Adaptive Mesh Refinementライブラリの開発研究

Adaptive Mesh Refinement Library

山田 良透[1)]　　宮下 尚[2)]
Yoshiyuki YAMADA　MIYASHITA Hisashi

1) 京都大学大学院理学研究科物理学第二教室天体核物理学研究室（〒606-8502 京都市左京区北白川追分町 E-mail: yamada@amesh.org）

2) 京都大学大学院理学研究科物理学第一教室非平衡研究室（〒606-8502 京都市左京区北白川追分町 E-mail: himi@meadowy.org）

ABSTRACT. We construct support tools for numerical simulation. This library supports to solving the problem, which is written by the system of partial differential equations, with high accuracy by using Adaptive Mesh Refinement technique with the lowest cost. The Adaptive Mesh Refinement method is applicable in various simulations with mesh.

1　背景

Adaptive Mesh Refinement 法と呼ばれるメッシュを用いたシミュレーションを高解像度で行うための、シミュレーションサポートツールのライブラリ開発を行う。我々の申請は、具体的な問題に依存したシミュレーションコードそのものの作成ではなく、汎用的なシミュレーションサポートツールの作成である。

1.1　実用分野から見た数値シミュレーション

製品開発においては、物理学の法則に基づくシミュレーションが用いられる場合が多い。ビルを建設することに伴う空気の流れの変化、列車や自動車などの空気力学的特性には流体力学シミュレーションが利用される。希薄な気体の中を飛行するスペースシャトルのような宇宙機や、小さい構造物であるハードディスクのヘッドまわりには希薄流体シミュレーションが用いられる。ガソリンエンジンの内部の燃焼効率を上げるためには、反応入りの流体シミュレーションが必要である。また、電源トランスの設計では磁場のシミュレーションが行われる。携帯電話の電磁波が医療器具や航空機のシステムに干渉することが問題となっているように、情報化社会が進めばますます電磁波の影響を正確に予測することは重要に成ってくるであろう。この他、天気予報や災害予測などにもシミュレーションは日常的に行われている。

物理法則をシミュレーションしようとすれば、その法則が表す偏微分方程式を解くことになる。近年、計算機の高速化により、シミュレーションは空間の対称性を仮定せず 3 次元で行われるようになってきた。また基礎研究では物理法則として例えば流体とか電磁場とかひとつの物理法則に着目するシミュレーションも意味を持つ場合もあるが、製品開発など現実問題に適用する場合には、いろいろな効果を取り入れた形のより現実的なシミュレーションが要求される。材料特性や他の物理的特性の要求から形状が複雑になる場合もある。このような複雑な構造の効果を取り入れる方法として有限要素法が用いられるが、有限要素法は要素の形やつながり方が複雑である分計算スキームに制限を受け、スキーム上の精度向上が一般的には難しい。また粒子的な計算スキームも研究されている。しかし、計算スキームの精度に関しては単純な形状のメッシュ計算の分野がもっとも進んでおり、より高精度の計算結果を得ようと思えば直交格子によるメッシュ計算で複雑な形状を扱うことができるのが望ましい。

しかし、メッシュ計算を行う場合、空間の次元が 3 次元である場合、シミュレーションの解像度を 2 倍高めようとすれば、メモリーで 8 倍、時間発展計算が必要な場合は時間解像度も関係するから計算時間では少なくとも 16 倍必要となる。亜音速気体や電磁場では Poisson 方程式を含むので、計算量はメモリー量の 1.5 乗以上のコストを必要とする。従って、単純にメッシュを細かく切って大規模シミュレーションを行う場合、解像度を 2 倍向上させるだけで 50 倍以上、3 倍向上させるために 500 倍もの計算コストの増大となる。まだ、多次元計算で十分な解像度のシミュレーションを行うことは容易なことではない。そこで、約 15 年前から米国の研究者が Adaptive Mesh Refinement 法という計算法を開発した。この計算法は、必要な部分に細かいメッシュを張り、全体は粗いメッシュで解くとにより、計算機リソースを節約しつつ高解像度の計算を実現しようというものである。この方法は、基礎物理分野では流体現象やプラズマ現象における特異性の発生問題、宇宙の構造形成問題などに応用され、実用的には航空機の機体設計や NASA で天気予報のためのライブラリ開発にも用いられている。

まず、このような複雑な計算方法の必要性と有効性を示す前に、物理法則のシミュレーションとはいかにして行われるかという点を整理してみることにする。これは、今後シミュレーション分野でプロジェクト的研究をする上で、重要な指針となる。

物理法則のシミュレーションを行う場合、典型的な場合は、以下の手順が踏まれることになる。

(1). 対象のモデル化に基づく、偏微分方程式の導出　この段階では、対象の研究手法を下に、偏微分方程式を記述する。例えば、物理学的な手法で対象をモデル化する場合は、物理学の理論に基づいた方程式が導出される。

(2). 空間・時間の適切な離散化に基づく、計算スキームの導出　この段階では、数値計算の技術が要求される。与えられた連続的な偏微分方程式を、どのように空間的・時間的に離散化するか、また、時間積分などの方法はど

のようにして行うかをこの段階で決定する。分解能、解像度が、数値計算結果に与える精度への影響は、この計算スキームが左右することになる。

(3). 計算スキームに基づいて、実際のコードを作成。
(2) で、決定したスキームを実際の FORTRAN や C 等の計算機言語で記述する。

(4). コードの実行 計算結果の収集　作成したコードを実行し、計算結果を取得する。計算の量や質に応じた計算機リソースが要求される。

(5). 計算結果の誤差評価　計算結果の誤差は、要求される精度の範囲内に収まっているかを検証する。

(6). 計算結果の解釈　もともとのモデルの観点から、計算結果を解釈する。例えば、物理学においては、物理学的な観点から、計算結果を捉えなおす。

一般にシミュレーションを行うためには、(2) で示したとおり、空間、時間の離散化が不可欠であり、精度を向上させるためには、その分解能および解像度を上げなくてはならない。しかし、時間分解能を a 倍に向上させると、当然、その計算量も a 倍に増えてしまう。また、空間解像度を b 倍に向上させると、b の d 乗 (d は次元数) 倍で計算量及び、要求記憶量が増してしまう。しかも、近年、計算機の高速化に伴い、(2) を出来るだけ自然な形で実装し、導出された偏微分方程式を、そのまま解く事例、及び、要求が多くなっている。例えば、従来であれば、計算量を劇的に減らすために空間の対称性を仮定して次元数を減らすなどの細工を必要とした。しかし、このような手法は一般的なものではなく、多様化していくシミュレーションの要求に答えきれなくなっている。

シミュレーションの分野で近年のハードウエアの進歩を十分に生かすためには、メッシュの張りなおしなどの操作を行うシミュレーションサポートツールの整備が急がれる。しかし、Adaptive Mesh Refinement ライブラリの作成にはシミュレーションの対象となる現象の知識、計算法の知識以外に高度なソフトウエア工学上の知識が必要となるため、従来シミュレーションを行っている研究者や技術者が手を出し難い分野である。申請者らは、我々が従来行ってきた技術開発をより広範に利用できるようにするために、この時期に大きな予算を獲得していっきにライブラリ開発を進展させたいと考えている。

1.2　従来の高解像度計算へのアプローチ

現在、複雑な形状を含む場合、あるいは部分的に高解像度が必要となるシミュレーションには、2 次元では直交格子ではなく形状適合格子を用いる手法が使われる。しかし、この方法には三つの欠点がある。第一に形状が複雑になると格子形成にもかなりのコストがかかること、第二に形状が時間的に変化する系を扱えないこと、第三に時間的には計算空間全体にわたって一様な離散化を行うため、もっとも小さな空間離散化度で時間離散化度が決まり、その結果場所によっては不必要に時間離散化度を小さくとることになることである。また、第一の欠点は三次元計算となると決定的となり、三次元の計算では形状適合格子はあまり用いられていない。

有限要素法を用いるアプローチも行われている。これは、上の第二の欠点を補う可能性がある。しかし、第一の欠点に対しては、やはり格子形成にかなりのコストをかけざるを得ないのが現状である。また、有限要素法は形状に対する柔軟性をもつ反面、有限要素の複雑な形状から計算精度の向上に限界があることが知られている。

格子的離散化ではなく粒子的離散化を行うアプローチも試されている。流体では、この方法は SPH(Smoothed Particle Hydrodynamics) として、最初天文学の研究者が観測の解釈を手軽に行うために開発した手法がある [Gingold and Monaghan]。実装の手軽さから宇宙物理学では広く用いられているが、精度が必要な実用的な計算でこの手法を用いるのは、プロペラやスクリューなど格子形成が困難な一部の問題に限定されており、航空機や自動車などの開発でこの手法を用いているという話は聞いたことが無い。現在では、SPH 法の改良版として精度を向上する手法も提案されて入るが [Inutsuka]、粒子的離散化を行う最大のメリットである実装の手軽さは全く失われてしまっている。

Adaptive mesh 法は、これらの方法の欠点を補完する以下の性質を持つ。

- 時間的にも空間的にも必要精度に応じた離散化を行うことにより、計算機リソースを有効に利用できる。
- 形状の複雑さによらず、離散化手続きが自動で行われる。
- 基本的に直交格子を使っているため、計算精度の向上についてはノーハウが蓄積されている。

このように、ユーザーにとってはメリットが大きい方法ではあるが、反面実装は複雑である。従って、個別の分野でそれぞれの計算コードを Adaptive mesh 法に載せる努力を行うことは、実装コストがかかることから敬遠される。

1980 年代の論文で、この方法の原理はほぼ確立した。ただ、この時代には、計算機能力のせいもあるが、精度評価はメッシュ間隔で 3 段階程度の Adaptive Mesh 計算で行なわれている。しかし、実際の計算では 10 段階、あるいはそれ以上の細かいメッシュを張り、詳細な計算を行なっている。

1990 年代は, メッシュ分割アルゴリズムの改良, 輻射輸送など流体以外の系への拡張などに関する研究がいくつかあるようである。また, この方法を応用した研究が増えてきている。これら場合にも, 実装のチェックと言う意味合いで精度評価を行なっている場合がある。しかし, 我々がサーチした範囲ではその評価は格子間隔のレベルで 3 段階までに留まっており, 実際の run で使っているように, 10 段階程度のメッシュを作った場合の精度評価をしたものはない。

同じく Adaptive Mesh Refinement 法として呼ばれる [De Zeeuw and Powell, Khokhlov] らの方法がある。これらを区別するため、今回我々が実装している方法をメッシュベースの AMR、[De Zeeuw and Powell] らの方法をグリッドベースの AMR と呼んで区別している。この方法は、長方形 (あるいは直方体) のメッシュを使うのではなく、格子点そのものを解像度比が 2 倍の場合 2 次元で 4 分割、3 次元では 8 分割した格子点を用意するものである。この方法は、我々の方法に比べて二つの優れた点を有する。第一にメモリー使用量が効率的である。もう一つは基本的にグリッドのレベルで階層化されているため、同一の点を代表するグリッドポイント上の物理量の整合性の確保が簡単である。一方、三つの大きなデメリットがある。第一に、我々の方法はメッシュを構造化の基礎にとっておりその要素数は高々1000 程度であるが、[De Zeeuw and Powell] らの方法では構造化するのは格子点であり、その数は 100 万以上に達する。実装上は、適切な Garbage collection を実装し、検索などの操作にデータベースソフトウエアのノーハウを利用するなどしなければ、効率良いプログラムを作成することは不可能である。第二に、同一レベルでの計算が集中するため、メモリーアクセスの効率向上のためには適切なタイミングでデータ点をソートする必要があり、そのためにオーバーヘッドがかかる。第三に、工業的応用を意識する場合制限方程式の実装が不可欠であることを述べたが、制限方程式の効率良い実装は非常に難しい。実際宇宙物理学でこの方法を用いている研究者は、制限方程式に粒子的方法を併用しており、生成した構造化格子の中では制限方程式を解いていない [Yahagi and Yoshii]。このデ

メリットは、汎用的なライブラリにすることに障害となるため、今回我々はこの方法を採用しなかった。

2　目的

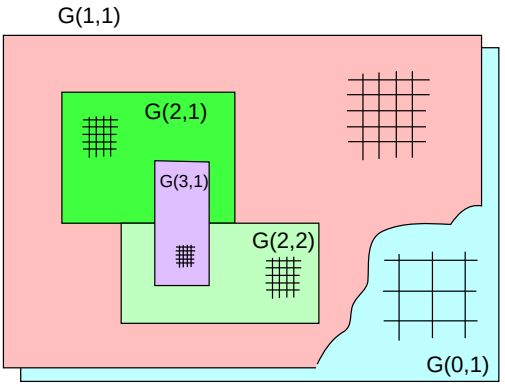

図 1: Adaptive Mesh 法におけるメッシュの空間配置の概念図。様々な離散化度のメッシュG(i,j)が、空間的に重なり合い、同時時間発展を行なう。時間発展においては、それぞれのメッシュは、近傍の他のメッシュから適切に初期条件と境界条件を与えられる必要がある。

この方法は、一般的な偏微分方程式の時間発展問題を、離散化誤差の大きいところに図 1 のように空間的に、また時間的にも局所的に分解能を向上させることにより、わずかな計算機資源しか消費しないにも関わらず、全体的に小さな離散化誤差で計算を行い、計算時間を著しく短縮させることを目的とする。即ち、同程度の離散化誤差の計算を行う場合においては著しい計算時間の短縮、計算の高速化をもたらすことになる。この方法では、離散化誤差の大きな部分を含む長方形領域に、格子間隔の小さな新たな長方形メッシュを張り、格子間隔の大きなメッシュと格子間隔の小さなメッシュの時間発展計算を同時に行う。格子間隔の小さなメッシュを作成する場所は、空間的時間的に限定的であるため、計算機リソースの消費も少なく抑えることができる。更に、計算の途中で格子間隔の大きなメッシュでも離散化誤差が十分に小さいと判断すれば、その領域の格子間隔の小さなメッシュは取り除くことにより、計算機リソースを解放する。この方法ではメッシュの生成消滅、整合性の確保にはオーバーヘッドがあるが、空間的・時間的に限定された範囲で格子間隔の小さなメッシュを生成することによりオーバーヘッドを相殺して余りある効果がある。

等間隔メッシュでのシミュレーションは、様々な分野で実績があり、またそのようなライブラリはすでに市販品や会社の研究所や大学の研究室などでも蓄積がある。この方法でのメッシュ生成は長方形の等間隔のメッシュを基本とするため、従来の計算法上の様々な知識をそのまま適用できる利点がある。しかし、個別のライブラリを Adaptive Mesh Refinement 法の枠組みに載せることは容易ではない。我々は、格子の生成消滅、物理量の新たに生成された格子への複写、同じ空間領域に重ねて張られた格子上での物理量の整合性の確保、格子の境界での境界条件の適用などこの方法特有の操作を、極力具体的なシミュレーションコードの内容と独立性を保った形でライブラリ化し、シミュレーションの汎用的な道具としての Adaptive Mesh Refinement ライブラリを作成しようと考えている。これとともに、動的に必要な部分にメッシュをはるために、あらゆる問題に対して少ない計算機リソースで高解像度の解を得ることができる利点がある。

我々が現在実装中のプロトタイプライブラリは、既に提案されているものに対して原理的な部分で二つの改良を行った。

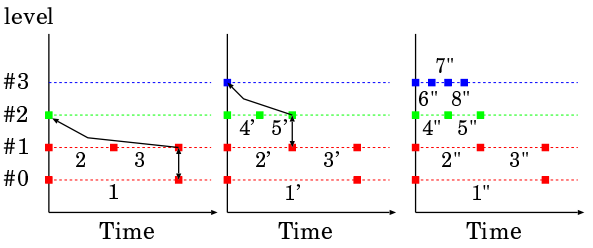

図 2: Rewind の概念図。グラフ軸の# 数字で書かれたものは、離散化度のレベル (本文参照) を表す。グラフ中に書かれた数字は時間発展順序を表す。図で、# 0 と# 2 は 1 から 3 の時間発展プロセスの後同期し、誤差評価を行なう。この段階で誤差が閾値より大きい場合、従来は 1 あるいは 3 の終了時刻で# 2 のメッシュを生成し、計算を開始した。我々の改良点は、# 2 の生成を 1 及び 2 の時間発展の開始時刻で行ない、続く 4 以降の計算を行なうことである。

巻き戻し　計算途中で誤差があると判定された場合、従来は比較対照となった計算の終了時点で離散化度の小さなメッシュを生成して計算を継続していた。我々は、比較対照となった計算の開始時点に戻って離散化度の小さなメッシュを生成して再計算を行う点を改良した。(図 2)

FCEE　誤差評価に、従来は異なる離散化度の小さい計算の平均値と離散化度の大きな計算の値を比較していた。我々は、離散化度の大きな計算結果から離散化度の小さい格子点での値を生成する内挿手続きを行った後で、離散化度の小さい計算結果と比較する点を改良した。

今回、このプロトタイプのライブラリに対して 3 点の改良を行うことを計画した。

ユーザーインターフェースの整理　このライブラリは基礎物理学の研究のために作成されたものであり、工学的目的などで他の問題に適用する場合にはライブラリの内容を熟知していなければならない。今回我々は、ユーザーインターフェースに相当する部分を整理しなおして、より汎用的にこのライブラリを使えるように改良すること。

汎用可視化ツールの構築　Adaptive Mesh Refinement 法のデータ構造は複雑で、可視化も汎用可視化ツールがサポートするデータ構造にダイレクトに当てはまるものではない。何らかのデータの加工を施さなければ、計算結果を見ることができない。しかし、この加工作業は Adaptive Mesh Refinement という枠組みを使う限りは汎用的に設計可能なため、可視化のためのインターフェースの設計及び実装を行う。
シミュレーションにおいて、特に高次元や高精度の計算でデータ量が多い場合、数値を書き出すだけでは現象を把握するだけでなく、プログラム上のバグを追跡する上でもコストがかかりすぎ、現実的ではない。我々のライブラリを多くの分野の技術者・研究者に利用してもらうためにも、組み込み上の不具合を修正するために可視化ツールは必須である。

real time 可視化　最近のパーソナルコンピューターの高速化に伴い、可視化ツールはかなり高機能になってきている。そこで、real time に可視化を行う可能性が開

けており、real time 可視化は現象の理解にも大変役に立つ。そこで今回、可視化ツールの機能を生かして real time に可視化できるようにするためのインターフェースを開発、実装する。

3 開発内容

3.1 本申請までの我々の開発状況

我々は、精度が不足した計算の「前」で細かいメッシュを生成するための改良を行った。概念図は図 2 に示したとおりである。これは、単にプログラムステップ中の精度チェックの位置を替えることにとどまらず、「前」から計算しなおすのに十分なだけ過去の計算結果を保存する必要があり、そのためにデータ構造の変更が必要である。

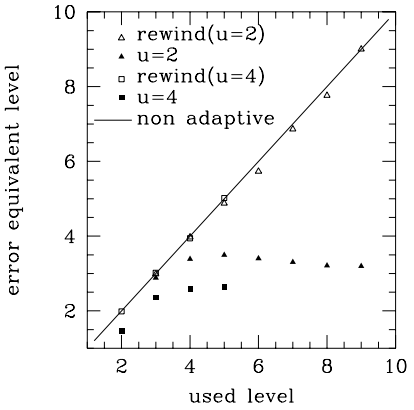

図 3: AMR 巻き戻し法の精度チェック

我々の改良した Adaptive Mesh Refinement 法の有効性を、数学的に厳密な解を持つ問題にこの方法を適用することで検証した。実際 10 段の階層で精度評価を行なった例は、他にない。従来の時間発展アルゴリズムだと、3 段階程度の計算でほぼ計算精度は収束してしまう。10 段階メッシュを作ったとしても、その計算の精度は 3 段階の計算程度であることを明らかにした。そこで、我々は時間発展の方法を改良し、精度の不十分な計算の部分は細かいメッシュを生成した後で、精度の不十分な計算の「前」から再計算するように改良した。この再計算のためにはデータ構造の変更が必要となり、かなり大がかりな変更となっている。それにより、一次元の衝撃波管問題では、2 倍づつ細かいメッシュを 10 段作っても、等間隔で 1024 倍細かいメッシュを使った計算と同程度の精度を得ることができた。図 3 にその結果を示す。

われわれは、この手法を「二次元自由対流における特異性の発生問題」と呼ばれる基礎物理学の問題に適用した。温度勾配が 1000 倍も向上するような場合について、特異性に近づく状況を安定かつ精度良くシミュレーションできることが分かった。なお、Adaptive mesh 法の原理、および流体現象への応用については、我々は「物性研究」に日本語の解説記事を書いた [Yamada and Miyashita]。

我々の開発してきた Adaptive Mesh Refinement コードは、離散化誤差に頭打ちが無く、階層を重ねてゆけばそれだけ十分な精度で計算ができる点が、既存の Adaptive Mesh Refinement ライブラリより優れている点である。また、既存のライブラリは流体計算だけを意識してかかれている場合が多いが、我々のライブラリは様々な分野のシミュレーションに応用可能なように、メッシュ操作と物理計算の独立性を極めて高く維持している点でも特徴がある。

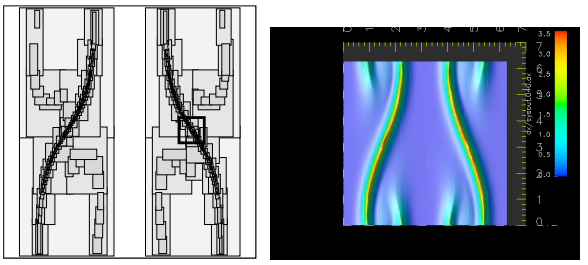

図 4: 二次元自由対流における得意性の発生問題の計算結果。右は温度勾配の分布図、左はある計算スナップショットにおけるメッシュ生成の例。

我々は、流体力学や相転移問題などの基礎物理学の問題に興味を持ってライブラリを開発しているが、上のように独立性と汎用性の高いライブラリ設計になっているため、工学的な様々な分野への応用も容易である。

また、少なくとも基礎科学分野では、シミュレーションには実行速度の向上が最優先されたため、オブジェクト指向技術に代表される先進的なソフトウエア技術を積極的に取り入れてくる努力が行われてこなかった。Adaptive mesh の実装においてはこれらのソフトウエア技術は不可欠だが、直交格子を基本にしているため、実行性能に大きくかかわる計算部分の独立性が高く、離散化度や空間配置が異なるメッシュの間の整合性を保つ操作など、抽象化が威力を発揮する部分が多少オーバーヘッドがかかる実装になっても、全体の性能に大きな影響を及ぼさない。従って、汎用ライブラリにすることが容易である。粒子法や有限要素法をベースにした場合には抽象化可能な部分と計算コストにかかわる部分の分離が難しいため、汎用ライブラリ開発には向かない。

3.2 本計画で行なった改良

我々のライブラリ開発では、以下の点が特に重視されている。

(A) 解くべき基礎偏微分方程式、また、そのための計算スキームから、ライブラリは完全に独立した設計にする。依存部が不可欠である時でも、適切な抽象化層 (abstraction layer) を用意し、上位層の依存を避けるようにする。

(B) ライブラリの使用者は、使用者の問題に関して選択された計算スキームに基づく実際のコード化の作業だけに集中できるようにする。つまり、計算スキーム及び、離散化の手法、変数の性質に基づく平均化の部分の実装だけが必要とされるように、充分なサービスを、一般性を失わない範囲で提供する。

(C) 性能、および、その拡張性 (scalability) を重視する。各 component は、出来るだけ効率の良い実装方法を追求する。また、並列化についても重視して設計しており、各 object の相互依存性を出来るだけ排除するように設計している。実際、現状で、POSIX thread library に基づく、並列化が実装されている。また、thread model に基づく並列化だけでなく、message passing model に基づく並列化も考慮中である。

(D) (C) に悪影響を及ぼさない範囲で、可搬性を重視する。中枢部のコードは、ANSI C++ および、ANSI C の標準を逸脱しないように書かれており、特定の machine architecture への依存は完全に排除されている。また、コンパイル作業も GNU autoconf を用いて可搬性を高めており、一般的な POSIX system には、容易に移植することが出来るようになっている。また、Windows 上でも Visual C++で動作するようになっている。

我々のライブラリは、整合格子メッシュを基本としている。利用者はライブラリ側から与えられた整合格子メッシュに対して適切な時間発展方法と、誤差評価方法、離散化方法を記述するだけで、利用できるように設計されている。ライブラリは、ユーザーの指示に基づき、適切なメッシュの生成、消滅を繰り返すことによって、誤差を許容範囲内に抑えながら、時間発展を自動的に行うようになっている。ここまでは、現状のライブラリで、既に実現されていることである。

申請により実現しようとする内容には、主に二つある。一つは、ユーザーインターフェースを整理し、より広範囲な問題に適用可能な汎用の Adaptive Mesh Refinement ライブラリを作ることである。もう一つは、計算内容の表示、およびプログラム開発のサポートの双方の目的から、可視化ツールを構築することである。以下、その内容について説明する。

このタイプのライブラリの開発は、「現象」、「計算法」、「ライブラリ設計」の３つの分野に関してそれぞれ深い専門的知識を持った上で、それぞれの要求の妥協点を模索してゆかなければならない。今までプロトタイプの開発において、山田は宇宙物理学における爆発現象の計算などで計算法の専門知識を持っており、また宮下は emacs を始めとする数々のソフトウエア開発に関わった経験からライブラリ設計に関する専門的知識を持っていた。これまでライブラリの中枢部分に関しては、二人が議論して合意することにより設計が進められてきて、このことが、計算方法上もライブラリ設計上も妥協できるバランスの良い設計を実現してきた。今後も中枢部分の開発にはこの方法を継続するのが最も良い。

一方精度を追求するには可視化が避けられないし、またより汎用性の高いものを書く場合ユーザーインターフェースやドキュメンテーションといった作業も必要となる。可視化、ユーザーインターフェース、ドキュメンテーションに関しては、仕様を決定すれば外注も可能であり、　また我々二人がこれらのライブラリ開発に関わることは、ライブラリプログラム全体の開発効率から言って望ましいことではない。そこで、今回これらの部分は外注することとし、それにより開発者が中枢部分の開発に集中できる環境を作ることが、このライブラリプログラムの発展にとって非常に重要である。

3.2.1　汎用ライブラリ化

現在、我々はこれらのライブラリを更に高精度かつ汎用性の高いものに改良する作業を行っている。このタイプの計算方法の問題点は、局所的に解く事のできる問題には強いが大局的な影響をもつ問題に対する具体的な処方箋がかなり複雑であるという点である。そして、実際特に工学的な側面で興味をもたれるような状況では Poisson 方程式が含まれる。超音速気体では方程式は移流型だが、通常我々の周りに存在する亜音速気体では制限方程式を含む。また、電磁場も Poisson 方程式に支配される。つまり、大局的な力を容易に導入できるようにライブラリを改良することは、工学的な問題に我々の計算スキームを適用する場合には必須である。

申請段階において、既にライブラリは局所的に格子の解像度を上げる手続きは提供していたが、Poisson 方程式を含むような大局的な作用がある場合には、新たな手続きが必要になる。今回我々は Multi Grid Iteration[Martin] と呼ばれる大局的な作用に有効な手続きの実装を行った。

また、今後保存則の精度を向上させる際、一般的に用いられている予測子修正子法が必要になる可能性がある。この実装には、現在ある mesh のデータ構造とは異なる新たなデータタイプとして、数値流束を格納するデータ構造が必要となる。このため、mesh_flux を新たに実装した。

3.2.2　汎用可視化ツール

問題が複雑になると、可視化が必要となる。これまで申請者らは、山田が主に計算法を、宮下が主にライブラリ設計部分を担当したが、全体設計に関わる部分では常に両者が議論して合意の上でその設計や実装方法を決定し、自らコードを書いてきた。今後もその開発手法は維持される。しかし、3 次元可視化は未経験者には非常に困難であり、また世の中にはこれに関わるノーハウが溢れている。また、開発ライブラリの中枢部分に比べると比較的仕様を明確にしやすく、また独立に構築できる部分も含む。さらに、3 次元可視化は今後このライブラリを 3 次元問題に適用するには避けて通れない部分であり、その開発は急がれる。

可視化には、可搬性を考慮し、OpenDX あるいはこれに相当する高機能な再配布可能なソフトウエアをベースにする形でかかれることが望ましい。実際申請者らは 2 次元の可視化を OpenDX の上で行っているが、本年は、この機能強化のための外注費用に予算を利用することとする。

今回、我々が開発を進めてきた Adaptive Mesh Refinement ライブラリをより一般性の高いライブラリに改良するため、ユーザーインターフェースとグラフィックスの部分の開発、およびドキュメンテーションを行う。これにより、今後他の分野の計算に応用するための計算法上の新たな要求をできるだけ反映し、より汎用性の高いシミュレーションライブラリにしてゆく基盤を作ることで、シミュレーションライブラリの標準化に結び付けられると良いと考えている。

我々が OpenDX を選択した理由は、以下のようなものである。

Open source
導入コスト
透明性
設計の良さ

以上のような利点は、どれも、非常に我々が重視したポイントで、これらに関しては OpenDX 以上の可視化システムは見当たらなかった。フリーな可視化システムは、もちろん、フリーなものも存在するが、全体的には OpenDX 以上の利点を持っているシステムは見つからなかったというのが、我々の結論である。

3.3　今後の改良点

本ソフトウエアをより優れたものにするため、今後我々は以下の改良を行なう予定である。

第一に、応用を視野に入れた場合、構造物の境界は避けて通れない課題である。可視化ライブラリが完成したので、構造物境界の実装作業のためのデバッグ環境は整った。今後、境界データの形式に関する資料を収集し、できるだけ汎用性のある構造物データを AMR ライブラリに実装することを検討している。二次元で単純な場合については、すでに [De Zeeuw and Powell] などに紹介されている構造データの実装方法がある。今後、CAD の開発担当者などとも議論する機会を作り、一般的な構造データの本ライブラリへの組み込みを行ないたいと考えている。

第二に、化学反応の実装は、このライブラリの応用範囲を広げる。適当な例題を探し、化学反応の実装に必要な改良について検討し、是非実装したい。

第三に、応用には流体の他に構造解析 (破壊) が重要である。従来、構造解析は有限要素法の独壇場と考えられていたが、ソフトウエア業者との議論により、有限要素法を持ってしても相当なコストがかかっていることを知った。適当な実装を考えると、構造解析にも有用なソフトウエアになる可能性があるので、今後これについて検討したい。

第四に、熱解析も応用分野が広い。熱解析には、環境要素にもよるが、人工衛星などでは輻射輸送が関わる。輻射は流体よりはるかに高速に伝搬するため、計算方法も特殊

であり、計算的に新たなツールを付け加える必要がある。AMR のこの方面での応用を考えることは、さらに応用範囲を広げる上で重要である。

第五に、本ライブラリは偏微分方程式で記述される系には適用可能なので、物理現象のみならず社会現象のモデル化にも有用である。そのためには、ランダムな擾乱が入り込む系を扱う必要がある。格子の整合性を確保する上で、ランダムな擾乱の適切な入れ方を検討したいと思う。

これらは、今後の課題となる。

4 全体構成

4.1 シミュレーションに於ける基本要求の抽出

シミュレーションを行なう場合、ユーザーはまず以下のものを決定する。

基礎変数　解きたい系の物理量を選択する。
座標系　問題に適した計算座標を選択する。直行座標表示か極座標表示か、あるいは物体整合座標系の特殊な座標系かと言うことを意味する。
基礎方程式　基礎変数と選択した座標系で書かれた方程式
初期値・境界値　この方程式を解くのに必要な初期条件と境界条件
要求精度　問題を解決するのに必要な計算精度

これらは、ユーザーが選択するべきものであり、ライブラリの内部に実装されるものではない。ライブラリはこれらを適切に引き渡してもらうインターフェースを用意する必要がある。

シミュレーションに於ける基本的要求を、実用シミュレーションと基礎科学に分けて、表 1 にまとめる。表を見

	制限方程式 境界値	移流拡散方程式 初期値境界値	構造物	外部境界
実用シミュレーション				
亜音速翼	非圧縮条件	非圧縮流体	形状データ	自由境界
超音速翼		圧縮性流体	形状データ	自由境界
熱解析		熱伝導方程式	形状データ	
磁場形状	電磁場		形状データ	
構造解析	構造方程式		形状データ	
エンジン	反応流体		形状データ	
プラズマ	MHD 方程式		形状データ	
天文学・基礎物理学				
I 型超新星	自己重力＋反応流体		形状データ	自由境界
降着円盤	自己重力	圧縮性流体		流入条件
銀河形成	自己重力	圧縮性流体		周期境界
流体特異性	非圧縮条件	流体		周期境界
誘電体	電磁場			周期境界

表 1: シミュレーションの基本要求に関して、表にまとめる。

ると、結局基本要求は

- 制限方程式と移流拡散方程式に対する適切な初期値・境界値を与える
- 形状データに対応する

の二点に絞られ、この要求は基礎科学においても実用シミュレーションにおいても同じであることが分かる。インターフェースを整理するため、方程式系が二階の偏微分方程式に限られ、かつ制限方程式と移流拡散タイプの方程式に分離して記述できることを仮定する。この仮定は一般的である。

AMR では、計算途中にシステムが判断した条件に応じて動的にメッシュを生成する。時間発展方程式においては、その初期値・境界値を適切にユーザープログラムに引き渡す操作が必要である。また、多数のメッシュに対して適切な順序で時間発展を行なうこと、及び同じ物理領域をカバーするメッシュに対しては、物理量の整合性を確保することが必要である。ただし、個別の物理量の扱い、及び時間発展スキームは問題依存であり、ライブラリからの独立性を確保する必要がある。

初期値の与え方については、新たにメッシュを生成した場合に離散化度の粗いメッシュデータから離散化度の細かいメッシュのデータを適切な内挿により生成する必要がある。我々のライブラリでは、IBTransfer と言う形でこの内挿手続きをユーザーがライブラリに渡すように設計されている。また、物理量として採用した量が示量変数 (質量のように 2 つの物質を合わせると和になる) であるか示強変数 (温度のように 2 つの物質を合わせると平均値になる) であるかということも、ライブラリのユーザーしか知らない。従って、ここで言う適切な平均値は和、あるいは n 個に分割する場合 1/n になる場合もある。従って、平均値を算出する関数も、ユーザーが実装し、ライブラリに教える必要がある。

数値スキームの上で境界条件がどのような形で与えられるかは、ライブラリのユーザーしか分からない。そこで、ライブラリの実装方法としては、必要な境界領域の値をユーザープログラムに引き渡す手続きのみを与えることとする。本ライブラリでは、境界領域の厚みをユーザーがパラメータとしてライブラリに与え、ユーザーはそのデータを参照して適切に数値スキームに組み込む必要がある。この厚みは、境界条件のセットだけでなく、内挿手続きで必要とするものより大きくとる必要がある。

また、構造物の境界を取り扱う必要がある。構造物が固定している場合、この構造を表すデータとの適切なインターフェースが必要となる。ユーザープログラムに対しては、各グリッドが構造物の内部、境界、外部のいずれにあるかを教えることにより、ユーザープログラムに処理を委譲する。

数値流束を計算する際、精度向上の要求から適当な内挿手続きが行われることがある。これは、一次式や二次式といった単純なものではなく、計算の安定性を保証するように多項式に適切な修正を施したものである場合が一般的である。ただ、この修正の仕方は安定性の理論に基づくものであるから、物理量を保存量になるように選択しておけば、一般性を失わない。

また、制限方程式を解く場合には Multi Grid Iteration を実装する必要があることを述べた。これについても、ライブラリは、計算途中に存在するメッシュの配置にしたがって、適切な順序で緩和プログラムを呼び、緩和中の配列の整合性を確保する必要がある。しかし、実際の緩和がどの基礎方程式に対応し、どの緩和方法で緩和を行なうのが適切であるかは、ユーザーしかわからない。従って、実際の緩和方法の実装は、ライブラリとの独立性を確保し、ユーザーが実装するものとする。

5 AMR ライブラリの外部仕様

たびたび述べているように、我々が作成したコードはライブラリである。すなわち、これだけでは動作せず、ユーザーがこのライブラリを適切に呼び出す必要がある。この章は外部仕様、すなわちユーザーがこのライブラリを利用したプログラムを記述する場合に最低限知っておく必要のある、AMR library 側の機能を説明する。より詳細な内部仕様の記述は後の章で多少行なうが、ライブラリ自体が今後も機能拡張のための機能修正を必要とする事情が予測されるため、開発者側は Web ページによる公開を行なうことを計画している。このライブラリの名称は、lupin[*1]である。また LDX は、本プロジェクトで外注された可視化ライブラリの名称 LUPIN DX ライブラリの意味である。

なお、ライセンス方針の項目で改めて記述するが、本ライブラリをどのような形で公開するかは検討中である。configure や make を行う方法は、この仕様が確定次第変更されることとなる。

5.1 概要

本ライブラリは、mesh_bisection と呼ばれるクラスの継承クラスのオブジェクトを用い、ユーザーがオーバーロー

[*1] Library Useful in Physics, Instrument of Numerical computation

ドする仮想関数を適切な順序でかつ適切なパラメータを引き渡すことにより呼び出すことにより、時間発展方程式及び制限方程式の解を得ることが出来るように設計されている。また、ユーザーのクラスのオブジェクトをユーザーがオーバーロードする仮想関数の動作にしたがって適切に生成、あるいは消滅させ、複数のオブジェクト間の必要な整合性を保持する機能を持つ。

5.2 機能仕様

5.2.1 mesh クラス

mesh は、複数階層に分かれた class 構造をしているが、ユーザーは mesh_bisection から継承されるユーザークラスを作成する必要がある。現在、mesh_restructure の継承クラスは mesh_bisection だけが実装されている。これは、大きなメッシュを効率良く小さなメッシュに分割する方法毎に実装されるもので、今後 bisection 以外の方法が実装された場合には、どのクラスから継承するかをユーザーが選択することができる。ここでは、mesh_bisection クラスから継承されるユーザークラスを作成する上で必要な mesh_bisection クラスのメソッドについて述べる。

mesh 構造は、図 5 にあるように矩形構造の geometry 情報を持っている。

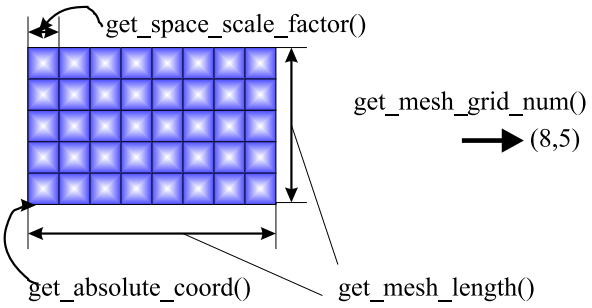

図 5: mesh_framework 関係の geometry 取得関数

5.2.2 mapper クラス

mesh のグリッド, map, elems は、図 6 のような関係になっている。

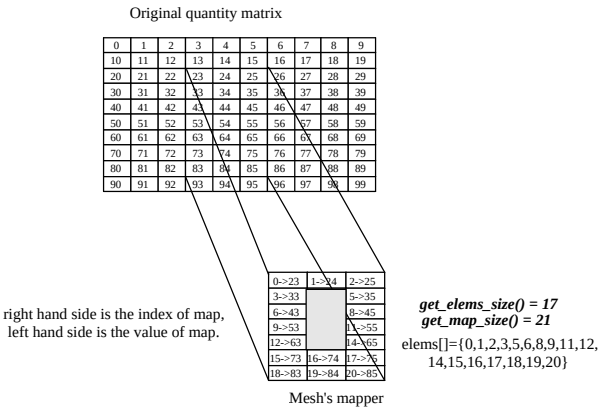

図 6: mesh, map, elems の関係

5.2.3 relaxation_array クラス

緩和法では、quantity の場合の構造体の配列ではなく、stored_object という配列の構造体を用いるのが便利であることを述べた。しかし、緩和法が用いる配列の構造体が持つべき値は、quantity と整合するものでなければならない。quantity の形式を知っているのはユーザーだけなので、緩和法の配列と quantity の整合性を確保する関数は、ユーザー関数として実装する必要がある。

緩和法のための stored_object として、現在 relaxation_arrays とその継承クラスである mgi_arrays が定義されている。まず、ベースクラスの relaxation_arrays について説明する。このクラスは、緩和行列である k 対角行列を格納する二つの配列 M と ls、k を表す変数、M が n 次行列であるとしてこの n、ベースとなるメッシュとの大きさの違いを表すパラメータ、それに、実際に緩和に用いる大きさ n の 3 つの配列を、メンバーとして持つ。

5.2.4 vector クラス

AMR library では、次元依存を排除するために vector class を導入している。図 7 を参照のこと。これ

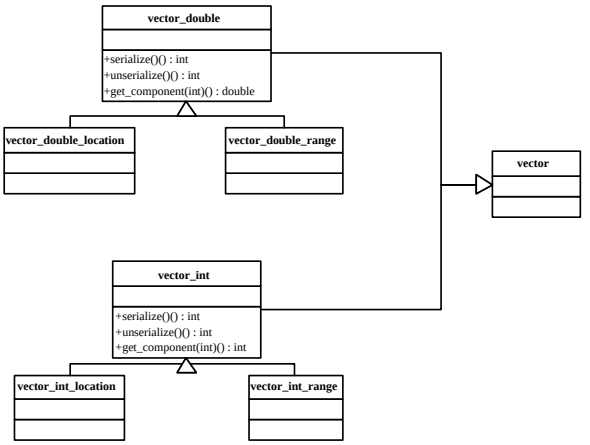

図 7: vector class ダイアグラム

は、ユーザーメッシュクラスのコンストラクタの中で mesh_bisection のコンストラクタを呼び出す場合に必要となる。vector クラスには、位置ベクトルを表すか長さベクトルを表すか、及び成分が整数か倍精度実数かにより、図に示す 4 つのクラスがある。

ユーザーは、コンストラクタの使用方法だけ知っていれば、このクラスの詳細の知識は不要である。

5.2.5 bind_mesh クラス

bind_meshes は、現在の mesh の集合を保持する class である。IT は、自分の担当する bind_meshes を格納しており、以下の method を用いることにより bind_meshes から適切に mesh を取り出すことができる。

5.2.6 IT クラス

IT は、bind_mesh のコントロールクラスである。動作によって、以下のクラスが用意されている。単一メッシュでの発展を行なう場合、

生成した bind_meshes を引数に渡して適切な IT のコンストラクタを呼び出し、その後 IT::do_operation() を実行すれば、目的の動作をするように設計されている。

5.2.7 IBTransfer クラス

IBTransfer(Image Block Transfer) は、配列から配列へのコピーを抽象化する目的で実装されたクラスである。配列の要素は構造体である場合もあるし、単純な倍精度実数である場合もある。型の違いは、適切な opif(operation interface) をユーザーが引き渡すことにより解決される。IBTransfer はメッシュ上の任意の型のデータをコピーできるよう実装するため、opif で定義される関数の引数は、`void*`型を取る必要がある。

IBTransfer の静的構造を、図 8 に示す。

6 操作仕様

6.1 dipole の結果

我々の MGI の実装結果を示す。

MGI は、制限方程式を整合的にとく手法である。従っ

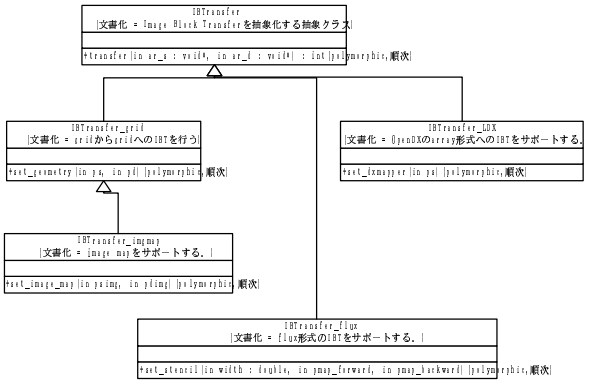

図 8: IBTransfer の静的構造

て、Poisson 方程式に従う系を扱えば、その性能が理解できる。我々は、双極子の作る電場を計算した。この計算ではベースのグリッド数は 32 × 32、upper_level_factor を 2 として 12 階層のメッシュを生成した場合のもので、双極子の間隔は計算空間の 1 万分の 1 である。従って、9 階層目くらいから双極子のプラスとマイナスの極が分離可能である。この例では、最も下の階層でもちゃんと双極子電場が見えていることが分かる。

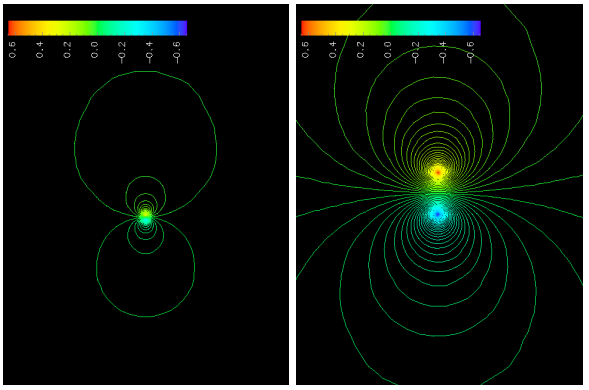

図 9: dipole の結果、左が全体像、右が双極子の近くを拡大したもの

6.2 OpenDX で LDX を用いた例

OpenDX を用いて、LDX ライブラリを利用する場合の方法を示す。

OpenDX は、ネットワークエディターを持つグラフィックソフトウエアである。LDX は、トリガーを持つことができ、我々の使用ではトリガーを 1 とすればプログラムが実行され、0 とすればプログラムは動作せずに OpenDX に操作を引き渡すことになっている。もしトリガーが 1 になっていなければ、LDX を編集して 1 にする。この他の入力は、現在使われていない。

出力には、data, mesh, current time がある。data はフィールドデータである。フィールドデータとしてどのようなものを出力するかは、userqunatity.h でユーザーが指定するマクロにより決定される。mesh は connection データが出力され、メッシュがどのように生成されているかを観察するのに便利である。サンプルネットワークを図 10 に示す。

7 参加企業及び機関

- フィックス株式会社 (札幌市東区北 23 条東 18 丁目 2-3 コルメナ石山)

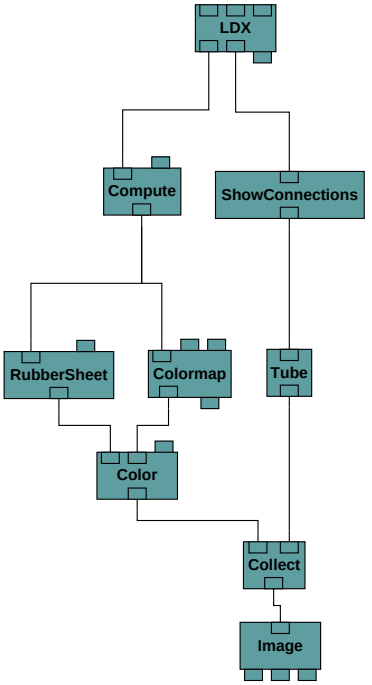

図 10: LDX を用いた描画方法

- 「OpenDX による基本的な描画部分の開発 (実装水準 I)」
- 「OpenDX による基本的な描画部分の開発 (実装水準 II)」

参考文献

[Gingold and Monaghan] R. A. Gingold and J. J. Monaghan Smoothed particle hydrodynamics – Theory and application to non spherical stars *Mon. Not. R. astron. Soc.*, 181:375–389, 1977.

[De Zeeuw and Powell] Darren De Zeeuw and Kenneth G. Powell. An adaptively refined cartesian mesh solver for the euler equations. *J. Comput. Phys.*, 104:56–68, 1993.

[Inutsuka] S. Inutsuka Godunov-type SPH *Memorie della Societa Astronomia Italiana*, 65:1027, 1994.

[Yamada and Miyashita] 山田良透、宮下尚 Adaptive Mesh Refinement 法とその応用–究極の高解像度計算を目指して– 物性研究, 77:73–112(2001).

[Martin] Dan Martin and Keith Cartwright Solving Poisson's Equation using Adaptive Mesh Refinement *http://www.barkley.berkeley.edu/~martin/AMR.ps*

[Khokhlov] A. M. Khokhlov. Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations. *J. Comput. Phys.*, 143:519–543, 1998.

[Yahagi and Yoshii] H. Yahagi and Y. Yoshii N-Body Code with Adaptive Mesh Refinement *Astrohpys. J.*, 558:463–475, 2001.

なお、本開発に関しては、現在ホームページを `http://www.amesh.org/` にて作成中である。