Flow Problem”. *J. Comp. Phys.*, No. 2, pp. 12–26, 1967.

[11] Roe, P.L. “Approximate Riemann Solvers, Parameters Vectors and Difference Schemes”. *J. Comp. Phys.*, No. 43, pp. 357–372, 1981.

[12] Yoon, S. and Jameson, A. “Lower-upper Symmetric-Gauss-Seidel Method for the Euler and Navier-Stokes Equations”. *AIAA J.*, No. 26, pp. 1025–1026, 1988.