GPU実装によるインタラクティブ波動伝搬シミュレーションに関する研究

2015年3月

河田　直樹

首都大学東京

目 次

図 目 次

表 目 次

第1章　序論

1.1　本研究の背景

近年の計算機環境の向上と共に，時間領域の波動数値シミュレーションは重要な技術となってきている．図 1.1 に日本のスーパーコンピュータの性能の推移を示す [1]．従来，数値シミュレーションの高速化には，スーパーコンピュータやクラスタなどの大型の計算機が利用されてきた．しかし，最近では GPU（Graphics Processing Unit）を並列処理アーキテクチャとして用いて汎用的な数値計算を行う GPGPU（General Purpose computation on GPUs）という技術が様々な分野で用いられてきている [2–5]．

GPU の性能は 2005 年前後から急速に向上し，今では CPU の数倍以上の性能を持っている．しかし，GPU が高い性能を持っていても GPGPU は大きく普及しなかった．当時は GPU 用プログラムを記述するためにシェーダ言語を用いなければならないため，GPU の構造について熟知している必要があったからである．そこで，高い性能を持った GPU を簡単に数値計算に適用するための開発環境として CUDA [6] が 2006 年にリリースされた．CUDA がリリースされたことにより GPU プログラミングが非常に容易となった [7,8]．CUDA は現在改良を重ねられており，倍精度浮動小数点数演算が可能になるなど機能が向上している．

現在の GPU の演算性能は数年前と比較して飛躍的に向上しており，最新の GPU ボードは 1 機あたりに 3000 個程度のコアを搭載し，計算アクセラレータとして CPU クラスタや小型スーパーコンピュータに匹敵する演算性能を秘めていると考えられる．すなわち，計算高速化を実現する手段において，GPU はコストパフォーマンスの高いアクセラレータとして位置づけられると言える．波動伝搬シミュレーションにおいても，生体内音波伝搬解析や建築音響・騒音解析，アンテナ解析，屋外電波伝搬解析，弾性体の音場特性解析などへの応用・実用化が期待されている．

時間領域の波動数値解析手法としては，FDTD（Finite-Difference Time-Domain）法が広く利用され，大型計算機によるベクトル化や並列化の研究も行われている．FDTD 法は Yee [9] によって電磁界解析に応用されて以来，コンピュータの発達と共に発展してきた [10,11]．今では電磁界解析以外に音響，弾性波等の様々な分野で FDTD 法は用いられ

ている [12–19]．また，特性曲線法の一つである CIP（Constrained Interpolation Profile）法を用いた数値解析の研究が進められ，成果が報告されている [20–29]．特性曲線は移流方程式などを解く際に用いられることが多い．特性曲線上では値が一定になることを利用する．特性曲線に CIP 法を組み合わせることで精度の良い移流が可能となる．CIP 法は場の空間微分値の値も同時に計算することで，同様の離散化条件において，従来法に比べて数値分散性が非常に小さい手法であることが明らかになっている．しかし，CIP 法は FDTD 法と比較すると数値散逸誤差が大きいことも明らかになっており，大規模領域を解析する際にはこの数値散逸誤差は非常に重要な問題となる．数値分散とは高周波数帯域で位相速度が遅れ波形が歪むことであり，数値散逸とは高周波数帯域で振幅の減少が起こりエネルギーが減少することである．

一方，計算機で数値解析と同時に解析結果の可視化を行う場合，特に，パーソナルコンピュータに代表される１ノード計算機において解析する場合には，３次元空間をモデル化するためのメモリ容量の問題や，メモリが確保できたとしてもその計算を行うための計算時間を考慮すると，現実問題として３次元解析は困難な場合が多かった．さらに，シミュレーションの実行中に，同時に解析結果を可視化する，いわゆるリアルタイム可視化を行う場合は，十分な描画スピードを維持するためには，３次元シミュレーションの実現は１ノード計算機ではほぼ不可能であった．

GPU はこの課題の解決に大きく貢献するアーキテクチャと言える．すなわち，GPU を用いた実装は単に計算速度が向上するだけでなく，ビデオメモリ（VRAM）上に計算領域を確保することで，一般的な可視化のデータ転送とは異なり，PCIe（Peripheral Component Interconnect Express）バスの転送を回避してビデオカード上の描画用の VRAM 領域へ情報を直接書き込むことができる．これは高速可視化を目指す上で重要な点である．これにより，GPU 実装によるリアルタイム可視化シミュレーション [30,31]，さらにこれを発展させたインタラクティブシミュレーションが可能となる．ここでインタラクティブシミュレーションとは，数値解析を行いながらリアルタイムに可視化を行い，その結果を確認しながらインタラクティブに解析パラメータ等を操作するシミュレーションのフレームワークを指している．

数値解析した結果を可視化する場合，従来であれば解析終了後に可視化を行うのが一般的だった．しかし，解析結果が期待通りの結果ではなかった時にはシミュレーションを最初からやり直さなければならなかった．これはコストの面から考えて重要な問題である．解析を行いながら結果をリアルタイムに可視化してパラメータ等の修正を行えるインタラクティブシミュレーションが望まれるのは当然の成り行きだろう．

インタラクティブシミュレーションを実現するために克服する課題として，波動伝搬

数値解析の高速化と波動伝搬の高速可視化が挙げられる．インタラクティブシミュレーションを 30fps/15fps(Frames Per Second) で行うためには高速化が不可欠だからである．シミュレーションのインタラクティブ性を効果的に発揮するためには 30fps/15fps が必要だと考える．30fps/15fps という基準はテレビ放送のフレームレートを参考とした．インタラクティブシミュレーションは大きく分けて波動伝搬解析を行うソルバと波動伝搬結果を可視化する描画のルーチンから成り立つ．数値解析と描画を 1 ステップとして考えた時，30fps/15fps を達成するには 1 ステップを 33ms/66ms 以下で実行する必要がある．そこで，インタラクティブシミュレーションの実現のため，波動伝搬数値解析の高速化と波動伝搬の高速可視化を目的とする．

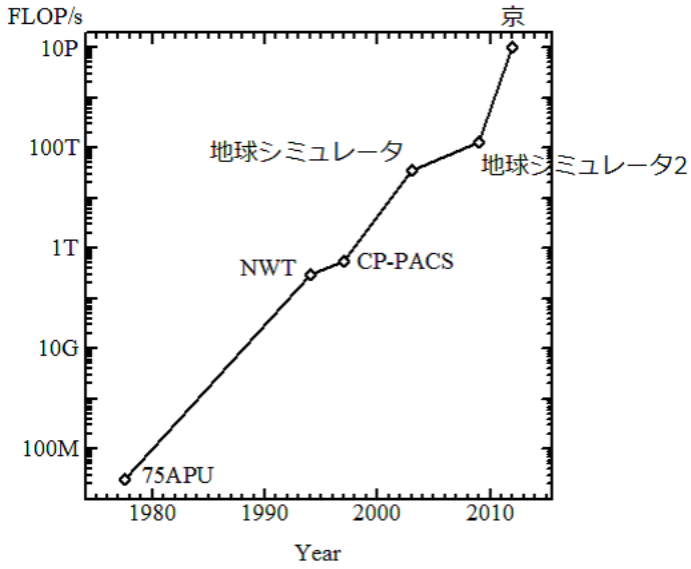

図 1.1 日本のスーパーコンピュータの性能の推移 (地球シミュレータ開発史より引用，加筆)

1.2 本研究の目的

本研究の目的はまず初めに波動伝搬数値解析手法の精度評価を行うことである．次に，GPU を用いた FDTD 法，MM-MOC 法の高速化を行う．そして，インタラクティブシミュレーションに適した可視化手法の提案と評価を行う．最後に，GPU 高速計算と高速可視化を組み合わせたインタラクティブシミュレーションの評価を行う．

目的を以下にまとめる．

(1) FDTD 法，MM-MOC 法の精度比較評価

- 数値散逸誤差，数値分散誤差における評価

(2) GPU を用いた FDTD 法，MM-MOC 法の高速化

- 数値解析手法の GPU への適用
- 数値解析手法に応じた GPU 実装の最適化
- マルチ GPU を用いたさらなる高速化

(3) 三次元数値解析における可視化手法の検討

- 既存の可視化手法の評価
- 新たな可視化手法として PMCC (Permeable Multi Cross-section Contours) の提案
- PMCC の評価
- マルチ GPU を用いたさらなる高速可視化の検討

である．

これらを通して最終的には，GPU により高速化された波動伝搬数値解析と PMCC を組み合わせたインタラクティブシミュレーションの実装および評価を行うことを目的とする．本論文は，波動伝搬数値解析に関するインタラクティブシミュレーションの可能性を論ずるものである．

1.3　本論文の構成

本論文は全 6 章よりなる．図 1.2 に本論文の構成を示す．2 章で波動伝搬数値解析手法について述べ，3 章でその手法の GPU による高速化を行う．4 章でインタラクティブシミュレーションのための可視化手法として PMCC を提案する．5 章で GPU 高速計算と高速可視化を組み合わせたインタラクティブシミュレーションの評価を行う．

第 1 章：序論

本章は序論である．本研究の目的と意義，位置付けを示す．

第 2 章：時間領域における波動伝搬数値シミュレーションのための数値解析手法

本章では波動伝搬解析における数値シミュレーション手法について述べる．初めに，音場・電磁場・弾性波それぞれの支配方程式に基づく，各手法の定式化を説明する．併せて，

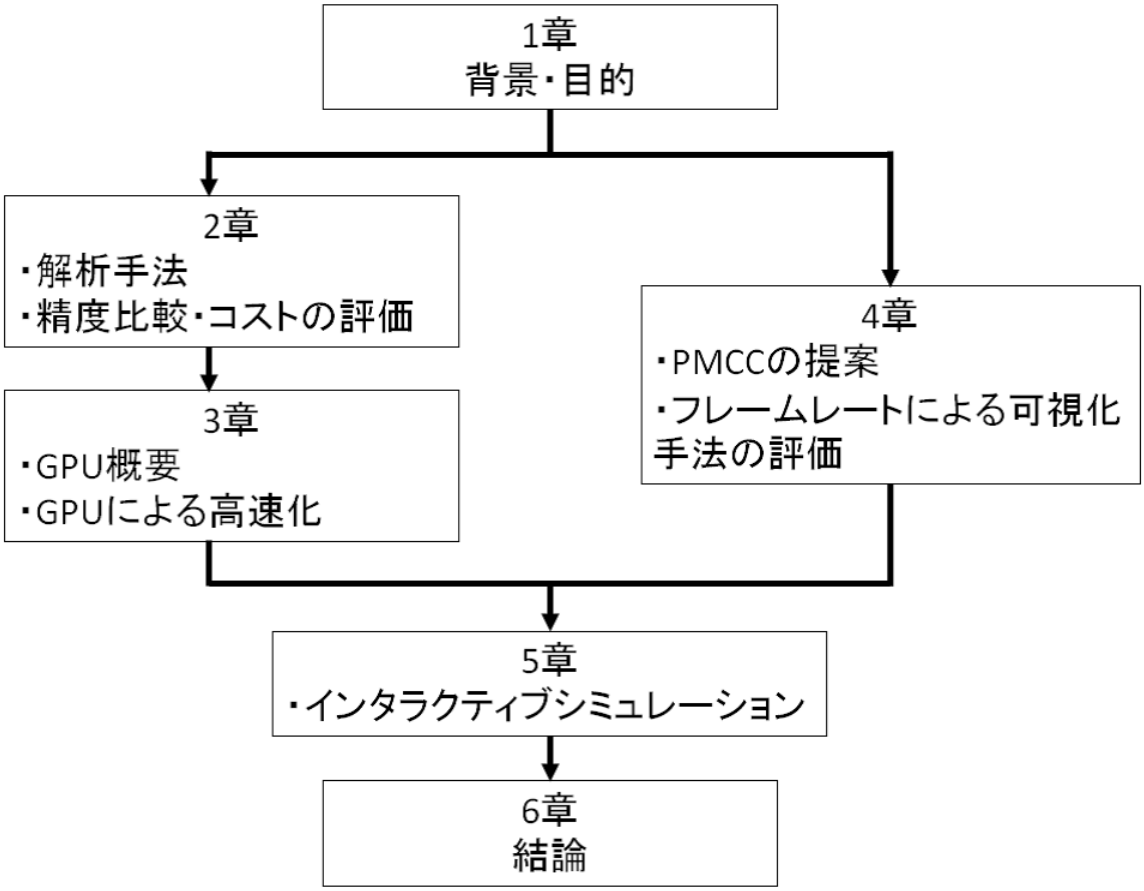

図 1.2 本論文の構成

MM-MOC 法の考え方及び定式化について述べる．さらに，FDTD 法と MM-MOC 法の精度比較評価を行い，第 3 章以降の GPU 実装に向けた準備をする．

第 3 章：GPU (Graphics Processing Unit) の進化と GPU を用いた数値解析の高速化

本章では数値シミュレーションの GPU 実装について述べる．初めに，GPU の概要について説明する．次に，FDTD 法・MM-MOC 法のそれぞれの手法に関して，解析手法に応じた GPU 実装のためのアルゴリズムの検討を行い，それによる計算時間の短縮について考察する．最後に，マルチ GPU を用いた更なる高速化の検討を行い，結果について考察する．

第 4 章：3 次元波動数値解析のための可視化手法の検討と評価

本章では可視化手法について述べる．初めに，従来用いられてきた可視化手法について説明する．続いて，それらの問題点を踏まえた上で，新しい可視化手法として PMCC（Permeable Multi Cross-section Contours）を提案し，この方法を用いた可視化の描画速度の比較評価を行う．さらに，マルチ GPU を用いたより高速な可視化についても検討し，考察を行う．

第 5 章：インタラクティブシミュレーションの概要とその応用について

本章では GPU を用いた高速並列計算とリアルタイム可視化を組み合わせたインタラクティブシミュレーションについて述べる．初めに，その概要について説明する．続いて 3 章及び 4 章において検討を行った解析手法と可視化手法を基盤とし，音場・電磁界・弾性波を対象としたインタラクティブシミュレーションの応用について検討を行う．

第 6 章：結論

本章は結論である．本研究で得られた成果をまとめると共に，残された課題と今後の展望について述べる．

第2章　時間領域における波動伝搬数値シミュレーションのための数値解析手法

2.1　序言

本章では，時間領域における波動伝搬数値シミュレーションのための数値解析手法について述べる．現在広く利用されている Finite-Difference Time-Domain (FDTD) 法の定式化について，および MM-MOC (Multi-Moment-Method of Characteristics) 法の定式化について説明する．次に，FDTD 法と MM-MOC 法の精度を比較し，評価を行う．精度比較評価の結果を踏まえて，インタラクティブシミュレーションで用いる数値解析手法を決定する．

波動伝搬数値解析において，音場・電磁界・弾性波はすべて波動方程式を満たし，伝搬速度と伝搬媒質は異なるが，統一的に解ける．つまり，これらの問題は類似性があると言える．一つの問題を解けば他の問題に簡単に応用できるので，今回は音場・電磁界・弾性波問題を対象とした．

2.2　FDTD 法を用いた時間領域波動伝搬数値解析

2.2.1　FDTD 法による音場波動伝搬数値解析

損失を無視した場合，音場の 2 つの支配方程式は，以下の式で表される．

$$\rho \frac{\partial \vec{v}}{\partial t} = -\nabla p \tag{2.1}$$

$$\nabla \cdot \vec{v} = -\frac{1}{K} \frac{\partial p}{\partial t} \tag{2.2}$$

ここで，p は音圧，$\vec{v}$ は粒子速度，ρ は媒質密度，K は体積弾性率である．

3 次元領域での定式化を行う．よって，$\vec{v} = (v_x, v_y, v_z)$ となる．これで，式 (2.1) 及び (2.2) から以下の式が得られる．

$$\frac{\partial v_x}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial x} = 0 \tag{2.3}$$

$$\frac{\partial v_y}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial y} = 0 \tag{2.4}$$

$$\frac{\partial v_z}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial z} = 0 \tag{2.5}$$

$$\frac{\partial p}{\partial t} + K\frac{\partial v_x}{\partial x} + K\frac{\partial v_y}{\partial y} + K\frac{\partial v_z}{\partial z} = 0 \tag{2.6}$$

また，式 (2.1) 及び (2.2) から粒子速度 $\vec{v}$ を消去すると波動方程式である式 (2.7) が得られる．

$$\frac{1}{c_0^2}\frac{\partial^2 p}{\partial t^2} = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} \tag{2.7}$$

ここで，c_0 は音速を表している．

2 次精度 FDTD 法による定式化

FDTD 法の特徴は，Staggered grid を用いている点である．Staggered grid とは，音圧 p と粒子速度 v を空間的に 1/2 グリッド，時間的に 1/2 タイムステップずらして配置しているグリッドモデルである．図 2.1 に音場 FDTD 法のグリッドモデルを示す．

式 (2.3)，(2.4)，(2.5) 及び (2.6) に，Staggered grid を用いて空間と時間で 2 次精度の中心差分近似を適用すると以下の式が得られる．

$$v_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k) = v_x^{n-\frac{1}{2}}(i+\frac{1}{2},j,k) - \frac{1}{\rho}\frac{\Delta t}{\Delta x}\big(p^n(i+1,j,k) - p^n(i,j,k)\big) \tag{2.8}$$

$$v_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k) = v_y^{n-\frac{1}{2}}(i,j+\frac{1}{2},k) - \frac{1}{\rho}\frac{\Delta t}{\Delta y}\big(p^n(i,j+1,k) - p^n(i,j,k)\big) \tag{2.9}$$

$$v_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2}) = v_z^{n-\frac{1}{2}}(i,j,k+\frac{1}{2}) - \frac{1}{\rho}\frac{\Delta t}{\Delta z}\big(p^n(i,j,k+1) - p^n(i,j,k)\big) \tag{2.10}$$

$$\begin{aligned} p^{n+1}(i,j,k) = p^n(i,j,k) &- K\frac{\Delta t}{\Delta x}\big(v_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k) - v_x^{n+\frac{1}{2}}(i-\frac{1}{2},j,k)\big) \\ &- K\frac{\Delta t}{\Delta y}\big(v_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k) - v_y^{n+\frac{1}{2}}(i,j-\frac{1}{2},k)\big) \\ &- K\frac{\Delta t}{\Delta z}\big(v_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2}) - v_z^{n+\frac{1}{2}}(i,j,k-\frac{1}{2})\big) \end{aligned} \tag{2.11}$$

ここで，Δx は x 方向の空間離散間隔，Δy は y 方向の空間離散間隔，Δz は z 方向の空間離散間隔，Δt は時間離散間隔である．また，$p^n(i,j,k)$ は格子点 $x = i\Delta x$，$y = j\Delta y$，$z = k\Delta z$ 上の時間が $t = n\Delta t$ のときの音圧を表している．

また，式 (2.7) に空間と時間で 2 次精度の中心差分近似を適用すると以下の式が得られる．

$$
\begin{aligned}
p^{n+1}(i,j,k) = 2p^n(i,j,k) - p^{n-1}(i,j,k) \\
+ (\frac{c_0\Delta t}{\Delta x})^2\big(p^n(i+1,j,k) - 2p^n(i,j,k) + p^n(i-1,j,k)\big) \\
+ (\frac{c_0\Delta t}{\Delta y})^2\big(p^n(i,j+1,k) - 2p^n(i,j,k) + p^n(i,j-1,k)\big) \\
+ (\frac{c_0\Delta t}{\Delta z})^2\big(p^n(i,j,k+1) - 2p^n(i,j,k) + p^n(i,j,k-1)\big)
\end{aligned}
\tag{2.12}
$$

式 (2.12) を Wave Equation FDTD(WE-FDTD) 法という [32]．WE-FDTD 法は FDTD 法と比較して演算量が少なく，使用するメモリ量も FDTD 法の半分となる．WE-FDTD 法がインタラクティブシミュレーションに適しているように思われるが，本論文では FDTD 法を使用する．理由として，粒子速度を可視化したい場合はその都度粒子速度を音圧から計算しなければならない．これはインタラクティブシミュレーションを行うにあたって大きなデメリットである．また，音響インテンシティなどを計算・可視化する場合にも粒子速度は不可欠である．これらの理由により，本論文では WE-FDTD 法ではなく FDTD 法を使用する．

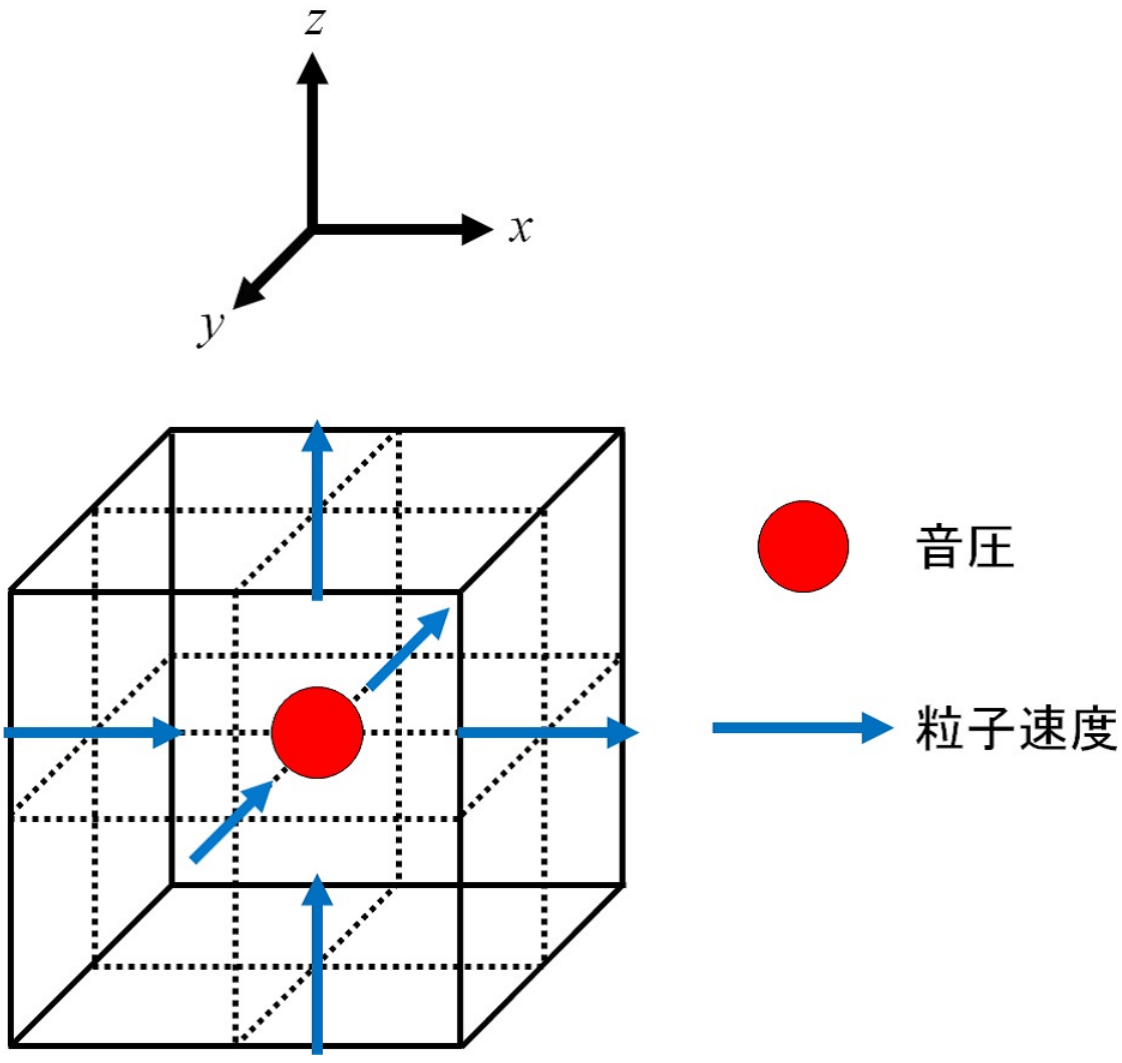

図 2.1　音場 FDTD 法のグリッドモデル

2.2.2 FDTD 法による電磁界波動伝搬数値解析

電磁界の 2 つの支配方程式は，以下の式で表される．

$$\frac{\partial \vec{E}}{\partial t} = -\frac{\sigma}{\varepsilon}\vec{E} + \frac{1}{\varepsilon}\nabla \times \vec{H} \tag{2.13}$$

$$\frac{\partial \vec{H}}{\partial t} = -\frac{1}{\mu}\nabla \times \vec{E} \tag{2.14}$$

ここで，$\vec{E}$ は電界，$\vec{H}$ は磁界，ε は誘電率，μ は透磁率，σ は導電率である．

3 次元領域での定式化を行う．よって，$\vec{E} = (E_x, E_y, E_z), \vec{H} = (H_x, H_y, H_z)$ となる．これで，式 (2.13) 及び (2.14) から以下の式が得られる．

$$\frac{\partial E_x}{\partial t} = -\frac{\sigma}{\varepsilon}E_x + \frac{1}{\varepsilon}(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z}) \tag{2.15}$$

$$\frac{\partial E_y}{\partial t} = -\frac{\sigma}{\varepsilon}E_y + \frac{1}{\varepsilon}(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x}) \tag{2.16}$$

$$\frac{\partial E_z}{\partial t} = -\frac{\sigma}{\varepsilon}E_z + \frac{1}{\varepsilon}(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}) \tag{2.17}$$

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu}(\frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z}) \tag{2.18}$$

$$\frac{\partial H_y}{\partial t} = -\frac{1}{\mu}(\frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x}) \tag{2.19}$$

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}) \tag{2.20}$$

2 次精度 FDTD 法による定式化

図 2.2 に電磁界 FDTD 法のグリッドモデルを示す．

式 (2.15)，(2.16)，(2.17)，(2.18)，(2.19) 及び (2.20) に，Staggered grid を用いて空間と時間で 2 次精度の中心差分近似を適用すると以下の式が得られる．

$$\begin{aligned} E_x^{n+\frac{1}{2}}(i, j+\frac{1}{2}, k+\frac{1}{2}) = {} & \frac{1-\frac{\sigma\Delta t}{2\varepsilon}}{1+\frac{\sigma\Delta t}{2\varepsilon}} E_x^{n-\frac{1}{2}}(i, j+\frac{1}{2}, k+\frac{1}{2}) \\ & + \frac{1}{1+\frac{\sigma\Delta t}{2\varepsilon}}\frac{1}{\varepsilon}\frac{\Delta t}{\Delta y}\big(H_z^n(i, j+1, k+\frac{1}{2}) - H_z^n(i, j, k+\frac{1}{2})\big) \\ & + \frac{1}{1+\frac{\sigma\Delta t}{2\varepsilon}}\frac{1}{\varepsilon}\frac{\Delta t}{\Delta z}\big(H_y^n(i, j+\frac{1}{2}, k+1) - H_y^n(i, j+\frac{1}{2}, k)\big) \end{aligned} \tag{2.21}$$

$$
\begin{aligned}
E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) &= \frac{1-\frac{\sigma\Delta t}{2\varepsilon}}{1+\frac{\sigma\Delta t}{2\varepsilon}} E_y^{n-\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) \\
&\quad + \frac{1}{1+\frac{\sigma\Delta t}{2\varepsilon}} \frac{1}{\varepsilon}\frac{\Delta t}{\Delta z}\big(H_x^n(i+\frac{1}{2},j,k+1) - H_x^n(i+\frac{1}{2},j,k)\big) \\
&\quad + \frac{1}{1+\frac{\sigma\Delta t}{2\varepsilon}} \frac{1}{\varepsilon}\frac{\Delta t}{\Delta x}\big(H_z^n(i+1,j,k+\frac{1}{2}) - H_z^n(i,j,k+\frac{1}{2})\big)
\end{aligned} \tag{2.22}
$$

$$
\begin{aligned}
E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) &= \frac{1-\frac{\sigma\Delta t}{2\varepsilon}}{1+\frac{\sigma\Delta t}{2\varepsilon}} E_z^{n-\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) \\
&\quad + \frac{1}{1+\frac{\sigma\Delta t}{2\varepsilon}} \frac{1}{\varepsilon}\frac{\Delta t}{\Delta x}\big(H_y^n(i+1,j+\frac{1}{2},k) - H_y^n(i,j+\frac{1}{2},k)\big) \\
&\quad + \frac{1}{1+\frac{\sigma\Delta t}{2\varepsilon}} \frac{1}{\varepsilon}\frac{\Delta t}{\Delta y}\big(H_x^n(i+\frac{1}{2},j+1,k) - H_x^n(i+\frac{1}{2},j,k)\big)
\end{aligned} \tag{2.23}
$$

$$
\begin{aligned}
H_x^{n+1}(i+\frac{1}{2},j,k) &= H_x^n(i+\frac{1}{2},j,k) \\
&\quad + \frac{1}{\mu}\frac{\Delta t}{\Delta y}\big(E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) - E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j-\frac{1}{2},k)\big) \\
&\quad + \frac{1}{\mu}\frac{\Delta t}{\Delta z}\big(E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) - E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k-\frac{1}{2})\big)
\end{aligned} \tag{2.24}
$$

$$
\begin{aligned}
H_y^{n+1}(i,j+\frac{1}{2},k) &= H_y^n(i,j+\frac{1}{2},k) \\
&\quad + \frac{1}{\mu}\frac{\Delta t}{\Delta z}\big(E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k+\frac{1}{2}) - E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k-\frac{1}{2})\big) \\
&\quad + \frac{1}{\mu}\frac{\Delta t}{\Delta x}\big(E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) - E_z^{n+\frac{1}{2}}(i-\frac{1}{2},j+\frac{1}{2},k)\big)
\end{aligned} \tag{2.25}
$$

$$
\begin{aligned}
H_z^{n+1}(i,j,k+\frac{1}{2}) &= H_z^n(i,j,k+\frac{1}{2}) \\
&\quad + \frac{1}{\mu}\frac{\Delta t}{\Delta x}\big(E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) - E_y^{n+\frac{1}{2}}(i-\frac{1}{2},j,k+\frac{1}{2})\big) \\
&\quad + \frac{1}{\mu}\frac{\Delta t}{\Delta y}\big(E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k+\frac{1}{2}) - E_x^{n+\frac{1}{2}}(i,j-\frac{1}{2},k+\frac{1}{2})\big)
\end{aligned} \tag{2.26}
$$

また，$\sigma = 0$ とした場合は以下の式となる．

$$
\begin{aligned}
E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k+\frac{1}{2}) &= E_x^{n-\frac{1}{2}}(i,j+\frac{1}{2},k+\frac{1}{2}) \\
&\quad + \frac{1}{\varepsilon}\frac{\Delta t}{\Delta y}\big(H_z^n(i,j+1,k+\frac{1}{2}) - H_z^n(i,j,k+\frac{1}{2})\big) \\
&\quad + \frac{1}{\varepsilon}\frac{\Delta t}{\Delta z}\big(H_y^n(i,j+\frac{1}{2},k+1) - H_y^n(i,j+\frac{1}{2},k)\big)
\end{aligned} \tag{2.27}
$$

$$
\begin{aligned}
E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) &= E_y^{n-\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) \\
&+ \frac{1}{\varepsilon}\frac{\Delta t}{\Delta z}\big(H_x^n(i+\frac{1}{2},j,k+1) - H_x^n(i+\frac{1}{2},j,k)\big) \\
&+ \frac{1}{\varepsilon}\frac{\Delta t}{\Delta x}\big(H_z^n(i+1,j,k+\frac{1}{2}) - H_z^n(i,j,k+\frac{1}{2})\big)
\end{aligned} \tag{2.28}
$$

$$
\begin{aligned}
E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) &= E_z^{n-\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) \\
&+ \frac{1}{\varepsilon}\frac{\Delta t}{\Delta x}\big(H_y^n(i+1,j+\frac{1}{2},k) - H_y^n(i,j+\frac{1}{2},k)\big) \\
&+ \frac{1}{\varepsilon}\frac{\Delta t}{\Delta y}\big(H_x^n(i+\frac{1}{2},j+1,k) - H_x^n(i+\frac{1}{2},j,k)\big)
\end{aligned} \tag{2.29}
$$

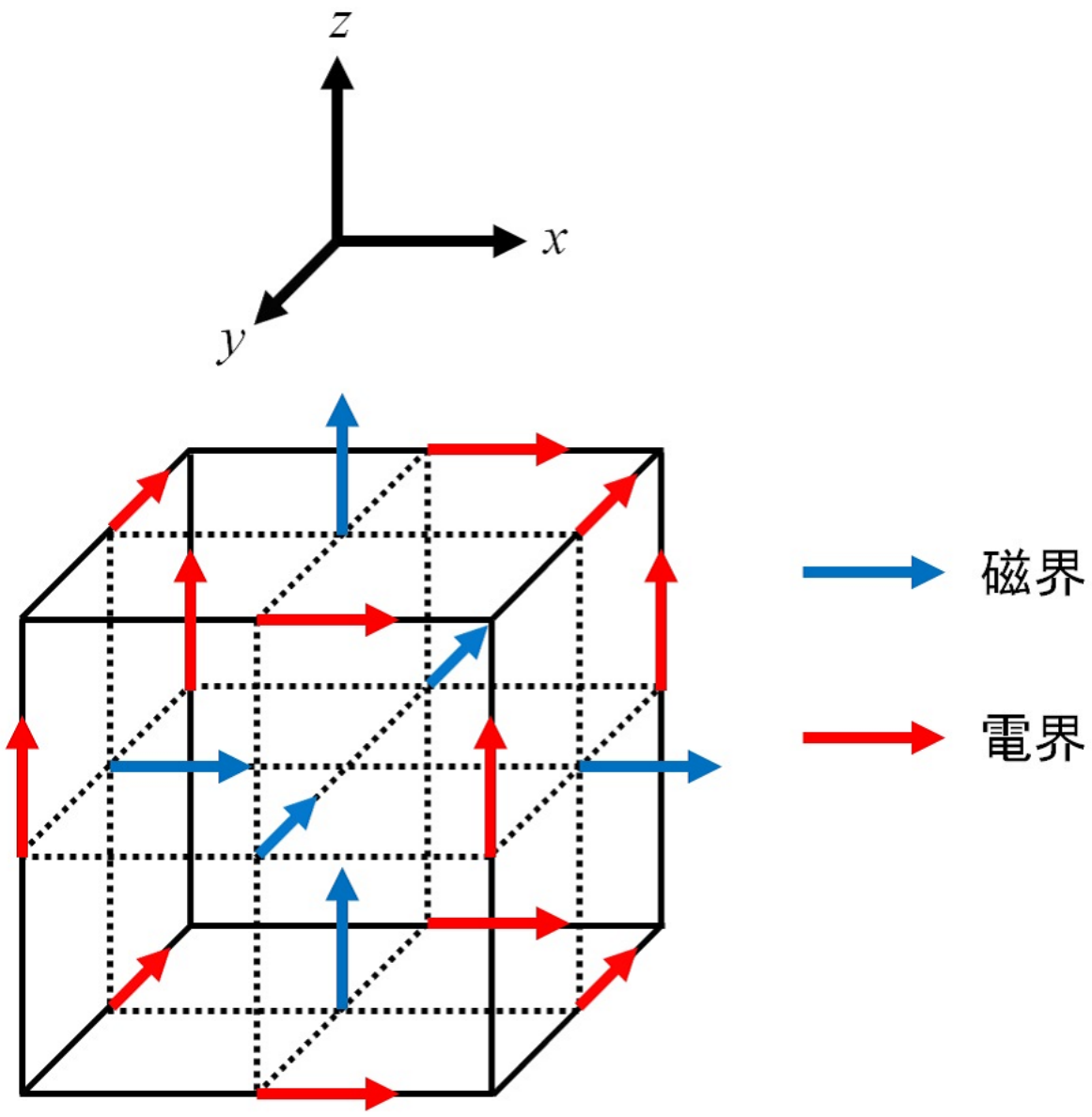

図 2.2　電磁界 FDTD 法のグリッドモデル

2.2.3 FDTD 法による弾性波波動伝搬数値解析

弾性体中の運動方程式は以下の式で表される.

$$\rho\frac{\partial\vec{u}}{\partial t}=\frac{\partial\sigma_{ij}}{\partial x_i} \tag{2.30}$$

ここで，σ_{ij} は応力テンソル，$\vec{u}$ は粒子速度ベクトル，x は位置ベクトルを表す.
式 (2.30) より以下の式が得られる.

$$\frac{\partial u_x}{\partial t}+\frac{1}{\rho}(\frac{\partial T_x}{\partial x}+\frac{\partial T_{s_{xy}}}{\partial y}+\frac{\partial T_{s_{zx}}}{\partial z})=0 \tag{2.31}$$

$$\frac{\partial u_y}{\partial t}+\frac{1}{\rho}(\frac{\partial T_y}{\partial y}+\frac{\partial T_{s_{xy}}}{\partial x}+\frac{\partial T_{s_{yz}}}{\partial z})=0 \tag{2.32}$$

$$\frac{\partial u_z}{\partial t}+\frac{1}{\rho}(\frac{\partial T_z}{\partial z}+\frac{\partial T_{s_{yz}}}{\partial y}+\frac{\partial T_{s_{zx}}}{\partial x})=0 \tag{2.33}$$

$$\frac{\partial T_x}{\partial t}+\rho\big\{C_l^2\frac{\partial u_x}{\partial x}+(C_l^2-2C_s^2)(\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z})\big\}=0 \tag{2.34}$$

$$\frac{\partial T_y}{\partial t}+\rho\big\{C_l^2\frac{\partial u_y}{\partial y}+(C_l^2-2C_s^2)(\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z})\big\}=0 \tag{2.35}$$

$$\frac{\partial T_z}{\partial t}+\rho\big\{C_l^2\frac{\partial u_z}{\partial z}+(C_l^2-2C_s^2)(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y})\big\}=0 \tag{2.36}$$

$$\frac{\partial T_{s_{xy}}}{\partial t}+\rho C_s^2(\frac{\partial u_x}{\partial y}+\frac{\partial u_y}{\partial x})=0 \tag{2.37}$$

$$\frac{\partial T_{s_{yz}}}{\partial t}+\rho C_s^2(\frac{\partial u_y}{\partial z}+\frac{\partial u_z}{\partial y})=0 \tag{2.38}$$

$$\frac{\partial T_{s_{zx}}}{\partial t}+\rho C_s^2(\frac{\partial u_z}{\partial x}+\frac{\partial u_x}{\partial z})=0 \tag{2.39}$$

ここで，T は応力，T_s は剪断応力，u は粒子速度，ρ は媒質密度，C_l は縦波音速，C_s は横波音速である.

2 次精度 FDTD 法による定式化

図 2.3 に弾性波 FDTD 法のグリッドモデルを示す.

式 (2.31)，(2.32)，(2.33)，(2.34)，(2.35)，(2.36)，(2.37)，(2.38) 及び (2.39) に，Staggered grid を用いて空間と時間で 2 次精度の中心差分近似を適用すると以下の式が得られる.

$$
\begin{aligned}
u_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k) = u_x^{n-\frac{1}{2}}(i+\frac{1}{2},j,k) \\
&- \frac{1}{\rho}\frac{\Delta t}{\Delta x}\big(T_x^n(i+1,j,k) - T_x^n(i,j,k)\big) \\
&- \frac{1}{\rho}\frac{\Delta t}{\Delta y}\big(T_{s_{xy}}^n(i+\frac{1}{2},j+\frac{1}{2},k) - T_{s_{xy}}^n(i+\frac{1}{2},j-\frac{1}{2},k)\big) \\
&- \frac{1}{\rho}\frac{\Delta t}{\Delta z}\big(T_{s_{zx}}^n(i+\frac{1}{2},j,k+\frac{1}{2}) - T_{s_{zx}}^n(i+\frac{1}{2},j,k-\frac{1}{2})\big)
\end{aligned}
\tag{2.40}
$$

$$
\begin{aligned}
u_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k) = u_y^{n-\frac{1}{2}}(i,j+\frac{1}{2},k) \\
&- \frac{1}{\rho}\frac{\Delta t}{\Delta y}\big(T_y^n(i,j+1,k) - T_y^n(i,j,k)\big) \\
&- \frac{1}{\rho}\frac{\Delta t}{\Delta x}\big(T_{s_{xy}}^n(i+\frac{1}{2},j+\frac{1}{2},k) - T_{s_{xy}}^n(i-\frac{1}{2},j+\frac{1}{2},k)\big) \\
&- \frac{1}{\rho}\frac{\Delta t}{\Delta z}\big(T_{s_{yz}}^n(i,j+\frac{1}{2},k+\frac{1}{2}) - T_{s_{yz}}^n(i,j+\frac{1}{2},k-\frac{1}{2})\big)
\end{aligned}
\tag{2.41}
$$

$$
\begin{aligned}
u_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2}) = u_z^{n-\frac{1}{2}}(i,j,k+\frac{1}{2}) \\
&- \frac{1}{\rho}\frac{\Delta t}{\Delta z}\big(T_z^n(i,j,k+1) - T_z^n(i,j,k)\big) \\
&- \frac{1}{\rho}\frac{\Delta t}{\Delta y}\big(T_{s_{yz}}^n(i,j+\frac{1}{2},k+\frac{1}{2}) - T_{s_{yz}}^n(i,j-\frac{1}{2},k+\frac{1}{2})\big) \\
&- \frac{1}{\rho}\frac{\Delta t}{\Delta x}\big(T_{s_{zx}}^n(i+\frac{1}{2},j,k+\frac{1}{2}) - T_{s_{zx}}^n(i-\frac{1}{2},j,k+\frac{1}{2})\big)
\end{aligned}
\tag{2.42}
$$

$$
\begin{aligned}
T_x^{n+1}(i,j,k) = T_x^n(i,j,k) \\
&- \rho C_l^2\frac{\Delta t}{\Delta x}\big(u_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k) - u_x^{n+\frac{1}{2}}(i-\frac{1}{2},j,k)\big) \\
&- \rho(C_l^2-2C_s^2)\frac{\Delta t}{\Delta y}\big(u_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k) - u_y^{n+\frac{1}{2}}(i,j-\frac{1}{2},k)\big) \\
&- \rho(C_l^2-2C_s^2)\frac{\Delta t}{\Delta z}\big(u_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2}) - u_z^{n+\frac{1}{2}}(i,j,k-\frac{1}{2})\big)
\end{aligned}
\tag{2.43}
$$

$$
\begin{aligned}
T_y^{n+1}(i,j,k) = T_y^n(i,j,k) \\
&- \rho C_l^2\frac{\Delta t}{\Delta y}\big(u_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k) - u_y^{n+\frac{1}{2}}(i,j-\frac{1}{2},k)\big) \\
&- \rho(C_l^2-2C_s^2)\frac{\Delta t}{\Delta x}\big(u_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k) - u_x^{n+\frac{1}{2}}(i-\frac{1}{2},j,k)\big) \\
&- \rho(C_l^2-2C_s^2)\frac{\Delta t}{\Delta z}\big(u_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2}) - u_z^{n+\frac{1}{2}}(i,j,k-\frac{1}{2})\big)
\end{aligned}
\tag{2.44}
$$

$$
\begin{aligned}
T_z^{n+1}(i,j,k) = T_z^n(i,j,k) \\
&- \rho C_l^2 \frac{\Delta t}{\Delta z}\big(u_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2}) - u_z^{n+\frac{1}{2}}(i,j,k-\frac{1}{2})\big) \\
&- \rho (C_l^2 - 2C_s^2)\frac{\Delta t}{\Delta x}\big(u_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k) - u_x^{n+\frac{1}{2}}(i-\frac{1}{2},j,k)\big) \\
&- \rho (C_l^2 - 2C_s^2)\frac{\Delta t}{\Delta y}\big(u_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k) - u_y^{n+\frac{1}{2}}(i,j-\frac{1}{2},k)\big)
\end{aligned} \tag{2.45}
$$

$$
\begin{aligned}
T_{s_{xy}}^{n+1}(i+\frac{1}{2},j+\frac{1}{2},k) = T_{s_{xy}}^n\big(i+\frac{1}{2},j+\frac{1}{2},k\big) \\
&- \rho C_s^2 \frac{\Delta t}{\Delta y}\big(u_x^{n+\frac{1}{2}}(i+\frac{1}{2},j+1,k) - u_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k)\big) \\
&- \rho C_s^2 \frac{\Delta t}{\Delta x}\big(u_y^{n+\frac{1}{2}}(i+1,j+\frac{1}{2},k) - u_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k)\big)
\end{aligned} \tag{2.46}
$$

$$
\begin{aligned}
T_{s_{yz}}^{n+1}(i,j+\frac{1}{2},k+\frac{1}{2}) = T_{s_{yz}}^n\big(i,j+\frac{1}{2},k+\frac{1}{2}\big) \\
&- \rho C_s^2 \frac{\Delta t}{\Delta z}\big(u_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k+1) - u_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k)\big) \\
&- \rho C_s^2 \frac{\Delta t}{\Delta y}\big(u_z^{n+\frac{1}{2}}(i,j+1,k+\frac{1}{2}) - u_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2})\big)
\end{aligned} \tag{2.47}
$$

$$
\begin{aligned}
T_{s_{zx}}^{n+1}(i+\frac{1}{2},j,k+\frac{1}{2}) = T_{s_{zx}}^n\big(i+\frac{1}{2},j,k+\frac{1}{2}\big) \\
&- \rho C_s^2 \frac{\Delta t}{\Delta x}\big(u_z^{n+\frac{1}{2}}(i+1,j,k+\frac{1}{2}) - u_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2})\big) \\
&- \rho C_s^2 \frac{\Delta t}{\Delta z}\big(u_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k+1) - u_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k)\big)
\end{aligned} \tag{2.48}
$$

図 2.4 に FDTD 法で弾性波数値解析を行う際のグリッドモデルを示す．同図より，応力，粒子速度と剪断応力が $\frac{1}{2}$ ずれて配置されていることが分かる．

弾性波と音場の連成解析を行う方法としては解析空間全領域で圧力を定義し，弾性体内では応力，空気中では圧力を計算するようにすれば良い．弾性体内と空気の媒質境界上では，粒子速度を計算する際に応力と圧力を用いて差分すれば良い．図 2.5 に FDTD 法で弾性波・音場連成数値解析を行う際の媒質境界付近のグリッドモデルを示す．

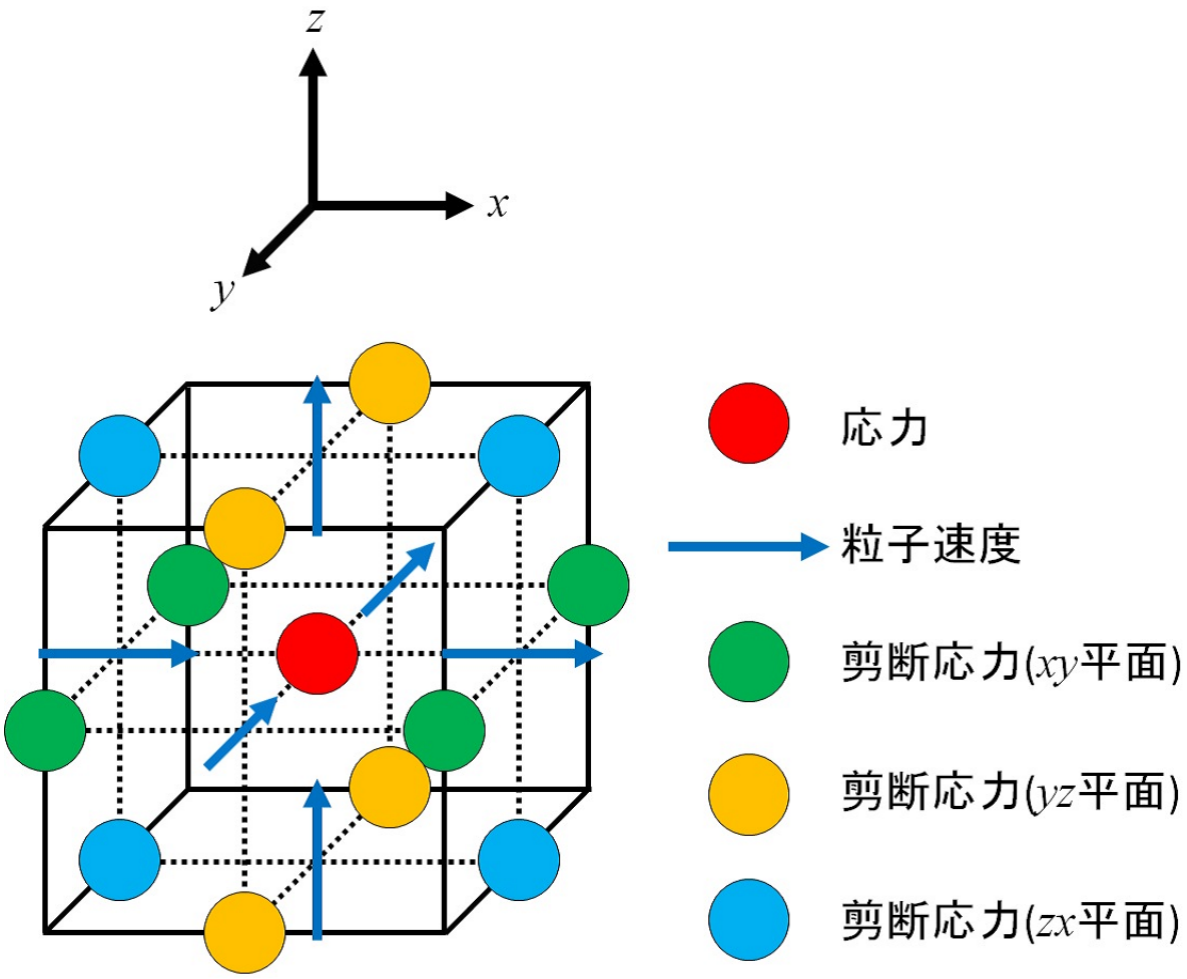

図 2.3　弾性波 FDTD 法のグリッドモデル

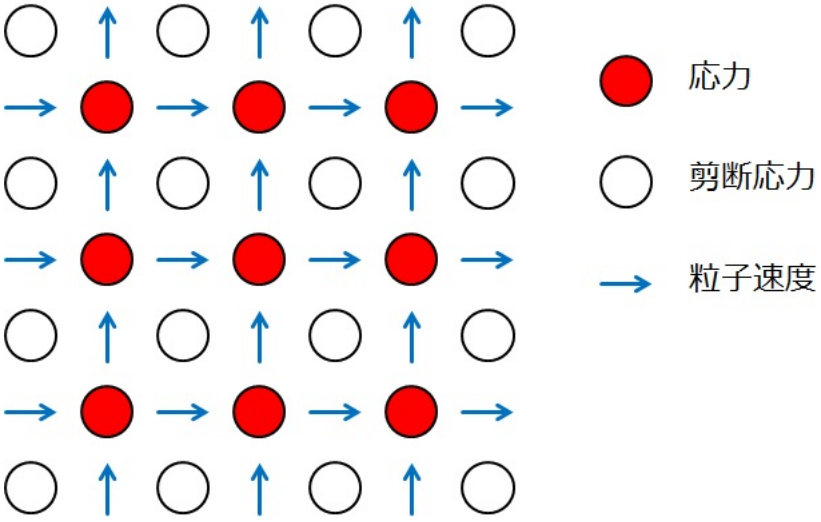

図 2.4 弾性波 FDTD 法における二次元グリッドモデル

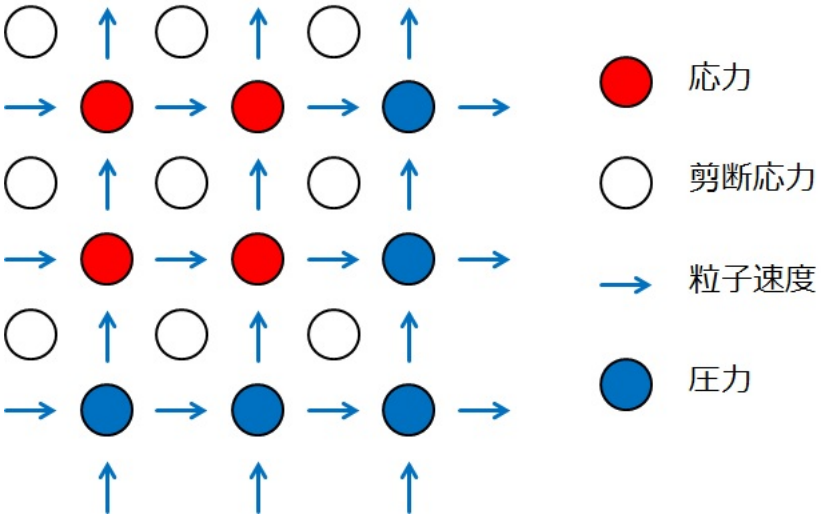

図 2.5 弾性波 FDTD 法における二次元グリッドモデル（媒質境界付近）

2.3 特性曲線法（MOC 法）を用いた時間領域波動伝搬数値解析

2.3.1 MOC 法による音場波動伝搬数値解析

MOC 法による定式化

音場の支配方程式を再掲する．

$$\rho\frac{\partial\vec{v}}{\partial t} = -\nabla p \tag{2.49}$$

$$\nabla\cdot\vec{v} = -\frac{1}{K}\frac{\partial p}{\partial t} \tag{2.50}$$

$$\frac{\partial v_x}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial x} = 0 \tag{2.51}$$

$$\frac{\partial v_y}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial y} = 0 \tag{2.52}$$

$$\frac{\partial v_z}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial z} = 0 \tag{2.53}$$

$$\frac{\partial p}{\partial t} + K\frac{\partial v_x}{\partial x} + K\frac{\partial v_y}{\partial y} + K\frac{\partial v_z}{\partial z} = 0 \tag{2.54}$$

上式より，x 方向について，

$$\frac{\partial}{\partial t}p + c\frac{\partial}{\partial x}Zv_x = 0 \tag{2.55}$$

$$\frac{\partial}{\partial t}Zv_x + c\frac{\partial}{\partial x}p = 0 \tag{2.56}$$

が得られる．Z は特性インピーダンス，c は媒質中の音速である（$Z=\sqrt{\rho K}$，$c=\sqrt{\frac{K}{\rho}}$）．さらに，この 2 つの式の和と差を計算して，

$$\frac{\partial}{\partial t}\left(p \pm Zv_x\right) \pm c\frac{\partial}{\partial x}\left(p \pm Zv_x\right) = 0 \tag{2.57}$$

以上より，x 方向に対して $p \pm Zv_x$ の移流方程式が得られる．

同様にして，y および z 方向についても式を導出し，それぞれの方向についてまとめると，以下のようになる．

x 方向について，

$$\frac{\partial}{\partial t}F_{x\pm} \pm c\frac{\partial}{\partial x}F_{x\pm} = 0 \tag{2.58}$$

y 方向について，

$$\frac{\partial}{\partial t}F_{y\pm} \pm c\frac{\partial}{\partial y}F_{y\pm} = 0 \tag{2.59}$$

z 方向について，

$$\frac{\partial}{\partial t}F_{z\pm} \pm c\frac{\partial}{\partial z}F_{z\pm} = 0 \tag{2.60}$$

ここで，

$$F_{x\pm} = p \pm Zv_x \tag{2.61}$$

$$F_{y\pm} = p \pm Zv_y \tag{2.62}$$

$$F_{z\pm} = p \pm Zv_z \tag{2.63}$$

となる．

これより，3 次元 MOC 法では，x 方向の伝搬について $F_{x\pm}$ に対して補間を適用し，y 方向の伝搬については $F_{y\pm}$ に対して補間を適用し，z 方向の伝搬については $F_{z\pm}$ に対して補間を適用することで音場解析を行う．

本手法では音圧，粒子速度ともに同一グリッド上に配置する．したがって本手法は，FDTD 法とは異なり音圧と粒子速度の半セルのずれは存在しない．

次に，MOC 法を用いて，タイムステップ n におけるグリッド上の値からタイムステップ $n+1$ におけるグリッド上の値を求める方法を述べる．まず，$\pm x$ 方向への伝搬を考える．$\pm x$ 方向の変数として $F_{x\pm}$ については，$+x$ 方向へ F_{x+} が伝搬し，$-x$ 方向へ F_{x-} が伝搬する．$\pm x$ 方向について，タイムステップ $n+1$ の $F_{x\pm}^{n+1}$ は以下の式で与えられる．

$$F_{x\pm}^{n+1}(i,\,j,\,k) = a\xi \pm F_{x\pm}^{n}(i,\,j,\,k) \tag{2.64}$$

となる．ただし，

$$a = \frac{\mp F_{x\pm}^{n}(i,\,j,\,k) \pm F_{x\pm}^{n}(i \mp 1,\,j,\,k)}{\mp \Delta x} \tag{2.65}$$

$$\xi = \mp c\,\Delta t \tag{2.66}$$

上式は 2 点の値で定義される 1 次精度の補間を用いている．一方，$\pm y$ 方向についても $F_{y\pm}$ と置き換えることで同様に求めることができる．同じように，$\pm z$ 方向についても $F_{z\pm}$ と置き換えることで求めることができる．

CIP 法による定式化

式 (2.57) の両辺を x で偏微分して,

$$\frac{\partial}{\partial t}\left(\partial_x p \pm Z\partial_x v_x\right) \pm c\frac{\partial}{\partial x}\left(\partial_x p \pm Z\partial_x v_x\right) = 0 \tag{2.67}$$

また，y で偏微分して

$$\frac{\partial}{\partial t}\left(\partial_y p \pm Z\partial_y v_x\right) \pm c\frac{\partial}{\partial x}\left(\partial_y p \pm Z\partial_y v_x\right) = 0 \tag{2.68}$$

また，z で偏微分して

$$\frac{\partial}{\partial t}\left(\partial_z p \pm Z\partial_z v_x\right) \pm c\frac{\partial}{\partial x}\left(\partial_z p \pm Z\partial_z v_x\right) = 0 \tag{2.69}$$

さらに，この両辺を x および y で偏微分して

$$\frac{\partial}{\partial t}\left(\partial_{xy} p \pm Z\partial_{xy} v_x\right) \pm c\frac{\partial}{\partial x}\left(\partial_{xy} p \pm Z\partial_{xy} v_x\right) = 0 \tag{2.70}$$

さらに，この両辺を x および z で偏微分して

$$\frac{\partial}{\partial t}\left(\partial_{xz} p \pm Z\partial_{xz} v_x\right) \pm c\frac{\partial}{\partial x}\left(\partial_{xz} p \pm Z\partial_{xz} v_x\right) = 0 \tag{2.71}$$

さらに，この両辺を y および z で偏微分して

$$\frac{\partial}{\partial t}\left(\partial_{yz} p \pm Z\partial_{yz} v_x\right) \pm c\frac{\partial}{\partial x}\left(\partial_{yz} p \pm Z\partial_{yz} v_x\right) = 0 \tag{2.72}$$

最後に，この両辺を x, y および z で偏微分して

$$\frac{\partial}{\partial t}\left(\partial_{xyz} p \pm Z\partial_{xyz} v_x\right) \pm c\frac{\partial}{\partial x}\left(\partial_{xyz} p \pm Z\partial_{xyz} v_x\right) = 0 \tag{2.73}$$

ここで, $\partial_x = \frac{\partial}{\partial x}$, $\partial_y = \frac{\partial}{\partial y}$, $\partial_z = \frac{\partial}{\partial z}$, $\partial_{xy} = \frac{\partial^2}{\partial x \partial y}$, $\partial_{xz} = \frac{\partial^2}{\partial x \partial z}$, $\partial_{yz} = \frac{\partial^2}{\partial y \partial z}$ 及び $\partial_{xyz} = \frac{\partial^3}{\partial x \partial y \partial z}$ としている．以上より，x 方向に対して $p \pm Z v_x$，$\partial_x p \pm Z\partial_x v_x$，$\partial_y p \pm Z\partial_y v_x$，$\partial_z p \pm Z\partial_z v_x$，$\partial_{xy} p \pm Z\partial_{xy} v_x$，$\partial_{xz} p \pm Z\partial_{xz} v_x$，$\partial_{yz} p \pm Z\partial_{yz} v_x$ 及び $\partial_{xyz} p \pm Z\partial_{xyz} v_x$ の移流方程式が得られる．

同様にして，y および z 方向についても式を導出し，それぞれの方向についてまとめると，以下のようになる．

x 方向について，

$$\frac{\partial}{\partial t}F_{x\pm} \pm c\frac{\partial}{\partial x}F_{x\pm} = 0 \tag{2.74}$$
$$\frac{\partial}{\partial t}G_{x\pm} \pm c\frac{\partial}{\partial x}G_{x\pm} = 0 \tag{2.75}$$
$$\frac{\partial}{\partial t}H_{x\pm} \pm c\frac{\partial}{\partial x}H_{x\pm} = 0 \tag{2.76}$$
$$\frac{\partial}{\partial t}I_{x\pm} \pm c\frac{\partial}{\partial x}I_{x\pm} = 0 \tag{2.77}$$
$$\frac{\partial}{\partial t}J_{x\pm} \pm c\frac{\partial}{\partial x}J_{x\pm} = 0 \tag{2.78}$$
$$\frac{\partial}{\partial t}K_{x\pm} \pm c\frac{\partial}{\partial x}K_{x\pm} = 0 \tag{2.79}$$
$$\frac{\partial}{\partial t}L_{x\pm} \pm c\frac{\partial}{\partial x}L_{x\pm} = 0 \tag{2.80}$$
$$\frac{\partial}{\partial t}M_{x\pm} \pm c\frac{\partial}{\partial x}M_{x\pm} = 0 \tag{2.81}$$

y 方向について，

$$\frac{\partial}{\partial t}F_{y\pm} \pm c\frac{\partial}{\partial y}F_{y\pm} = 0 \tag{2.82}$$
$$\frac{\partial}{\partial t}G_{y\pm} \pm c\frac{\partial}{\partial y}G_{y\pm} = 0 \tag{2.83}$$
$$\frac{\partial}{\partial t}H_{y\pm} \pm c\frac{\partial}{\partial y}H_{y\pm} = 0 \tag{2.84}$$
$$\frac{\partial}{\partial t}I_{y\pm} \pm c\frac{\partial}{\partial y}I_{y\pm} = 0 \tag{2.85}$$
$$\frac{\partial}{\partial t}J_{y\pm} \pm c\frac{\partial}{\partial y}J_{y\pm} = 0 \tag{2.86}$$
$$\frac{\partial}{\partial t}K_{y\pm} \pm c\frac{\partial}{\partial y}K_{y\pm} = 0 \tag{2.87}$$
$$\frac{\partial}{\partial t}L_{y\pm} \pm c\frac{\partial}{\partial y}L_{y\pm} = 0 \tag{2.88}$$
$$\frac{\partial}{\partial t}M_{y\pm} \pm c\frac{\partial}{\partial y}M_{y\pm} = 0 \tag{2.89}$$

z 方向について,

$$\frac{\partial}{\partial t}F_{z\pm} \pm c\frac{\partial}{\partial z}F_{z\pm} = 0 \tag{2.90}$$
$$\frac{\partial}{\partial t}G_{z\pm} \pm c\frac{\partial}{\partial z}G_{z\pm} = 0 \tag{2.91}$$
$$\frac{\partial}{\partial t}H_{z\pm} \pm c\frac{\partial}{\partial z}H_{z\pm} = 0 \tag{2.92}$$
$$\frac{\partial}{\partial t}I_{z\pm} \pm c\frac{\partial}{\partial z}I_{z\pm} = 0 \tag{2.93}$$
$$\frac{\partial}{\partial t}J_{z\pm} \pm c\frac{\partial}{\partial z}J_{z\pm} = 0 \tag{2.94}$$
$$\frac{\partial}{\partial t}K_{z\pm} \pm c\frac{\partial}{\partial z}K_{z\pm} = 0 \tag{2.95}$$
$$\frac{\partial}{\partial t}L_{z\pm} \pm c\frac{\partial}{\partial z}L_{z\pm} = 0 \tag{2.96}$$
$$\frac{\partial}{\partial t}M_{z\pm} \pm c\frac{\partial}{\partial z}M_{z\pm} = 0 \tag{2.97}$$

ここで,

$$F_{x\pm} = p \pm Zv_x \tag{2.98}$$
$$G_{x\pm} = \partial_x p \pm Z\partial_x v_x \tag{2.99}$$
$$H_{x\pm} = \partial_y p \pm Z\partial_y v_x \tag{2.100}$$
$$I_{x\pm} = \partial_{xy} p \pm Z\partial_{xy} v_x \tag{2.101}$$
$$J_{x\pm} = \partial_z p \pm Z\partial_z v_x \tag{2.102}$$
$$K_{x\pm} = \partial_{xz} p \pm Z\partial_{xz} v_x \tag{2.103}$$
$$L_{x\pm} = \partial_{yz} p \pm Z\partial_{yz} v_x \tag{2.104}$$
$$M_{x\pm} = \partial_{xyz} p \pm Z\partial_{xyz} v_x \tag{2.105}$$

$$F_{y\pm} = p \pm Zv_y \tag{2.106}$$
$$G_{y\pm} = \partial_y p \pm Z\partial_y v_y \tag{2.107}$$
$$H_{y\pm} = \partial_x p \pm Z\partial_x v_y \tag{2.108}$$
$$I_{y\pm} = \partial_{xy} p \pm Z\partial_{xy} v_y \tag{2.109}$$
$$J_{y\pm} = \partial_z p \pm Z\partial_z v_y \tag{2.110}$$
$$K_{y\pm} = \partial_{yz} p \pm Z\partial_{yz} v_y \tag{2.111}$$
$$L_{y\pm} = \partial_{xz} p \pm Z\partial_{xz} v_y \tag{2.112}$$
$$M_{y\pm} = \partial_{xyz} p \pm Z\partial_{xyz} v_y \tag{2.113}$$

$$F_{z\pm} = p \pm Zv_z \tag{2.114}$$
$$G_{z\pm} = \partial_z p \pm Z\partial_z v_z \tag{2.115}$$
$$H_{z\pm} = \partial_x p \pm Z\partial_x v_z \tag{2.116}$$
$$I_{z\pm} = \partial_{xz} p \pm Z\partial_{xz} v_z \tag{2.117}$$
$$J_{z\pm} = \partial_y p \pm Z\partial_y v_z \tag{2.118}$$
$$K_{z\pm} = \partial_{yz} p \pm Z\partial_{yz} v_z \tag{2.119}$$
$$L_{z\pm} = \partial_{xy} p \pm Z\partial_{xy} v_z \tag{2.120}$$
$$M_{z\pm} = \partial_{xyz} p \pm Z\partial_{xyz} v_z \tag{2.121}$$

となる．

これより，3 次元 CIP 法では，x 方向の伝搬について $F_{x\pm}$ と $G_{x\pm}$，$H_{x\pm}$ と $I_{x\pm}$，$J_{x\pm}$ と $K_{x\pm}$，及び $L_{x\pm}$ と $M_{x\pm}$ に対して CIP 法（3 次エルミート補間）をそれぞれ適用し，y 方向の伝搬については $F_{y\pm}$ と $G_{y\pm}$，$H_{y\pm}$ と $I_{y\pm}$，$J_{y\pm}$ と $K_{y\pm}$，及び $L_{y\pm}$ と $M_{y\pm}$ に対して CIP 法をそれぞれ適用し，z 方向の伝搬については $F_{z\pm}$ と $G_{z\pm}$，$H_{z\pm}$ と $I_{z\pm}$，$J_{z\pm}$ と $K_{z\pm}$，及び $L_{z\pm}$ と $M_{z\pm}$ に対して CIP 法をそれぞれ適用することで音場解析を行う．

本手法では音圧，粒子速度ともに同一グリッド上に配置する．したがって本手法は，FDTD 法とは異なり音圧と粒子速度の半セルのずれは存在しない．また，各グリッド上にそれぞれの成分の微分値 $\partial_x p$，$\partial_x v_x$，$\partial_y p$，$\partial_y v_y$，$\partial_z p$ 及び $\partial_z v_z$ を配置する．

次に，CIP 法を用いて，タイムステップ n におけるグリッド上の値からタイムステップ $n+1$ におけるグリッド上の値を求める方法を述べる．まず，$\pm x$ 方向への伝搬を考える．$\pm x$ 方向の変数として $F_{x\pm}$ 及び $G_{x\pm}$ については，$+x$ 方向へ F_{x+} 及び G_{x+} が伝搬し，$-x$ 方向へ F_{x-} 及び G_{x-} が伝搬する．$\pm x$ 方向について，タイムステップ $n+1$ の $F_{x\pm}^{n+1}$ 及び $G_{x\pm}^{n+1}$ は以下の式で与えられる．

$$F_{x\pm}^{n+1}(i,\, j,\, k) = a\xi^3 \pm b\xi^2 \pm G_{x\pm}^{n}(i,\, j,\, k)\xi \pm F_{x\pm}^{n}(i,\, j,\, k) \tag{2.122}$$
$$G_{x\pm}^{n+1}(i,\, j,\, k) = 3a\xi^2 \pm 2b\xi \pm G_{x\pm}^{n}(i,\, j,\, k) \tag{2.123}$$

となる．ただし，

$$
\begin{aligned}
a &= \frac{G^n_{x\pm}(i,\,j,\,k) \pm G^n_{x\pm}(i \mp 1,\,j,\,k)}{(\mp\Delta x)^2} \\
&\pm \frac{2(F^n_{x\pm}(i,\,j,\,k) \mp F^n_{x\pm}(i \mp 1,\,j,\,k))}{(\mp\Delta x)^3} \qquad (2.124) \\
b &= \frac{3(F^n_{x\pm}(i \mp 1,\,j,\,k) \mp F^n_{x\pm}(i,\,j,\,k))}{(\mp\Delta x)^2} \\
&\mp \frac{2G^n_{x\pm}(i,\,j,\,k) \pm G^n_{x\pm}(i \mp 1,\,j,\,k)}{\mp\Delta x} \qquad (2.125) \\
\xi &= \mp c\,\Delta t \qquad (2.126)
\end{aligned}
$$

上式は 2 点の値と微分値で定義される 3 次エルミート補間となっている．$H_{x\pm}$ と $I_{x\pm}$，$J_{x\pm}$ と $K_{x\pm}$ 及び $L_{x\pm}$ と $M_{x\pm}$ についても $F_{x\pm}$ と $G_{x\pm}$ と入れ替えることで，同様に 3 次エルミート補間を用いて計算できる．一方，$\pm y$ 方向についても $F_{y\pm}$ と $G_{y\pm}$，$H_{y\pm}$ と $I_{y\pm}$，$J_{y\pm}$ と $K_{y\pm}$ 及び $L_{y\pm}$ と $M_{y\pm}$ と置き換えることで同様に求めることができる．同じように，$\pm z$ 方向についても $F_{z\pm}$ と $G_{z\pm}$，$H_{z\pm}$ と $I_{z\pm}$，$J_{z\pm}$ と $K_{z\pm}$ 及び $L_{z\pm}$ と $M_{z\pm}$ と置き換えることで求めることができる．

上の式 (2.122) から式 (2.126) は，グリッド上の境界条件より求められるが，通常，補間計算を行う場合は，以下のように積和計算の形にして値を求める．

$$
\begin{aligned}
F^{n+1}_{x\pm}(i,\,j) &= C_{\pm}(1)F^n_{x\pm}(i\mp1,\,j) \\
&+ C_{\pm}(2)F^n_{x\pm}(i,\,j) \\
&+ C_{\pm}(3)G^n_{x\pm}(i\mp1,\,j) \\
&+ C_{\pm}(4)G^n_{x\pm}(i,\,j) \qquad (2.127) \\
G^{n+1}_{x\pm}(i,\,j) &= C'_{\pm}(1)F^n_{x\pm}(i\mp1,\,j) \\
&+ C'_{\pm}(2)F^n_{x\pm}(i,\,j) \\
&+ C'_{\pm}(3)G^n_{x\pm}(i\mp1,\,j) \\
&+ C'_{\pm}(4)G^n_{x\pm}(i,\,j). \qquad (2.128)
\end{aligned}
$$

ただし，

$$
\begin{aligned}
C_{\pm}(1) &= -2\chi^3 + 3\chi^2, && (2.129)\\
C_{\pm}(2) &= 2\chi^3 - 3\chi^2 + 1, && (2.130)\\
C_{\pm}(3) &= -2\xi_{\pm}(\chi^2 - \chi), && (2.131)\\
C_{\pm}(4) &= -2\xi_{\pm}(\chi^2 - 2\chi + 1), && (2.132)\\
C'_{\pm}(1) &= 6(-\chi^3 + \chi^2)/\xi_{\pm}, && (2.133)\\
C'_{\pm}(2) &= 6(\chi^3 - \chi^2)/\xi_{\pm}, && (2.134)\\
C'_{\pm}(3) &= 3\chi^2 - 2\chi, && (2.135)\\
C'_{\pm}(4) &= 3\chi^2 - 4\chi + 1 && (2.136)
\end{aligned}
$$

とする．ここで，$\chi = c\,\Delta t/\Delta x$ である．$C_{\pm}()$ 及び $C'_{\pm}()$ は媒質で決まる定数であるので，積和演算の形で実装することで比較的高速な計算が可能となる．

さらに直交微分値の取り扱いについても，エルミート補間を用いる場合には同様の考え方ができる．すなわち，

$$
\begin{aligned}
H^{n+1}_{x\pm}(i,\,j) &= C_{\pm}(1)H^{n}_{x\pm}(i\mp1,\,j)\\
&+ C_{\pm}(2)H^{n}_{x\pm}(i,\,j)\\
&+ C_{\pm}(3)I^{n}_{x\pm}(i\mp1,\,j)\\
&+ C_{\pm}(4)I^{n}_{x\pm}(i,\,j) && (2.137)\\
I^{n+1}_{x\pm}(i,\,j) &= C'_{\pm}(1)H^{n}_{x\pm}(i\mp1,\,j)\\
&+ C'_{\pm}(2)H^{n}_{x\pm}(i,\,j)\\
&+ C'_{\pm}(3)I^{n}_{x\pm}(i\mp1,\,j)\\
&+ C'_{\pm}(4)I^{n}_{x\pm}(i,\,j). && (2.138)
\end{aligned}
$$

なる式で，$F^{n+1}_{x\pm}$ 及び $G^{n+1}_{x\pm}$ と全く同様に求められる．

CIP 法から MM-MOC 法への発展

前節のとおり，CIP 法を用いた音場シミュレーションでは，音場の波動方程式を移流方程式に変換し，進行波と後退波のそれぞれに対して伝搬を解く．このとき，次のタイムステップの音場を計算するために，グリッド上の音圧と粒子速度およびそれらの微分値を用いて 3 次エルミート補間を適用する．

しかし，エルミート補間補間多項式は 3 次だけでなく，一般形は

$$\begin{aligned} F(x) &= \sum_i h_i^{(0)}(x)\, f(x_i) + \sum_i h_i^{(1)}(x)\, f^{'}(x_i) \\ &+ \sum_i h_i^{(2)}(x)\, f^{''}(x_i) + \cdots \end{aligned} \tag{2.139}$$

で与えられる．ただし，$h_i^{(n)}(x)$ は m 階微分値に対する補間関数とする．一方，$F'(x)$ や $F''(x)$ の補間値についても，式（2.139）と同様により，積和演算の形で計算できる．

ここで multi-moments の考え方について述べる．この考え方は CIP 法によって初めて与えられたのもであるが，これは物理量だけでなく，その微分値も変数として定義し，それらを同時に計算する方法のことをいう．すなわち，音圧や粒子速度という物理量に加えて，もう 1 つ，それらの 1 階微分値というモーメントと利用していることを示しており，さらに 2 階微分値も用いれば，3 つのモーメントを利用していることになる．したがって，一般に m 階微分値までを用いる場合は $(m+1)$ のモーメントを用いた手法となる．式（2.139）の補間式を見てみると，まさにエルミート補間の一般形はこのマルチモーメントを利用するのに最適な補間法ということができる．

以上より，Generalized CIP 法として，l 次多項式を用いた $m+1$ モーメントの計算方法を MM(l,m)-MOC 法と呼ぶことができるだろう．この定義によると，MM(3,1)-MOC 法が従来の CIP 法にあたり，MM(5,1)-MOC 法は従来の IDO 法となる．また 4 点のサポートポイントを持つ 7 次エルミート補間を用いる手法は MM(7,1)-MOC 法にあたる．さらに，MM(1,0)-MOC 法，MM(2,0)-MOC 法及び MM(3,0)-MOC 法はそれぞれ 1 次風上法，Lax-Wendroff（LW）法および QUICKEST 法にあたる．ここで，MM(l,m)-MOC 法のサポート点は $(l+1)/(m+1)$ 点となることは明らかである．

先ほど定義した MM-MOC 法の補間関数と補間多項式を示す．MM-MOC 法では，一般形エルミート補間を用いて，高次の補間多項式を定義することが可能である．いま，l 次エルミート補間の作用を H^l 及び $\mathsf{H}^{l\prime}$ とすると，

$$F_{x\pm}^{n+1}(i) \leftarrow \mathsf{H}^l(F_{x\pm}^n, G_{x\pm}^n), \tag{2.140}$$

$$G_{x\pm}^{n+1}(i) \leftarrow \mathsf{H}^{l\prime}(F_{x\pm}^n, G_{x\pm}^n). \tag{2.141}$$

となる．

したがって，例えば MM(3,1)-MOC 法は，式（2.139）より

$$F_{x\pm}^{n+1}(i) = \sum_{k=i\mp1}^{i} h_k^{(0)}(x)\, F_{x\pm}^n + \sum_{k=i\mp1}^{i} h_k^{(1)}(x)\, G_{x\pm}^n \tag{2.142}$$

MM(7,1)-MOC 法は，

$$F_{x\pm}^{n+1}(i) = \sum_{k=i\mp2}^{i\pm1} h_k^{(0)}(x)\, F_{x\pm}^{n} + \sum_{k=i\mp2}^{i\pm1} h_k^{(1)}(x)\, G_{x\pm}^{n} \tag{2.143}$$

一方，2 階微分値まで使用する MM(5,2)-MOC 法は

$$\begin{aligned} F_{x\pm}^{n+1}(i) &= \sum_{k=i\mp1}^{i} h_k^{(0)}(x)\, F_{x\pm}^{n} + \sum_{k=i\mp1}^{i} h_k^{(1)}(x)\, G_{x\pm}^{n} \\ &+ \sum_{k=i\mp1}^{i} h_k^{(2)}(x)\, H_{x\pm}^{n} \end{aligned} \tag{2.144}$$

ただし，

$$H_{x\pm} = \partial_{xx} p \pm Z \partial_{xx} v_x \tag{2.145}$$

のようになる．

さらに，$n+1$ における p と v_x とその微分値は以下の式で求められる．

$$p^{n+1}(i) \leftarrow \frac{F_+^{n+1}(i) + F_-^{n+1}(i)}{2} \tag{2.146}$$

$$v_x^{n+1}(i) \leftarrow \frac{F_+^{n+1}(i) - F_-^{n+1}(i)}{2Z} \tag{2.147}$$

$$\partial_x p^{n+1}(i) \leftarrow \frac{G_+^{n+1}(i) + G_-^{n+1}(i)}{2} \tag{2.148}$$

$$\partial_x v_x^{n+1}(i) \leftarrow \frac{G_+^{n+1}(i) - G_-^{n+1}(i)}{2Z} \tag{2.149}$$

$$\partial_{xx} p^{n+1}(i) \leftarrow \frac{H_+^{n+1}(i) + H_-^{n+1}(i)}{2} \tag{2.150}$$

$$\partial_{xx} v_x^{n+1}(i) \leftarrow \frac{H_+^{n+1}(i) - H_-^{n+1}(i)}{2Z} \tag{2.151}$$

以上のように，各タイムステップでエルミート補間を繰り返して，次の時刻の音圧と粒子速度を求めることができる．注目すべき点は，多項式の次数が増えたとしても，補間計算は積和演算の形であらわされるため，比較的高速な計算が可能となる．特に GPU を利用した並列計算では高速化が期待できる．また実際に計算コードを記述するときは，Zv_x，Zv_y，Zv_z は積の形で変数として扱えるため，計算時間を削減できる．

図 2.6 に MOC 法のグリッドモデルを示す．

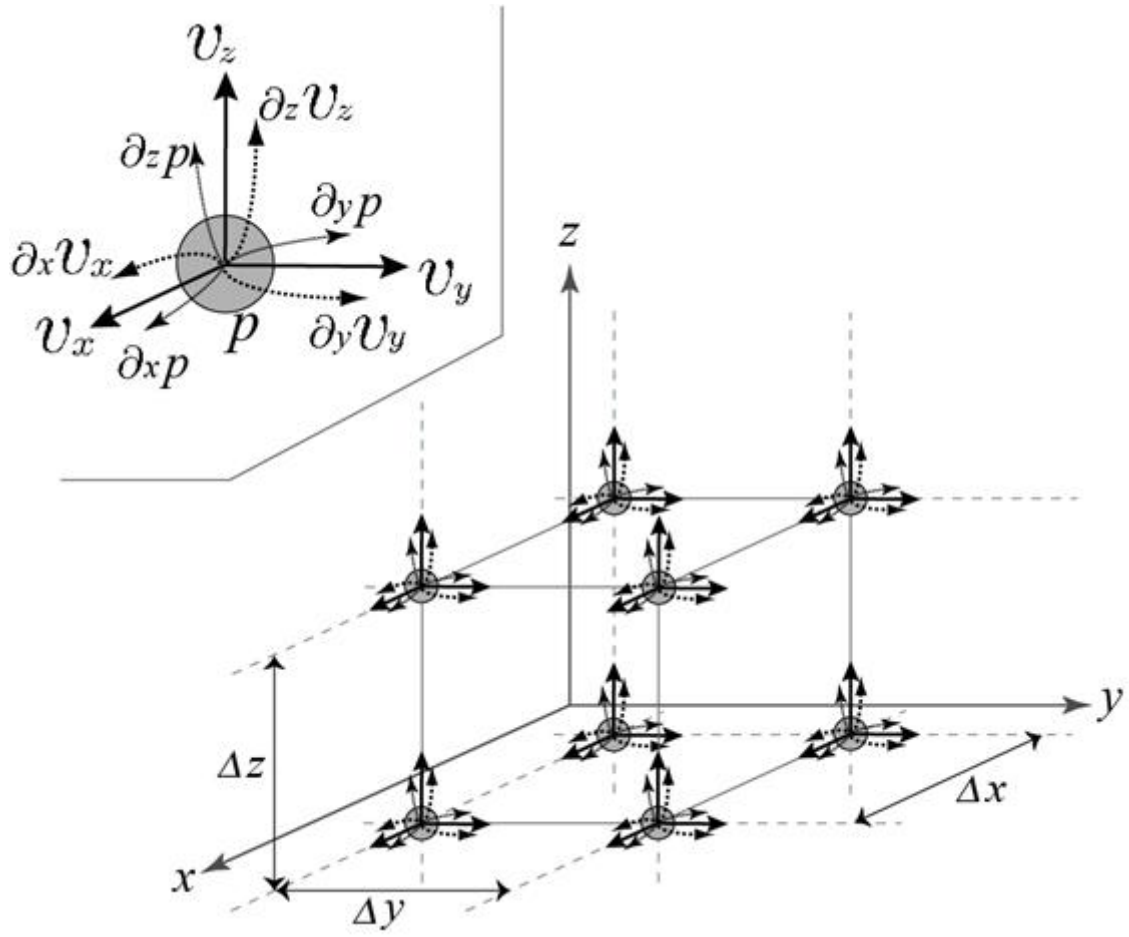

図 2.6　MOC 法のグリッドモデル

2.4 FDTD 法と MM-MOC 法の精度比較評価と考察

FDTD 法と MM-MOC 法の精度比較を行う．二次元音場波動伝搬数値解析における伝搬精度を比較する．解析空間の音源において線音源で駆動し，各方位角における 2 点の音圧を比較して，数値散逸および数値分散誤差を計算する．解析に用いたパラメータを表 2.1 にまとめる．

表 2.1 精度比較評価で用いた解析条件

タイムステップ	$\Delta t = 5.0 \times 10^{-5}$ s
グリッドサイズ	$\Delta x = \Delta y = 0.05$ m
密度 ρ	1.21 kg/m^3
体積弾性率 K	1.4235529 $\times 10^5$ N/m^2

波の伝搬誤差を評価する方法の概略図を図 2.7 に示す．解析空間の中心に波源を設置し，伝搬した波を観測する観測点 A と観測点 B を設定する．観測点 B は観測点 A と比較して波源から遠い位置に設定するものとする．観測点 A で観測された波形と観測点 B で観測された波形を比較し，A-B 間での伝搬について評価を行う．

初めに，数値散逸誤差についての比較を行う．数値散逸とは高周波数帯域で振幅の減少が起こりエネルギーが減少することである．図 2.9 に数値散逸の例を示す．図 2.10 に方位角ごとの数値散逸誤差を比較した図を示す．ただし同図は 679Hz(PPW $\fallingdotseq$ 10) の場合を示している．PPW とは Points Per Wavelength のことであり，1 波長をどれくらいのグリッドで表しているかを示している．PPW で規格化して波長分解能を揃えることで，波の伝搬速度や使用周波数帯が違う場合においても解が一致するため，統一的に精度の評価を行うことができる．横軸が波源から観測点への方位角，縦軸がエネルギーの減衰量を表す．そして，図 2.11 に周波数 (PPW) ごとの数値散逸誤差を比較した図を示す．ただし同図は方位角が 0° の場合を示している．横軸が PPW，縦軸がエネルギーの減衰量を表す．

図 2.10 から分かるように，FDTD 法の場合は数値散逸誤差はほぼ発生しない．しかし，MM(3, 1)-MOC 法（従来の CIP 法）では数値散逸が発生している．だが，MM(7, 1)-MOC 法や MM(5, 2)-MOC 法では数値散逸はほぼ発生せず，MM(3, 1)-MOC と比較して大幅に改善されている．使用する点の数や微分値の数を増やすことで数値散逸誤差は改善できることが分かった．

次に，数値分散誤差についての比較を行う．数値分散とは高周波数帯域で位相速度が遅れ波形が歪むことである．図 2.8 に数値分散の例を示す．図 2.12 に方位角ごとの数値分散誤差を比較した図を示す．ただし同図は 679Hz(PPW $\fallingdotseq$ 10) の場合を示している．横軸が

波源から観測点への方位角，縦軸が位相速度がどれだけずれているかを表す．そして，図 2.13 に周波数 (PPW) ごとの数値分散誤差を比較した図を示す．ただし同図は方位角が 0° の場合を示している．横軸が PPW，縦軸が位相速度がどれだけずれているかを表す．

図 2.12 から分かるように，MM-MOC 法の場合は数値分散誤差はほぼ発生しない．対して，FDTD 法では数値分散が発生していることが分かる．グリッドサイズやタイムステップを細かくすることで誤差を小さくできることが図 2.12 でも示されているが，そのぶんメモリを多く使用し計算時間も膨大になる．

FDTD 法と MM-MOC 法の比較を行った結果，数値散逸誤差が改善された MM(7, 1)-MOC 法は比較的少ない計算量で精度良く計算できる手法であることが分かった．

表 2.2 に FDTD 法と MM-MOC 法の計算コストをまとめた．計算領域のグリッド数を 1024 × 1024，計算回数を 1024 ステップとした場合の計算時間，メモリ量，1 ステップあたりの演算量を示している．CPU として Intel Core i7 920 を用い，OpenMP を用いて並列化を行った．

表 2.2　計算コストの比較

	FDTD 法	MM(3, 1)-MOC 法	MM(7, 1)-MOC 法
計算時間 [s]	3.05	58.63	101.00
メモリ量 [MByte]	12	48	48
1 ステップあたりの演算量	12	144	272

同表から同じ解析条件で解析を行った場合，MM-MOC 法は FDTD 法と比較して多くの計算コストがかかることが分かる．次章では，GPU を用いることでどれだけ高速化が行えるか検討を行う．

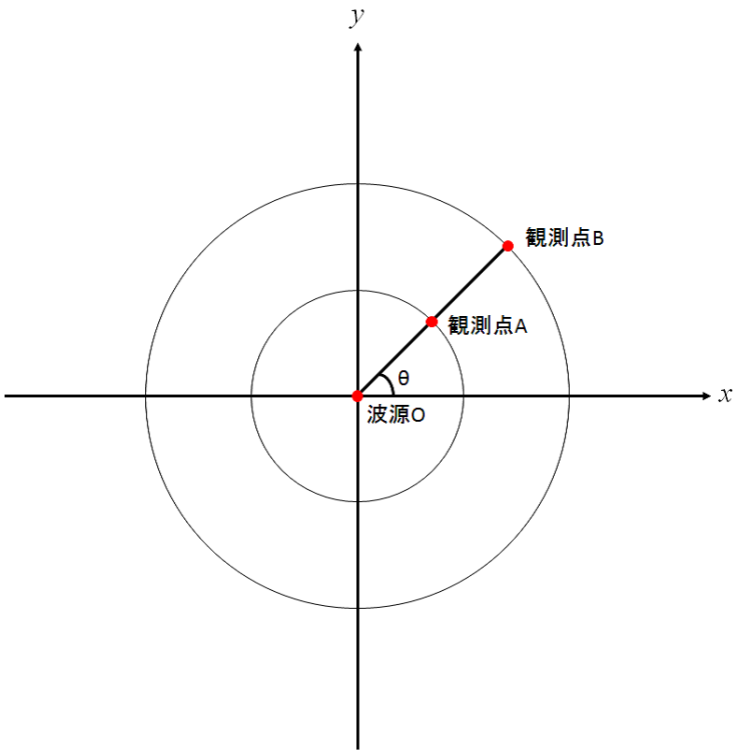

図 2.7 伝搬誤差の比較方法

図 2.8 数値分散の例

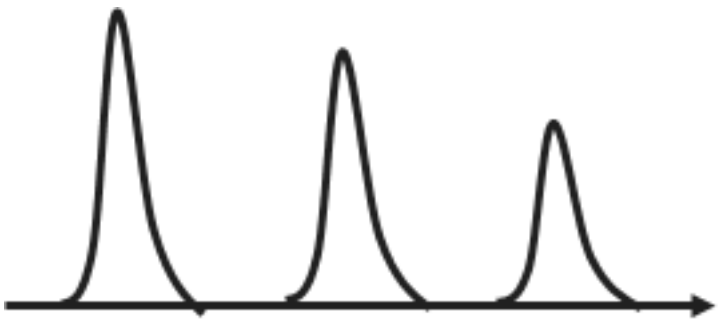

図 2.9 数値散逸の例

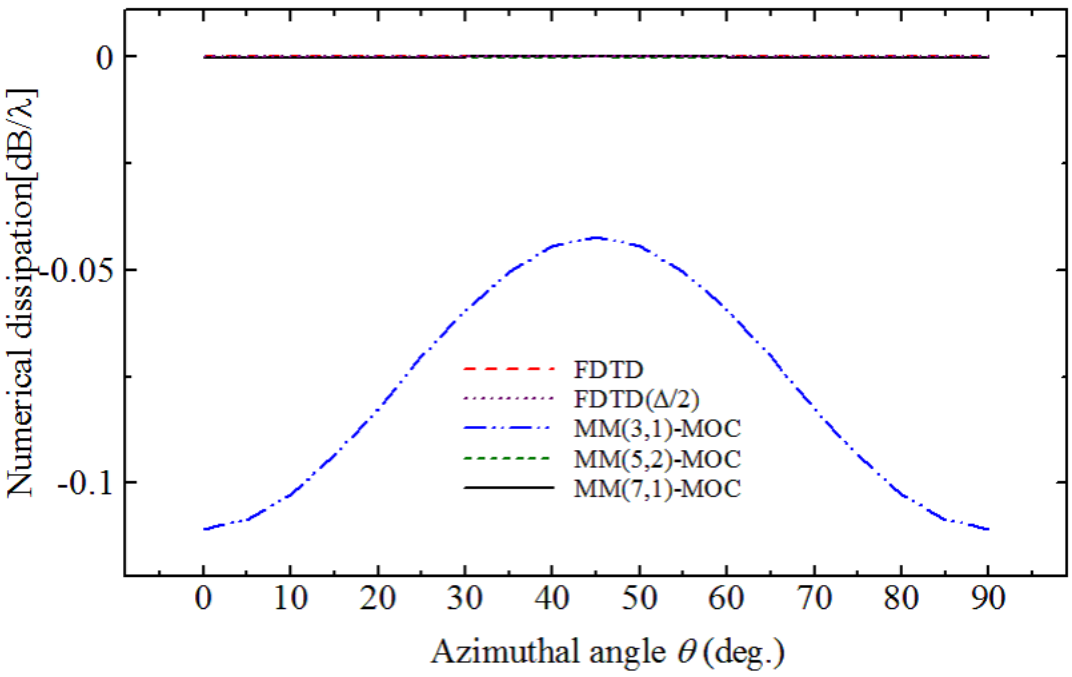

図 2.10　方位角ごとの数値散逸誤差

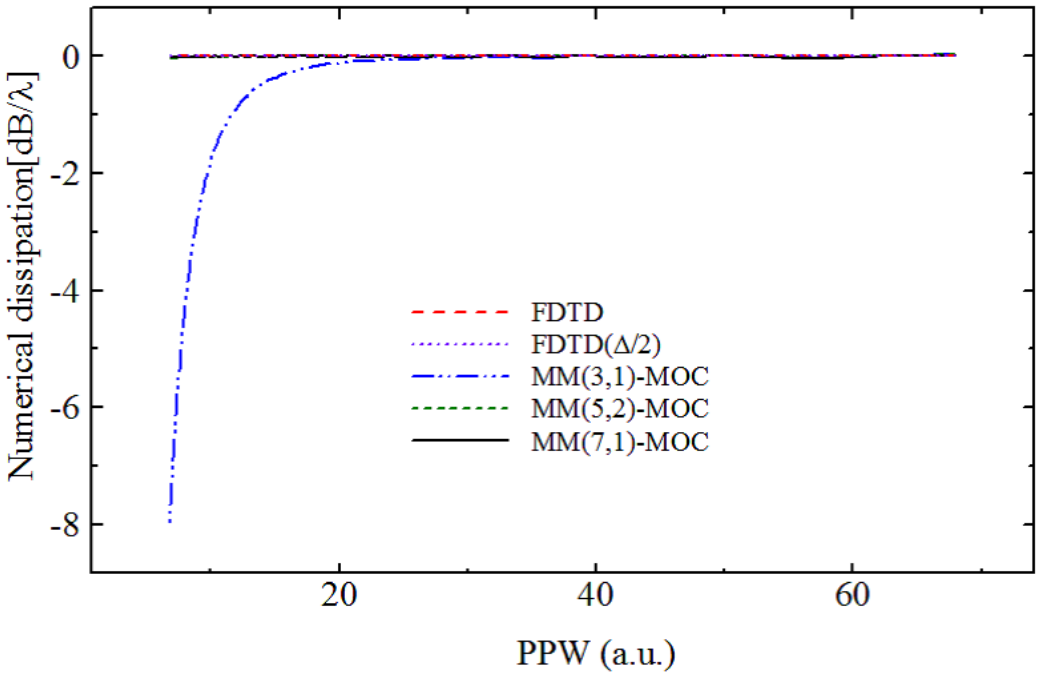

図 2.11　PPW ごとの数値散逸誤差

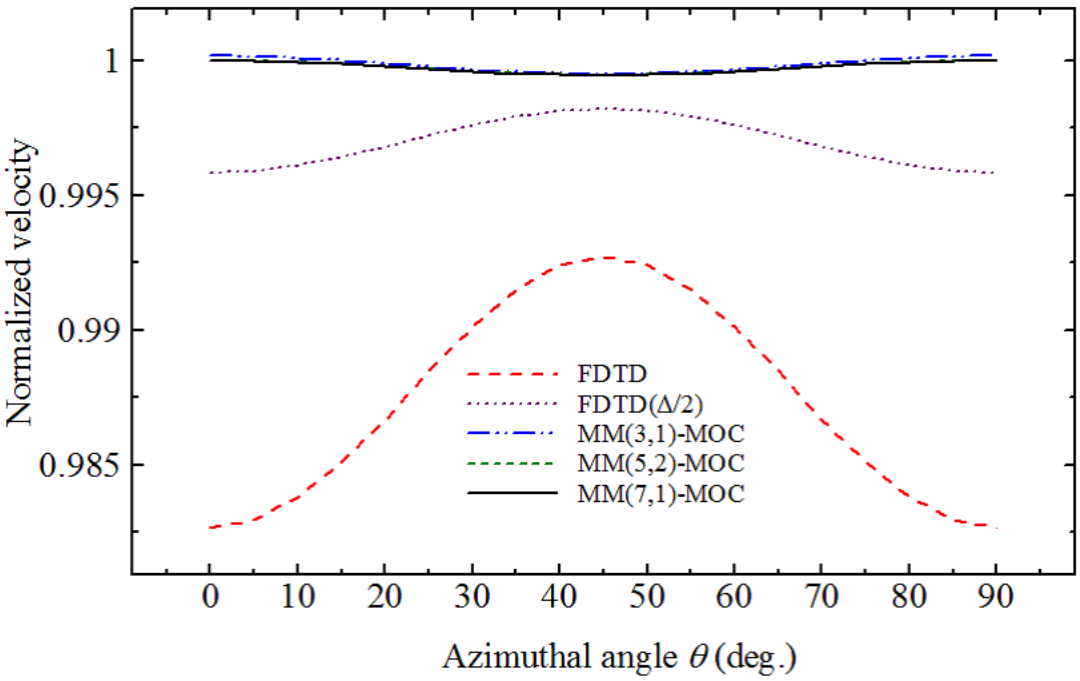

図 2.12　方位角ごとの数値分散誤差

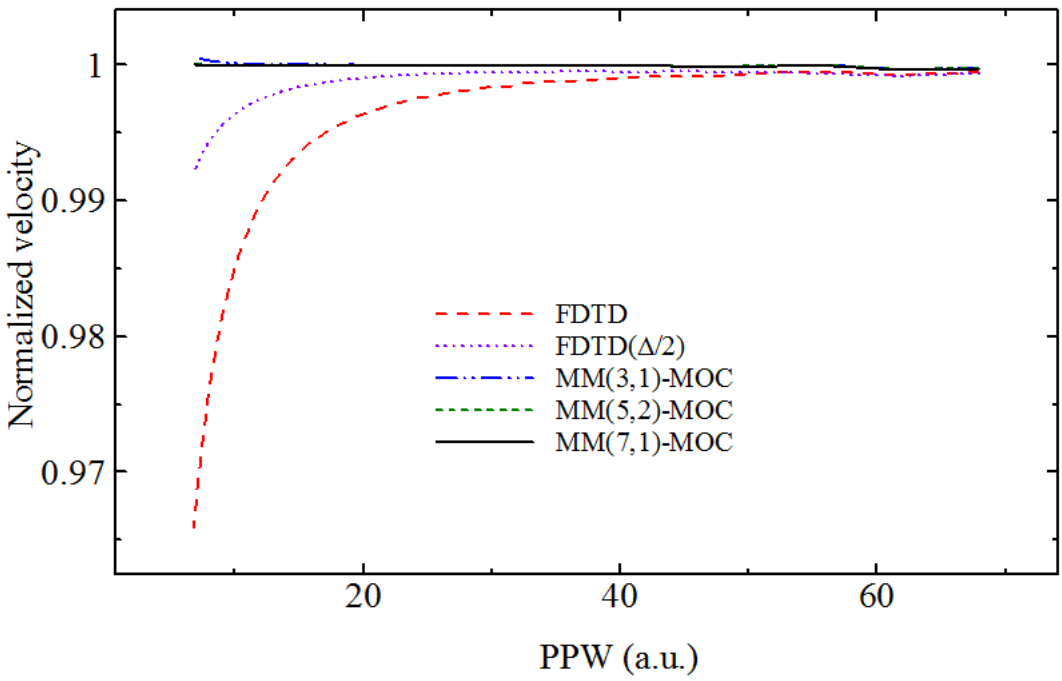

図 2.13　PPW ごとの数値分散誤差

2.5　結言

この章では，波動伝搬数値解析手法である FDTD 法，MM-MOC 法の概要について説明し，それぞれ定式化を行った．そして，FDTD 法と MM-MOC 法の精度を比較し評価を行った．その結果，MM(7, 1)-MOC 法が 2 次精度 FDTD 法に比べて高い精度で計算できることを示した．

しかし，アンテナ解析のような問題では MM-MOC 法を用いることは難しく，逆に非線形問題などは FDTD 法で解くことは難しい．解析する問題によって適切な解析手法を選択する必要がある．今回提案した MM-MOC 法は手法を選ぶ際の有効な選択肢となるだろう．

第3章　GPU (Graphics Processing Unit) の進化とGPUを用いた数値解析の高速化

3.1　序言

本章では，波動伝搬数値解析の GPU への実装について述べる．

初めに，3.2 節で GPU の概要について述べる．3.2.1 項で GPU の進化について述べ，3.2.2 項で開発環境に関して，3.2.3 項で GPU のアーキテクチャについて解説する．

次に，3.3 節で GPU を用いた波動伝搬数値解析の高速化について述べる．3.2.3 項で解説した GPU のアーキテクチャに適した波動伝搬数値解析の GPU への実装・最適化を行う．3.3.1 項では GPU を用いた FDTD 法の高速化と評価を，3.3.2 項では GPU を用いた MM-MOC 法の高速化と評価を行う．

次に，3.4 節では 3.3 節の高速化の結果を踏まえたマルチ GPU によるさらなる高速化について述べる．3.4.1 項でマルチ GPU 計算の必要性について説明し，3.4.2 項で数値解析をマルチ GPU 化するための手法を説明する．最後に，マルチ GPU 化による高速化の結果について解説し，考察を行う．

3.2　GPU 概要－歴史と進化について－

3.2.1　GPU の著しい進化

GPU とは Graphics Processing Unit の略称であり，グラフィックス専用ハードウェアである．近年，GPU の性能は著しい速度で向上している．図 3.1 及び 3.2 に GPU と CPU の性能の比較を示す．

2005 年前後から GPU の性能が向上し，GPGPU(General Purpose computation on GPUs) に注目が集まっても大きな広がりは見られなかった．それは，Cg に代表される当時の GPU プログラミング言語がグラフィックス処理のためのものであり，汎用的なプログラムの記述に不向きであったことが原因であると考えられる．

しかし，NVIDIA 社が自社の GPU 用に，C 言語を用いて GPU プログラムの記述をすることができる汎用コンピューティングに適した開発環境 CUDA(Compute Unified Device Architecture) を開発したことにより，様々な分野で GPU を用いた研究が行われるようになり，GPGPU は爆発的に広がった．

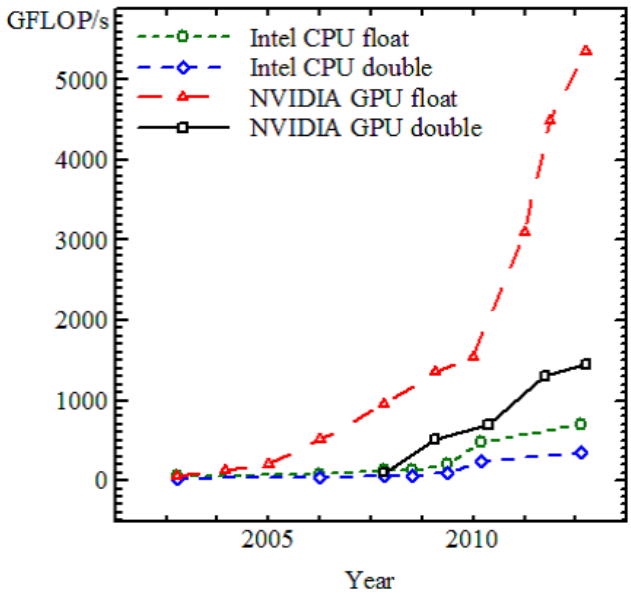

図 3.1　GPU の計算性能（出典：CUDA C Programming Guide）

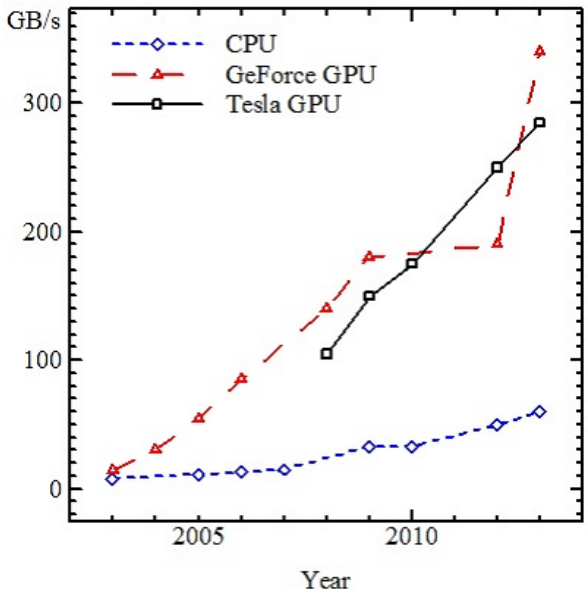

図 3.2　GPU のメモリ帯域幅（出典：CUDA C Programming Guide）

3.2.2 GPU 用プログラム開発環境 CUDA

CUDA は NVIDIA 社が提供している開発環境であり，NVIDIA 製 GPU で動作する GPU プログラムを作成することができる．C 言語標準の型が使えず開発の敷居が高かった Cg などのグラフィックス用言語とは異なり，C 言語の型を用いて C 言語ライクに開発できるため，汎用数値計算を容易に行えるようになった．

以下に，CUDA プログラムのサンプルを記述する．

```
1  #include <iostream>
2  #include <algorithm>
3  #include <cutil.h>
4
5  const int AryElem = 100;
6
7  __global__ void GpuFunction(float *input);
8
9  int main(int argc, char **argv)
10 {
11         CUT_DEVICE_INIT(argc, argv);
12
13         float *x = new float[AryElem];
14
15         for(int i = 0; i < AryElem; ++i)
16         {
17                 x[i] = static_cast<float>(i);
18         }
19
20         float *gpu_x;
21         int DataSize = sizeof(float) * AryElem;
22
23         CUDA_SAFE_CALL(cudaMalloc(reinterpret_cast<void **>(&gpu_x),
               DataSize));
24
25         CUDA_SAFE_CALL(cudaMemcpy(gpu_x, x, DataSize,
               cudaMemcpyHostToDevice));
26
27         dim3 threads(AryElem, 1, 1);
28         dim3 grid(1, 1, 1);
29
30         GpuFunction <<< grid, threads >>> (gpu_x);
31
32         CUDA_SAFE_CALL(cudaMemcpy(x, gpu_x, DataSize,
               cudaMemcpyDeviceToHost));
33
34         cudaFree(gpu_x);
35
36         std :: copy(x, x + AryElem, std :: ostream_iterator<float>(std :: cout, "\n"));
```

```
37
38          delete[] x;
39
40          CUT_EXIT(argc, argv);
41
42          return 0;
43  }
44
45  __global__ void GpuFunction(float *input)
46  {
47          input[threadIdx.x] += 100;
48
49          __syncthreads();
50  }
```

CUDA プログラムは，デバイス（GPU）を制御するホスト（CPU）用の記述と，デバイスで実行する関数（カーネル）用の記述に別れている．ホストが指定した数だけスレッドは起動し，カーネルはマルチスレッドで並列に動作する．デバイスの計算結果をホストに転送する関数なども用意されており，ホスト側が制御を行う．

このプログラムは配列の値に 100 を加算する単純なものである．23 行目で GPU の VRAM 上に配列を確保し，25 行目で VRAM 上の配列に RAM 上の配列のデータを転送する．27 行目でブロックのサイズを指定し，28 行目でグリッドのサイズを指定する．30 行目で GPU にカーネルを投げる．32 行目で計算結果を RAM 上の配列に転送する．34 行目で VRAM 上の配列を解放する．45 行目から 50 行目が GPU で動作するカーネル部分である．カーネルはマルチスレッドで動作するため，組み込み変数 threadIdx.x を用いて，それぞれのスレッドがアクセスすべき配列のインデックスを決定する．

このように，CUDA プログラムはホストとデバイスが相互に動作するヘテロジニアスなプログラムといえる．

GPU で動作するプログラムの流れの概略図を図 3.3 に示す．

3.2.3　GPU のアーキテクチャについて

図 3.4 に NVIDIA GPU のハードウェアモデルを示す．

本研究では GPU に GeForce GTX 285，GeForce GTX 580 と GeForce GTX TITAN を使用した．

GeForce GTX 285 [33] に搭載されている GT200 系チップは，30 個のストリーミングマルチプロセッサ (SMP) で構成され，各 SMP では 8 個のストリーミングプロセッサ (SP) が並列に動作する．各 SMP には高速にアクセス可能な共有メモリ (SM) があり，8 個の

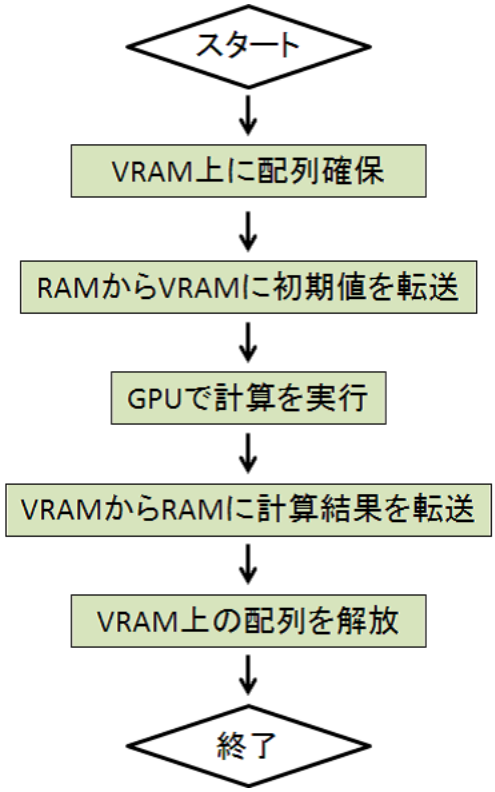

図 3.3 GPU プログラムの流れ

SP で共有されている．そして，全ての SP がレジスタを保有している．チップ上には他にはコンスタントキャッシュ・テクスチャキャッシュが置かれている．チップ外にはグローバルメモリ (VRAM) が設置されているが，チップの外に置かれているため，データ転送にかなりの時間がかかる．しかし，チップの外にあるために大容量である．

また，GeForce GTX 580 [34] に搭載されている GF100 系チップは，16 個の SMP で構成され，各 SMP では 32 個の SP が並列に動作する．メモリ周りの概要に関しては GT200 系チップと同様だが，大きな違いとしてキャッシュが搭載されていることが挙げられる．

そして，GeForce GTX TITAN [35] に搭載されている GK110 系チップは，14 個の SMP で構成され，各 SMP では 192 個の SP が並列に動作する．GeForce GTX 580 と同様にキャッシュが搭載されている．

CUDA における最小実行単位はスレッドと呼ばれ，スレッドを 32 個まとめたものをワープと呼ぶ．また，CUDA ではスレッドのまとまりをブロック，ブロックのまとまりをグリッドと呼び管理している．グリッドはホストから実行を指令する単位で，グリッド内の全スレッドは同一のカーネルを実行する．図 3.5 に CUDA のスレッドの階層構造を示す．

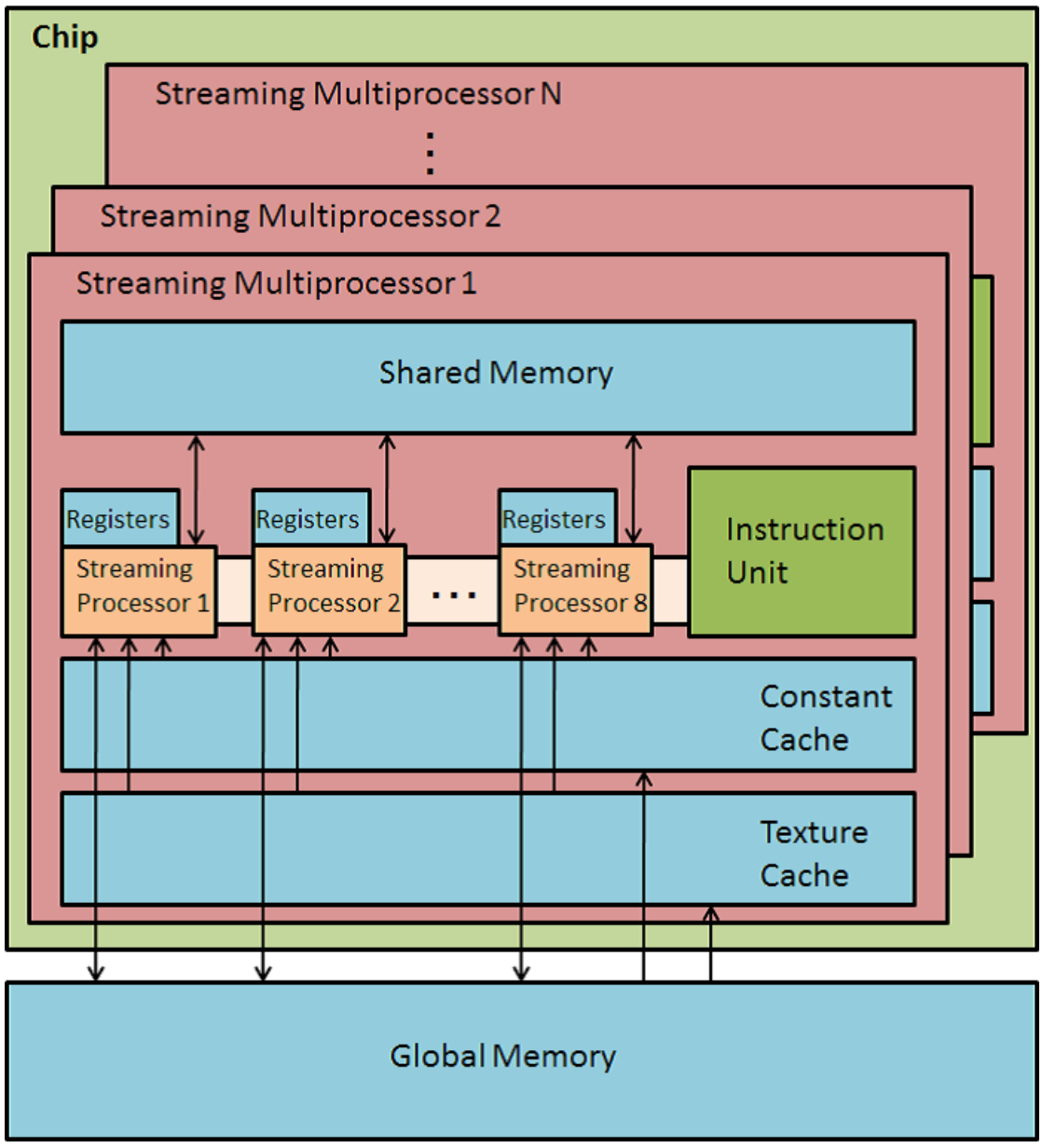

図 3.4　GPU のハードウェアモデル（出典：CUDA C Programming Guide）

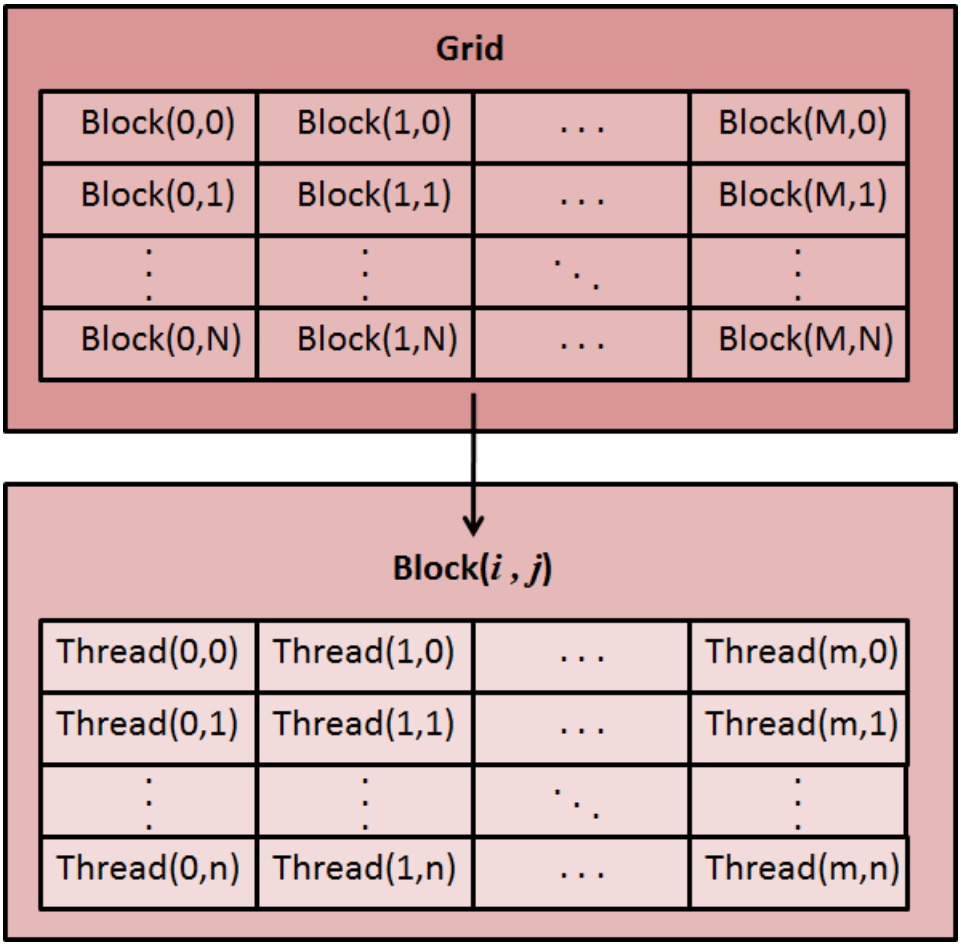

図 3.5 CUDA のスレッドの階層構造

3.3　GPU を用いた波動伝搬数値解析の高速化

本節では，FDTD 法，MM-MOC 法それぞれに対して GPU を用いて高速化した結果について述べる．初めに，モデルの簡単化のために二次元数値解析において最適化を施す．そして，その最適化を踏まえて更に三次元数値解析について最適化を行う．

以下，断りのない場合は単精度浮動小数点型の変数を用いて計算を行っている．また，比較のための CPU は CPU：Intel Core i7 920 2.67GHz を用いた．この CPU は 4 コアを有し，Hyper-Threading Technology を実装しており，最大で同時 8 スレッドの処理が可能である．以下の CPU の計算は全て OpenMP を用いた 8 スレッド並列化を行っている．

計算領域は正方形・立方体の領域として，吸収境界条件は除いた領域のみの評価を行っている．また，性能評価の指標として FLOPS(Floating point number Operations Per Second) と FUPS(Fields Update Per Second) を用いる．FLOPS は 1 秒間に浮動小数点演算が何回行われたか，FUPS は 1 秒間に何点の領域が更新されたかを表す．FDTD 法では FUPS と同様の指標として Cells/s を用いる場合もある．

3.3.1　GPU を用いた FDTD 法の高速化と評価

本論文では音場波動伝搬数値解析における FDTD 法を例に高速化と評価を行う．直交格子を用いた差分法であることは共通しているので，本論文の高速化手法は電磁界，弾性波ともに適用可能である．また，二次元における音場波動伝搬数値解析と電磁界波動伝搬数値解析は媒質の違いを考慮すればまったく同じように解くことができる．

FDTD 法は各グリッド上の値はそれぞれ独立して求めることができるため並列化可能である．解析領域のグリッド数と同数のスレッドを立ち上げて並列計算を行う．起動したブロックの実行順序は不定であり，1 時間ステップあたりの全ての計算を 1 つのカーネルに記述すると正常な計算結果は得られない．計算ステップごとのブロック間の同期をとるために，カーネルはある程度小分けにしなければならない．

GPU は GeForce GTX 285 及び GTX 580 を使用した．

二次元における高速化

表 3.1 に計算領域のグリッド数を 1024×1024，計算回数を 1024 と固定した時の FDTD 解析の計算時間を示す．

表 3.1 より，GTX 285 を用いた場合，8 スレッドの CPU と比較して FDTD 法は約 6 倍の高速化が実現出来ている．一方，GTX 580 を用いた場合，FDTD 法は約 12.7 倍の高速

表 3.1 FDTD 法の計算時間

計算環境	計算時間 [s]	GFLOPS	GFUPS
Core i7 920	3.05	4.226	0.352
GTX 285	0.51	25.265	2.105
GTX 580	0.24	53.687	4.474

化が実現出来ている．

共有メモリを用いた高速化

GPU は演算処理能力・メモリ帯域幅共に CPU の数倍の性能を持っているが，数値計算を行う場合，演算にかかる時間よりもグローバルメモリとのデータ転送にかかる時間のほうが遥かに大きい．シミュレーションの実行時間の大半はデータ転送に費やされているのである．データ転送を最小限に減らすことが GPU 計算の大幅な高速化に繋がる．

格子計算を行う場合，計算セルの値・その周辺の値が必要になる．1 ブロック内の全てのスレッドが計算セルの値を転送した場合，計算に必要な周辺の値は同ブロック内の他スレッドがすでに保持している場合がほとんどである．そのような場合でも，旧世代の GPU はキャッシュ機能を持たないため，グローバルメモリと通信を行ってしまう．このような無駄なメモリアクセスを減らすことが高速化には重要である．

しかし，スレッドが転送した値は SP 固有のレジスタに格納され，他スレッドがそれを直接参照することは不可能である．そこで，ブロック内のスレッドで値を共有するために共有メモリを使用する．共有メモリはバンクコンフリクトが起きなかった場合，レジスタと同等の速度でアクセスすることができる．グローバルメモリから転送した値を共有メモリに格納してブロックで共有することで，グローバルメモリへのアクセスを格段に減らすことが出来るので大幅な高速化が期待できる．

表 3.2 及び 3.3 に計算領域のグリッド数を 1024×1024，計算回数を 1024 と固定した時の FDTD 解析の計算時間を示す．この計算に使用したスレッドモデルは，1 ブロックあたり 128 個のスレッドが起動しており，x 方向に一列に並んでいる．

表 3.2 共有メモリを利用した計算時間 (GTX 285 の場合)

計算方法	計算時間 [s]	GFLOPS	GFUPS
FDTD 法	0.36	35.791	2.983

表 3.3　共有メモリを利用した計算時間 (GTX 580 の場合)

計算方法	計算時間 [s]	GFLOPS	GFUPS
FDTD 法	0.26	49.55 7	4.129

表 3.2 と表 3.1 を比べると，GTX 285 を用いた場合，高速化が達成出来ていることが分かる．共有メモリを利用することで，FDTD 法は約 1.4 倍高速化された．

一方，表 3.3 より，GTX 580 を用いた場合は，FDTD 法では共有メモリを用いない場合よりも計算速度が低下した．これは，共有メモリを用いない場合でもキャッシュが有効に働き，無駄なメモリアクセスが発生しなかったためであると考えられる．共有メモリを使うためのカーネル内での if 文によるオーバーヘッドが計算速度低下の原因であると思われる．

共有メモリを使用する場合，スレッドモデルの検討が重要となる．

スレッドモデルの検討

FDTD 法のように x, y 方向を同時に計算するような解析方法の場合，x 方向の共有・袖領域の重複，y 方向の共有・袖領域の重複のトレードオフを考えなければならない．

FDTD 法は粒子速度を計算するカーネル，音圧を計算するカーネルに分かれているが，どちらのカーネルも x, y 方向同時に計算するので，スレッドモデルの検討が重要になる．スレッドが x 方向に一列に長く並んだモデルは x 方向の袖領域の重複が少なくなり，無駄なデータ転送が減るが y 方向の値の共有ができない．一方，x 方向へのスレッドの長さを短くし，y 方向への厚みを持たせた正方形に近いようなモデルは y 方向の値の共有に対しては有効であるが，x 方向の袖領域の重複が多くなり，無駄なデータ転送が増えてしまう．

図 3.6 及び 3.7 にそれぞれのスレッドモデルの概略図を示す．

以上の点から，解析方法それぞれに適したスレッドモデルがあると考えられるので，FDTD 法に関してスレッドモデルの検討を行った．

表 3.4，3.5 及び 3.6 に計算領域のグリッド数を 1024×1024，計算回数を 1024 と固定した時の FDTD 解析の計算時間を示す．なお，GPU では 1 ワープ (32 スレッド) が同期して動作するため，1 ブロックで起動しているスレッドが 1 ワープに達していない場合，SP がスレッドを処理していない無駄な時間が発生してしまう．そのため，1 ブロックで最低 1 ワープが起動するようなスレッドモデルの検討のみを行った．表 3.4 の x は x 方向に並んだスレッドの数，y は y 方向に並んだスレッドの数である．また，同表中の数字の単位はすべて秒 [s] である．

表 3.4 FDTD 法のスレッドモデルの検討 (GTX 285 の場合)

y \ x	1	2	4	8	16	32	64	128	256	512
1						0.730	0.441	0.364	0.376	0.389
2					0.720	0.438	**0.355**	0.362	0.377	
4				0.787	0.441	0.356	0.361	0.369		
8			1.253	0.550	0.357	0.364	0.370			
16		2.746	1.095	0.508	0.369	0.385				
32	8.394	2.538	1.068	0.544	0.408					
64	8.478	2.573	1.233	0.599						
128	9.197	4.453	2.087							
256	14.931	7.448								
512	23.556									

表 3.5 FDTD 法のスレッドモデルの検討 (GTX 580 の場合)

y \ x	1	2	4	8	16	32	64	128	256	512	1024
1							0.392	0.265	**0.238**	0.250	0.347
2						0.409	0.275	0.244	0.250	0.336	
4					0.463	0.289	0.266	0.271	0.343		
8				0.567	0.331	0.255	0.278	0.436			
16			0.849	0.442	0.296	0.260	0.353				
32		1.417	0.792	0.447	0.324	0.363					
64	2.271	1.534	0.847	0.463	0.427						
128	2.769	1.581	0.896	0.646							
256	3.000	1.704	1.067								
512	3.447	2.062									
1024	3.840										

表 3.6 FDTD 法のスレッドモデルの検討 (GTX 580，共有メモリを使わない場合)

y \ x	1	2	4	8	16	32	64	128	256	512	1024
1							0.313	0.243	**0.236**	0.238	0.295
2						0.355	0.265	0.245	0.239	0.294	
4					0.393	0.268	0.278	0.265	0.322		
8				0.571	0.322	0.249	0.273	0.447			
16			1.001	0.527	0.311	0.246	0.327				
32		1.853	1.038	0.550	0.331	0.334					
64	3.796	2.107	1.139	0.550	0.389						
128	3.773	2.141	1.097	0.682							
256	4.113	2.253	1.230								
512	4.569	2.498									
1024	4.864										

表 3.4 より，GTX 285 を用いた場合は $(x, y) = (64, 2)$ のモデルの時が最速であった．これより，FDTD 法においてはスレッドを x 方向に長く並べ x 方向の袖領域の重複を削減し無駄なデータ転送を減らすのが最適であることがわかった．

表 3.5 及び 3.6 より，GTX 580 を用いた場合は $(x, y) = (256, 1)$ のモデルの時が最速であった．これにより，GTX 580 においても傾向は同様であり，スレッドを x 方向に長く並べるモデルが最適であることがわかった．

また，表 3.4 より x 方向のスレッドの数が 16 より小さくなると急激に計算が遅くなることがわかる．これは，グローバルメモリ上の配列が x 方向を基準に線形的に並んでいるため，グローバルメモリからのデータ転送がコアレスでなくなったからだと考えられる．SMP での演算は 1 ワープが同期して実行されるが，グローバルメモリへのアクセスはハーフワープ (16 スレッド) ごとにスケジュリーングされる．そして，16 スレッドが連続したメモリ番地へアクセスしたした際，メモリアクセスはコアレス化されバースト転送が行われ，16 スレッドが要求した 16 の要素が一回のアクセスで転送される．16 スレッドが不連続なメモリ番地へアクセスすると，メモリアクセスはコアレス化されずノンコアレスアクセスとなりバースト転送は行われない．すると，16 スレッドが要求した 16 の要素は順次転送されデータ転送が複数回行われる．GPU はグローバルメモルと太いメモリバスで繋がっているが，1 要素ごとに順次転送されるのでは太いバスが意味を為さない．GPU は CPU の数倍以上のメモリ帯域幅を有しているが，それはコアレスアクセスで太いメモリバスを有効利用した場合の数値であり，ノンコアレスアクセスで非効率にデータ転送を行うと CPU 以下の数値しか出せない場合がある．x 方向にスレッドを 1 個しか並べない場合，計算時間が CPU より遅くなってしまうのはそのためである．これは FDTD 法に限った話ではなく，どのような解析法にも適用される理論である．スレッドモデルを検討する際はコアレスアクセスを考慮することが重要である．

レジスタを用いたキャッシング

前項までは，1 スレッドが 1 セルを計算していた．しかし，三次元への拡張を考えた場合，これは非常に強い制限となる．そこで，1 スレッドが複数のセルを計算することを考える．

x 方向にスレッドを長く一列に並べることで x 方向の袖領域の重複を削減できる．そして，スレッドが計算セルを y 方向に動かしていくことで y 方向の要素の再利用を可能にし，y 方向の要素の共有の問題も解決できる．

ここで重要となるのは「1 スレッドが何セル計算するのか」ということである．1 スレッドが y 方向の境界から境界までを計算するのが要素の再利用という面では一番効率的で

ある．しかし，その場合は解析領域が x 方向に長くない限り起動するブロック数が少なくなってしまい，並列性が失われ計算速度が低下してしまう．要素の再利用の効率とブロック数のトレードオフを見極め，最適なスレッドモデルを決定することが重要となる．

FDTD 法に関してスレッドモデルを検討した結果，x 方向にスレッドを長く並べるモデルが最適であるとわかった．しかし，このモデルが y 方向の値の共有を犠牲にしていることは確かである．

そこで，1 スレッドが 1 セルを計算するモデルから離れ，1 スレッドが複数のセルを計算するモデルを考える．そうすることで，スレッドを x 方向に長く並べ x 方向の袖領域の重複を削減しながらも，計算セルを y 方向に動かすことで y 方向の要素を再利用することができるので共有性も高く保てる．1 スレッドが値を再利用しながら複数のセルを計算するので使用するハードウェアリソースも少量で済み，1 ブロック内で大量のスレッドが起動するわけではないので，CUDA や GPU ハードウェアの制限から開放される．そのため，さらに自由にスレッドモデルを考えることができる．

この計算モデルの場合，データの共有性と計算の並列性にトレードオフが発生するので，最適なスレッドモデルを検討することが重要となる．そこで，新計算モデルを用いた FDTD 法のスレッドモデルの検討を行った．

表 3.7 に計算領域のグリッド数を 1024×1024，計算回数を 1024 と固定した時の FDTD 解析の計算時間を示す．表 3.7 中の x は x 方向に並んだスレッドの数，y は y 方向に計算セルを何セル動かすかを表している．また，同表中の数字の単位はすべて秒 [s] である．

表 3.7 より，GTX 285 を用いた場合は $(x, y) = (256, 4)$ のモデルの時が最速であった．表 3.4 と比較して，要素の再利用により FDTD 法がさらに高速化されたことがわかる．

一方，表 3.8 より，GTX 580 を用いた場合は $(x, y) = (256, 1)$ のモデルの時が最速であった．

表 3.9, 3.26 及び 3.27 に計算領域のグリッド数を 1024×1024，計算回数を 1024 と固定した時の FDTD 解析の最も速かった計算時間を示す．

表 3.9 より最適化の結果，GTX 285 を用いた場合は 8 スレッドの CPU と比較して FDTD 法は約 9.3 倍の高速化が実現出来た．

一方，GTX 580 を用いた場合は 8 スレッドの CPU と比較して，FDTD 法は約 12.7 倍の高速化が実現出来た．

三次元への拡張

次に，FDTD 法を三次元へ拡張する．

共有メモリを使用せずに計算した場合の計算時間を表 3.10 に示す．解析領域を 256^3 グ

表 3.7　FDTD 法の新スレッドモデルの検討 (GTX 285 の場合)

y \ x	32	64	128	256	512
1	0.535	0.446	0.416	0.414	0.423
2	0.480	0.373	0.342	0.341	0.362
4	0.417	0.329	0.330	**0.329**	0.349
8	0.389	0.331	0.349	0.346	0.357
16	0.416	0.368	0.398	0.391	0.402
32	0.531	0.510	0.588	0.569	0.595
64	0.945	0.883	0.816	0.683	0.679
128	1.031	0.666	0.669	0.686	0.603
256	0.708	0.648	0.662	0.694	0.831

表 3.8　FDTD 法の新スレッドモデルの検討 (GTX 580 の場合)

y \ x	64	128	256	512	1024
1	0.278	0.235	**0.235**	0.236	0.285
2	0.262	0.235	0.236	0.237	0.295
4	0.259	0.238	0.248	0.249	0.278
8	0.253	0.247	0.251	0.252	0.273
16	0.255	0.252	0.255	0.254	0.265
32	0.255	0.252	0.258	0.264	0.264
64	0.257	0.261	0.260	0.247	0.263
128	0.255	0.255	0.261	0.276	0.410
256	0.361	0.371	0.399	0.470	0.698

表 3.9　FDTD 法の計算時間

計算環境	計算時間 [s]	GFLOPS	GFUPS
Core i7 920	3.05	4.226	0.352
GTX 285	0.33	39.164	3.264
GTX 580	0.24	53.687	4.473

リッドとして，吸収境界を除いて 1024 ステップ計算した．スレッドモデルは一次元モデルである．また，GPU は GeForce GTX TITAN を使用した．

表 3.10 共有メモリを利用しない計算時間

y \ x	32	64	128	256
1	8.361	5.108	4.086	4.117

共有メモリを使用して計算した場合の計算時間を表 3.11 に示す．

表 3.11 共有メモリを利用した計算時間

y \ x	32	64	128	256
1	9.786	5.637	4.284	4.376

表 3.11 と表 3.10 を比較すると，共有メモリを使用すると計算時間が遅くなったことが分かる．TITAN にはキャッシュが搭載されており，共有メモリを使用した効果が現れず，共有メモリを使用したオーバーヘッドがかかってしまったからである．

次に，スレッドモデルを二次元にして計算を行う．共有メモリを使用せずに二次元スレッドモデルを用いた場合の計算時間を表 3.12 に示す．

表 3.12 2 次元スレッドモデルを用いて共有メモリを利用しない計算時間

y \ x	1	2	4	8	16	32	64	128	256
1							5.101	4.089	4.121
2						5.129	4.092	4.13	4.137
4					6.112	4.106	4.127	4.137	4.267
8				9.323	5.38	4.142	4.142	4.266	
16				10.084	5.428	4.168	4.299		
32				10.083	5.429	4.293			
64				10.008	5.491				
128				10.193					

表 3.12 より，二次元スレッドモデルで計算した場合，一次元スレッドモデルで計算した場合よりも計算時間が遅くなることが分かる．

共有メモリを使用して二次元スレッドモデルを用いた場合の計算時間を表 3.13 に示す．

表 3.13　2 次元スレッドモデルを用いて共有メモリを利用した計算時間

y \ x	1	2	4	8	16	32	64	128	256
1							6.053	4.454	4.51
2						6.18	4.462	4.523	4.547
4					7.013	4.493	4.531	4.555	4.661
8				9.275	5.438	4.572	4.581	4.698	
16				9.25	5.507	4.601	4.731		
32				9.105	5.54	4.766			
64				9.32	5.948				
128				9.62					

表 3.13 より，共有メモリを使用すると計算時間が遅くなったことが分かる．これは，一次元スレッドモデルの場合と同様に TITAN にキャッシュが搭載されているためである．

次に，スレッドモデルを三次元にして計算を行う．共有メモリを使用せずに三次元スレッドモデルを用いた場合の計算時間を表 3.14 に示す．ここでは z 方向にスレッドを 64 スレッド並べたモデルを使用した．

表 3.14　3 次元スレッドモデルを用いて共有メモリを利用しない計算時間

y \ x	1	2	4	8	16
1	224.457	115.939	63.789	45.062	43.022
2	252.705	124.965	65.037	49.067	
4	278.907	142.151	94.896		
8	319.791	186.334			
16	252.007				

表 3.14 より，三次元スレッドモデルで計算した場合，一次元スレッドモデルで計算した場合よりも大幅に計算時間が遅くなることが分かる．これは，x 方向に並べられたスレッドが少ないため，コアレスアクセスが効果的に行われなかったためである．

共有メモリを使用して三次元スレッドモデルを用いた場合の計算時間を表 3.15 に示す．ここでは z 方向にスレッドを 64 スレッド並べたモデルを使用した．

表 3.15 より，共有メモリを使うことで計算時間は改善されたが，一次元スレッドモデルで計算した場合よりも大幅に計算時間が遅いことは変わらなかった．

以上より，三次元 FDTD 法においてもスレッドモデルは x 方向に長い一次元モデルが

表 3.15 3 次元スレッドモデルを用いて共有メモリを利用した計算時間

y \ x	1	2	4	8	16
1	176.492	83.702	45.979	33.929	37.994
2	162.526	77.898	44.608	39.059	
4	162.05	89.203	62.844		
8	193.376	113.877			
16	152.157				

最適であることが分かった,

次に，要素の再利用を用いた高速化を検討する.

表 3.16 にスレッドを x 方向に並べて y 方向に要素の再利用を適用した計算時間を示す.

表 3.16 y 方向のみに要素の再利用を用いた計算時間

y \ x	64	128	256
1	5.716	4.331	4.393
2	5.245	4.262	4.298
4	5.296	4.392	4.434
8	5.238	4.32	4.377
16	5.333	4.372	4.418
32	5.288	4.354	4.391
64	5.277	4.357	4.382
128	5.373	4.364	4.39
256	5.386	4.589	4.594

表 3.16 より，要素の再利用を適用することで計算時間が遅くなってしまったことが分かる．これは，TITAN にキャッシュが搭載されているためであると考えられる.

次に，y, z 方向に要素の再利用を適用する．表 3.17 に x 方向にスレッドを 128 スレッド並べた場合の計算時間を，表 3.18 に x 方向にスレッドを 256 スレッド並べた場合の計算時間を示す.

表 3.17 と表 3.18 より，要素の再利用を適用しても一次元スレッドモデルで共有メモリを使わない場合と比較してそれほど高速化されないことが分かった.

表 3.19 に三次元 FDTD 法の計算時間の CPU と GPU の比較を示す.

同表より，GPU を用いることで CPU で計算した場合に比べて，約 26 倍高速化される

表 3.17 y, z 方向に要素の再利用を用いた計算時間（x 方向にスレッドを 128 並べた場合）

z \ y	1	2	4	8
1	4.186	4.033	4.213	4.592
2	4.304	4.062	4.172	4.428
4	4.159	4.16	4.211	4.574
8	4.113	4.117	4.23	4.546
16	4.103	4.115	4.233	4.58
32	4.116	4.158	4.27	4.757
64	4.176	4.227	4.493	4.874
128	4.231	4.486	5.075	4.989
256	4.496	5.092	4.928	6.921

表 3.18 y, z 方向に要素の再利用を用いた計算時間（x 方向にスレッドを 256 並べた場合）

z \ y	1	2	4	8
1	4.248	4.088	4.262	4.871
2	4.351	4.111	4.247	4.677
4	4.197	4.21	4.278	4.819
8	4.156	4.167	4.289	4.8
16	4.145	4.167	4.294	4.857
32	4.161	4.21	4.331	4.937
64	4.224	4.283	4.549	5.041
128	4.281	4.538	5.169	5.141
256	4.551	5.155	5.129	7.234

表 3.19　最適化を施した計算時間

計算環境	計算時間 [s]	GFLOPS	GFUPS
Core i7 920	104.62	2.955	0.164
GTX TITAN	4.033	76.677	4.259

ことが分かった.

以上より，GeForce GTX 280 ではメモリアクセスを減らす最適化は効果があった．しかし，GeForce GTX 580, GeForce GTX TITAN のようにキャッシュが実装された最新のアーキテクチャが搭載された GPU を用いた場合，2 次精度 FDTD 法においてはメモリアクセスを減らす最適化がそれほど期待できないことが分かった．チューニングを行うことなく最高の結果を得られるため，プログラミングがより容易になったと言って良いだろう．しかし，アーキテクチャが更新されキャッシュの効果がそれほど得られなくなってしまった場合はメモリアクセスを減らす最適化は再び有効となるだろう.

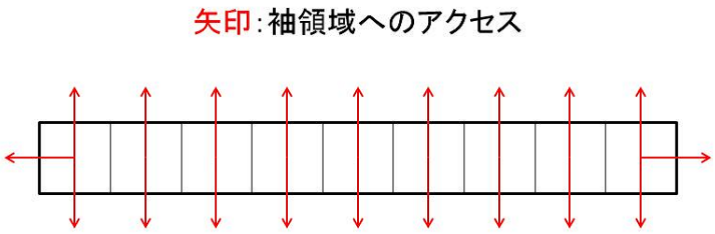

図 3.6　x 方向にスレッドを長く並べた場合

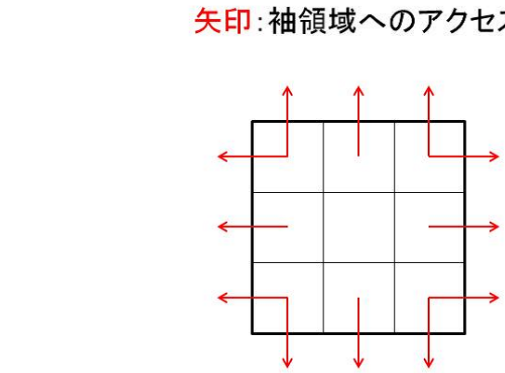

図 3.7　y 方向に厚みを持たせてスレッドを並べた場合

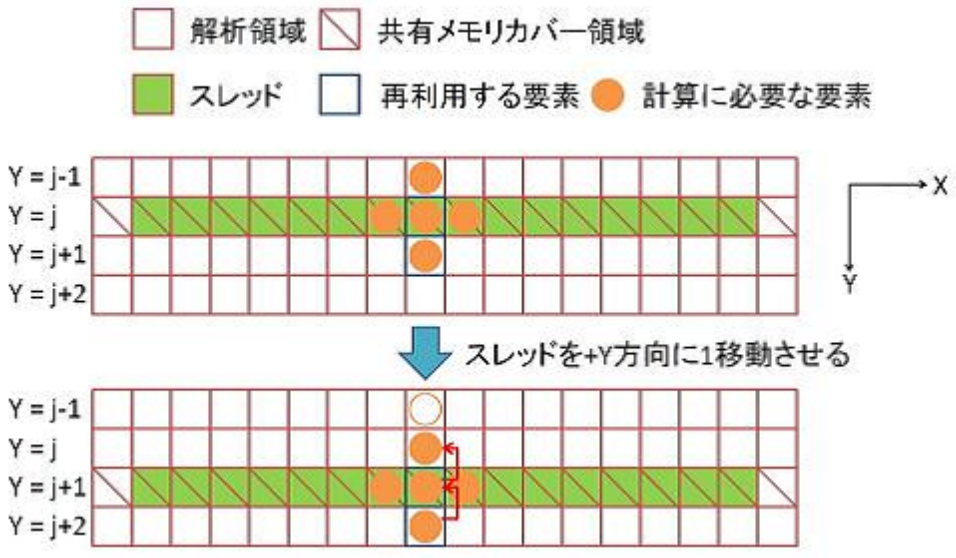

図 3.8　要素の再利用の概念図

3.3.2 GPU を用いた MM-MOC 法の高速化と評価

本論文では音場波動伝搬数値解析における MM-MOC 法の高速化と評価を行う．

MM-MOC 法では各グリッド上の値はそれぞれ独立して求めることができるため並列化可能である．解析領域のグリッド数と同数のスレッドを立ち上げて並列計算を行う．起動したブロックの実行順序は不定であり，1 時間ステップあたりの全ての計算を 1 つのカーネルに記述すると正常な計算結果は得られない．計算ステップごとのブロック間の同期をとるために，カーネルはある程度小分けにしなければならない．

GPU は GeForce GTX 285 及び GTX 580 を使用した．

二次元における高速化

表 3.20 及び 3.21 に計算領域のグリッド数を 1024 × 1024，計算回数を 1024 と固定した時の MM-MOC 解析の計算時間を示す．

表 3.20 MM(3, 1)-MOC 法の計算時間

計算環境	計算時間 [s]	GFLOPS	GFUPS
Core i7 920	58.63	2.784	0.018
GTX 285	13.37	12.207	0.080
GTX 580	1.74	88.861	0.617

表 3.21 MM(7, 1)-MOC 法の計算時間

計算環境	計算時間 [s]	GFLOPS	GFUPS
Core i7 920	101.00	3.145	0.011
GTX 285	23.91	13.292	0.045
GTX 580	2.26	129.229	0.475

表 3.20 及び 3.21 より，GTX 285 を用いた場合，8 スレッドの CPU と比較して MM(3, 1)-MOC 法は約 4.4 倍，MM(7, 1)-MOC 法では約 4.2 倍の高速化が実現出来ている．一方，GTX 580 を用いた場合，MM(3, 1)-MOC 法は約 34.2 倍，MM(7, 1)-MOC 法では約 43.2 倍の高速化が実現出来ている．

共有メモリを用いた高速化

次に，共有メモリを使用した高速化を検証する．表 3.22 及び 3.23 に計算領域のグリッド数を 1024 × 1024，計算回数を 1024 と固定した時の MM-MOC 解析の計算時間を示す．この計算に使用したスレッドモデルは，1 ブロックあたり 128 個のスレッドが起動しており，x 方向に一列に並んでいる．

表 3.22　共有メモリを利用した計算時間 (GTX 285 の場合)

計算方法	計算時間 [s]	GFLOPS	GFUPS
MM(3, 1)-MOC 法	12.68	12.871	0.085
MM(7, 1)-MOC 法	21.68	14.660	0.050

表 3.23　共有メモリを利用した計算時間 (GTX 580 の場合)

計算方法	計算時間 [s]	GFLOPS	GFUPS
MM(3, 1)-MOC 法	1.899	81.42 1	0.565
MM(7, 1)-MOC 法	2.152	135.714	0.498

表 3.22 と表 3.20 及び 3.21 を比べると，GTX 285 を用いた場合，全ての手法において高速化が達成出来ていることが分かる．共有メモリを利用することで，MM(3, 1)-MOC 法は約 1.05 倍，MM(7, 1)-MOC 法法は約 1.1 倍高速化された．

一方，表 3.23 より，GTX 580 を用いた場合は，MM(3, 1)-MOC 法では共有メモリを用いない場合よりも計算速度が低下した．これは，共有メモリを用いない場合でもキャッシュが有効に働き，無駄なメモリアクセスが発生しなかったためであると考えられる．共有メモリを使うためのカーネル内での if 文によるオーバーヘッドが計算速度低下の原因であると思われる．

共有メモリを使用する場合，スレッドモデルの検討が重要となる．

MM-MOC 法は x 方向の移流のカーネルと y 方向の移流のカーネルが分離しているので，それぞれの方向に最大限スレッドを並べればそれが最良のモデルとなる．

コンスタントメモリを用いた高速化

解析領域の全セルで媒質密度 ρ と体積弾性率 K，セルサイズ $\Delta x, \Delta y$ が一定で，時間ステップ Δt も不変の場合，MM-MOC 法の移流計算に用いる畳込み係数は全領域，全ステッ

プで一定となる．このような場合，コンスタントメモリを使用すると高速化できる場合がある．

コンスタントメモリはグローバルメモリ上に配置され，初回のデータ転送にはグローバルメモリからのデータ転送と同等の時間がかかる．しかし，一度チップに転送された値はチップ上のコンスタントキャッシュにキャッシングされ，次回利用時からはレジスタと同等の速度でアクセスできる．MM-MOC 法のように畳込み係数が多数ある場合，コンスタントメモリの利用は有効であると言えよう．コンスタントメモリを利用した最適化は FDTD 法には適用せず，MM-MOC 法のみに適用した．

表 3.24 に計算領域のグリッド数を 1024 × 1024，計算回数を 1024 と固定した時の MM-MOC 解析の計算時間を示す．なお，共有メモリを用いた最適化は適用済みである．

表 3.24　コンスタントメモリを利用した計算時間 (GTX 285 の場合)

計算方法	計算時間 [s]	GFLOPS	GFUPS
MM(3, 1)-MOC 法	3.05	53.511	0.352
MM(7, 1)-MOC 法	3.28	96.899	0.327

表 3.25　コンスタントメモリを利用した計算時間 (GTX 580 の場合)

計算方法	計算時間 [s]	GFLOPS	GFUPS
MM(3, 1)-MOC 法	1.81	85.424	0.593
MM(7, 1)-MOC 法	1.90	153.714	0.565

表 3.24 と表 3.2 を比較すると，GTX 285 を用いた場合，コンスタントメモリを利用することで MM-MOC 法が格段に高速化されたことが分かる．今回は，MM-MOC 法の移流に必要な畳込み係数をコンスタントメモリに配置した．このことから，コンスタントメモリを利用しない従来の実装の場合，グローバルメモリに配置された畳込み係数の転送が計算時間の大部分を占めていたことが分かった．コンスタントメモリに畳込み係数を配置したことで，コンスタントキャッシュが有効に働き，データ転送を大幅に削減できたことが大幅な高速化に繋がった．

一方，表 3.25 より，GTX 580 を用いた場合はコンスタントメモリを用いても大きな計算速度向上は見られなかった．これは，畳込み係数の読み込みにおいてキャッシュが有効に働いたため，コンスタントメモリの効果が現れなかったものと思われる．

FDTD 法と同様にして MM-MOC 法にも要素の再利用を適用して最適化を行ったので，最速となったものを最後にまとめる．

表 3.26 及び 3.27 に計算領域のグリッド数を 1024 × 1024，計算回数を 1024 と固定した時の MM-MOC 解析の計算時間を示す.

表 3.26　MM(3, 1)-MOC 法の計算時間

計算環境	計算時間 [s]	GFLOPS	GFUPS
Core i7 920	58.63	2.784	0.018
GTX 285	2.58	63.210	0.416
GTX 580	1.28	120.795	0.838

表 3.27　MM(7, 1)-MOC 法の計算時間

計算環境	計算時間 [s]	GFLOPS	GFUPS
Core i7 920	101.00	3.145	0.011
GTX 285	2.99	106.226	0.359
GTX 580	1.36	214.748	0.789

表 3.26 及び 3.27 より, 最適化の結果 GTX 285 を用いた場合は 8 スレッドの CPU と比較して MM(3, 1)-MOC 法は約 22.7 倍，MM(7, 1)-MOC 法では約 33.8 倍の高速化が実現出来た.

一方，GTX 580 を用いた場合は 8 スレッドの CPU と比較して，MM(3, 1)-MOC 法は約 45.8 倍，MM(7, 1)-MOC 法では約 74.3 倍の高速化が実現出来た.

三次元への拡張

MM-MOC 法の三次元への拡張は FDTD 法よりも容易である．元々MM-MOC 法は方向分離で解析していたので，同様のアルゴリズムを z 方向にも適用するだけである.

表 3.28 に解析領域を 256^3 グリッド，吸収境界を除いて 1024 ステップ計算した時の MM(3, 1)-MOC 法の計算時間を示す.

表 3.28　MM(3, 1)-MOC 法の計算時間

計算環境	計算時間 [s]	GFLOPS	GFUPS
Core i7 920	3775.05	1.966	0.00455
GTX TITAN	34.74	213.642	0.495

同表より，GPU を用いることで CPU で計算した場合に比べて，約 108 倍高速化されることが分かった.

次に，表 3.29 に解析領域を 256^3 グリッド，吸収境界を除いて 1024 ステップ計算した時の MM(7, 1)-MOC 法の計算時間を示す．

表 3.29 MM(7, 1)-MOC 法の計算時間

計算環境	計算時間 [s]	GFLOPS	GFUPS
Core i7 920	6035.89	2.323	0.00285
GTX TITAN	36.48	384.266	0.471

同表より，GPU を用いることで CPU で計算した場合に比べて，約 165 倍高速化されることが分かった．

図 3.9 に FDTD 法と MM-MOC 法の 1 タイムステップあたりの計算時間をまとめたものを示す．同図より，計算領域が大きくなると MM-MOC 法ではインタラクティブシミュレーションの目安となる 33/66ms を超えてしまうことが分かる．この結果より，三次元波動伝搬数値解析で MM-MOC 法を用いることは難しいと考えられる．

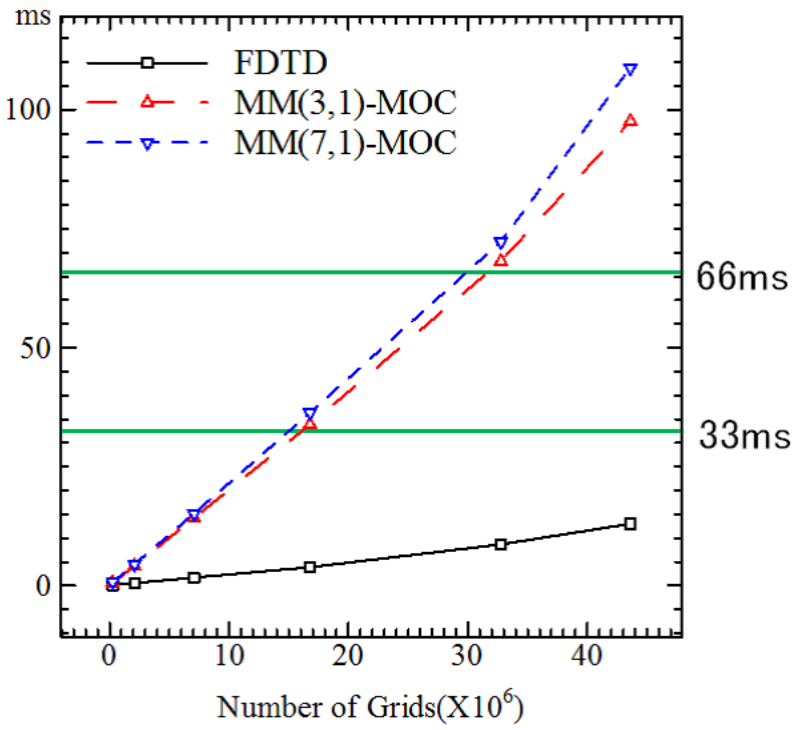

図 3.9 1 タイムステップあたりの計算時間

3.4　マルチ GPU を用いた数値解析の更なる高速化について

3.4.1　シングル GPU からマルチ GPU へ

GPU を用いることで波動数値解析が大きく高速化されることが分かった．しかし，ビデオカードに搭載されている VRAM の量は有限であり，スーパーコンピュータ等と比較すると極僅かである．大規模領域での計算ができなくてはいくら高速化されても無意味となってしまうだろう．大規模領域を計算する方法として，RAM と VRAM でデータを相互に転送して GPU で計算するデータをその都度入れ替えるという方法が考えられるが，PCI Express バスの帯域幅を考えると現実的ではない．データの転送に大きく時間がかかってしまい，計算が高速化された効果が消えてしまうからである．

GPU の高速化を活かしたままで大規模領域を計算するにはどのようにすればよいのか．その答えが複数の GPU を並列に動作させ計算させるマルチ GPU である．パソコンのマザーボードには PCI Express バスのスロットが複数搭載されているものがあり，大型のビデオカードを最大で 4 基搭載できる．パソコンに GPU を追加するだけで更なる高速化が期待できる．

3.4.2　数値解析をマルチ GPU 化する手法

パソコンに GPU を複数載せれば自動でプログラムがマルチ GPU 化されるわけではない．プログラムをマルチ GPU に対応させなければならない．

対応させるためにしなければならないことは様々あるが，まず初めに複数の GPU を並列に動かすために CPU 側のスレッドをマルチスレッドにする必要がある．本研究では，マルチスレッドのための並列化ライブラリとして OpenMP を使用した．OpenMP を使用すると，ディレクティブを挿入するだけで簡単に並列化ができる．CPU 側をマルチスレッドにすると複数の GPU と一対一で対応づけることができるので，マルチ GPU を並列に動作させることが出来るようになる．

次に，マルチ GPU で解析する方法について述べる．VRAM はビデオカード毎に独立しており，一つの大容量メモリとして使用することはできない．そこで，解析空間を分割して GPU 毎に割り当てて計算することでマルチ GPU 解析が可能となる．しかし，領域を分割することで新たな問題が発生する．差分法等の隣接格子点の値を参照して計算する手法の場合，分割領域の境界の格子点では違う VRAM 上に存在する隣接格子点の値を参照しなければならない．この問題を解決するためには配列を余分に確保し，境界上の格子点の値を 1 タイムステップ毎に GPU 間で相互に交換する必要がある．図 3.10 に解析空間

を分割するイメージを示す.

解析空間が小規模の場合は GPU 間のデータ通信のコストが計算時間に対して大きいため，マルチ GPU の効果はあまり大きくない．しかし，解析空間が大規模になればその分 GPU 間のデータ通信のコストが計算時間に対して小さくなるため，マルチ GPU による高速化の効果は大きくなる．マルチ GPU は大規模領域での数値解析でこそ真価を発揮すると言ってよいだろう.

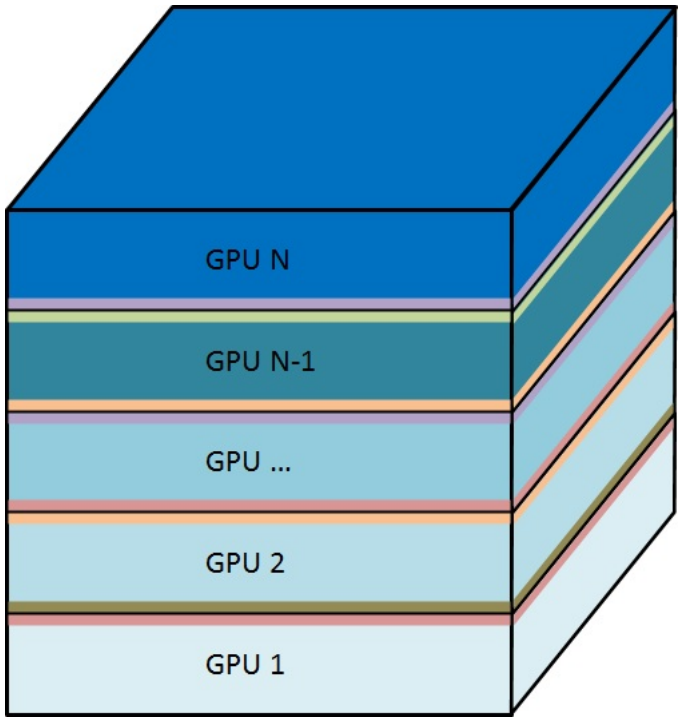

図 3.10 解析空間を z 軸に垂直に分割

3.4.3 マルチ GPU 化による高速化の結果

図 3.11 に FDTD 法による三次元音場数値解析の 1GPU と 2GPU を用いた場合の領域サイズに対する計算時間の変化を示す．GPU は GeForce GTX 580 を用いた.

この図より，領域サイズが小さい場合は 1GPU の方が 2GPU よりもむしろ速く，領域サイズが大きくなるにつれて 2GPU での計算が 1GPU に比べて高速になっていることが分かる.

これより，マルチ GPU 解析は領域サイズが大きい問題で有効であることが証明された.

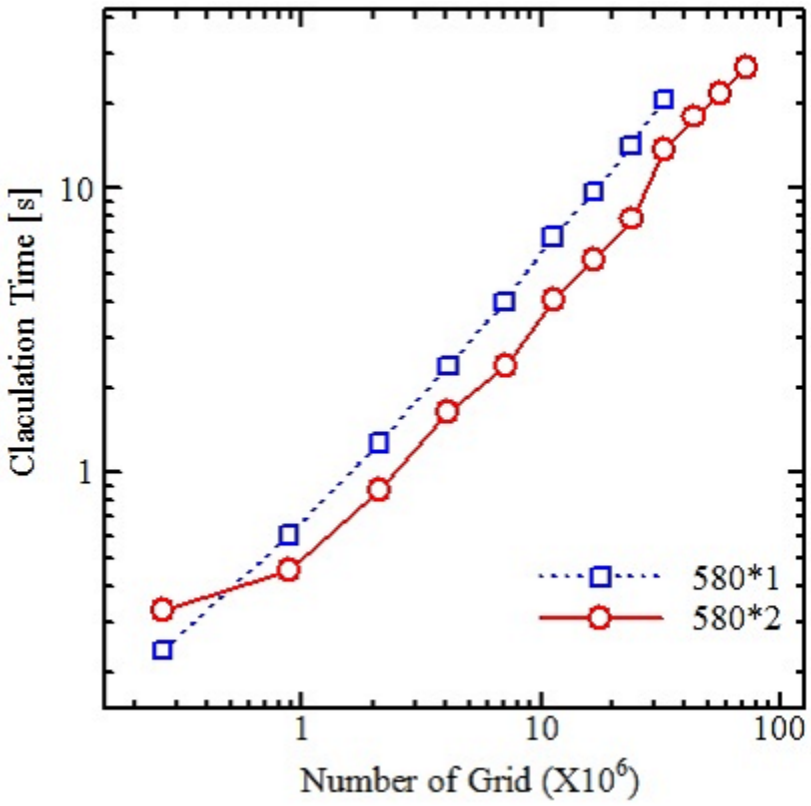

図 3.11　マルチ GPU 計算の領域サイズに対する計算時間の変化

3.5　結言

本章では，波動伝搬数値解析の GPU への実装について述べた．GPU への基礎的な実装からアーキテクチャを意識した最適化までを実装し，その有効性を検証した．その結果，最適化を施すことで GPU 計算が更に高速化されることが分かった．最終的に，数十倍から百数十倍の高速化を達成することができた．さらに，マルチ GPU を用いることでより高速に，より大規模な領域を計算できることを示した．

第4章　3次元波動数値解析のための可視化手法の検討と評価

4.1　序言

本章では，3次元波動伝搬数値解析の可視化手法について述べる．初めに，従来用いられてきた可視化手法について説明を行う．そして，既存手法の特徴と欠点を踏まえた提案手法である PMCC (Permeable Multi Cross-section Contours) について述べ，既存手法との比較を行う．次に，PMCC を用いて波動伝搬数値解析をリアルタイム可視化した場合の可視化速度について検討する．最後に，マルチ GPU を用いて可視化したときの高速化について説明する．

4.2　既存の可視化手法についての考察

本節では，従来用いられてきた可視化手法について述べる．

4.2.1　Contour & Surface Plot による可視化法

従来，三次元空間の解析結果を可視化する場合，初歩的な手法として三次元空間のある面のみを取り出し表示する方法が挙げられる．

Contour は一番シンプルな手法で，取り出した断面をそのまま二次元的に表示する方法である．Contour で可視化した様子を図 4.1 に示す．

加えて，取り出した断面に対して z 軸方向に音圧などの強度を割り当て，Contour よりも視認性を向上させた Surface Plot がある．Surface Plot で可視化した様子を図 4.2 に示す．

これらの表示方法は描画のための計算負荷や転送負荷が小さく，ある特定の面のみが見たい場合には有効であると言えるかもしれない．

しかし，三次元空間を解析したにもかかわらず，二次元の情報しか表示できていないため，三次元空間での波動の広がりを理解するには効果的な方法とは言えない．

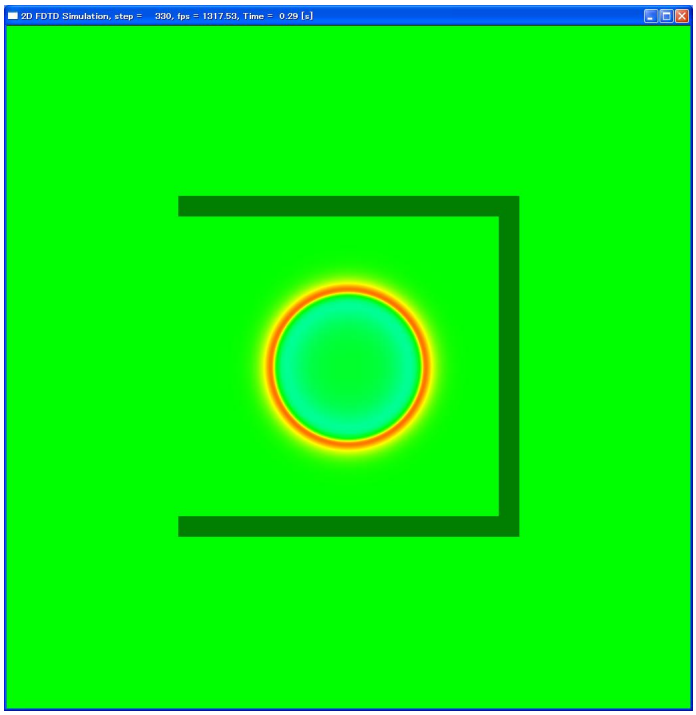

図 4.1　Contour による可視化

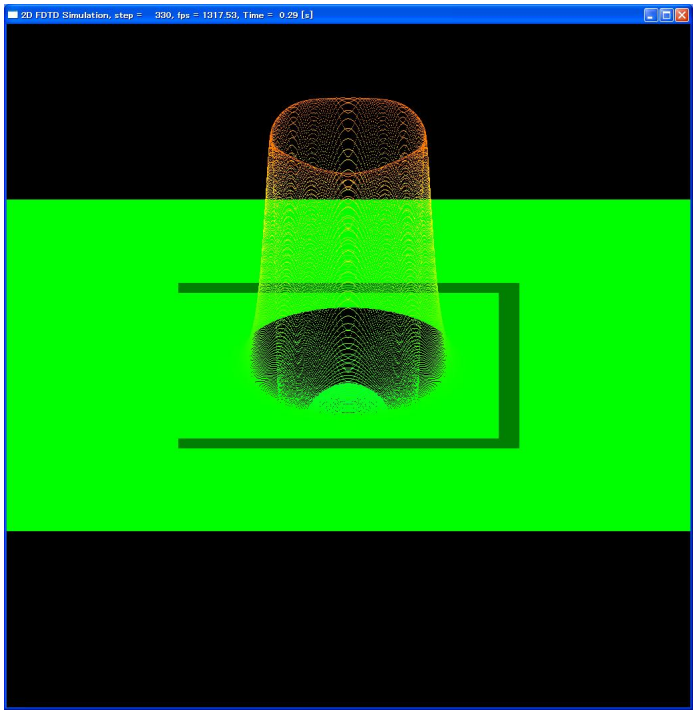

図 4.2　Surface Plot による可視化

4.2.2 Multi Cross-section Contours による可視化法

Contour の発展として，三次元空間の複数の断面を同時に表示する方法がある．ここでは，この方法を Multi Cross-section Contours (MCC) と呼ぶ．この方法は，三次元の情報のうち，見たい範囲を中心に複数の断面を同時に見る事ができる点で前節の方法よりも三次元での波動の広がりを理解しやすい．MCC で可視化した様子を図 4.3 及び 4.4 に示す．

しかし，平行な複数の面を表示する場合には，オクルージョンの影響で効果が半減してしまう．

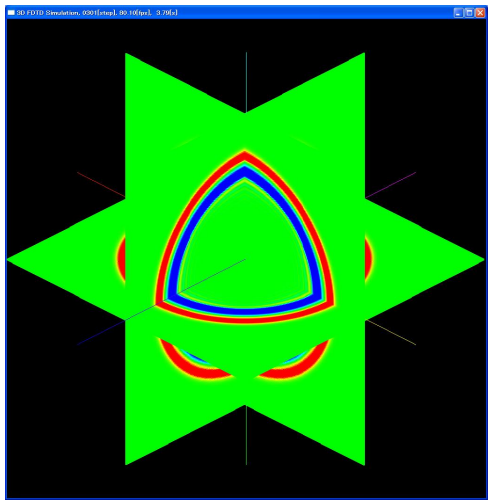

図 4.3 Multi Cross-section Contours による可視化

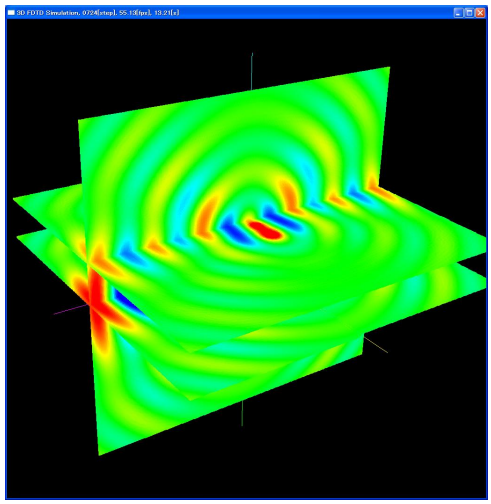

図 4.4 Multi Cross-section Contours による可視化 (平行な面を表示した場合)

4.2.3　ボリュームレンダリングによる可視化法

近年では，不透過度を利用するボリュームレンダリング (Volume Rendering, VR) という手法が提案されている．不透過度とは，ポリゴンがどれだけ透明化を表す値で，不透過度の値が小さければポリゴンは透明に近づく．指定した閾値以下の不透過度を持つポリゴンを描画しないように設定することも可能である．三次元波動数値解析の可視化にこのボリュームレンダリングを利用する場合，三次元空間の全領域における各ボクセルに対して，音圧値などの強度に合わせた不透過度を設定し，それをある視点からのパス上で積分することで三次元の空間を表す方法である．不透過度の値が小さい箇所を透過または非表示にすることで空間全体を可視化したとしても強度が強く不透過度の値が大きい部分のみ可視化することができる．ボリュームレンダリングで可視化した様子を図 4.5，4.6 及び 4.7 に示す．この方法は，近年，医療画像の表示や流体力学分野などで利用されることがある．特徴として，三次元解析空間中の全ての情報を利用しており，三次元空間は一気に表示できる点でとても優れている．しかし，解析空間中に壁などの障害物があったり，閉空間を解析したりと複雑な干渉・散乱を繰り返す波動伝搬を可視化すると，視点によっては描画された結果が複雑すぎて，かえって理解ができないという場合がある．また，波動現象そのものは，もともと煙などのようにまとまった動きをすることは少なく，干渉・散乱を繰り返すことを考えると，ボリュームレンダリングを波動伝搬シミュレーションの可視化に利用し，可視化の効果を明確にするには，かなりのノウハウが必要となるだろう．

加えて，この方法は三次元空間の各ボクセルの全ての情報を処理するため，描画のために多くの計算負荷がかかるという欠点もある．

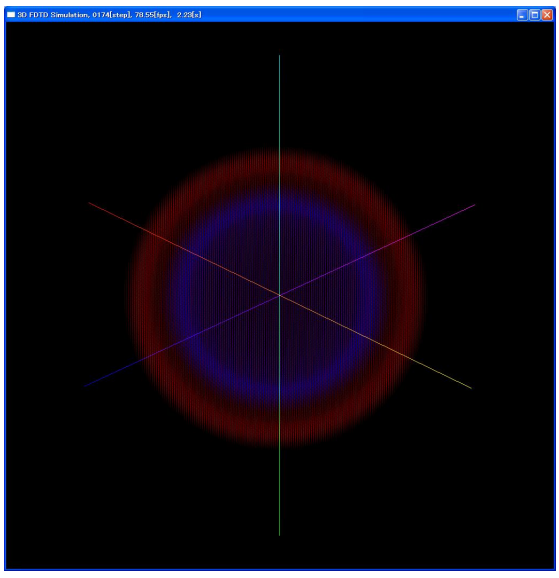

図 4.5　ボリュームレンダリングによる可視化

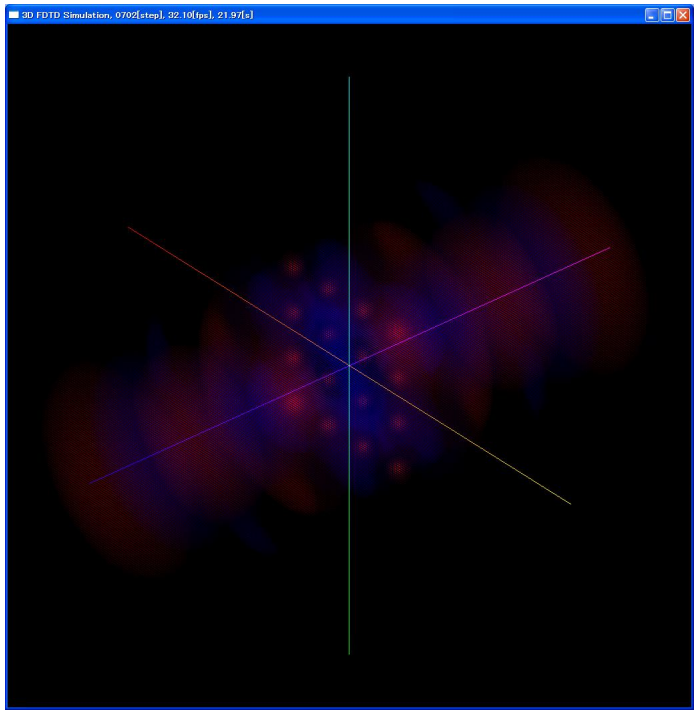

図 4.6　ボリュームレンダリングによる可視化 (音源をアレイにした場合)

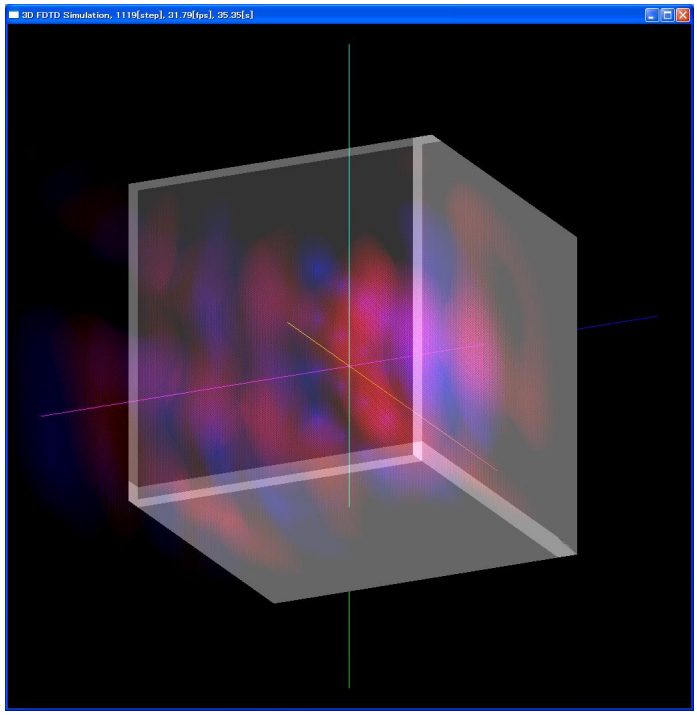

図 4.7　ボリュームレンダリングによる可視化 (壁を設置した場合)

4.3 PMCC (Permeable Multi Cross-section Contours) を用いた 3 次元波動伝搬数値解析の可視化

そこで，これらの問題点を解決するための方法として PMCC (Permeable Multi Cross-section Contours) を提案する．

4.3.1 PMCC の概要について

この PMCC は一言で言えば，MCC において表示する複数の断面に対して，表示する強度に合わせて不透過度を設定する手法である．描画の手順をまとめると以下のようになる．

(1) 三次元空間のある断面に対して，音圧値などをカラー表示する．

(2) 更に，音圧値の強度に合わせて不透過度（α 値）を設定する（音圧値が大きい場合に不透過度は 1 に近づく）．

(3) 不透過度（α 値）の与え方は計算対象によって変動させることができるので，表示させる強度の下限や強度と α 値の関係式などを自由に設定できる．

(4) 同様の手順で表示させたい断面を複数描画する．

この方法の特徴としては

・複数の断面を表示しても，音圧値が小さい点は不透過度が小さくなるため，オクルージョンが発生しにくく，ある断面の裏に隠れてしまっていた情報も見ることができる．

・したがって，平行な複数の断面も同時に表示可能である．

・断面の回転及び移動も可能である．

・断面のみの不透過度を利用しているため，ボリュームレンダリングに比べて，描画のための計算負荷が少ない．

・断面表示を基にしているため，複雑な波動伝搬現象も把握しやすい．

ということが挙げられる．アニメーションにより視点を変更することで，より三次元空間を把握することが容易になる．PMCC で可視化した様子を図 4.8，4.9，4.10，4.11 及び 4.12 に示す．

しかし，PMCC にもデメリットは存在する．複雑な形状をした音源やアンテナ近傍の波動の様子を可視化することが難しいという点である．PMCC は断面で表示を行うため，複雑な物体の内部および近傍の様子は面のみではうまく可視化できない場合がある．反対に，物体から離れた遠方解の様子については PMCC で分かり易く可視化できる．近傍解の様子についてはボリュームレンダリングを用いて可視化を行った方が分かり易い場合があり，適宜適した手法を用いることが重要である．

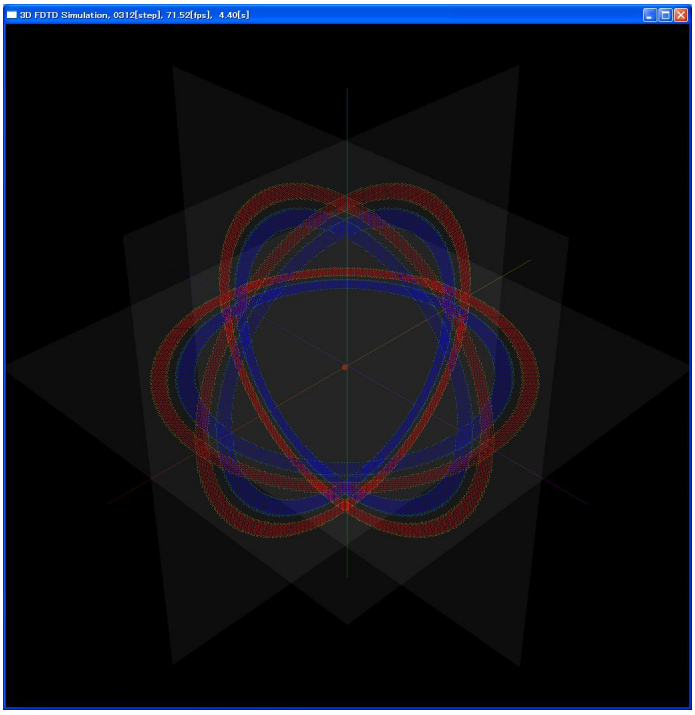

図 4.8 PMCC による可視化

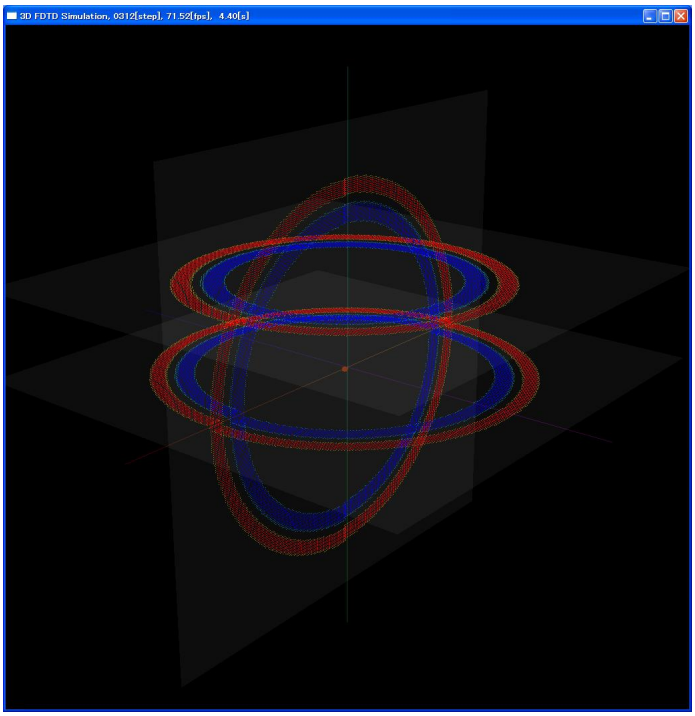

図 4.9 PMCC による可視化 (平行な面を表示した場合)

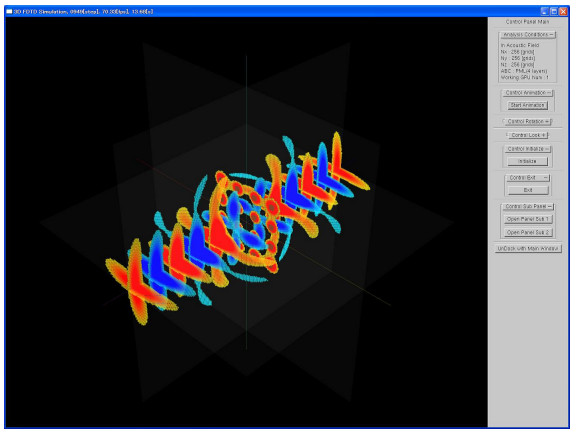

図 4.10 PMCC による可視化 (音源をアレイにした場合)

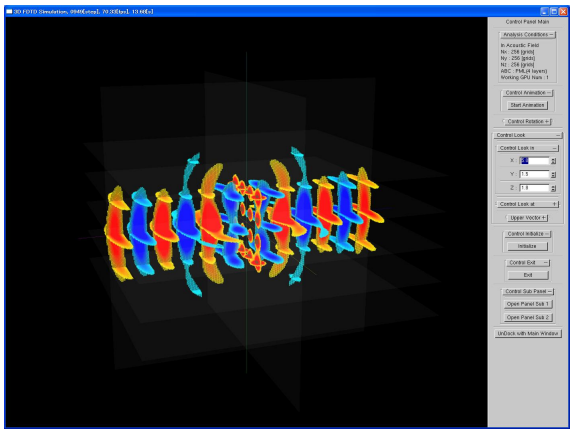

図 4.11 PMCC による可視化 (zx 平面を複数表示した場合)

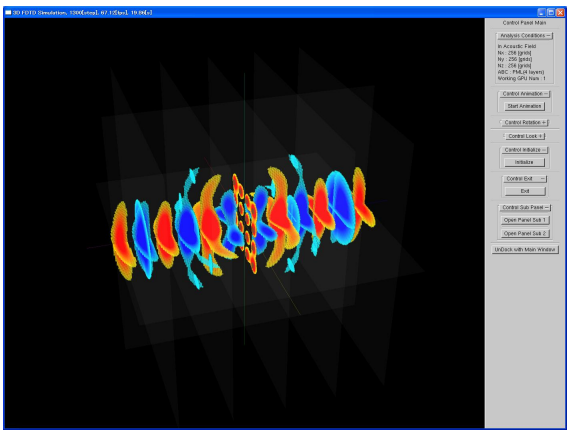

図 4.12 PMCC による可視化 (xy 平面を複数表示した場合)

4.3.2 PMCC と既存の可視化手法との比較

前節で提案した PMCC と従来法とを比較し，検討を行う．

まず初めに，描画に必要なデータ量を比較する．三次元解析空間の x 方向の格子点数を N_x，y 方向の格子点数を N_y，z 方向の格子点数を N_z とする．

ボクセル一つにつき，座標の情報として (x, y, z, w) の四要素，カラー情報として (R, G, B, α) の四要素，合計で八要素が必要となる．データは float 型で一つの要素が 4Byte のデータ量を持つので，一つのボクセルが 32Byte のデータを持つ．

ボリュームレンダリングの場合は，解析空間中の全ての格子点を描画用データとして扱うので，必要なデータ量は $N_x \times N_y \times N_z \times 32\text{Byte}$ となる．

PMCC の場合は，xy 平面，yz 平面及び zx 平面それぞれの断面を一つずつ表示した場合，必要なデータ量は $(N_x \times N_y + N_y \times N_z + N_z \times N_x) \times 32\text{Byte}$ となる．

ここで，$N_x = N_y = N_z = N$ とした場合，必要なデータ量は以下のようになる．

表 4.1 必要なデータ量

	データ量 [Byte]
二次元表示	$32N^2$
ボリュームレンダリング	$32N^3$
MCC, PMCC	$96N^2$

ここで，$N = 256$ とすると，ボリュームレンダリングの場合は 512MByte ものデータ量になるが，PMCC の場合は 6MByte で済む．

従来であれば，GPU 上の VRAM をどれだけ使用しても数値解析に不便はなかった．しかし，次章で述べる GPU を用いた高速可視化の場合には，とても重要な問題となる．なぜならば，GPU を用いて数値解析を行う場合，数値解析に用いるデータも可視化に用いるデータも全て VRAM 上に確保するからだ．可視化に用いるデータ量が大きければ，VRAM を圧迫し数値解析用のデータを十分に確保できず，大規模な空間を解析することができなくなる．可視化に用いるデータ量が小さいほど，GPU を用いた可視化に有効な方法であると言える．

4.4 PMCC を用いた可視化の可視化速度の評価

次に，可視化した際の描画速度についての比較を行う．描画速度に関しての評価は，フレームレートを用いて行う．図 4.13 に示すように，フレームレートは 1 秒間に何フレーム画面が更新されたかを表す．リアルタイムな可視化においては，1 ステップあたりの計算が高速であればあるほど，また，描画負荷が小さければ小さいほどフレームレートは上昇する．このフレームレートを指標とし，可視化手法の比較を行う．フレームレートの基準として，インタラクティブシミュレーションを行うためには最低でも 15fps 以上，理想として 30fps 程度が必要であると考える．インタラクティブシミュレーションには滑らかな可視化が必要であり，fps が下がるとインタラクティブシミュレーションとしての操作性に支障をきたす．波動伝搬数値解析のソルバと解析結果の可視化を 1 ステップとして考えると，30fps を達成するには 1 ステップが 33ms 以内に実行できる必要がある．インタラクティブシミュレーションの実現のためには可視化の部分をどれだけ高速に行えるかが重要となる．

FDTD 法を用いて解析を行い，PMCC を使用して可視化した際のフレームレートを図 4.14 に示す．

FDTD 法を用いて解析を行い，ボリュームレンダリングを使用して可視化した際のフレームレートを図 4.15 に示す．

図 4.14 と図 4.15 より，吸収境界条件として 4 層 PML を，デバイスとして GeForce GTX 580 を使用した際のフレームレートを比較したものを図 4.16 に示す．

同図より，PMCC がボリュームレンダリングに比べて 2 倍の速度で可視化を行えることが分かる．

以上より，提案手法である PMCC が視認性と描画速度を両立した可視化手法であることが分かった．

従来，CPU で計算を行いそのまま可視化を行う場合，計算結果が格納されている RAM から描画用データを VRAM に転送してやる必要があった．この転送は PCI Express Bus を通して行われるが，このバスは帯域幅がそれほど広くないのでデータの転送量によってはボトルネックになる場合がある．

しかし，GPU で計算を行う場合，計算結果は VRAM に格納されており，PCI Express Bus を通した転送は必要ない．データ転送はグラフィックボード上で完結しており，なおかつ GPU の広い帯域を用いてデータ転送が行えるため非常に高速である．CPU で計算する場合のデータ転送の概略図を図 4.17 に，GPU で計算する場合のデータ転送の概略図を図 4.18 に示す．

CPU で計算して可視化した場合と GPU で計算して可視化した場合のフレームレートの比較を表 4.2 に示す.

表 4.2 フレームレート (CPU と GPU の比較)

計算環境	フレームレート [fps]
Core i7 930	2.3
GeForce GTX 580M	36.2
GeForce GTX 580	69.1

同表より，GPU で計算・可視化した場合，CPU の場合に比べて約 30 倍の速度で可視化を行えることが分かる.

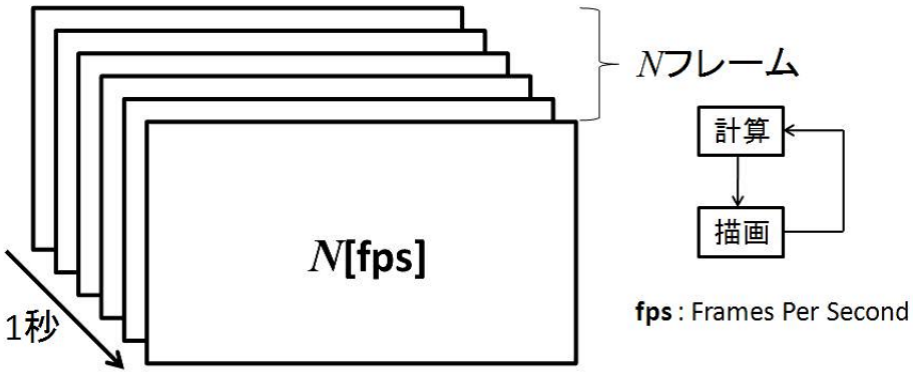

図 4.13 フレームレートの考え方

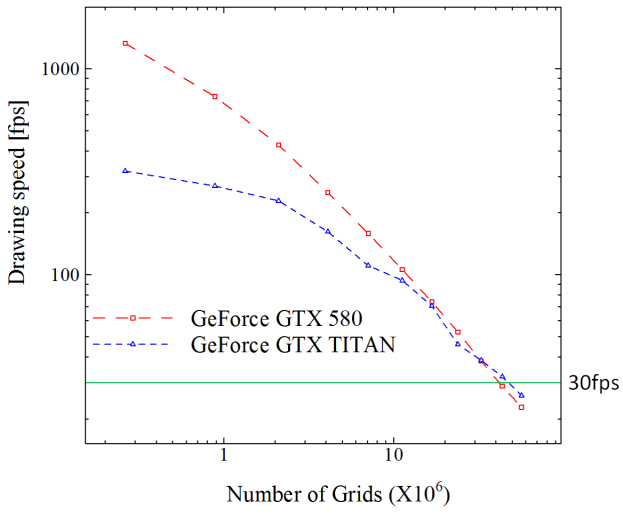

図 4.14 PMCC のグリッド数に対するフレームレートの変化

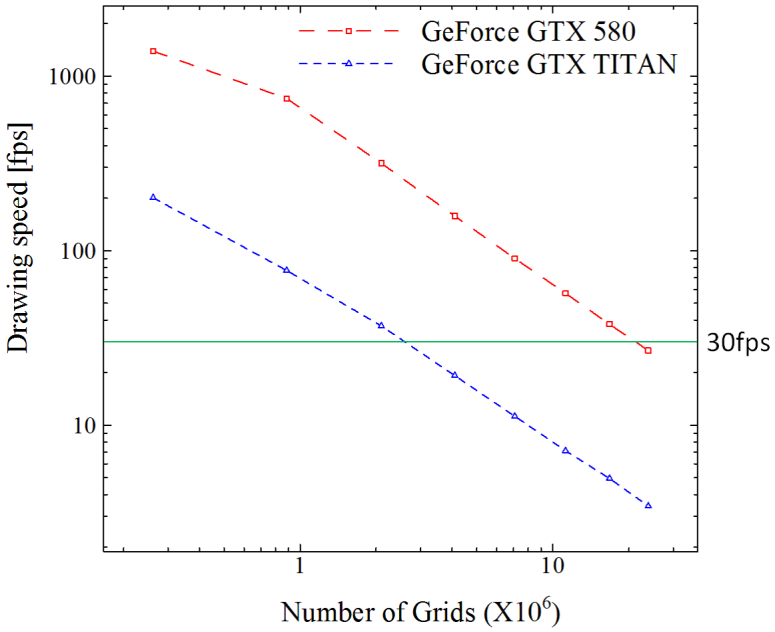

図 4.15 ボリュームレンダリングのグリッド数に対するフレームレートの変化

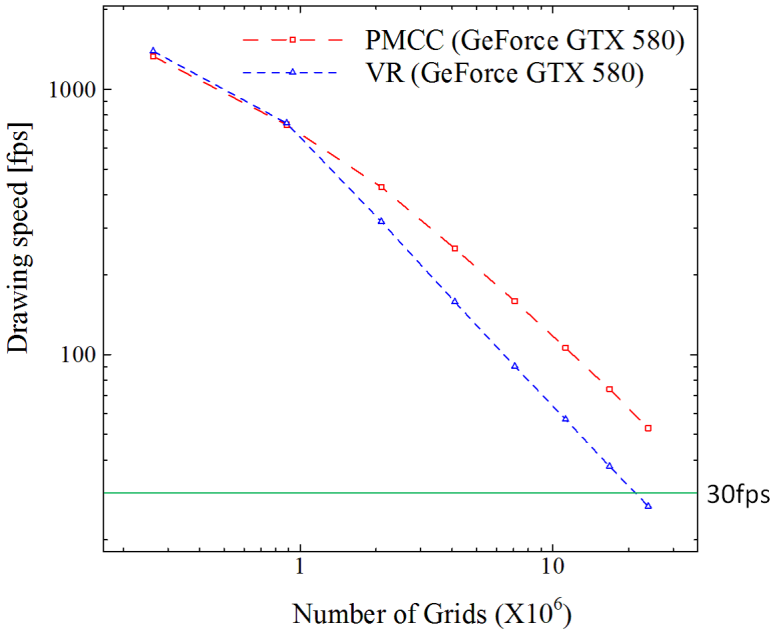

図 4.16 フレームレートの比較

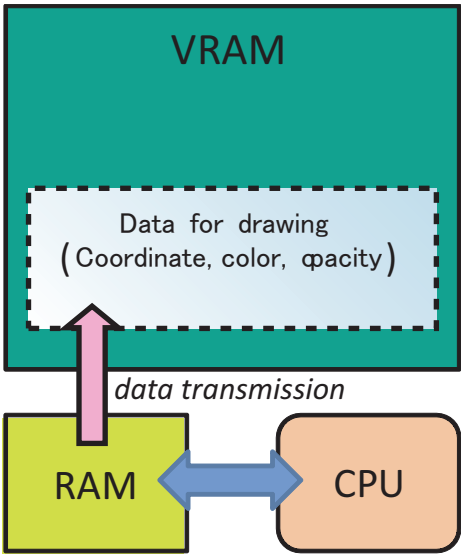

図 4.17 CPU で計算をする場合

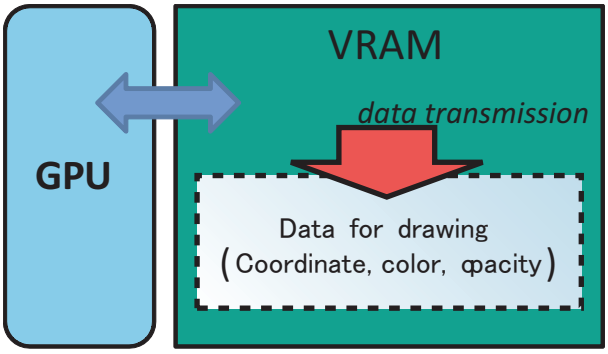

図 4.18 GPU で計算をする場合

4.5　マルチ GPU を用いた場合の可視化

GPU1 基あたりに搭載されているメモリはそれほど多くない．そのため，大規模な空間を可視化しようとした際には複数の GPU を用いる必要がある．マルチ GPU の構成を図 4.19 に示す．マルチ GPU で計算するためには，解析空間を GPU ごとに分割してやる必要がある．本研究では，図 4.20 に示したように z 軸に垂直に分割した．解析領域を 256^3 グリッドとして，吸収境界には 4 層の PML を適用した．

シングル GPU における可視化では単に GPU 上の計算領域から描画領域に描画用のデータを書き込むことで可視化が可能であった．しかし，PCI バスで接続された複数の GPU を用いて計算を行う場合（いわゆる単一ノード上のマルチ GPU 計算），計算したデータを描画するには，描画を担当する GPU に PCI バスを通して，描画用データを転送する必要がある．したがって，マルチ GPU での可視化では GPU 間の計算のための境界データの転送に加えて，描画のために必要な GPU 間の描画データ転送作業が必要となる．

2GPU で計算して可視化した場合のフレームレートの比較を表 4.3 に示す．

表 4.3　フレームレート (2GPU との比較)

計算環境	フレームレート [fps]
Core i7 930	2.3
GeForce GTX 580M	36.2
GeForce GTX 580	69.1
GeForce GTX 580 × 2	120

同表より，2GPU で計算・可視化した場合，CPU の場合に比べて約 52 倍の速度で可視化を行えることが分かる．

2GPU を用いた可視化が 1GPU の時に比べて更に高速であることは分かった．しかし，改良すべき余地がまだ残されている．

その一つが，非同期転送を用いた GPU 間通信の隠ぺいである．マルチ GPU を用いて解析する場合，解析空間が GPU 毎に分割されているので，解析の 1 ステップごとに GPU 間の境界の値を交換する必要がある．これまでは，図 4.23 のようにカーネルとデータ転送を同期して実行していた．

しかし，図 4.24 に示すように，初めに境界部のみカーネルを実行し，それ以外の部分のカーネル実行とデータ転送を非同期で行うことでデータ転送を隠ぺいすることができる．

これにより，数値解析に要していた時間を短縮できるため，フレームレートが上昇すると考えられる．

2GPU を用いた可視化に非同期転送を適用した場合のフレームレートの比較を表 4.4 に示す.

表 4.4 フレームレート (非同期転送を適用)

計算環境	フレームレート [fps]
Core i7 930	2.3
GeForce GTX 580M	36.2
GeForce GTX 580	69.1
GeForce GTX 580 × 2	120
GeForce GTX 580 × 2 with Async	124

同表より，非同期転送を適用した場合，適用前と比較して 4fps 高速化されたことが分かる.

もう一つの改良点が GPUDirect 2.0 を用いた高速 GPU 間通信である．従来の GPU 間のデータ転送は図 4.21 のようにホストの RAM を介さなければ不可能であった．しかし，CUDA 4.0 から RAM を通さない直接 GPU 間通信 GPUDirect 2.0 が実装された．図 4.25 または図 4.26 のように直接データを転送することが可能となった．RAM を介さない分，従来の転送よりも高速な通信が行えるようになった.

そして，GPUDirect 2.0 による高速通信を描画用データ通信部分に適用する．マルチ GPU を用いて可視化する場合，描画用データをセカンダリ GPU からプライマリ GPU へ転送する必要があるが，前節の場合と違い，この転送に非同期転送は適用できない．そこで，GPUDirect 2.0 を用いて描画用データ通信に要する時間を圧縮すれば，フレームレートが上昇すると考えられる.

2GPU を用いた可視化に非同期転送を適用後に，更に GPUDirect 2.0 を適用した場合のフレームレートの比較を表 4.5 に示す.

同表より，GPUDirect 2.0 を適用した場合，適用前と比較して 10fps 高速化されたことが分かる．そして，CPU の場合に比べて約 58 倍の速度で可視化を行えることが分かる.

マルチ GPU を用いて可視化した際のグリッド数に対するフレームレートの変化を図 4.27 に示す.

同図より，グリッドサイズが大きくなるに連れて，2GPU 可視化と改良を施した 2GPU 可視化のフレームレートの差が縮まっていくことが分かる．これは，解析空間が大きくなるに連れて FDTD 法による数値解析に要する時間が支配的になっていき，改良した箇所の占める割合が小さくなっていくためである.

表 4.5 フレームレート (GPUDirect 2.0 を適用)

計算環境	フレームレート [fps]
Core i7 930	2.3
GeForce GTX 580M	36.2
GeForce GTX 580	69.1
GeForce GTX 580 × 2	120
GeForce GTX 580 × 2 with Async	124
GeForce GTX 580 × 2 with Async, Direct	134

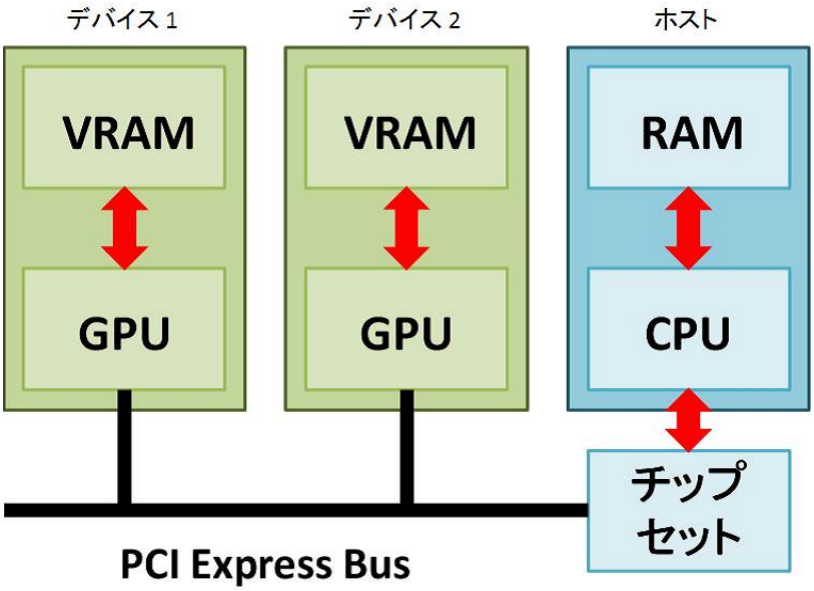

図 4.19 マルチ GPU の構成

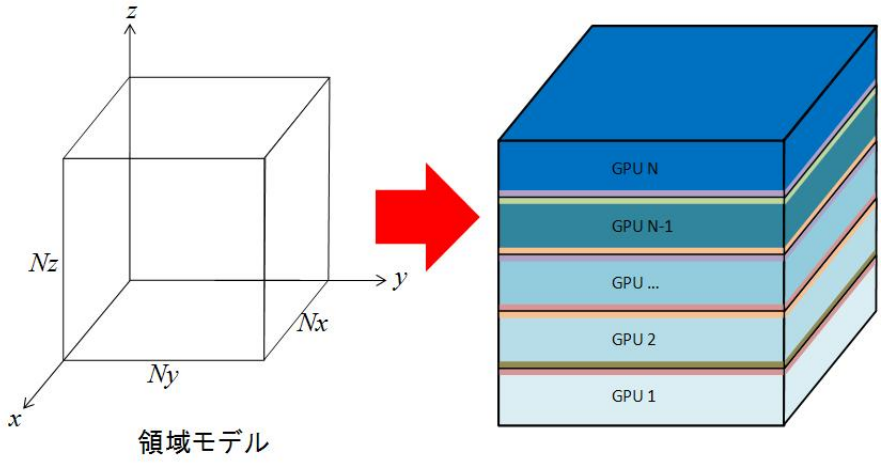

図 4.20 解析空間を GPU 毎に分割

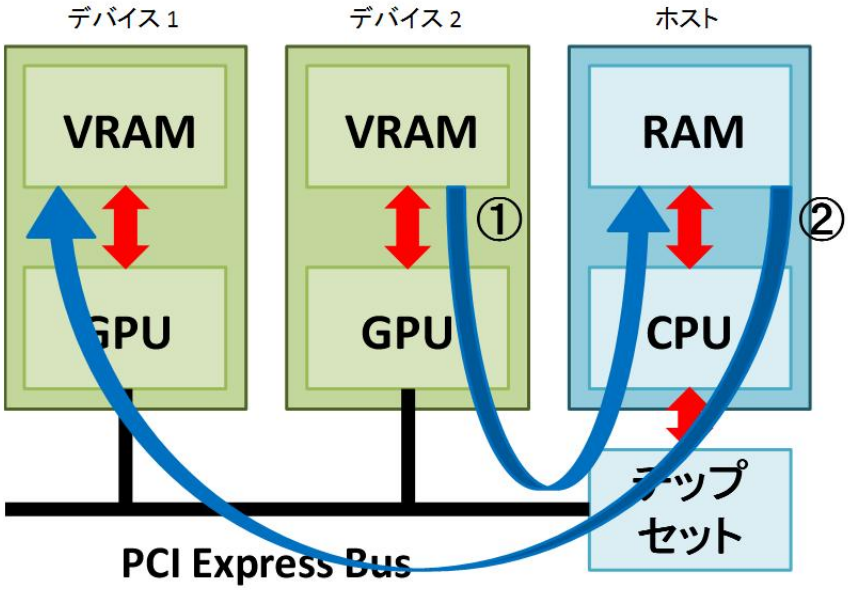

図 4.21 マルチ GPU 間でのデータ通信

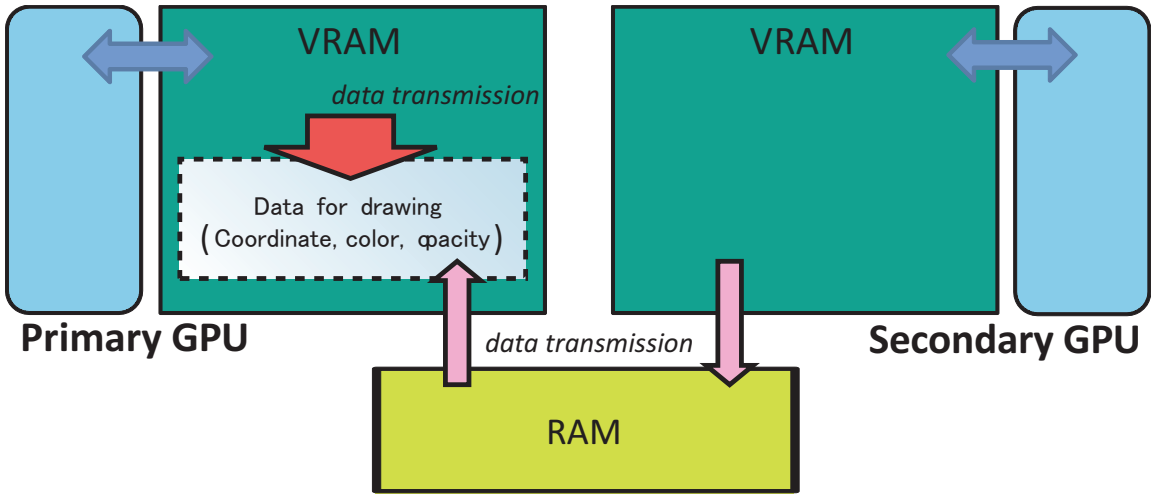

図 4.22　2GPU 間での描画データ転送

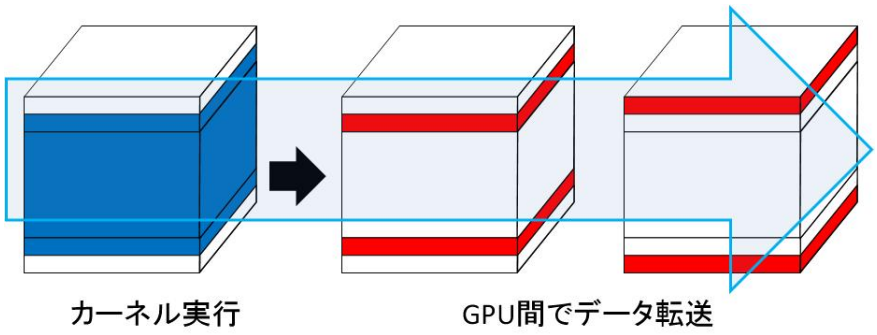

図 4.23　同期通信

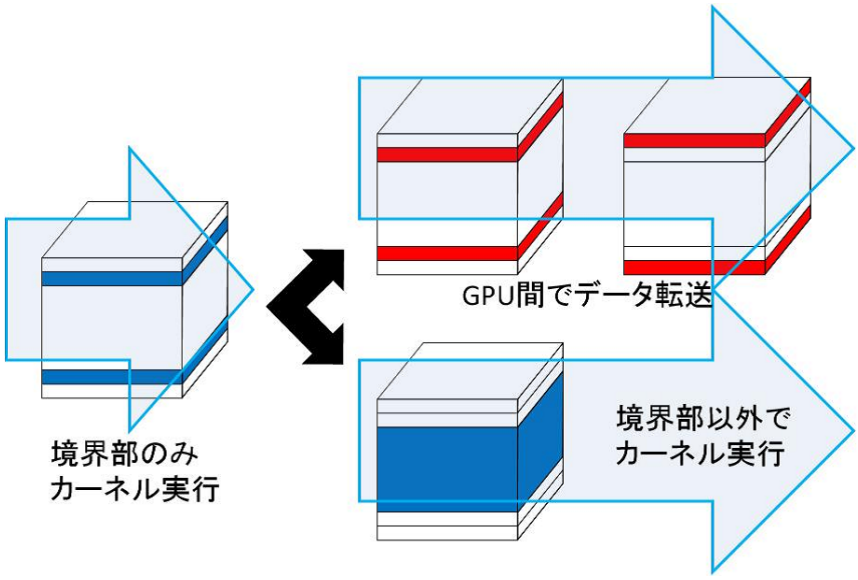

図 4.24　非同期通信

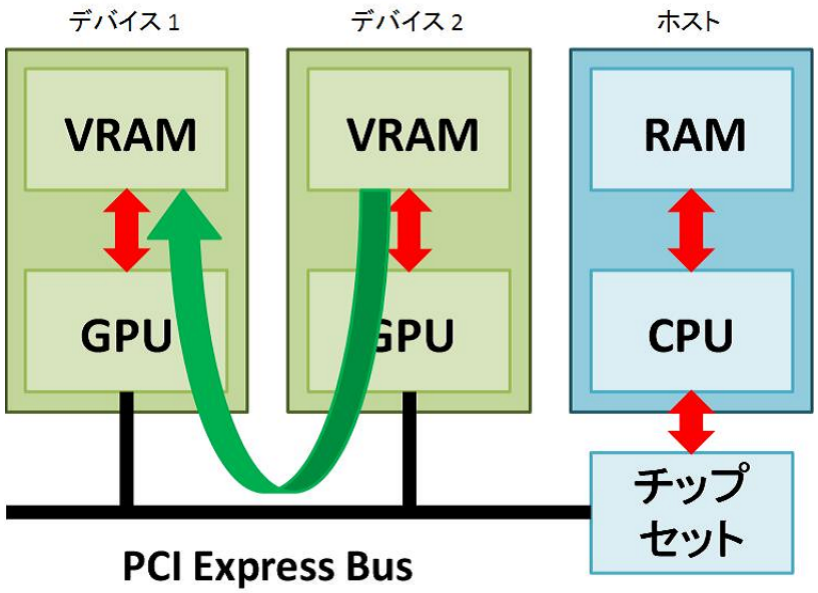

図 4.25　マルチ GPU 間でのデータ通信 (GPUDirect 2.0 適用)

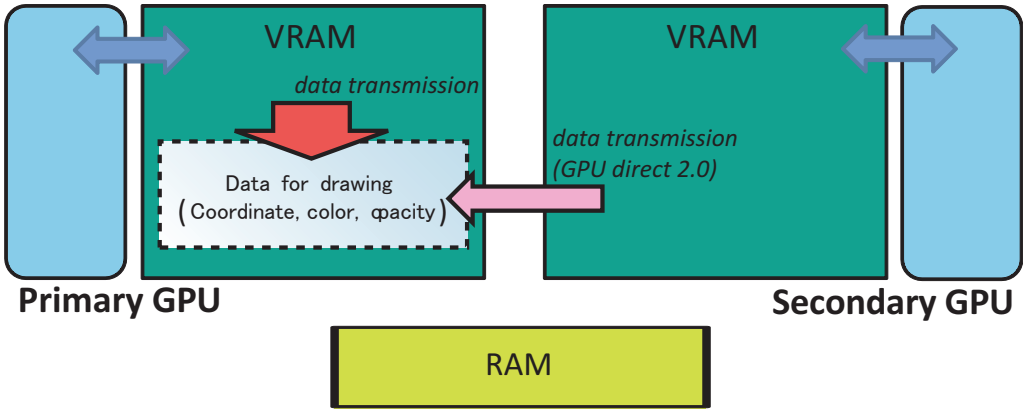

図 4.26　2GPU 間での描画データ転送 (GPUDirect 2.0 適用)

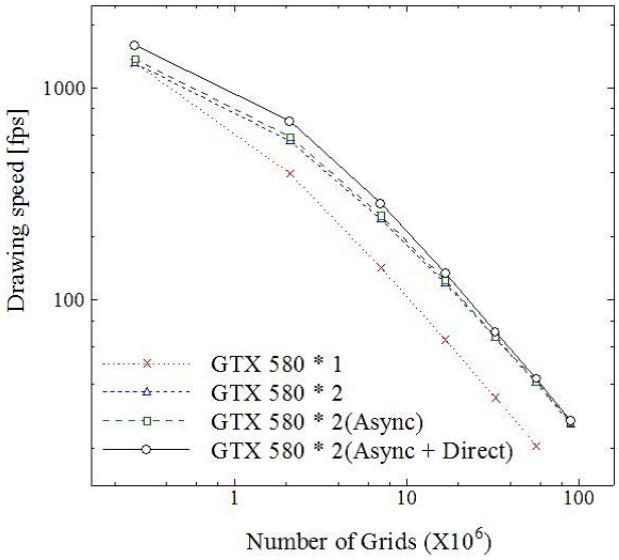

図 4.27　マルチ GPU 可視化のグリッド数に対するフレームレートの変化

4.6　結言

本章では，3 次元波動伝搬数値解析の可視化について述べた．3 次元波動伝搬シミュレーションの有効な可視化手法として PMCC を提案し，既存手法との比較を行った．そして，PMCC を用いたリアルタイム可視化に GPU を組み合わせることで従来よりもずっと高速に可視化を行えることを示した．さらに，マルチ GPU を用いることでシングル GPU の場合に比べてより高速に可視化できることを示した．

第5章　インタラクティブシミュレーションの概要とその応用について

5.1　序言

本章では，GPU による高速並列計算と PMCC を組み合わせたインタラクティブシミュレーションについて述べる．初めに，インタラクティブシミュレーションの概要を述べる．従来の数値解析および可視化について説明し，問題点を述べる．その問題点を解決する方法としてインタラクティブシミュレーションを提案する．GPU によって高速化されたことで可能となった，シミュレーションの実行中におけるインタラクティブな解析パラメータ操作について説明し，インタラクティブシミュレーションで実現できることを解説する．そして，音場・電磁界・弾性波それぞれの数値解析におけるインタラクティブシミュレーションの応用例について説明する．

5.2　インタラクティブシミュレーションの概要と特徴について

インタラクティブシミュレーションとは，GPU による高速並列計算とリアルタイム可視化を組み合わせたシミュレーションのことである．

従来，数値シミュレーションの可視化を行う場合にはまずシミュレーションを走らせ，シミュレーションが終了した後にシミュレーション中に書き出したファイルを用いて可視化していた．しかし，解析結果に満足がいかない場合はシミュレーションをその都度やり直す必要があり，コストの増大が問題であった．数値解析を行いながらその結果をリアルタイムに可視化して，内容によって解析パラメータをインタラクティブに操作したいという願望が生まれるのは当然の成り行きである．

だが，これまでの CPU を用いた計算では計算速度が遅く，リアルタイム可視化を行うと可視化速度が不十分であるためにインタラクティブなシミュレーションは不可能であった．スーパーコンピュータを用いれば高速に計算することはできるが，スーパーコンピュータは多くのユーザでのタイムシェアリング方式が一般であり，一人のユーザが占有するよう

なシミュレーションは難しい．しかし，GPU を用いることでこれらの問題は一気に解決した．GPU をパソコンに接続するだけでスーパーコンピュータに匹敵する計算速度が手に入り，それを占有して使うことができる．GPU 高速並列計算とリアルタイム可視化を組み合わせることでインタラクティブシミュレーションは現実のものとなった．

インタラクティブシミュレーションにより以下のことが可能となる．

(1) 可視化視点，可視化断面位置の変更
(2) 表示する成分の切替
(3) 表示する成分の可視化強度のスケール変更
(4) PMCC とボリュームレンダリングの切替
(5) サブウインドウによるある観測点の時系列変化の観測
(6) 時間方向における逆回転計算への切替
(7) 入力波形の振幅・周波数の段階的変化

上記のことをシミュレーションの状況に合わせてインタラクティブに変更することができる．

本研究では，インタラクティブシミュレーションを実装するにあたって解析手法として FDTD 法，主な可視化手法として PMCC を用いた．両手法を採用した大きな理由はシミュレーションの可視化速度である．解析手法は FDTD 法，MM-MOC 法ともに GPU を利用することで大幅に高速化されたが，同じ領域サイズでの計算速度は MM-MOC 法と比較すると FDTD 法のほうが速い．シミュレーションの可視化速度を考慮して解析手法に FDTD 法を採用した．また，PMCC は他の可視化手法と比較して高い視認性を持ちながらも高速に可視化を行える手法である．インタラクティブシミュレーションでは高い視認性と高速な可視化が非常に重要となるため，可視化手法として PMCC を用いた．

インタラクティブシミュレーションのフレームワークの流れを図 5.1 に示す．OpenGL を用いた可視化の場合，開発には FreeGLUT を使用した [36]．FreeGLUT ではマウスによるクリック，ドラッグやキーボードのキー押下などのイベントに応じた関数をコールバックとして登録することができる．イベント待ちループをスタートさせると発生したイベントに応じてコールバックが動作する．CPU がアイドル時に発生するアイドルイベントというイベントがあり，このイベントに対応するコールバックとして波動数値解析のソルバを登録することでシミュレーションが実行される．そして，シミュレーション中にマウス操作などのイベントを発生させることでコールバックが動作し，インタラクティブな操作が可能となる．

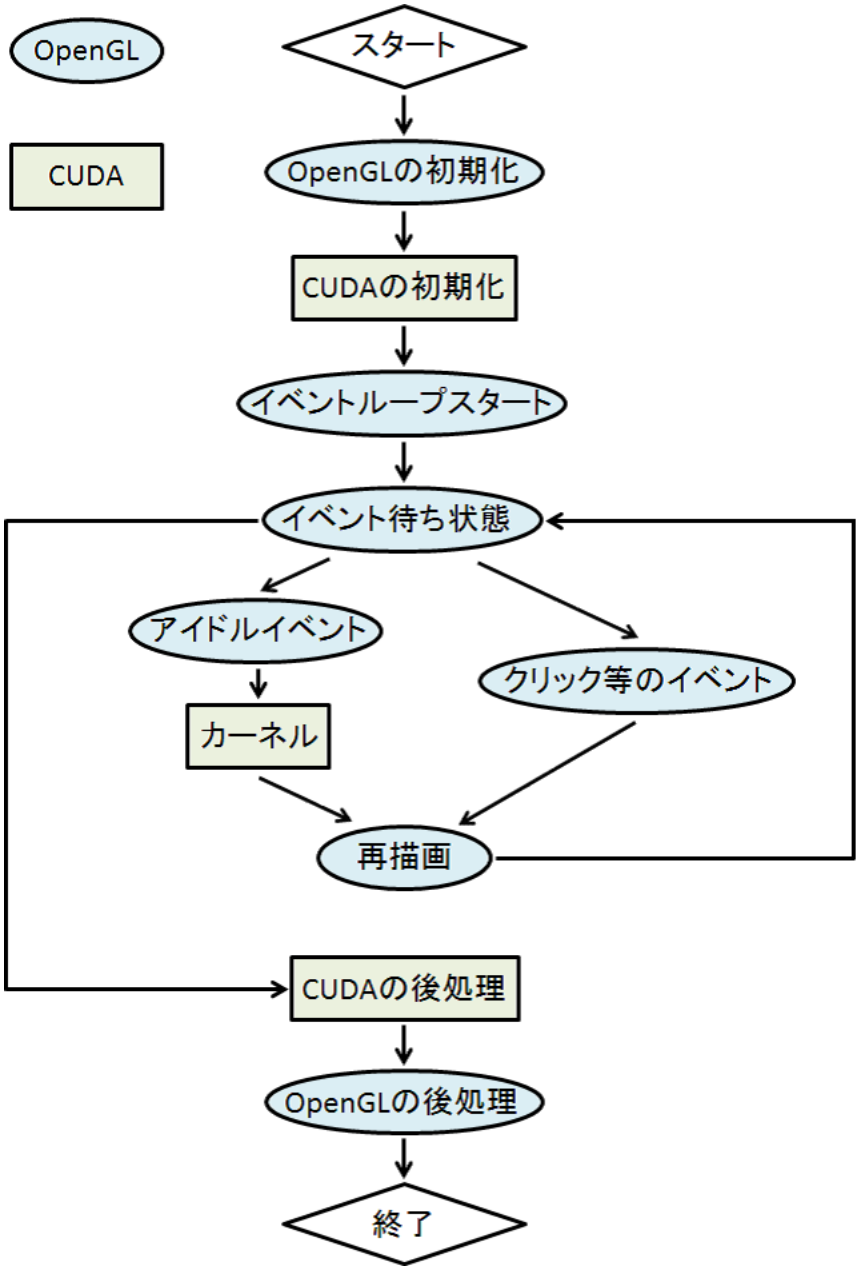

図 5.1 インタラクティブシミュレーションの流れ

5.3　インタラクティブシミュレーションの音場解析における応用

初めに，インタラクティブシミュレーションの音場数値解析における適用について説明する．

5.3.1　室内音場解析への適用

ここでは，室内音場での音波伝搬解析について述べる．室内音場解析解析で用いたパラメータを表 5.1 にまとめる．

表 5.1　室内音場解析で用いた解析条件

タイムステップ	$\Delta t = 5.0 \times 10^{-5}$ s
グリッドサイズ	$\Delta x = \Delta y = \Delta z = 0.05$ m
密度 ρ	1.21 kg/m^3
体積弾性率 K	1.4235529 $\times 10^5$ N/m^2

図 5.2 及び 5.3 に室内音場数値解析におけるインタラクティブシミュレーションの例を示す．外壁・棚にはインピーダンス境界を適用している．

図から分かるように，サブウインドウを展開することで反射を繰り返す音波が減衰していく様子を確認できる．

また，図 5.4 及び 5.5 に同じく室内音場数値解析におけるインタラクティブシミュレーションの例を示す．

図から分かるように，音波の様子によって可視化する視点を変更可能であり，可視化強度のスケールを変更することにより微弱な音波を可視化することができる．すなわち，音波の伝搬・反射・散乱の状況により注意して観測を行いたい部分を思い通りに可視化することができる．

このように，インタラクティブシミュレーションを音場数値解析に適用することで様々なことが可能となる．

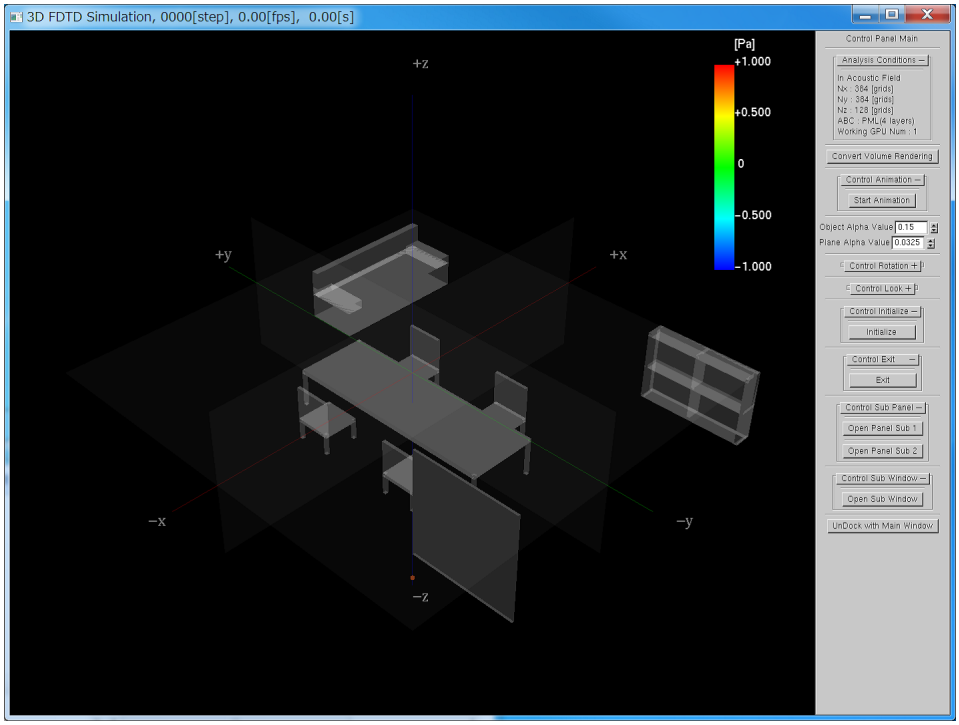

図 5.2　設定した室内モデル

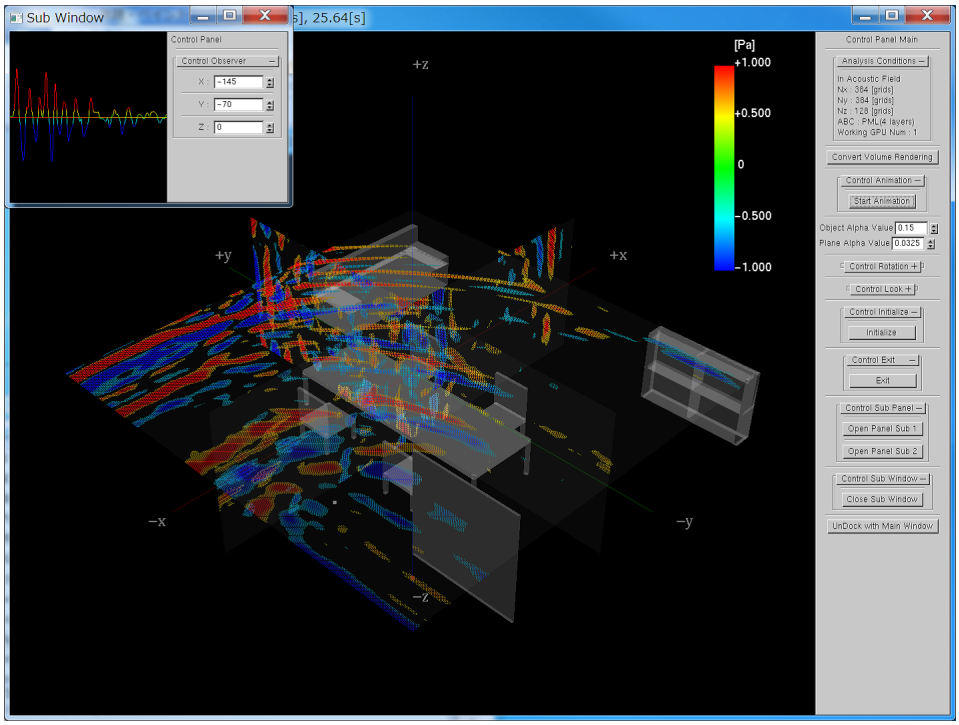

図 5.3　音波が壁で反射し散乱している様子

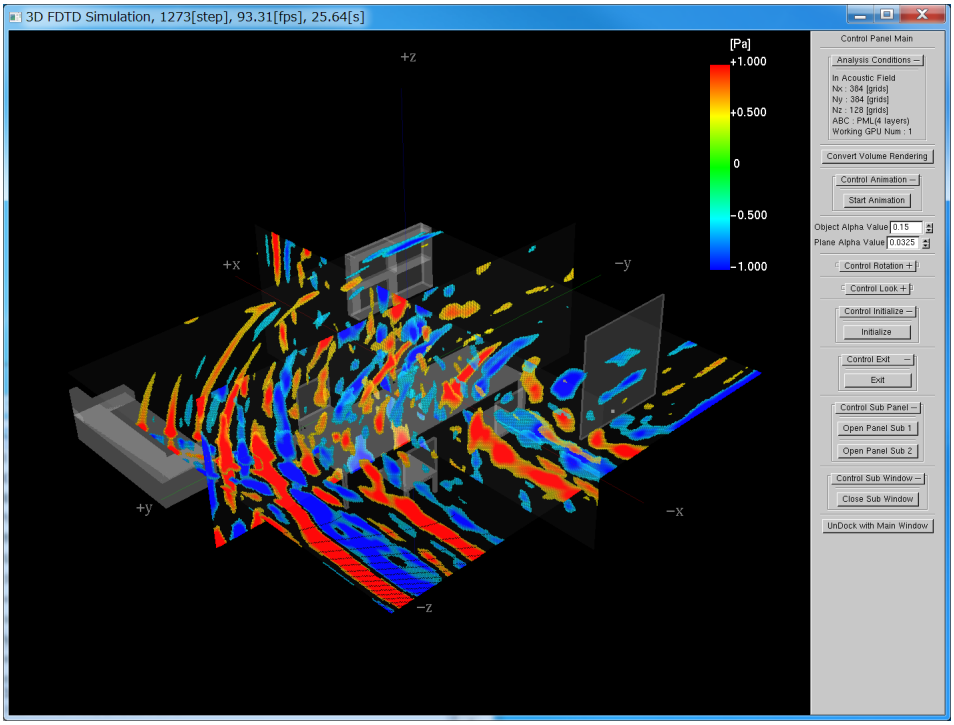

図 5.4　別の視点からの様子

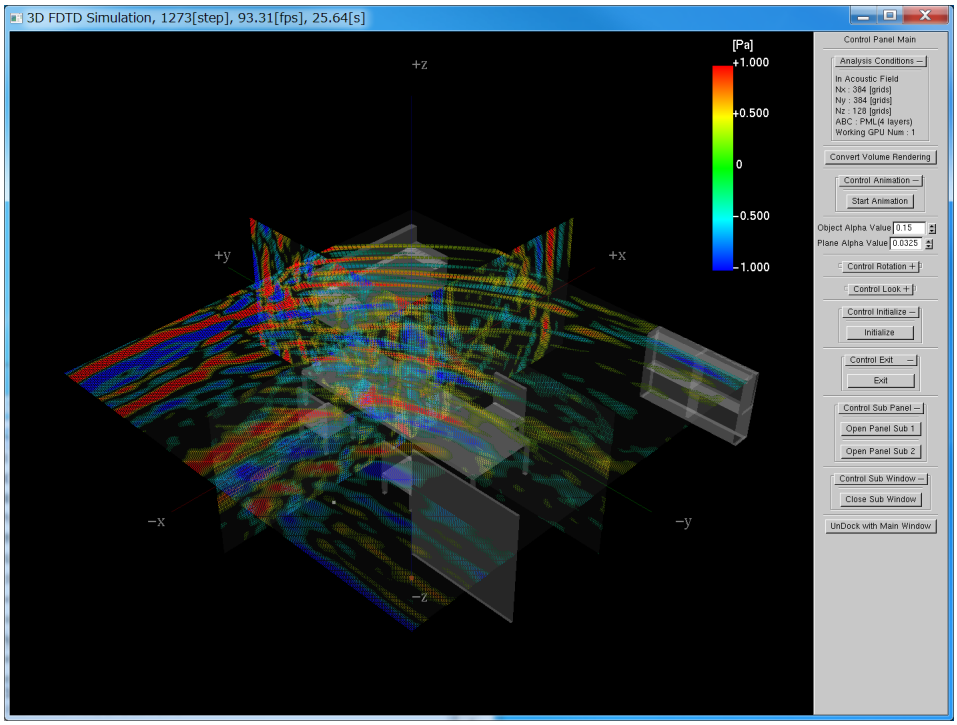

図 5.5　描画する α 値の閾値を変更した場合の様子

5.4 インタラクティブシミュレーションの電磁界解析における応用

次に，インタラクティブシミュレーションの電磁界数値解析における適用について説明する．

5.4.1 アンテナ解析への適用

ここでは，アンテナ解析におけるインタラクティブシミュレーションの例を示す．

ダイポールアンテナ

初めに，ダイポールアンテナをモデリングし，シミュレーションした例を示す．ダイポールアンテナはアマチュア無線の自作アンテナ等で広く普及している．ダイポールアンテナ解析で用いたパラメータを表 5.2 にまとめる．

表 5.2 ダイポールアンテナ解析で用いた解析条件

タイムステップ	$\Delta t = 2.5 \times 10^{-12}$ s
グリッドサイズ	$\Delta x = \Delta y = \Delta z = 0.0025$ m
アンテナ長	0.075 m
給電周波数	2 GHz

ダイポールアンテナを解析空間の中心に設置した様子を図 5.6 に示す．

図 5.7 にダイポールアンテナから放射される電界の分布を，図 5.8 にダイポールアンテナから放射される磁界の分布を示す．

図 5.7 より，ダイポールアンテナがアンテナに対して垂直方向に指向性を持っていることが確認できる．これは，ダイポールアンテナの長さが半波長となるような給電周波数で入力しているためであり，インタラクティブシミュレーションでは給電周波数を操作することで電波の指向性が変化していく様を観測することができる．

図 5.9 にダイポールアンテナから放射されるポインティングベクトルを，図 5.10 にダイポールアンテナ上の電流の分布を示す．

図 5.10 から，ダイポールアンテナの長さが半波長となるような給電周波数で入力していることが確認できる．入力波形の周波数をインタラクティブに操作することで，アンテナ上の電流分布の様子が変わっていく様を観測することができる．

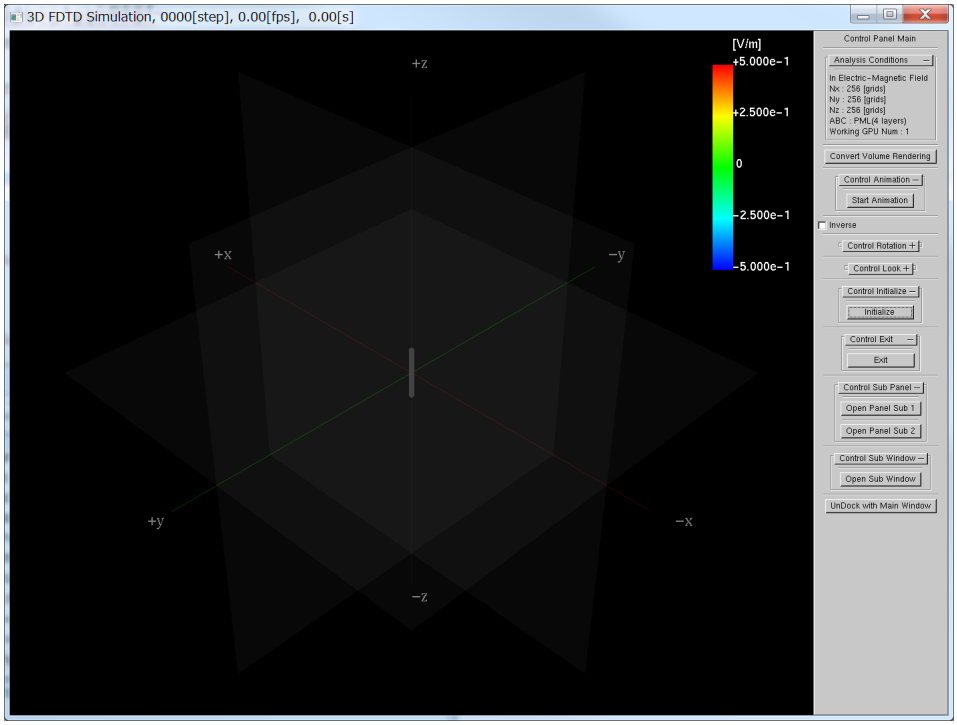

図 5.6　中心にダイポールアンテナを設置

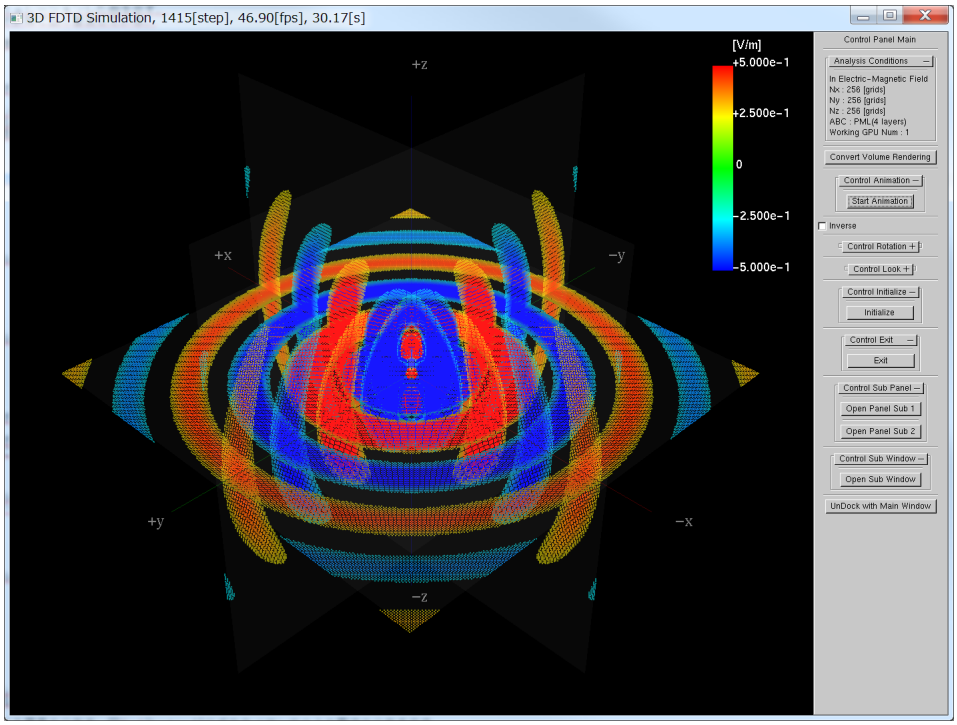

図 5.7　ダイポールアンテナ周辺の電界分布

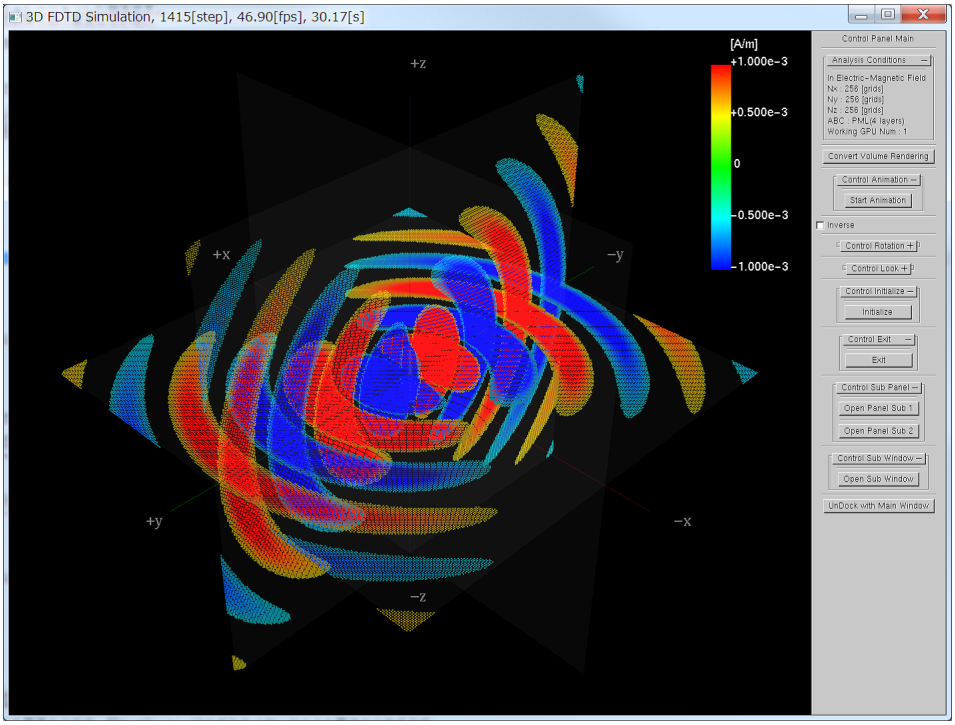

図 5.8　ダイポールアンテナ周辺の磁界分布

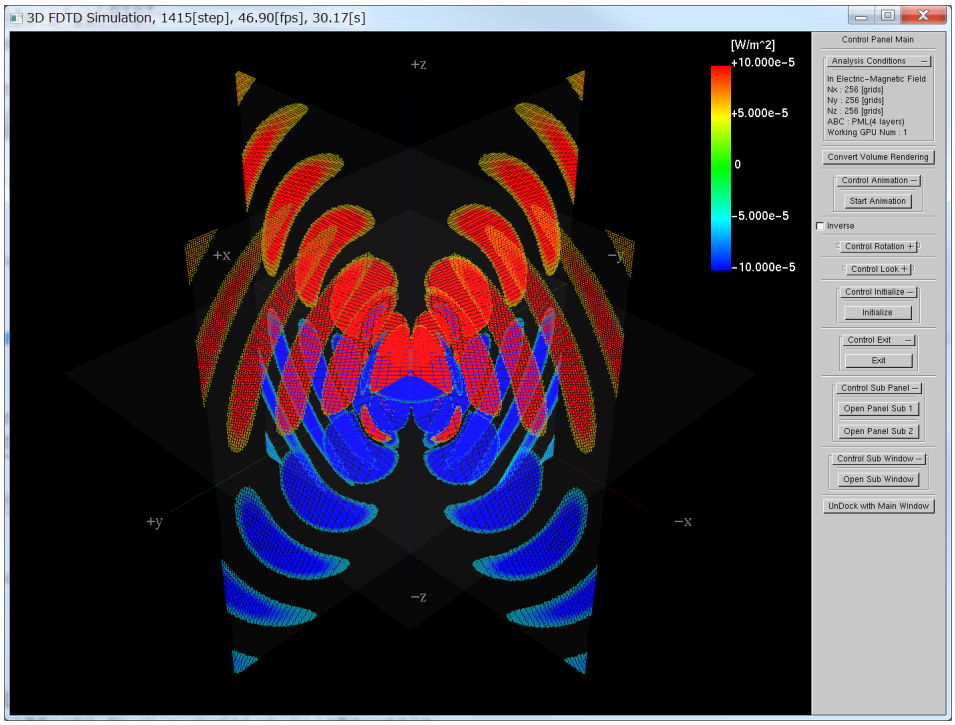

図 5.9　ダイポールアンテナ周辺のポインティングベクトル

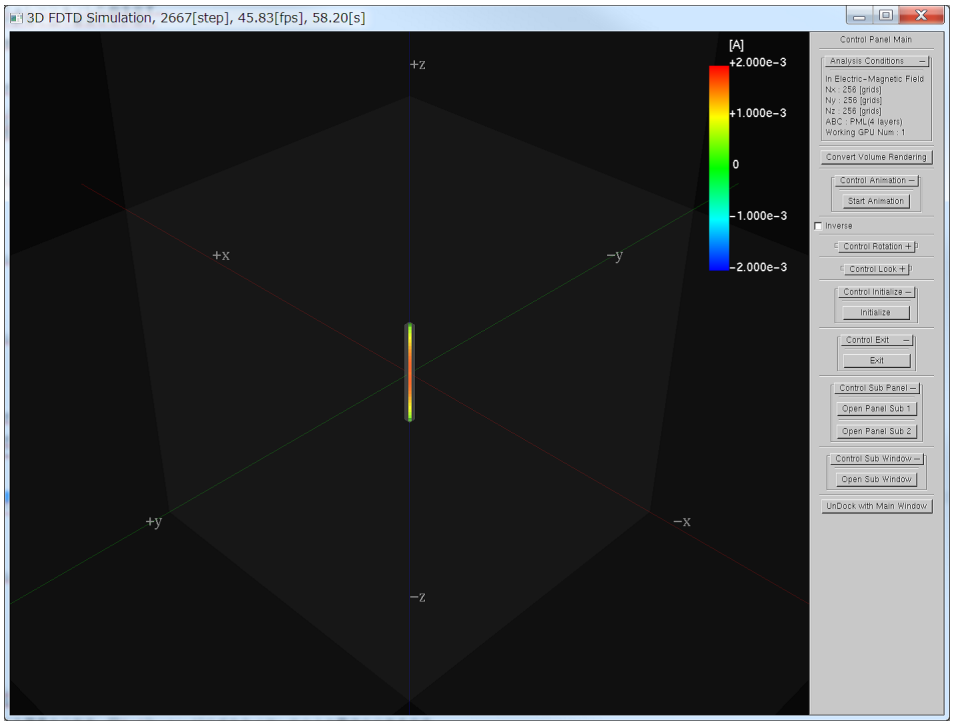

図 5.10　ダイポールアンテナ上の電流分布

ボウタイアンテナ

次に，ボウタイアンテナをモデリングし，シミュレーションした例を示す．ボウタイアンテナは地中レーダ等で用いられている．ボウタイアンテナ解析で用いたパラメータを表 5.3 にまとめる．

表 5.3　ボウタイアンテナ解析で用いた解析条件

タイムステップ	$\Delta t = 1.25 \times 10^{-12}$ s
グリッドサイズ	$\Delta x = \Delta y = \Delta z = 0.00125$ m
アンテナ長	0.075 m
給電周波数	2 GHz

ボウタイアンテナを解析空間の中心に設置した様子を図 5.11 に示す．図 5.12 にボウタイアンテナから放射される電界の分布を，図 5.13 にボウタイアンテナ上の電流の分布を示す．

図 5.12 より，ボウタイアンテナがアンテナに対して垂直方向に指向性を持っていることが確認できる．これは，ボウタイアンテナの中心の長さが半波長となるような給電周波数で入力している場合の指向性となる．ボウタイアンテナはダイポールアンテナよりも広帯域なアンテナである．入力波形の周波数をインタラクティブに操作することで，アンテナ上の電流分布の様子が変わっていく様を観測することができ，アンテナから放射される電界の指向性が変化することも確認できる．

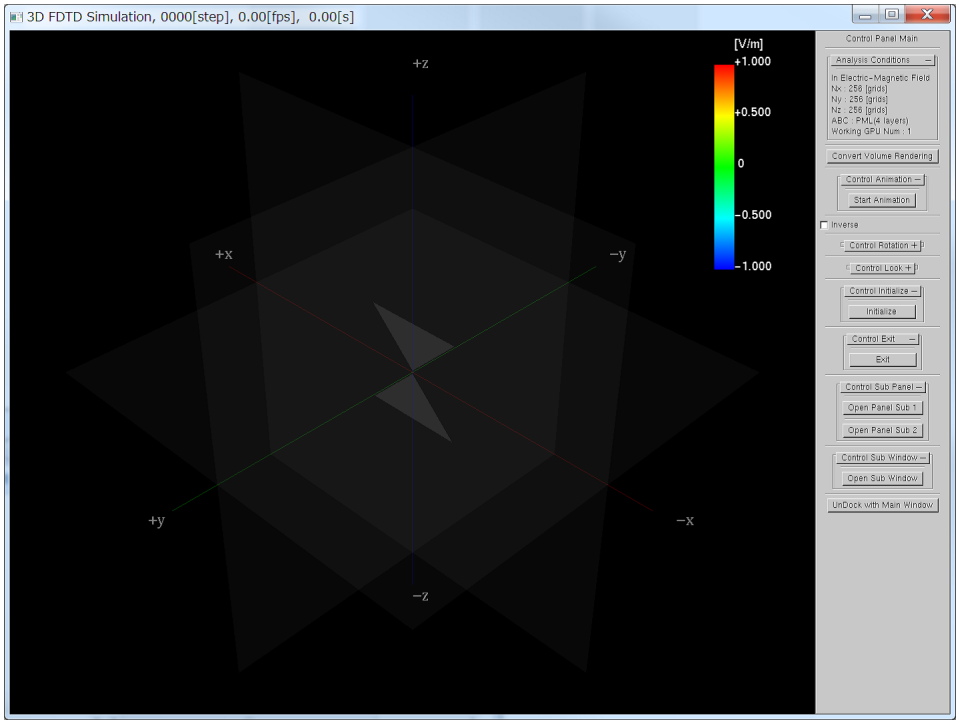

図 5.11　中心にボウタイアンテナを設置

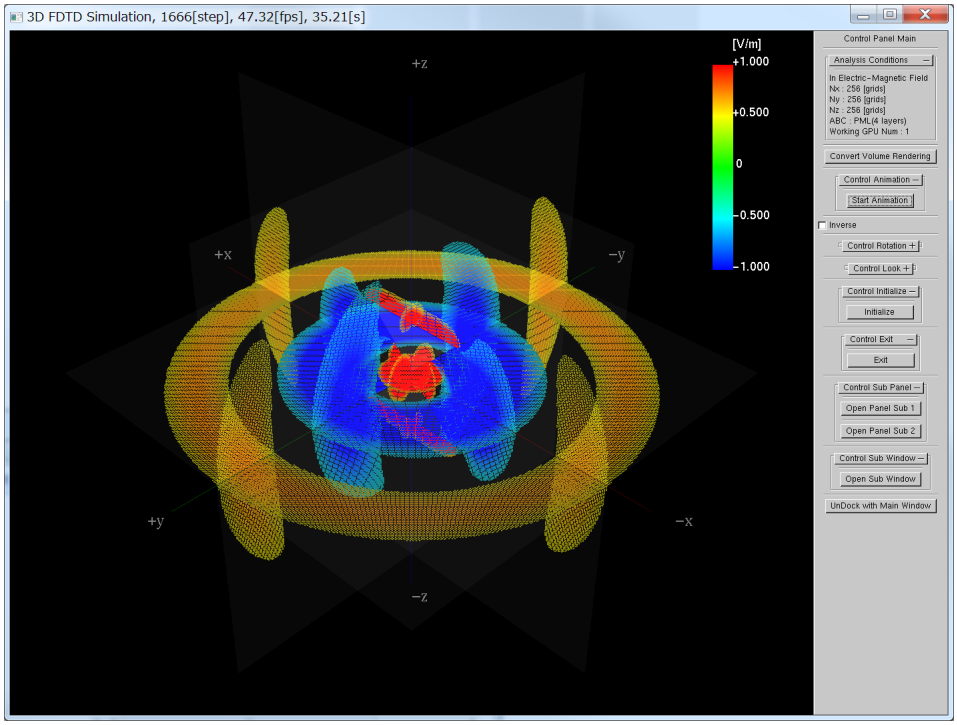

図 5.12　ボウタイアンテナ周辺の電界分布

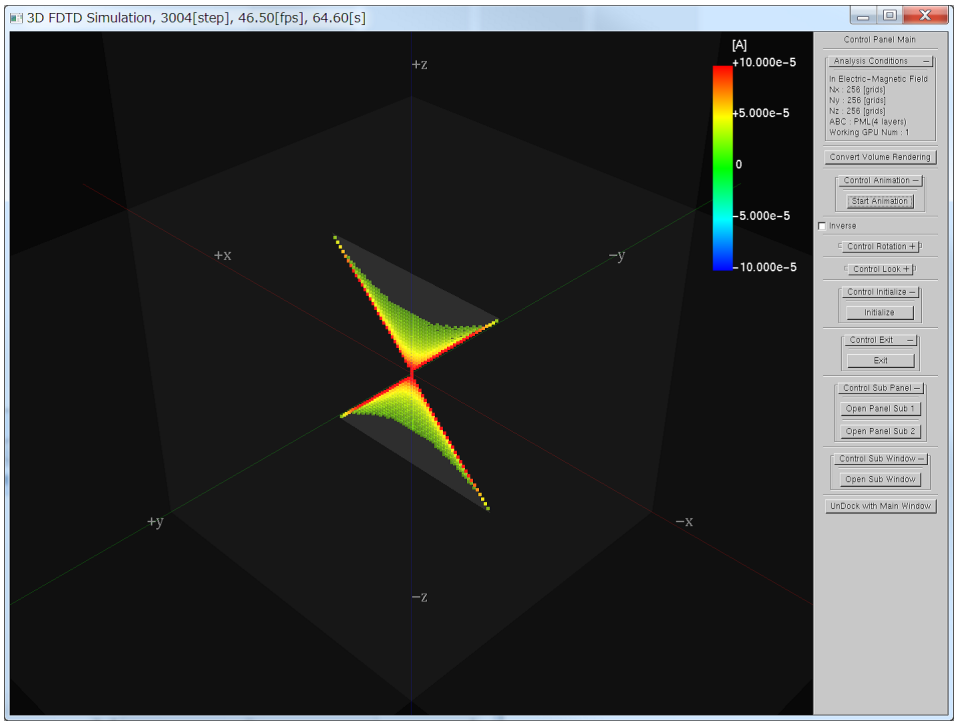

図 5.13　ボウタイアンテナ上の電流分布

ループアンテナ

次に，ループアンテナをモデリングし，シミュレーションした例を示す．ループアンテナは AM ラジオ受信のためなどに用いられている．ループアンテナ解析で用いたパラメータを表 5.4 にまとめる．

表 5.4　ループアンテナ解析で用いた解析条件

タイムステップ	$\Delta t = 2.5 \times 10^{-12}$ s
グリッドサイズ	$\Delta x = \Delta y = \Delta z = 0.0025$ m
アンテナ長（1 周）	0.3 m
給電周波数	1 GHz

ループアンテナを解析空間の中心に設置した様子を図 5.14 に示す．図 5.15 にループアンテナから放射される電界の分布を，図 5.16 にループアンテナ上の電流の分布を示す．

図 5.15 より，ループアンテナの指向性を確認できる．これは，ループアンテナの一周の長さが一波長となるような給電周波数で入力している場合の指向性となる．図 5.16 から，ループアンテナの一周の長さが一波長となるような給電周波数で入力していることが確認できる．入力波形の周波数をインタラクティブに操作することで，アンテナ上の電流分布の様子が変わっていく様を観測することができる．

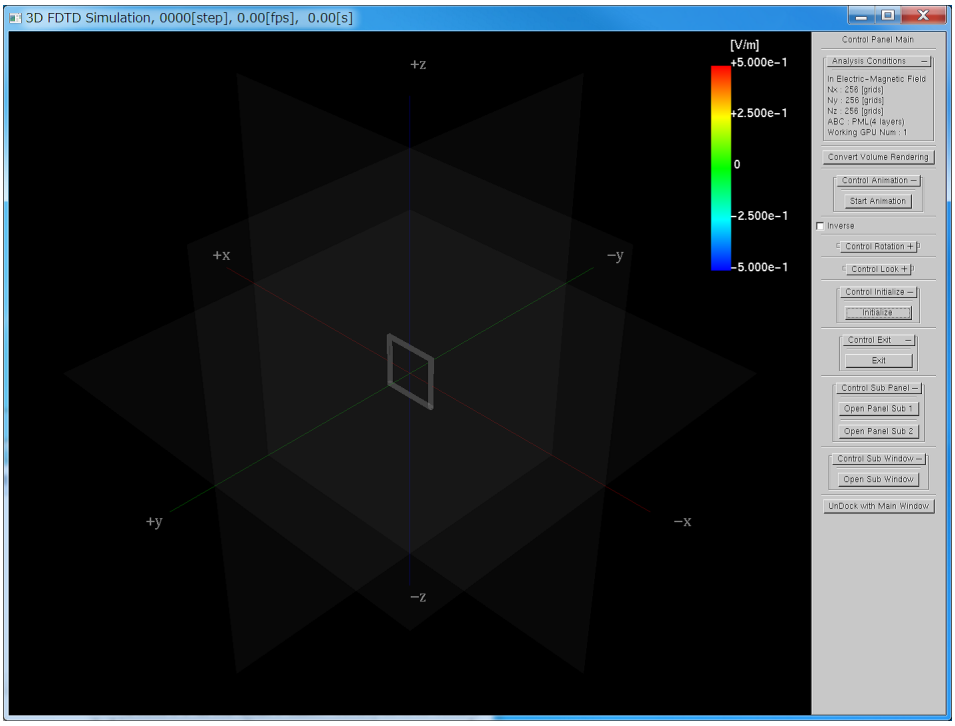

図 5.14　中心にループアンテナを設置

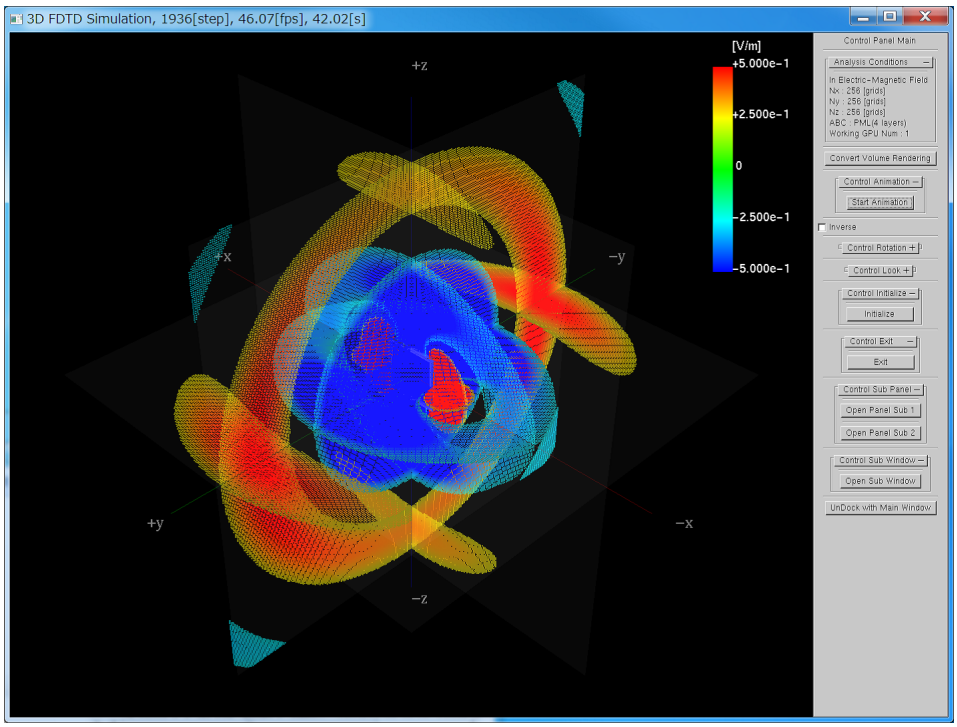

図 5.15　ループアンテナ周辺の電界分布

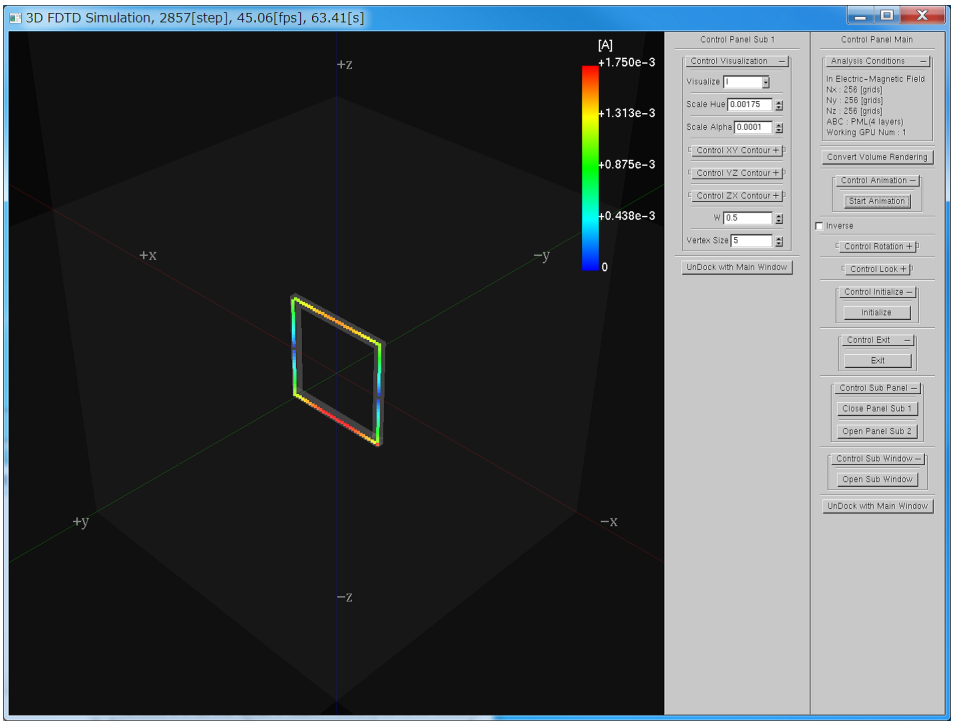

図 5.16　ループアンテナ上の電流分布

八木・宇田アンテナ

次に，八木・宇田アンテナをモデリングし，シミュレーションした例を示す．八木・宇田アンテナはテレビ放送，FM 放送の受信用やアマチュア無線，業務無線の基地局用などに用いられている．八木・宇田アンテナ解析で用いたパラメータを表 5.5 にまとめる．

表 5.5　八木・宇田アンテナ解析で用いた解析条件

タイムステップ	$\Delta t = 2.5 \times 10^{-12}$ s
グリッドサイズ	$\Delta x = \Delta y = \Delta z = 0.0025$ m
アンテナ長（輻射器）	0.075 m
給電周波数	2 GHz

八木・宇田アンテナを解析空間の中心に設置した様子を図 5.17 に示す．図 5.18 に八木・宇田アンテナから放射される電界の分布を，図 5.19 に八木・宇田アンテナ上の電流の分布を示す．

図 5.15 より，八木・宇田アンテナの指向性を確認できる．輻射器の周りに反射器，導波器を設置することでアンテナの指向性を導波器方向に設定することができる．PMCC を用いることで八木・宇田アンテナの指向性を分かり易く可視化することができる．

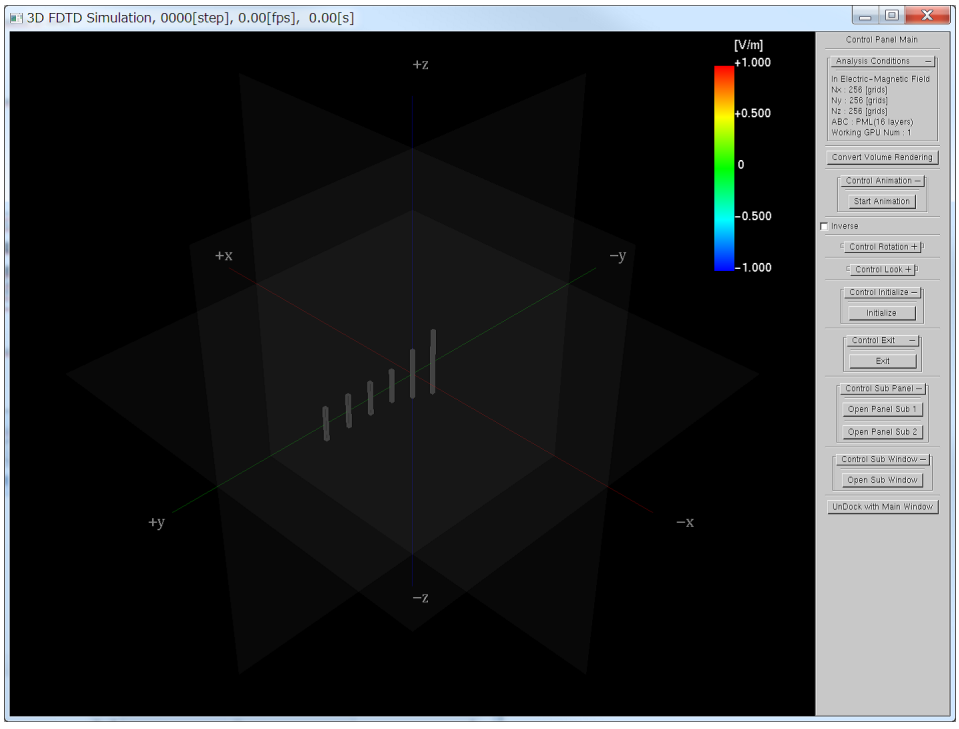

図 5.17　中心に八木・宇田アンテナを設置

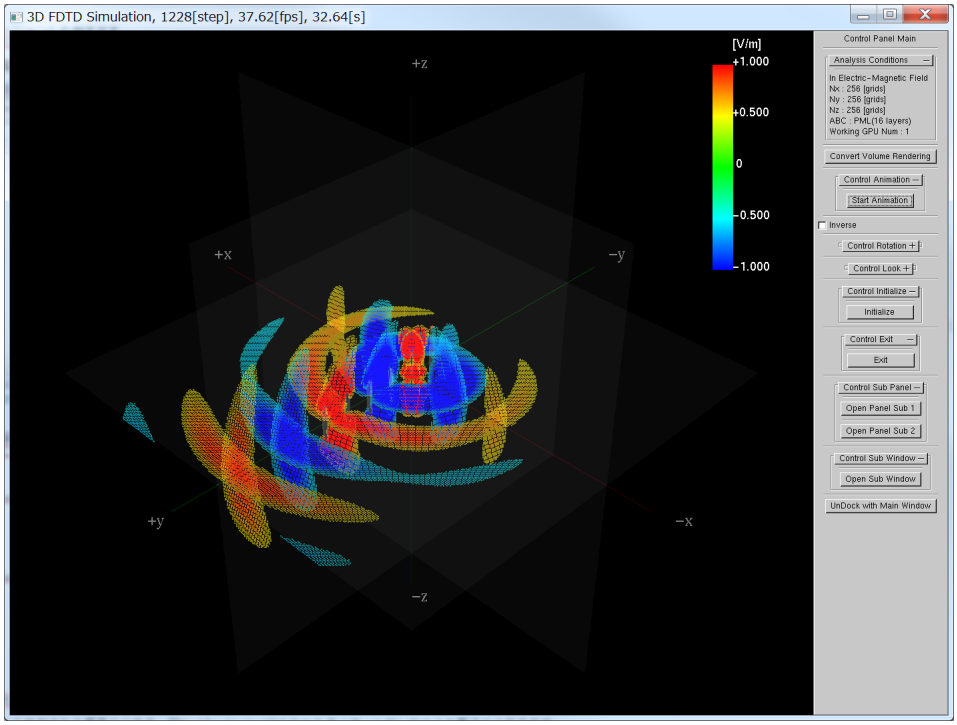

図 5.18　八木・宇田アンテナ周辺の電界分布

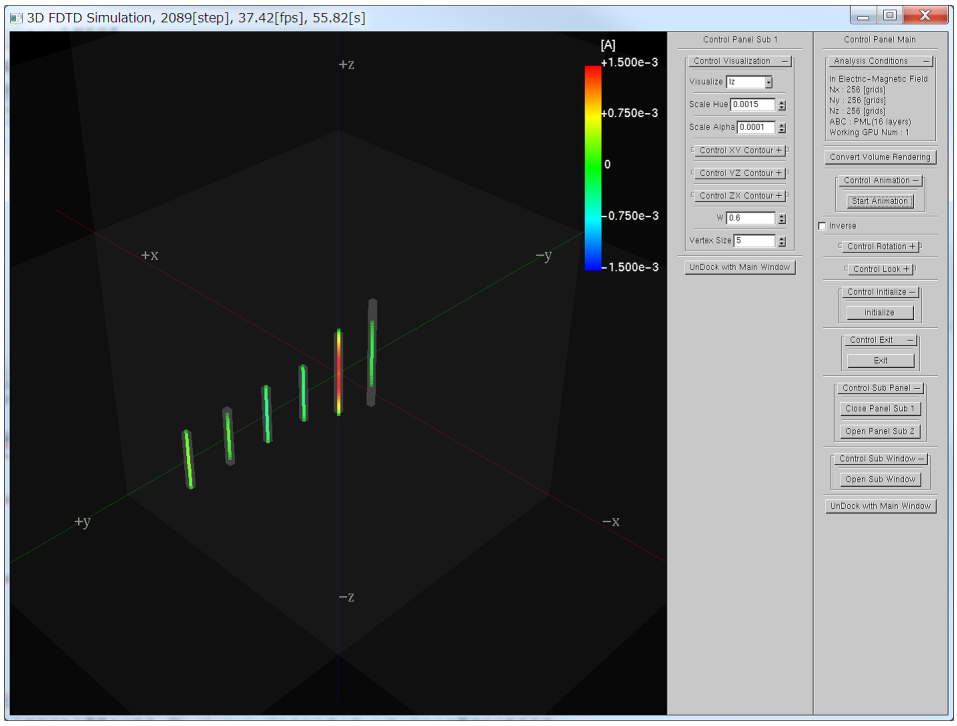

図 5.19　八木・宇田アンテナ上の電流分布

クロスダイポールアンテナ

次に，クロスダイポールアンテナをモデリングし，シミュレーションした例を示す．クロスダイポールアンテナは衛星通信等に用いられている．クロスダイポールアンテナ解析で用いたパラメータを表 5.6 にまとめる．

表 5.6　クロスダイポールアンテナ解析で用いた解析条件

タイムステップ	$\Delta t = 1.0 \times 10^{-12}$ s
グリッドサイズ	$\Delta x = \Delta y = \Delta z = 0.001$ m
アンテナ長	0.075 m
給電周波数	2 GHz

クロスダイポールアンテナを解析空間の中心に設置した様子を図 5.20 に示す．図 5.21 にクロスダイポールアンテナから放射される電界の分布を，図 5.22 にクロスダイポールアンテナ上の電流の分布を示す．

図 5.21 より，クロスダイポールアンテナの指向性を確認できる．図 5.22 から，クロスダイポールアンテナでは二本のダイポールアンテナに位相をずらして給電していることが確認できる．クロスダイポールアンテナは円偏波で電波を放射する．クロスした二本のダイポールアンテナから互い違いに放射される電波を観測する場合，PMCC で可視化を行うとよく理解できる．空間から切り取った断面に E_x, E_y, Ez の電界成分をそれぞれ個別に表示できるためである．このような可視化は他の手法では難しく，PMCC ならではと言えるだろう．

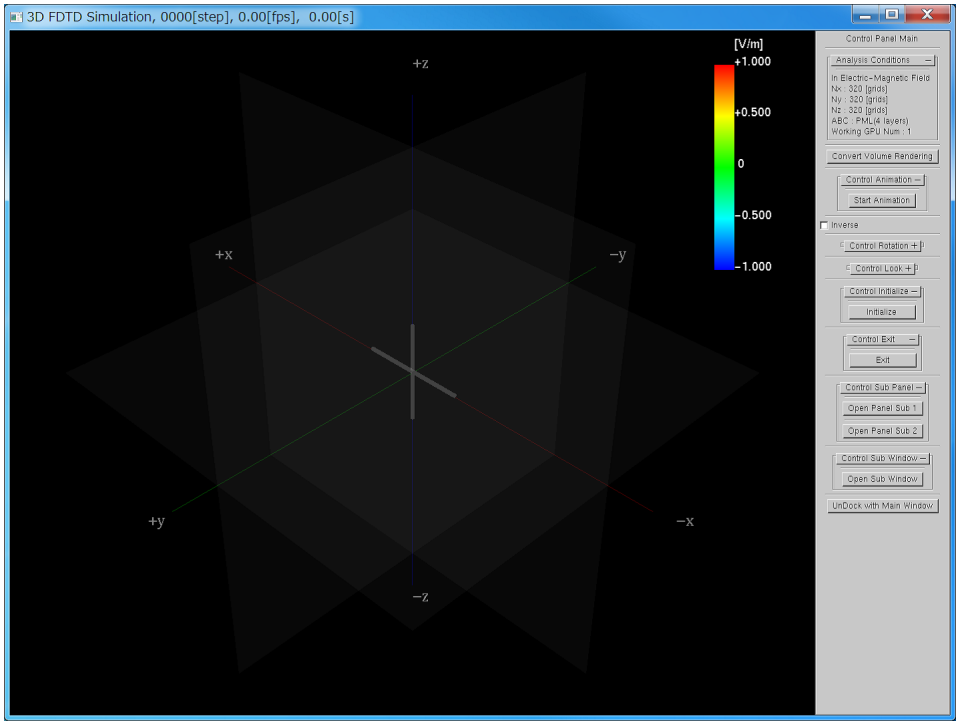

図 5.20　中心にクロスダイポールアンテナを設置

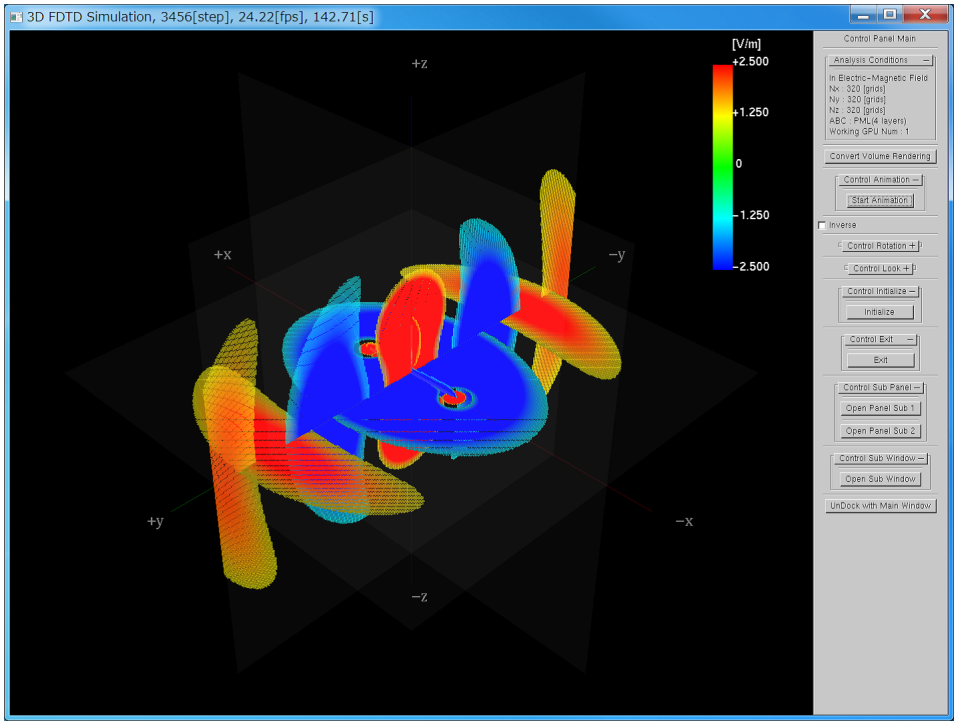

図 5.21　クロスダイポールアンテナ周辺の電界分布

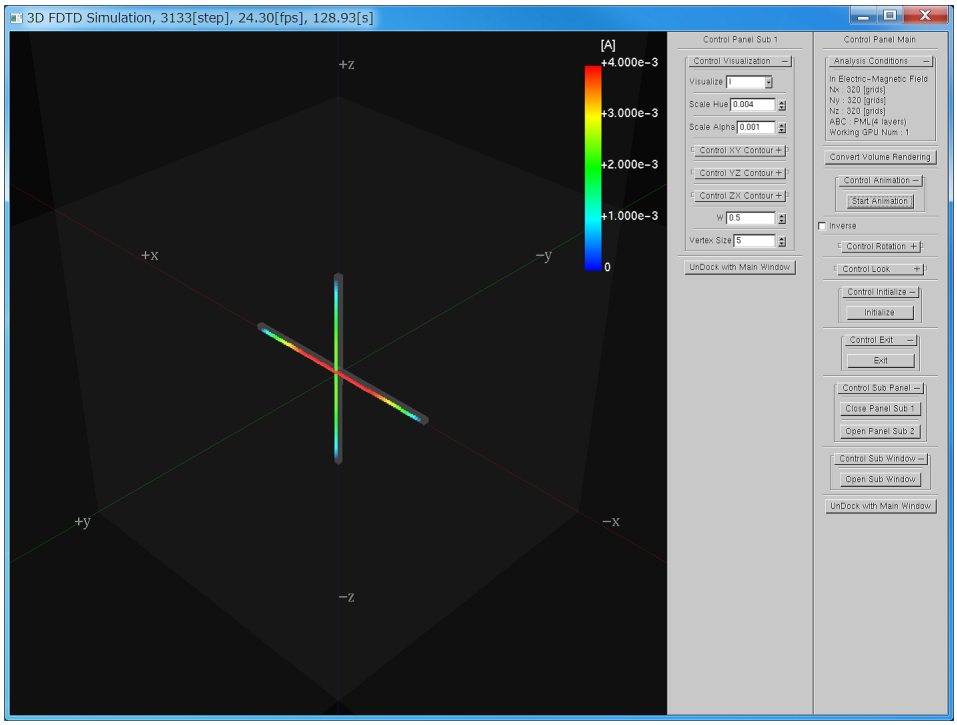

図 5.22　クロスダイポールアンテナ上の電流分布

ヘリカルアンテナ

最後に，ヘリカルアンテナをモデリングし，シミュレーションした例を示す．ヘリカルアンテナは衛星通信等に用いられている．ヘリカルアンテナ解析で用いたパラメータを表 5.7 にまとめる．

表 5.7 ヘリカルアンテナ解析で用いた解析条件

タイムステップ	$\Delta t = 1.0 \times 10^{-12}$ s
グリッドサイズ	$\Delta x = \Delta y = \Delta z = 0.001$ m
アンテナ長（1 周）	0.15 m
ピッチ	12°
反射板半径	0.048 m
給電周波数	2 GHz

ヘリカルアンテナを解析空間の中心に設置した様子を図 5.23 に示す．図 5.24 にヘリカルアンテナから放射される電界の分布を，図 5.25 にヘリカルアンテナ上の電流の分布を示す．

図 5.24 より，ヘリカルアンテナがヘリカル軸方向に指向性を持っていることが確認できる．ヘリカルアンテナもクロスダイポールアンテナと同様に円偏波で電波を放射する．しかし，ヘリカルアンテナの放射の様子を PMCC で可視化する場合，空間のどこを切り取ってもキレイに分かりやすく可視化することは難しい．ヘリカルアンテナの場合はボリュームレンダリングを用いて可視化した方が分かりやすく可視化できる．このことから，形状が複雑なアンテナ等の近傍解はボリュームレンダリング，遠方解は PMCC で可視化を行うのが分かり易いと考える．このように，解析するモデルによって可視化手法を切り替えられるということは重要である．

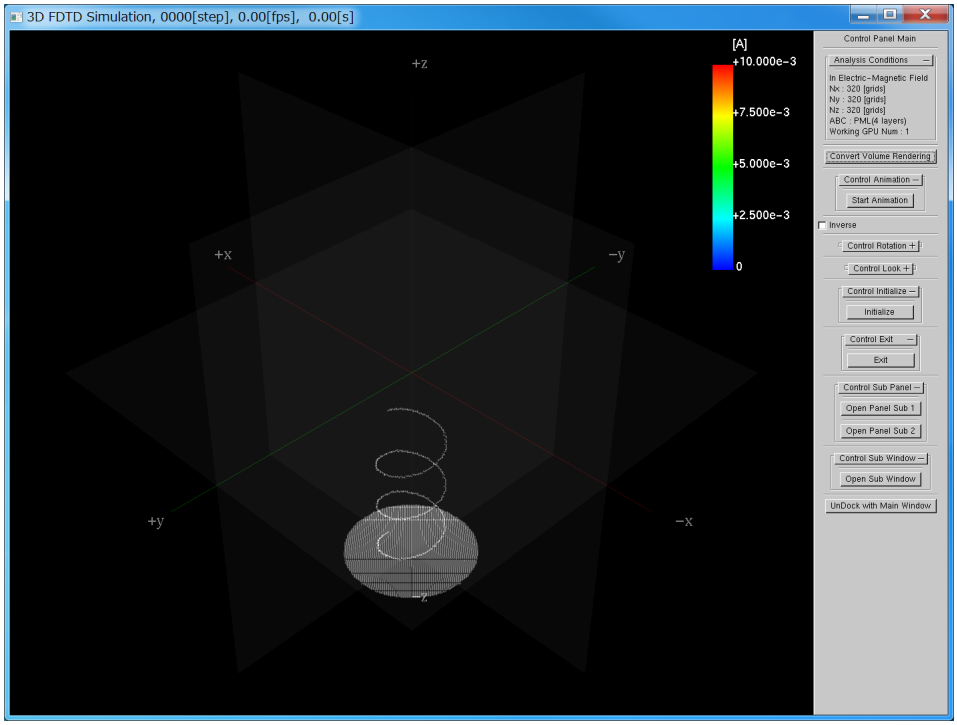

図 5.23　中心にヘリカルアンテナを設置

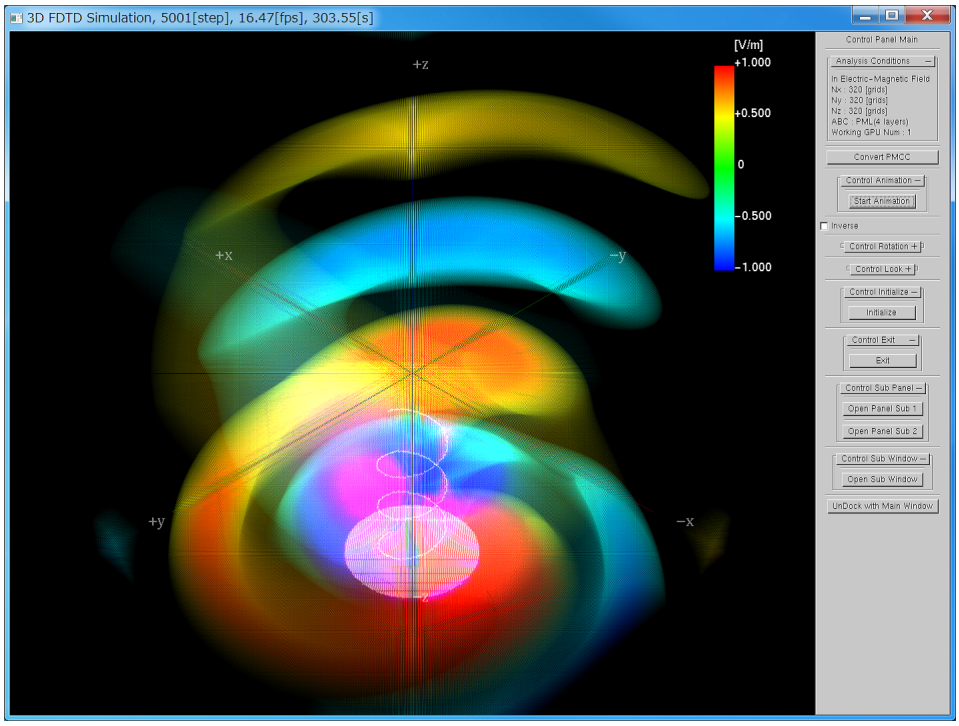

図 5.24　ヘリカルアンテナ周辺の電界分布

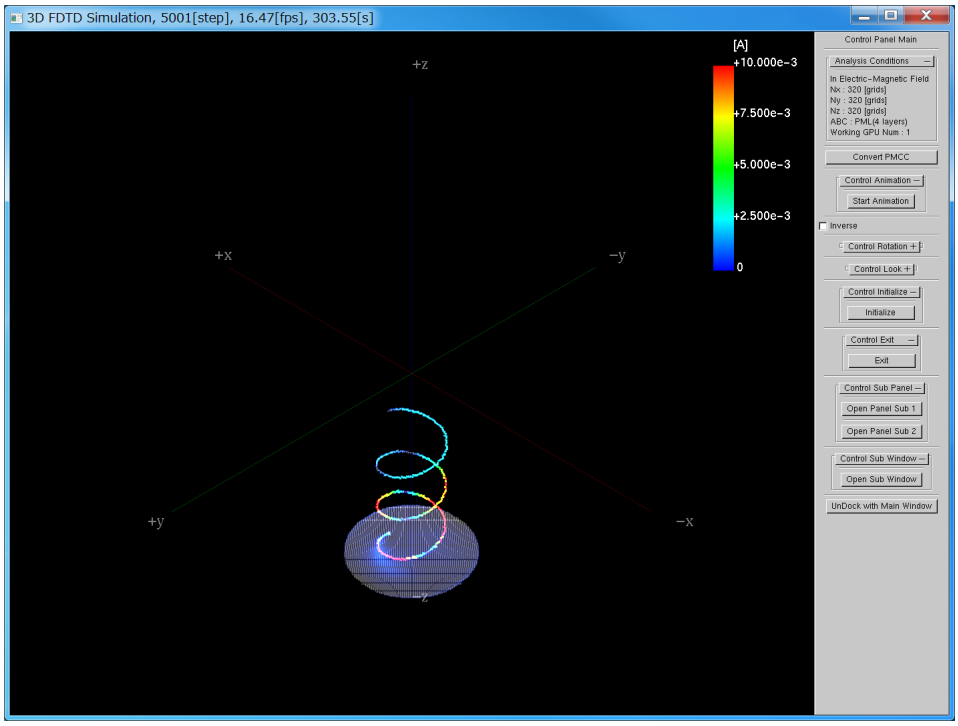

図 5.25　ヘリカルアンテナ上の電流分布

5.4.2　生体電磁界解析への適用

次に，生体電磁界解析におけるインタラクティブシミュレーションの例を示す．生体電磁界解析で用いたパラメータを表 5.8 にまとめる．

表 5.8　生体電磁界解析で用いた解析条件

タイムステップ	$\Delta t = 2.0 \times 10^{-12}$ s
グリッドサイズ	$\Delta x = \Delta y = \Delta z = 0.002$ m
使用周波数	1 GHz

図 5.26 に解析空間の中心に人体モデルを設置した様子を示す．今回は 1GHz の周波数に対応した媒質パラメータを持つ人体モデルデータを使用した．人体モデルをカラーで表示することで骨格・内臓などが分かり易くなるが，電磁界分布を重ねて描画すると電磁界分布の様子を理解することが難しい．そこで，電磁界分布を描画する際には図 5.27 のように白黒で人体モデルを描画する．

人体モデルに対して 1GHz の平面波を正面から曝露させた場合の人体内の電界分布を図 5.28 及び 5.29 に示す．同図より，PMCC を用いることで人体内の電界分布を分かり易く可視化できることが分かった．生体電磁界解析において PMCC は有効な可視化手法であると言える．

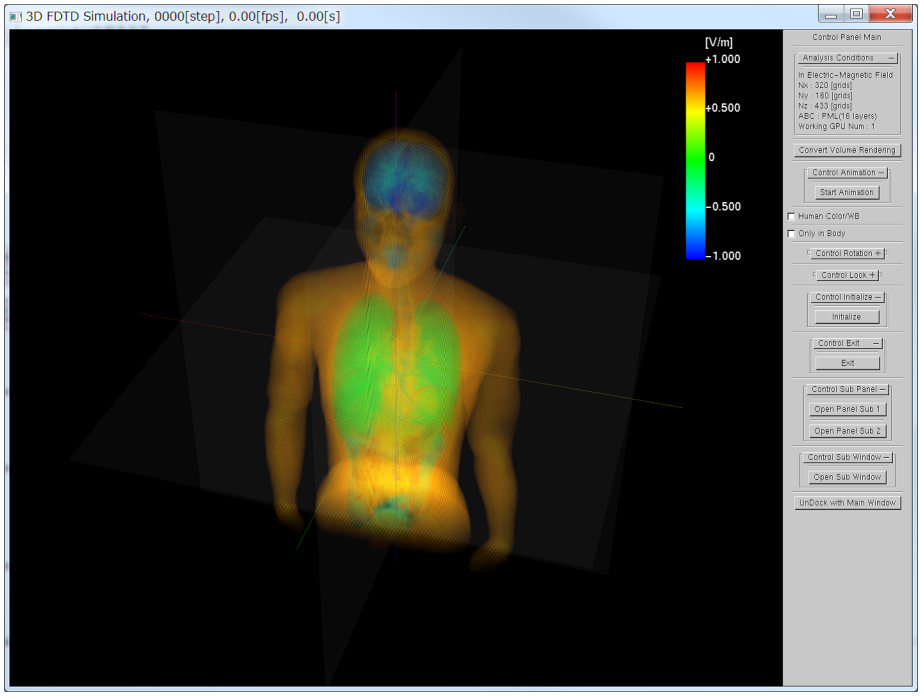

図 5.26　人体モデル（カラー）

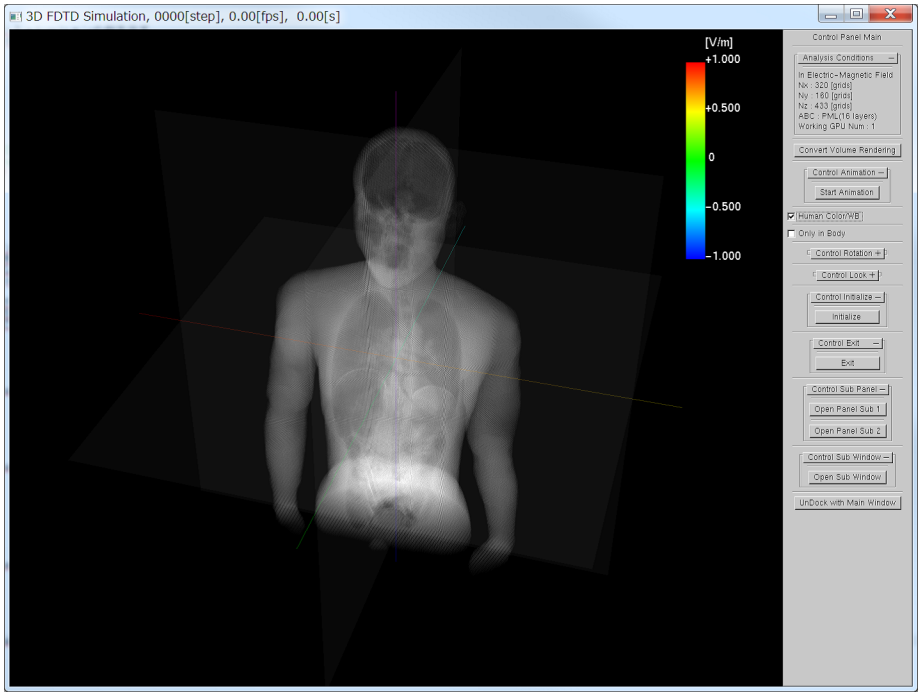

図 5.27　人体モデル（白黒）

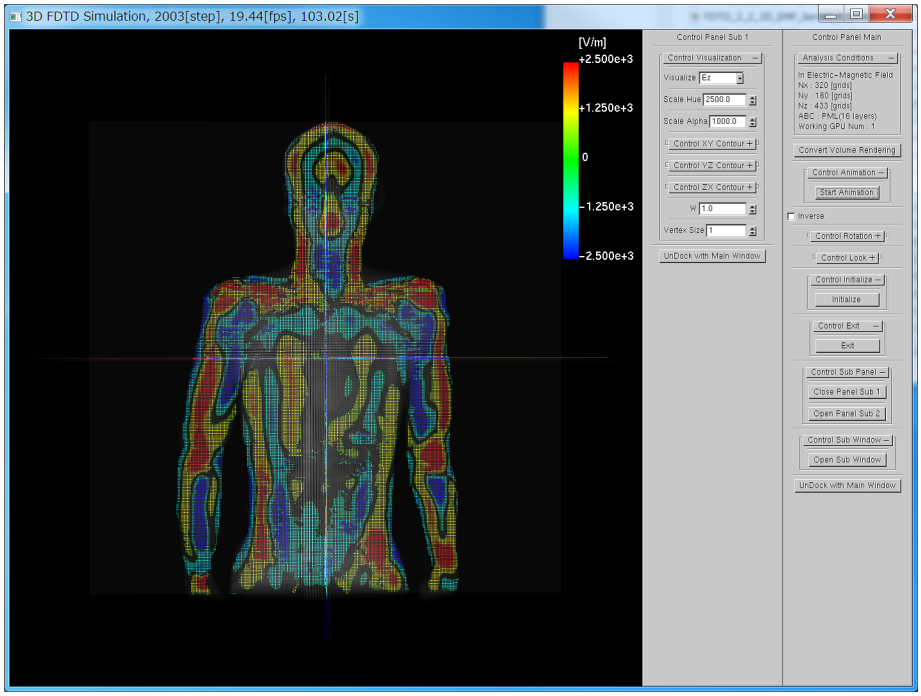

図 5.28　正面から見た人体内電界分布

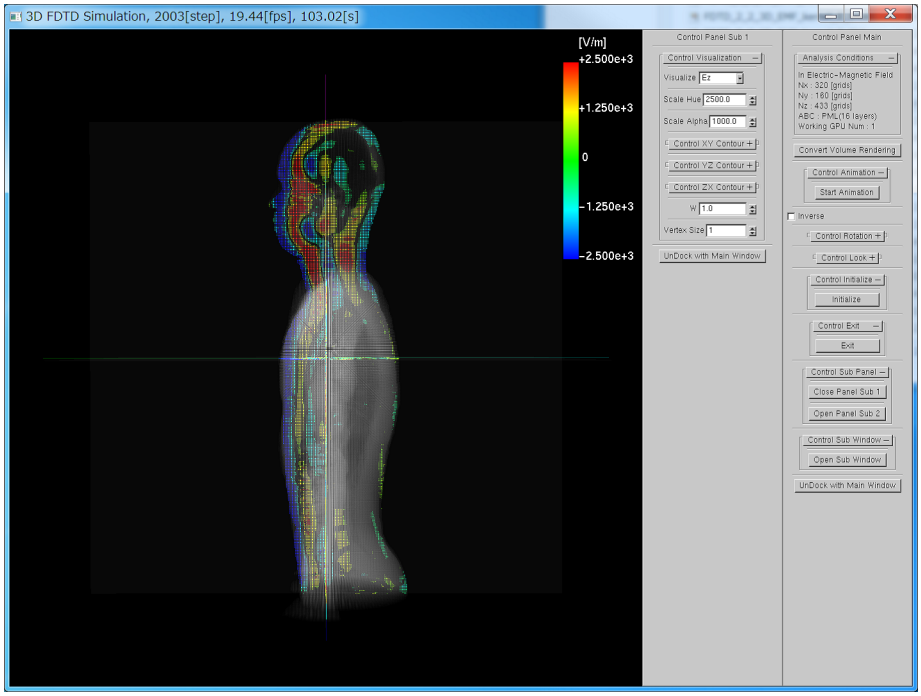

図 5.29　側面から見た人体内電界分布

5.5 インタラクティブシミュレーションの弾性波解析における応用

最後に，インタラクティブシミュレーションの弾性波数値解析における適用について説明する．

5.5.1 弾性波・音場連成解析への適用

ここでは，弾性波・音場連成解析におけるインタラクティブシミュレーションの例を示す．弾性波・音場連成解析で用いたパラメータを表 5.9 及び 5.10 にまとめる．弾性体は木を想定しており，木の周りは空気とする．

表 5.9 弾性波・音場連成解析で用いた解析条件

タイムステップ	$\Delta t = 1.0 \times 10^{-7}$ s
グリッドサイズ	$\Delta x = \Delta y = \Delta z = 0.002$ m

表 5.10 弾性波・音場連成解析で用いた媒質パラメータ

	空気	弾性体
密度 ρ	1.21 kg/m^3	467 kg/m^3
縦波音速 C_l	343 m/s	5114 m/s
横波音速 C_s	0 m/s	3131 m/s

初めに，弾性体を解析空間の中心に設置した様子を図 5.30 に示す．

木に応力を加えたときに生じる音波の様子を図 5.31 に示す．

図から分かるように，PMCC を用いることで弾性体周りに発生した音波を分かりやすく可視化することができる．ボリュームレンダリングを用いた場合，弾性体周辺の音波と伝搬した音波が重なってしまうため，音波伝搬の様子を正確に理解することは難しい．弾性体周りの音波を観測するには PMCC が適していることが分かる．

図 5.32 に弾性体内の応力分布を示す．

図より，PMCC を用いることで弾性体内の応力の伝搬の様子が正確に把握できる．さらに，描画する α 値の閾値を操作することでより小さい応力や大きい応力のみを可視化することができる．弾性体内のような狭い閉空間の分布を可視化するのにボリュームレンダリングは不向きである．

図 5.33 に弾性体内の剪断応力分布を示す.

図より，PMCC を用いることで弾性体内の剪断応力の伝搬の様子が正確に把握できる．剪断応力は平面に対して働く力であり，ボリュームレンダリングで分かりやすく可視化することは難しい.

図 5.34 に粒子速度の分布を示す.

図 5.34 より，弾性体内では空気中と比べて粒子速度が小さいことが分かる．これは空気中と比較して弾性体内は密度 ρ が大きいためである．インタラクティブシミュレーションでは弾性体内の粒子速度，空気中の粒子速度のどちらに焦点を当てるかを変える時に，可視化強度のスケールを変更し即座に適切な可視化に切り替えることができる.

図 5.35 に逆回転計算を行った例を示す.

図から分かるように，逆回転計算を用いることでタイムステップを戻すことができる．このように，インタラクティブシミュレーションではインタラクティブに逆方向計算へ切り替えることができる．これにより，観察したい時点まで伝搬を簡単に巻き戻すことが可能となる.

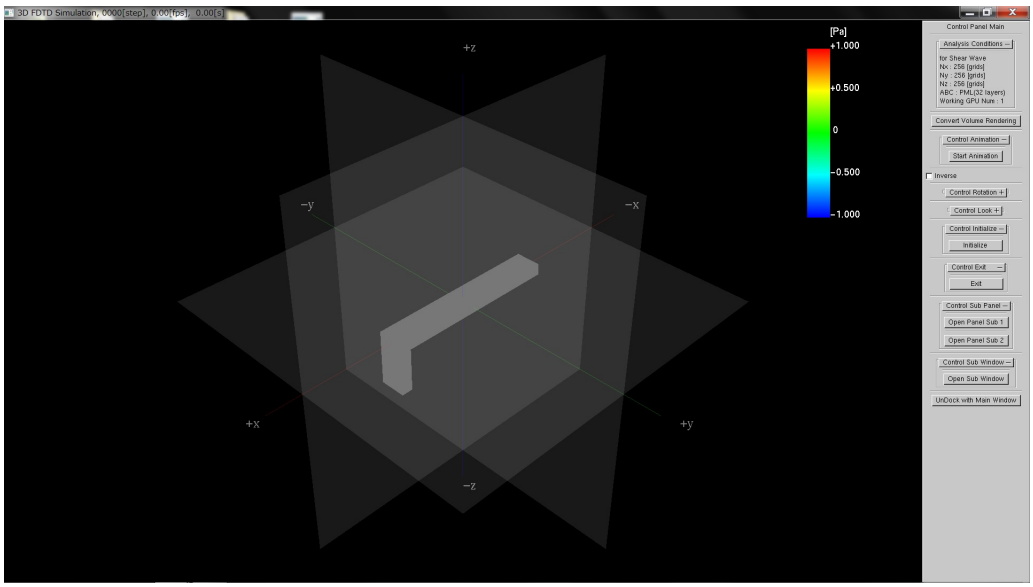

図 5.30　中心に弾性体を設置

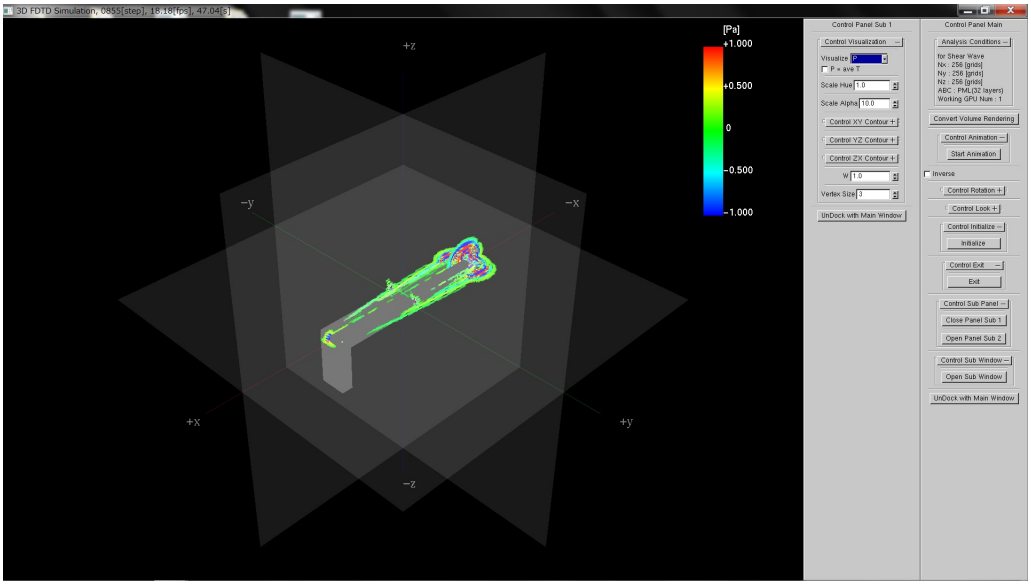

図 5.31　弾性体から生じる音波

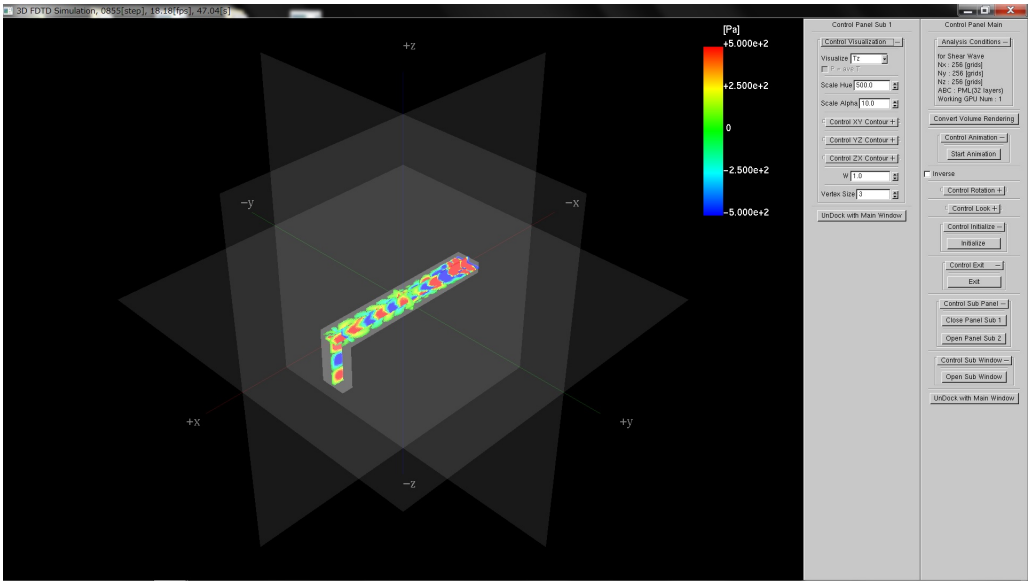

図 5.32　弾性体内の垂直応力分布

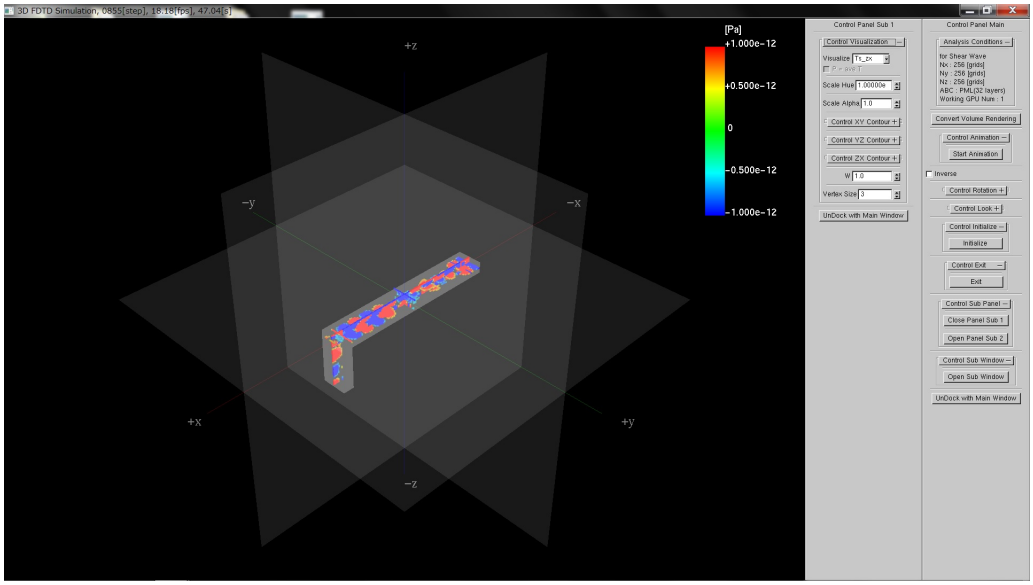

図 5.33　弾性体内の剪断応力分布

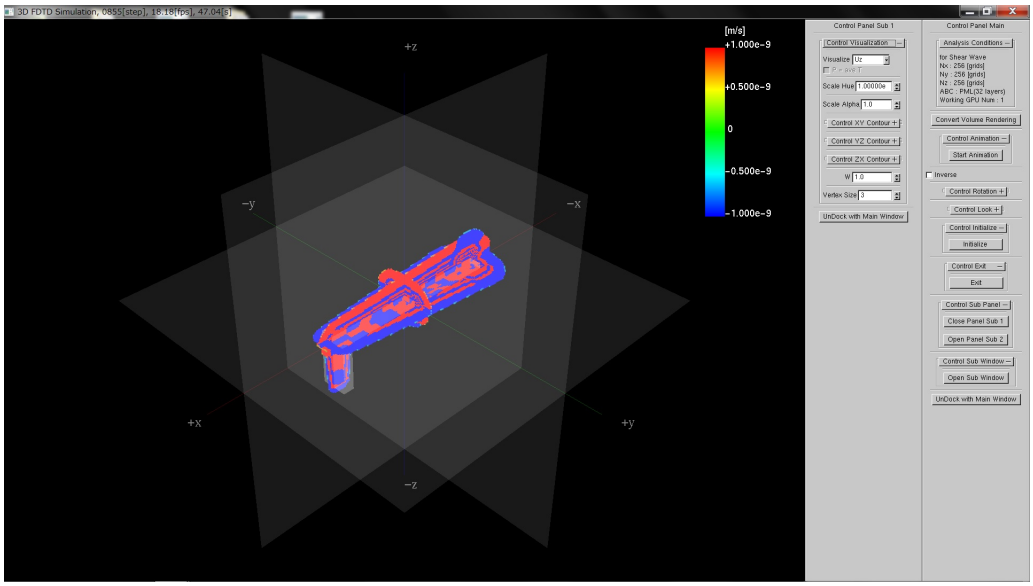

図 5.34　粒子速度の分布

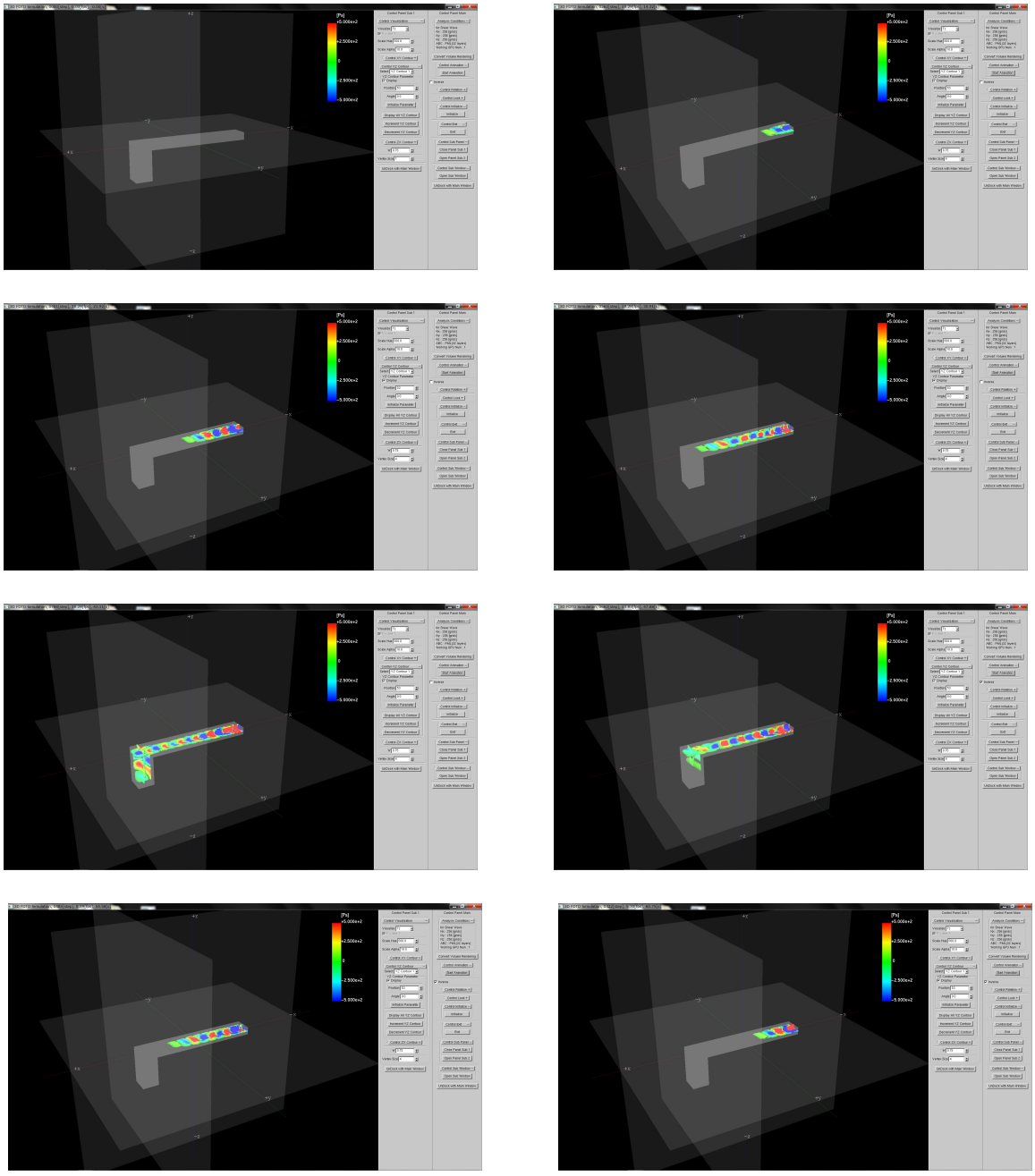

図 5.35　時間領域における順方向計算から逆方向計算へ切り替えた様子

5.6　結言

本章では，GPU 高速計算とリアルタイム可視化を組み合わせたインタラクティブシミュレーションについて述べた．インタラクティブシミュレーションを音場・電磁界・弾性波それぞれに適用し，インタラクティブシミュレーションの有効性を検証した．

その結果，PMCC を用いたインタラクティブシミュレーションは音場・電磁界・弾性波それぞれにおいて有効であることが分かった．PMCC を用いることで高速で分かりやすい可視化が可能であり，パラメータを操作するとすぐさま反映され，まさにインタラクティブなシミュレーションが実現できた．しかし，ボリュームレンダリングを用いることで分かりやすくなる場合もあるので，ボリュームレンダリングのアルゴリズムも今後検討していきたい．

本論文では，FDTD 法を用いてインタラクティブシミュレーションを実装した．波動伝搬数値解析の手法として FDTD 法，MM-MOC 法を GPU に実装し検討を行ったが，可視化速度を考慮して FDTD 法を使用した．今後は MM-MOC 法を用いたインタラクティブシミュレーションも検討していきたい．

第6章　結論

6.1　本研究の総括

本論文では，GPU 実装によるインタラクティブ波動伝搬シミュレーションについて述べた．

2 章では時間領域における波動伝搬数値シミュレーションのための数値解析手法について検討した．FDTD 法の音場・電磁界・弾性波における支配方程式に基づく差分を用いた定式化を説明した．また，CIP 法の音場における支配方程式に基づく移流方程式，エルミート補間を用いた定式化について述べた．そして，一般化したエルミート補間を用いる MM-MOC 法を提案した．FDTD 法と MM-MOC 法で数値分散誤差と数値散逸誤差についての精度比較評価を行った．その結果，同じ離散化条件では MM-MOC 法が FDTD 法と比較してより高精度に解析を行えることが分かった．しかし，同じ離散化条件では MM-MOC 法が FDTD 法と比較して計算時間と使用メモリ量が多く必要なことが分かった．

3 章では GPU の進化と GPU を用いた数値解析の高速化についてまとめた．近年の GPU の著しい性能向上について述べた．GPU 用プログラム開発環境 CUDA についてサンプルコードを交えて解説した．メニーコアプロセッサである GPU のアーキテクチャ，スレッドの階層構造について説明を行った．そして，GPU を用いた波動伝搬数値解析の高速化を行った．GPU を用いた高速化ではプロセッサ－メモリ間のデータ転送を減らすことが重要となるので，共有メモリ・コンスタントメモリ・レジスタを用いたキャッシングなどを駆使して最適化を行った．その結果，FDTD 法では CPU と比較して約 26 倍，MM-MOC 法では CPU と比較して MM(3,1)-MOC 法では約 80 倍，MM(7,1)-MOC 法では約 115 倍の高速化を達成した．さらに，マルチ GPU を用いることでより大きな解析領域をより高速に解析できることも分かった．GPU を用いることで MM-MOC 法は FDTD 法と比較してより大幅に高速化された．しかし，GPU で高速化されても同じ離散化条件では FDTD 法が MM-MOC 法と比較してより高速に計算できることが分かった．インタラクティブシミュレーションではソルバの計算が高速である必要があるので，インタラクティブシミュレーションで用いる解析手法として FDTD 法を選択した．

4 章では 3 次元波動数値解析のための可視化手法の検討と評価を行った．既存の可視化

手法として Contour や Surface Plot を用いた 2 次元的可視化手法，MCC を用いた可視化手法とボリュームレンダリングを用いた可視化手法について検討・評価を行った．既存の可視化手法のメリットとデメリットを考慮してインタラクティブシミュレーションに適した新たな可視化手法として PMCC を提案した．PMCC を視認性，可視化速度，使用メモリ量などに関して既存の可視化手法と比較を行った結果，非常に優れた可視化手法であることを示した．そして，マルチ GPU を用いることで PMCC を用いた可視化がより高速化されることを示し，GPU 間のデータの非同期転送や GPUDirect 2.0 を用いたデータ転送を用いることでさらなる可視化の高速化を達成した．

5 章ではインタラクティブシミュレーションの概要とその応用について説明した．GPU を用いた高速計算とリアルタイム可視化を組み合わせたインタラクティブシミュレーションの概要について説明を行い，インタラクティブシミュレーションにより実現できることを解説した．そして，インタラクティブシミュレーションを実装し，応用について検討を行った．音場，電磁界，弾性波における波動伝搬数値解析についてインタラクティブシミュレーションを実装した．音場波動伝搬数値解析では室内環境をモデリングした室内音場におけるインタラクティブシミュレーションを実装し有効性を検討した．電磁界波動伝搬数値解析ではアンテナ解析と人体モデルを用いた生体電磁界解析においてインタラクティブシミュレーションを実装した．アンテナ解析では様々なアンテナをモデリングしそれぞれについて視認性などを検討した．生体電磁界解析では人体モデルを使用して解析を行い，生体内の電界分布などをインタラクティブシミュレーションによって効果的に可視化することに成功した．弾性波波動伝搬数値解析では音場との連成解析を実装し，インタラクティブシミュレーションの有効性を検討した．音場・電磁界・弾性波それぞれにおいてインタラクティブシミュレーションが有効であることを示した．

6.2 今後の課題

本論文ではインタラクティブシミュレーションに FDTD 法を用いたが，解析する問題の種類によって FDTD 法が有効でない場合がある．MM-MOC 法を用いたインタラクティブシミュレーションを実装することを今後は行いたい．

また，今回は簡単なモデル化でのシミュレーションを行った．今後は，インタラクティブシミュレーションの更なる実用的な応用が望まれる．

GPU コンピューティングは今では多くの人が利用するようになった．しかし，まだ多くの課題が残されている．CUDA の登場により GPU 用プログラムの開発が容易になったとはいえ，開発が難しいと考えている人も多い．だが，最近では OpenACC という既存の

ソースコードに指示文を挿入するだけでGPU用プログラムが開発できる環境も提供されている．開発環境がより容易になることでGPUを利用する人が増え，さらにはインタラクティブシミュレーションもより利用されるようになるだろう．

また，1つのGPUが使用することができるメモリの量が少ないことも問題である．技術の発展により搭載できるメモリの量が増えるか，またはマルチGPUをより容易に利用できる環境の整備が望まれる．

そして，遠い将来にはGPUコンピューティング技術は別の技術に置き換えられているだろう．しかし，技術がどのように発展を遂げても並列計算技術の重要性は変わらないだろう．本論文は並列計算の重要性を示す一つの基盤技術となるだろう．

付 録A　4次精度中心差分を用いたFDTD法の定式化

A.1　4次精度FDTD法による音場波動伝搬数値解析

式 (2.3)，(2.4)，(2.5) 及び (2.6) に，Staggered grid を用いて空間で 4 次精度の中心差分近似，時間で 2 次精度の中心差分近似を適用すると以下の式が得られる．

$$v_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k) = v_x^{n-\frac{1}{2}}(i+\frac{1}{2},j,k) - \frac{1}{24\rho}\frac{\Delta t}{\Delta x}\Big\{ -p^n(i+2,j,k) + p^n(i-1,j,k) + 27\big(p^n(i+1,j,k) - p^n(i,j,k)\big)\Big\} \tag{A.1}$$

$$v_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k) = v_y^{n-\frac{1}{2}}(i,j+\frac{1}{2},k) - \frac{1}{24\rho}\frac{\Delta t}{\Delta y}\Big\{ -p^n(i,j+2,k) + p^n(i,j-1,k) + 27\big(p^n(i,j+1,k) - p^n(i,j,k)\big)\Big\} \tag{A.2}$$

$$v_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2}) = v_z^{n-\frac{1}{2}}(i,j,k+\frac{1}{2}) - \frac{1}{24\rho}\frac{\Delta t}{\Delta z}\Big\{ -p^n(i,j,k+2) + p^n(i,j,k-1) + 27\big(p^n(i,j,k+1) - p^n(i,j,k)\big)\Big\} \tag{A.3}$$

$$\begin{aligned} p^{n+1}(i,j,k) = p^n(i,j,k) &- \frac{K}{24}\frac{\Delta t}{\Delta x}\Big\{-v_x^{n+\frac{1}{2}}(i+\frac{3}{2},j,k) + v_x^{n+\frac{1}{2}}(i-\frac{3}{2},j,k) \\ &+ 27\big(v_x^{n+\frac{1}{2}}(i+\frac{1}{2},j,k) - v_x^{n+\frac{1}{2}}(i-\frac{1}{2},j,k)\big)\Big\} \\ &- \frac{K}{24}\frac{\Delta t}{\Delta y}\Big\{-v_y^{n+\frac{1}{2}}(i,j+\frac{3}{2},k) + v_y^{n+\frac{1}{2}}(i,j-\frac{3}{2},k) \\ &+ 27\big(v_y^{n+\frac{1}{2}}(i,j+\frac{1}{2},k) - v_y^{n+\frac{1}{2}}(i,j-\frac{1}{2},k)\big)\Big\} \\ &- \frac{K}{24}\frac{\Delta t}{\Delta z}\Big\{-v_z^{n+\frac{1}{2}}(i,j,k+\frac{3}{2}) + v_z^{n+\frac{1}{2}}(i,j,k-\frac{3}{2}) \\ &+ 27\big(v_z^{n+\frac{1}{2}}(i,j,k+\frac{1}{2}) - v_z^{n+\frac{1}{2}}(i,j,k-\frac{1}{2})\big)\Big\} \end{aligned} \tag{A.4}$$

A.2 4 次精度 FDTD 法による電磁界波動伝搬数値解析

式 (2.15)，(2.16)，(2.17)，(2.18)，(2.19) 及び (2.20) に，Staggered grid を用いて空間で 4 次精度の中心差分近似，時間で 2 次精度の中心差分近似を適用すると以下の式が得られる．

$$
\begin{aligned}
E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k+\frac{1}{2}) &= E_x^{n-\frac{1}{2}}(i,j+\frac{1}{2},k+\frac{1}{2}) \\
&+ \frac{1}{24\varepsilon}\frac{\Delta t}{\Delta y}\big\{-H_z^n(i,j+2,k+\frac{1}{2}) + H_z^n(i,j-1,k+\frac{1}{2}) \\
&+ 27\big(H_z^n(i,j+1,k+\frac{1}{2}) - H_z^n(i,j,k+\frac{1}{2})\big)\big\} \\
&+ \frac{1}{24\varepsilon}\frac{\Delta t}{\Delta z}\big\{-H_y^n(i,j+\frac{1}{2},k+2) + H_y^n(i,j+\frac{1}{2},k-1) \\
&+ 27\big(H_y^n(i,j+\frac{1}{2},k+1) - H_y^n(i,j+\frac{1}{2},k)\big)\big\}
\end{aligned}
\tag{A.5}
$$

$$
\begin{aligned}
E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) &= E_y^{n-\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) \\
&+ \frac{1}{24\varepsilon}\frac{\Delta t}{\Delta z}\big\{-H_x^n(i+\frac{1}{2},j,k+2) + H_x^n(i+\frac{1}{2},j,k-1) \\
&+ 27\big(H_x^n(i+\frac{1}{2},j,k+1) - H_x^n(i+\frac{1}{2},j,k)\big)\big\} \\
&+ \frac{1}{24\varepsilon}\frac{\Delta t}{\Delta x}\big\{-H_z^n(i+2,j,k+\frac{1}{2}) + H_z^n(i-1,j,k+\frac{1}{2}) \\
&+ 27\big(H_z^n(i+1,j,k+\frac{1}{2}) - H_z^n(i,j,k+\frac{1}{2})\big)\big\}
\end{aligned}
\tag{A.6}
$$

$$
\begin{aligned}
E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) &= E_z^{n-\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) \\
&+ \frac{1}{24\varepsilon}\frac{\Delta t}{\Delta x}\big\{-H_y^n(i+2,j+\frac{1}{2},k) + H_y^n(i-1,j+\frac{1}{2},k) \\
&+ 27\big(H_y^n(i+1,j+\frac{1}{2},k) - H_y^n(i,j+\frac{1}{2},k)\big)\big\} \\
&+ \frac{1}{24\varepsilon}\frac{\Delta t}{\Delta y}\big\{-H_x^n(i+\frac{1}{2},j+2,k) + H_x^n(i+\frac{1}{2},j-1,k) \\
&+ 27\big(H_x^n(i+\frac{1}{2},j+1,k) - H_x^n(i+\frac{1}{2},j,k)\big)\big\}
\end{aligned}
\tag{A.7}
$$

$$
\begin{aligned}
H_x^{n+1}(i+\frac{1}{2},j,k) &= H_x^n(i+\frac{1}{2},j,k) \\
&\quad + \frac{1}{24\mu}\frac{\Delta t}{\Delta y}\big\{-E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j+\frac{3}{2},k) + E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j-\frac{3}{2},k) \\
&\quad + 27\big(E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) - E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j-\frac{1}{2},k)\big)\big\} \\
&\quad + \frac{1}{24\mu}\frac{\Delta t}{\Delta z}\big\{-E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{3}{2}) + E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k-\frac{3}{2}) \\
&\quad + 27\big(E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) - E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k-\frac{1}{2})\big)\big\}
\end{aligned} \tag{A.8}
$$

$$
\begin{aligned}
H_y^{n+1}(i,j+\frac{1}{2},k) &= H_y^n(i,j+\frac{1}{2},k) \\
&\quad + \frac{1}{24\mu}\frac{\Delta t}{\Delta z}\big\{-E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k+\frac{3}{2}) + E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k-\frac{3}{2}) \\
&\quad + 27\big(E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k+\frac{1}{2}) - E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k-\frac{1}{2})\big)\big\} \\
&\quad + \frac{1}{24\mu}\frac{\Delta t}{\Delta x}\big\{-E_z^{n+\frac{1}{2}}(i+\frac{3}{2},j+\frac{1}{2},k) + E_z^{n+\frac{1}{2}}(i-\frac{3}{2},j+\frac{1}{2},k) \\
&\quad + 27\big(E_z^{n+\frac{1}{2}}(i+\frac{1}{2},j+\frac{1}{2},k) - E_z^{n+\frac{1}{2}}(i-\frac{1}{2},j+\frac{1}{2},k)\big)\big\}
\end{aligned} \tag{A.9}
$$

$$
\begin{aligned}
H_z^{n+1}(i,j,k+\frac{1}{2}) &= H_z^n(i,j,k+\frac{1}{2}) \\
&\quad + \frac{1}{24\mu}\frac{\Delta t}{\Delta x}\big\{-E_y^{n+\frac{1}{2}}(i+\frac{3}{2},j,k+\frac{1}{2}) + E_y^{n+\frac{1}{2}}(i-\frac{3}{2},j,k+\frac{1}{2}) \\
&\quad + 27\big(E_y^{n+\frac{1}{2}}(i+\frac{1}{2},j,k+\frac{1}{2}) - E_y^{n+\frac{1}{2}}(i-\frac{1}{2},j,k+\frac{1}{2})\big)\big\} \\
&\quad + \frac{1}{24\mu}\frac{\Delta t}{\Delta y}\big\{-E_x^{n+\frac{1}{2}}(i,j+\frac{3}{2},k+\frac{1}{2}) + E_x^{n+\frac{1}{2}}(i,j-\frac{3}{2},k+\frac{1}{2}) \\
&\quad + 27\big(E_x^{n+\frac{1}{2}}(i,j+\frac{1}{2},k+\frac{1}{2}) - E_x^{n+\frac{1}{2}}(i,j-\frac{1}{2},k+\frac{1}{2})\big)\big\}
\end{aligned} \tag{A.10}
$$

付 録B　MM-MOC法の精度評価に関する追加検討

3 階微分値までを用いて補間を行う MM(7,3)-MOC 法を実装し，数値散逸誤差・数値分散誤差に関して他の MM-MOC 法との比較を行う．

3 階微分値まで使用する MM(7,3)-MOC 法は

$$\begin{aligned} F_{x\pm}^{n+1}(i) &= \sum_{k=i\mp1}^{i} h_k^{(0)}(x)\, F_{x\pm}^{n} + \sum_{k=i\mp1}^{i} h_k^{(1)}(x)\, G_{x\pm}^{n} \\ &+ \sum_{k=i\mp1}^{i} h_k^{(2)}(x)\, H_{x\pm}^{n} + \sum_{k=i\mp1}^{i} h_k^{(3)}(x)\, I_{x\pm}^{n} \end{aligned} \tag{B.1}$$

ただし，

$$I_{x\pm} = \partial_{xxx}p \pm Z\partial_{xxx}v_x \tag{B.2}$$

のようになる．

さらに，$n+1$ における p と v_x とその微分値は以下の式で求められる．

$$p^{n+1}(i) \leftarrow \frac{F_+^{n+1}(i) + F_-^{n+1}(i)}{2} \tag{B.3}$$

$$v_x^{n+1}(i) \leftarrow \frac{F_+^{n+1}(i) - F_-^{n+1}(i)}{2Z} \tag{B.4}$$

$$\partial_x p^{n+1}(i) \leftarrow \frac{G_+^{n+1}(i) + G_-^{n+1}(i)}{2} \tag{B.5}$$

$$\partial_x v_x^{n+1}(i) \leftarrow \frac{G_+^{n+1}(i) - G_-^{n+1}(i)}{2Z} \tag{B.6}$$

$$\partial_{xx} p^{n+1}(i) \leftarrow \frac{H_+^{n+1}(i) + H_-^{n+1}(i)}{2} \tag{B.7}$$

$$\partial_{xx} v_x^{n+1}(i) \leftarrow \frac{H_+^{n+1}(i) - H_-^{n+1}(i)}{2Z} \tag{B.8}$$

$$\partial_{xxx} p^{n+1}(i) \leftarrow \frac{I_+^{n+1}(i) + I_-^{n+1}(i)}{2} \tag{B.9}$$

$$\partial_{xxx} v_x^{n+1}(i) \leftarrow \frac{I_+^{n+1}(i) - I_-^{n+1}(i)}{2Z} \tag{B.10}$$

初めに，数値散逸誤差についての比較を行う．図 B.1 に方位角ごとの数値散逸誤差を比較した図を示す．ただし同図は PPW $\fallingdotseq 10$ の場合を示している．横軸が波源から観測点への方位角，縦軸がエネルギーの減衰量を表す．そして，図 B.1 から MM(3,1)-MOC 法を除いたグラフを図 B.2 に示す．

さらに，図 B.3 に PPW ごとの数値散逸誤差を比較した図を示す．ただし同図は方位角が 0° の場合を示している．横軸が PPW，縦軸がエネルギーの減衰量を表す．そして，図 B.3 から MM(3,1)-MOC 法を除いたグラフを図 B.4 に示す．

図 B.2 と図 B.4 から分かるように，MM(7,3)-MOC 法は MM(5,2)-MOC 法や MM(7,1)-MOC 法と比較して高い精度で計算を行えることが分かった．方位角ごとの数値散逸誤差は低減し，PPW が小さい高周波数成分においても他の手法と比較して数値散逸誤差が大きく改善した．MM(7,3)-MOC 法を用いることで数値散逸誤差が改善されることが分かった．

次に，数値分散誤差についての比較を行う．図 B.5 に方位角ごとの数値分散誤差を比較した図を示す．ただし同図は PPW $\fallingdotseq 10$ の場合を示している．横軸が波源から観測点への方位角，縦軸が位相速度がどれだけずれているかを表す．そして，図 B.5 から MM(3,1)-MOC 法を除いたグラフを図 B.6 に示す．

さらに，図 B.7 に PPW ごとの数値分散誤差を比較した図を示す．ただし同図は方位角が 0° の場合を示している．横軸が PPW，縦軸が位相速度がどれだけずれているかを表す．そして，図 B.7 から MM(3,1)-MOC 法を除いたグラフを図 B.8 に示す．

図 B.6 と図 B.8 から分かるように，MM(7,3)-MOC 法は MM(5,2)-MOC 法や MM(7,1)-MOC 法と比較して高い精度で計算を行えることが分かった．方位角ごとの数値分散誤差に関しては改善は見られなかったが，PPW が小さい高周波数成分においては他の手法と比較して数値分散誤差が大きく改善した．MM(7,3)-MOC 法を用いることで数値分散誤差が改善されることが分かった．

以上のことから，3 階微分値まで使用する MM(7,3)-MOC 法は数値散逸誤差・数値分散誤差の両方において他の MM-MOC 法と比較して精度が高い手法であることが分かった．

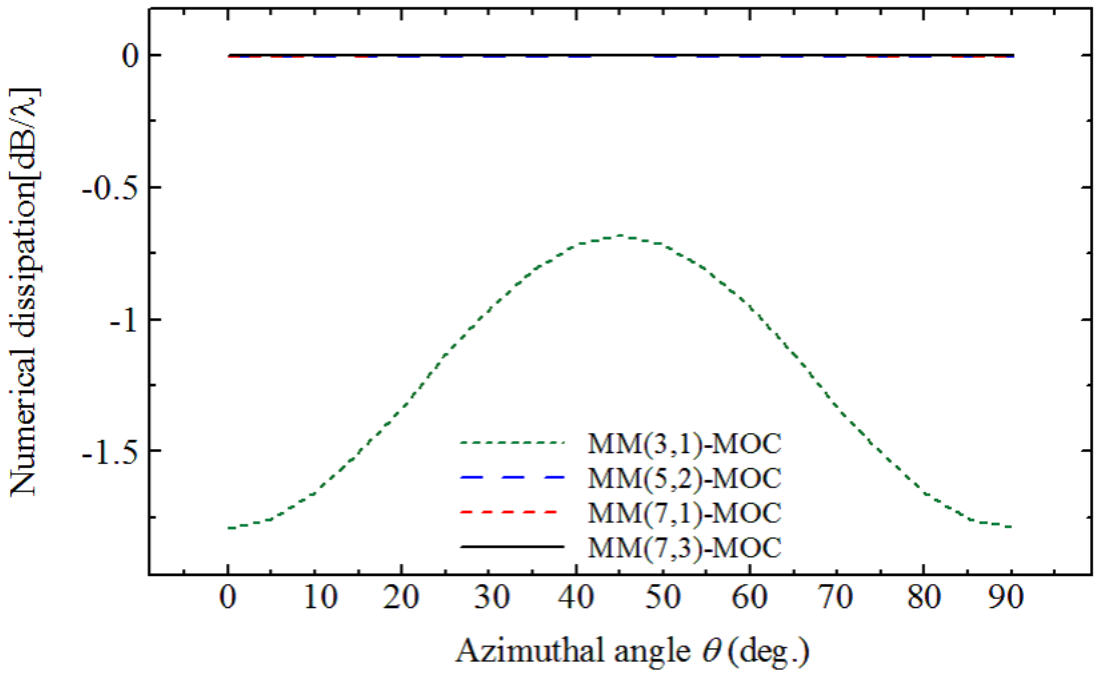

図 B.1 方位角ごとの数値散逸誤差

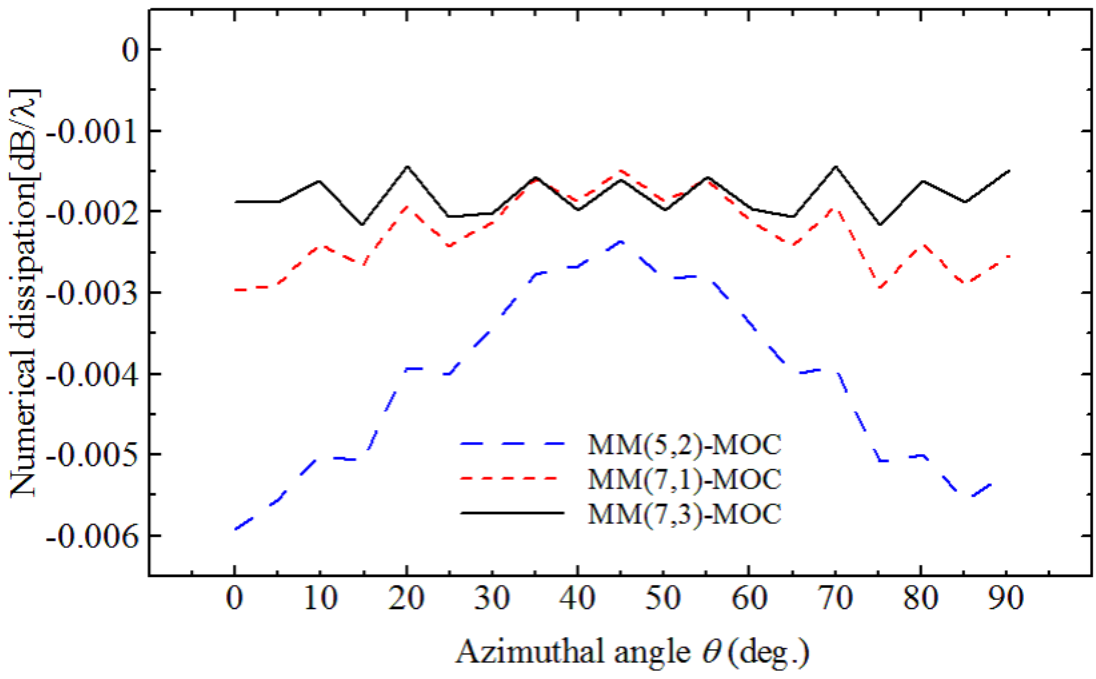

図 B.2 方位角ごとの数値散逸誤差 (MM(3,1)-MOC 法を除く)

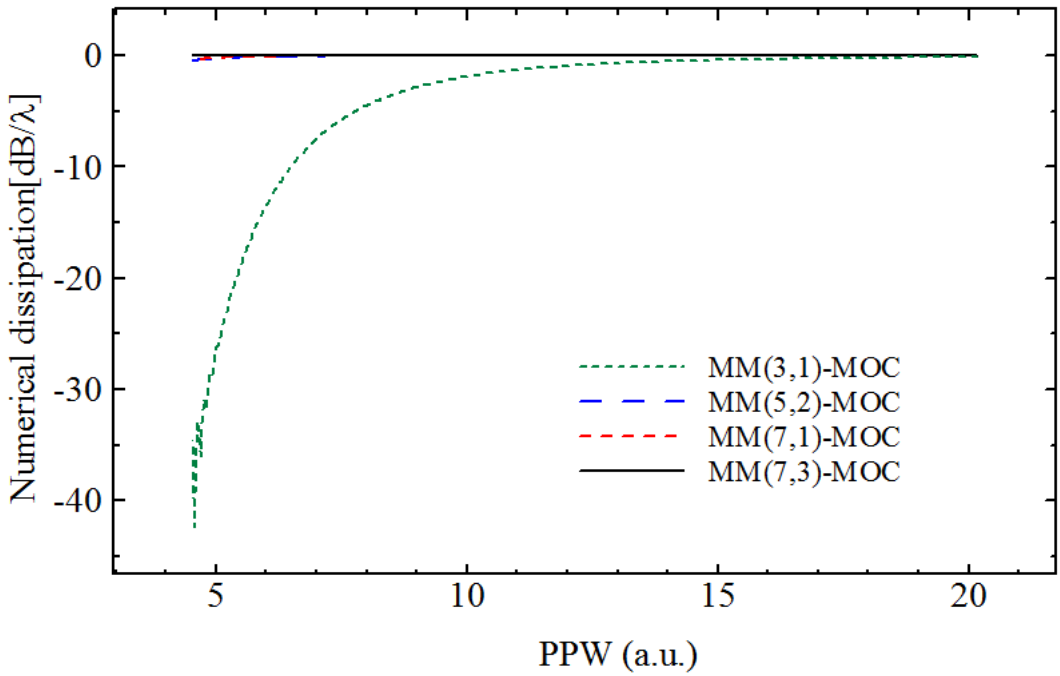

図 B.3　PPW ごとの数値散逸誤差

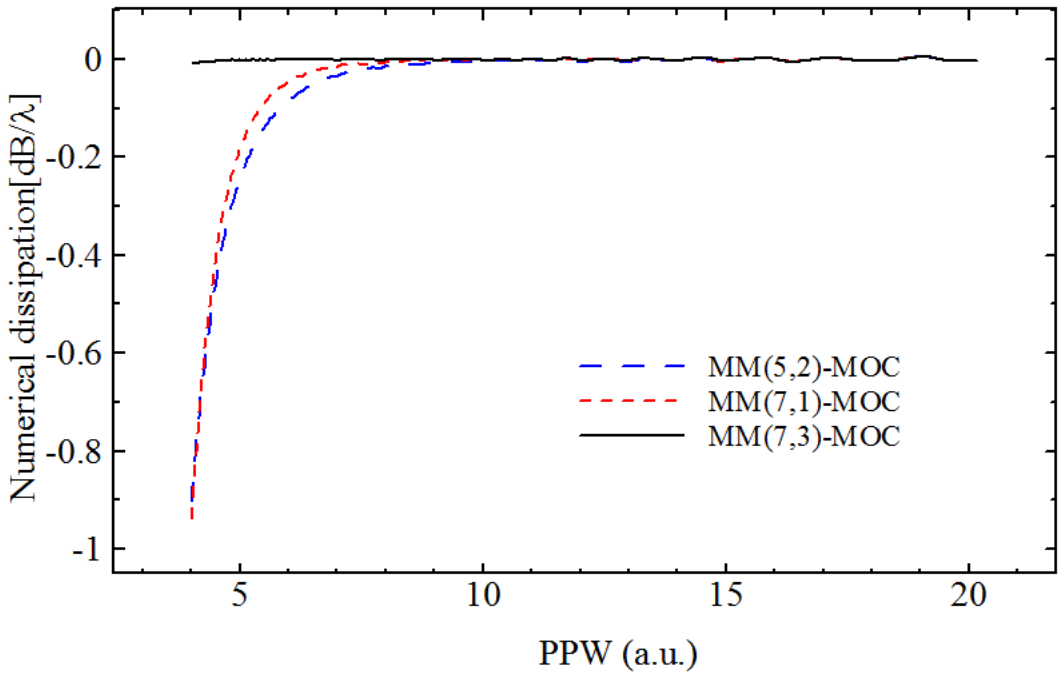

図 B.4　PPW ごとの数値散逸誤差 (MM(3,1)-MOC 法を除く)

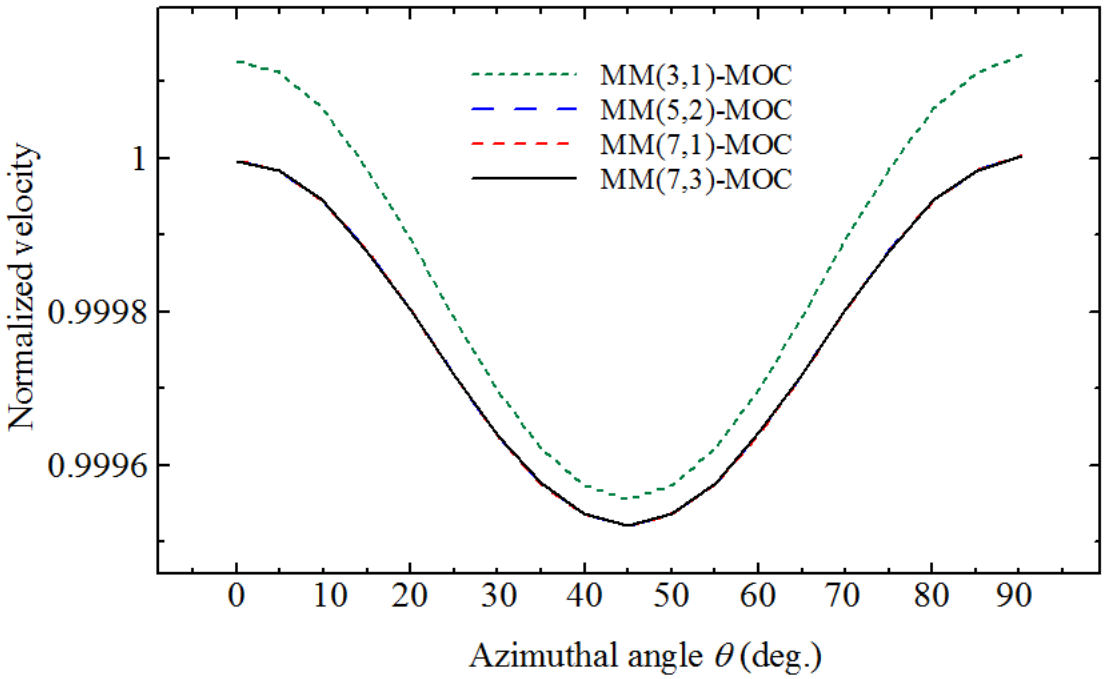

図 B.5 方位角ごとの数値分散誤差

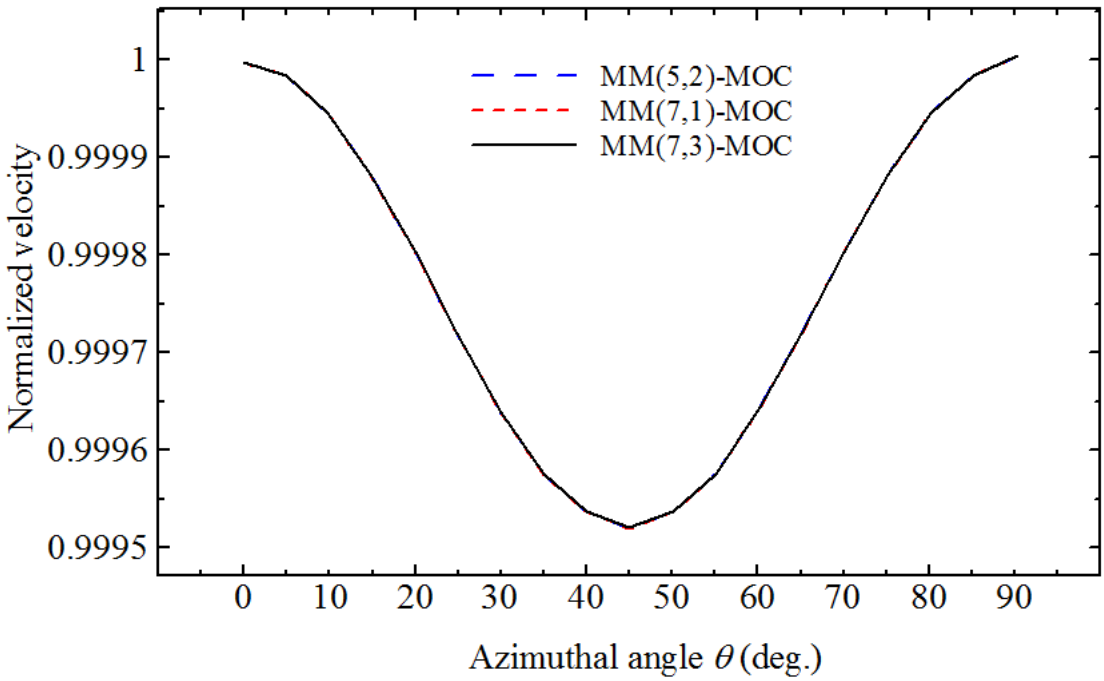

図 B.6 方位角ごとの数値分散誤差 (MM(3,1)-MOC 法を除く)

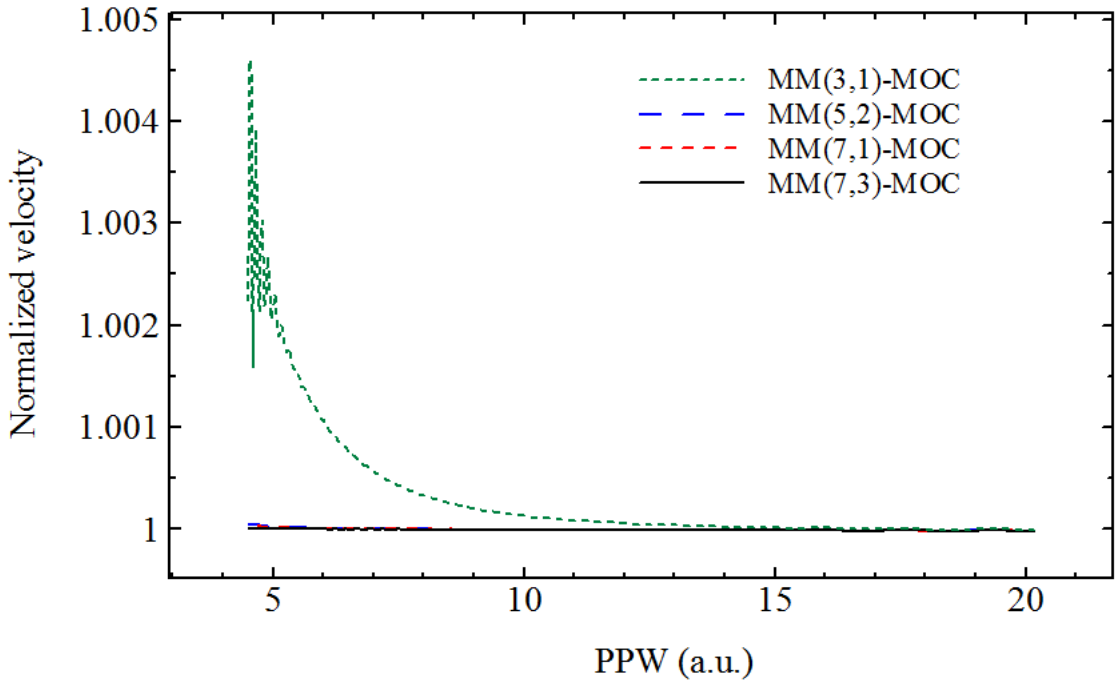

図 B.7　PPW ごとの数値分散誤差

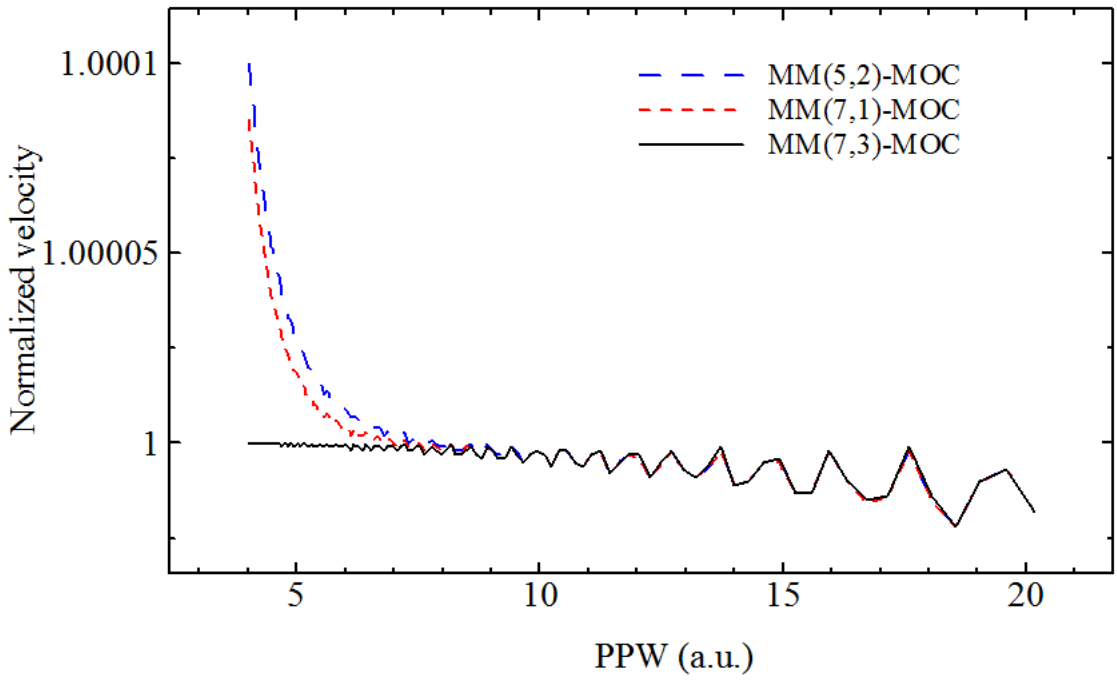

図 B.8　PPW ごとの数値分散誤差 (MM(3,1)-MOC 法を除く)

謝辞

本研究を遂行し学位論文をまとめるに当たり，多くのご支援とご指導を賜りました指導教官である本学大学院システムデザイン研究科情報通信システム学域 大久保寛准教授に深く感謝いたします.

また，本論文作成に当たり審査委員として多くのご助言をいただきました，本学大学院システムデザイン研究科情報通信システム学域 田川憲男教授，本学大学院システムデザイン研究科情報通信システム学域 福本聡教授，本学大学院理工学研究科電気電子工学専攻 鈴木敬久准教授，同志社大学理工学部情報システムデザイン学科 土屋隆生教授に深く感謝いたします.

新しい可視化手法を開発するに当たり，貴重なアドバイスをいただきました清水建設株式会社 石塚崇様に深く感謝いたします.

本論文の一部は，一般財団法人テレコム先端技術研究支援センター様よりの研究奨励金によります.

そして多くのご指摘を下さいました本学大学院システムデザイン研究科情報通信システム学域 大久保研究室，田川研究室の皆様には深く感謝しております.

最後に，温かく見守りそして支援してくださった両親に対して深い感謝の意を表して謝辞といたします.

参考文献

[1] http://www.jamstec.go.jp/es/jp/publication/pdf/Development_ES.pdf

[2] 青木尊之，情報処理学会誌, Vol.50, No.2, pp.107-115，2009.

[3] 河田直樹, 大久保寛, 田川憲男, 土屋隆生，“音響数値計算のための GPU によるパーソナルスーパーコンピュータの実現へ向けて，” 音響春季，2010 年 3 月.

[4] 河田直樹，大久保寛，土屋隆生，信学論 B，“GPU 計算を用いた時間領域電磁界数値解析の高速化性能に関する比較，”Vol.J94-B,No.3,pp.480–483,Mar. 2011.

[5] T. Tsuchiya and H. Sekoguchi, Journal of the Japan Society for Simulation Technology, 27, 4, pp.245-254, 2008.

[6] http://www.nvidia.co.jp/object/cuda_home_new_jp.html

[7] 青木尊之，額田彰，はじめての CUDA プログラミング，工学社，2009

[8] http://docs.nvidia.com/cuda/cuda-c-programmingguide/

[9] K.S. Yee, IEEE Trans. Antennas Propag., vol.AP- 14, no.4, pp.302-307, May 1966.

[10] 宇野　亨，FDTD 法による電磁界およびアンテナ解析，コロナ社，1998 .

[11] 計算電磁気学，電気学会編，東京，2003 .

[12] Toyoda, Masahiro, et al. ”Finite-difference time-domain method for heterogeneous orthotropic media with damping.” Acoustical Science and Technology 33.2 (2012): 77-85.

[13] Toyoda, Masahiro, and Daiji Takahashi. ”Prediction for architectural structure-borne sound by the finite-difference time-domain method.” Acoustical science and technology 30.4 (2009): 265-276.

[14] T. Tsuchiya and A. Kumagai: Jpn. J. Appl. Phys. 48 (2009) 07GN02.

[15] T. Yoda, K. Okubo, N. Tagawa, and T. Tsuchiya: Jpn. J. Appl. Phys. 50 (2011) 07HC12.

[16] H. Kudo, T. Kashiwa, T. Ohtani, ” The non-standard FDTD method for three-dimensional acoustic analysis and its numerical dispersion and stability condition, ” IEICE Trans., vol. J84-A, no. 6, pp. 736-744, June 2001.

[17] O. Chiba, T. Kashiwa, H. Shimoda, S. Kagami, I. Fukai ”Analysis of sound fields in three dimensional space by the time-dependent finite-difference method, ”Acoustical Science and Technology vol. 49 no. 8 pp. 551-562, Aug. 1993.

[18] S. K. Lele: J. Comput. Phys. 103 (1992) 16.

[19] Y. Tanaka, T. Tsuchiya, and N. Endoh: Jpn. J. Appl. Phys. 41 (2002) 3297.

[20] Takashi Yabe, Feng Xiao, Takayuki Utsumi, Jour. of Comput. Phys. 169, pp556–593, 2001.

[21] 矢部 孝，内海隆行，尾形陽一，CIP 法，森北出版，2003 .

[22] K. Okubo, S. Oh, T. Tsuchiya, and N. Takeuchi: Acoust. Sci. Technol. 30 (2009) 132.

[23] K. Okubo, S. Oh, T. Tsuchiya and N. Takeuchi: IEICE Trans. Fundamentals, vol. E90-A, No. 9, pp.2000-2005, Sept. 2007.

[24] K. Okubo and N. Takeuchi: IEEE Trans. Antennas Propag., vol. 55, No. 1, pp.111-119, Jan. 2007.

[25] 大久保寛，竹内伸直,‘‘ CIP 法による線電流源から発生する電磁界の数値解析, ” 信学技報，A・P2004-336 ’，March 2005 ,

[26] T. Tsuchiya, K. Okubo and N. Takeuchi: Jpn. J. Appl. Phys., 47 (2008) pp. 3952-3958

[27] M. Konno, K. Okubo, T. Tsuchiya, and N. Tagawa: Jpn. J. Appl. Phys. 47 (2008) 3962.

[28] M. Konno, K. Okubo, T. Tsuchiya, and N. Tagawa, Jpn. J. Appl. Phys. 48 (2009) 07GN01

[29] Y. Ara, K. Okubo, N. Tagawa, T. Tsuchiya, and T. Ishizuka: Jpn. J. Appl. Phys. 50 (2011) 07HC20.

[30] http://www.opengl.org/

[31] 河田直樹，大久保寛，田川憲男，土屋隆生，石塚崇，AI2010-5-03, 2011 年 2 月.

[32] 土屋隆生, 石井琢人, and 大久保寛. "波動方程式に基づく FDTD 法 (WE-FDTD 法) による音響レンダリング (医用超音波, アコースティックイメージング, 一般)." 電子情報通信学会技術研究報告. US, 超音波 111.88 (2011): 25-30.

[33] http://www.nvidia.co.jp/object/product_geforce_gtx_285_jp.html

[34] http://www.nvidia.co.jp/object/product-geforce-gtx-580-jp.html

[35] http://www.nvidia.co.jp/object/geforce-gtx-titan-jp.html

[36] http://freeglut.sourceforge.net/