FDTD 法に基づく声道音響解析における GPGPU 導入の効果*

☆福庭隆弘 (甲南大院), 北村達也 (甲南大)

1 はじめに

近年，差分法の一つである時間領域差分 (finite difference time domain, FDTD) 法は様々な分野のシミュレーションに利用されるようになっている．FDTD 法は比較的計算量が多い数値計算であり，特に 3 次元空間のシミュレーションを行う場合には非常に時間を要する手法であった．しかし，昨今の計算機環境の高性能化に伴い計算に要する時間も年々短縮され，現実的な時間で処理が行えるようになってきた．この背景には計算機のハードウェア面における性能向上とソフトウェア面における並列計算開発環境の普及の 2 つが大きく寄与している．ハードウェアの面では，メモリの大容量化，CPU コアの複数化, General Purpose computing on Graphics Processing Units (GPGPU) の登場などがある．CPU や GPU のコアが複数個搭載された計算機を用いることによって各プロセッサに処理を分割することができるようになり並列演算が可能になった．また，それぞれに並列演算のための開発環境が用意され, 以前より開発が比較的容易になってきている．本研究は，FDTD 法で計算する 3 次元声道音響シミュレータを 2 つの並列演算方法を用いて高速化し，計算時間の比較を行った．

2 3 次元音場シミュレーション

2.1 音響 FDTD 法

音響 FDTD 法とは音響の波動方程式を Yee アルゴリズム [1] で時間的，空間的に中心差分した差分法の一つである．FDTD 法の長所としてアルゴリズムが比較的簡単でプログラムがしやすいという点が挙げられる．短所として計算機のメモリを大量に使用し，計算時間を要するという点があり，特に 3 次元空間を扱う場合には大きな課題となる．以下に 1 次元空間における波動方程式を示す．

$$\frac{\partial p}{\partial t} = -\kappa \frac{\partial u}{\partial x}$$

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho} \frac{\partial p}{\partial x}$$

ここで p は音圧，u は粒子速度，κ は弾性体積率，t は時間，ρ は媒質の密度である．FDTD 法は上式を以下のように時間的，空間的に差分化したものである．

$$\begin{aligned} p^{n+\frac{1}{2}}(i) &= p^{n-\frac{1}{2}}(i) \\ &- \kappa \frac{\Delta t}{\Delta x} \left\{ u^n\left(i+\frac{1}{2}\right) - u^n\left(i-\frac{1}{2}\right) \right\} \end{aligned}$$

$$\begin{aligned} u^{n+1}\left(i+\frac{1}{2}\right) &= u^n\left(i+\frac{1}{2}\right) \\ &- \frac{1}{\rho} \frac{\Delta t}{\Delta x} \left\{ p^{n+\frac{1}{2}}(i+1) - p^{n+\frac{1}{2}}(i) \right\} \end{aligned}$$

ここで Δt は離散時間間隔，Δx は離散空間間隔，n は時間ステップ，i は空間離散点である．3 次元の式の場合は，この式に各軸方向の粒子速度の式が追加される．

2.2 吸収境界条件

解析領域が剛壁に囲まれている場合は，境界の粒子速度を 0 とすればよい．しかし，自由空間を含んだモデルを解析する場合は，無反射の境界 (吸収境界条件) を擬似的に設定する必要がある．以下に代表的な手法の 1 つである Mur の 1 次吸収境界条件を示す [2].

$$\begin{aligned} u^{n+1}(N) &= u^n(N-1) \\ &+ \frac{c\Delta t - \Delta x}{c\Delta t + \Delta x} \left\{ u^{n+1}(N-1) - u^n(N) \right\} \end{aligned}$$

ここで N は吸収境界条件を設定する点，c は音速である．3 次元空間で設定する場合は斜め入射に対する反射を抑制する必要があるため，上式を拡張した Mur の 2 次吸収境界を設定する．

3 GPU を用いた並列演算

GPU は，CPU よりも並列性が非常に高いデバイスである．この理由はアーキテクチャの違いにあり，メインメモリとの接続を比較してみると CPU では 64 bit，GPU では 512 bit 程度で，GPU ではデータ転送が高速にできる．

NVIDIA 製の GPU ボードは，主に GPU チップとデバイスメモリから構成されている．GPU チップの内部には Streaming Multiprocessor (SM) というマルチプロセッサが多数搭載されており，この SM には Streaming Processor (SP) という演算処理ユニット 8 個と共有メモリ 1 個が入っている．代表的な GPU では SM が 30 個，SP が 240 個入っている．これは，

*Acceleration by GPGPU for acoustic analysis of the vocal tract by FDTD method. by FUKUBA, Takahiro, KITAMURA, Tatsuya (Konan University)

1.4 GHz 程度の演算処理ユニットが 240 個あることになる.

また，NVIDIA 製の GPU を利用するための開発環境として CUDA が提供されている．CUDA における開発ではデバイス (GPU) 上で計算する処理を行う際に特有の記述が必要である．例えばホスト (CPU) で計算した結果をデバイスに転送する場合，デバイスにメモリを確保することやデバイスに結果を転送する際に API を用いる．API が用意されているとはいえハードウエアを意識したプログラミングが必要である. 高速化を行うにあたってはホストデバイス間のメモリ転送をできるだけ抑えることや効率的なメモリアクセスを行うことが重要である.

4 並列化の効果の評価実験

4.1 条件

声道形状は ATR 母音発話 MRI データベース [4] から抽出した．このデータベースは日本人成人男性を被験者として，磁気共鳴画像法 (Magnetic Resonance Imaging，MRI) によって母音発話時の頭頸部を撮像した画像である．母音発話時の声道形状に自由空間を付与し，声道のシミュレーションモデル (Fig. 1) を作成した．自由空間は，Mur の吸収境界条件によって音波を吸収しやすくするために十分な空間を設定した. このシミュレーションモデルのボクセルサイズは $2 \times 2 \times 2$ mm，総ボクセル数は 1,859,000 ($130 \times 130 \times 110$) とした．また，シミュレーションにおける離散時間間隔は 1.0×10^{-6} s とした.

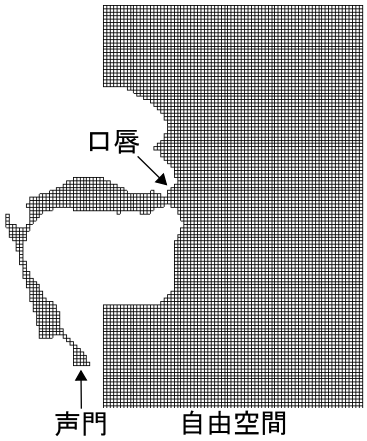

Fig. 1 母音/a/のシミュレーションモデルの正中矢状断面

上記の解析空間を (1) CUDA を用いた GPU 計算と (2) OpenMP を用いたマルチコア CPU による並列計算と (3) 逐次計算 (CPU1 コアによる計算) で実装した．実行環境は CPU として Intel Core i7 920 (2.66 GHz)，3 GB のメインメモリ，さらに GPU が搭載された PC を用いた．使用した GPU は NVIDIA 社の Tesla C1060 で CUDA のバージョンは 3.1 である．そのスペックを Table 1 に示す．OpenMP はコンパイラに gcc のバージョン 4.3 を使用し，最適化に関するコンパイラオプションとして`-O2` を指定した.

Table 1 Tesla C1060 のスペック

シェーダクロック	1,300 MHz
メモリクロック	16,000 MHz
メモリバンド幅	102 GB/s
デバイスメモリ	4 GB
SP 数	240
SM 数	30

4.2 結果

CUDA，OpenMP，逐次計算の 3 つの方法によって 200 ステップ計算した結果を Table 2 に示す. 逐次計算と比較すると OpenMP の 2.6 倍に対して CUDA では 35.0 倍の速度向上が実現された．また，CUDA の実行時には高いメモリバンド幅を引き出すことができた.

Table 2 各手法の計算時間

計算手法	計算時間 (sec)	高速化 (倍)
逐次計算	21.36	1.00
OpenMP	8.10	2.63
CUDA	0.61	35.01

5 まとめ

本研究では，FDTD 法を用いた 3 次元声道音響シミュレータにおいて，並列計算によって計算時間の短縮を試みた. GPU による並列計算の効果は大きく，逐次計算と比較すると 35 倍程度の速度が得られた.

謝辞 本研究は 2010 年度科研費 (21300071) の支援を受けた.

参考文献

[1] Yee, *IEEE Trans. Antennas Propag.*, 14, 302–307, 1966.

[2] Mur, *IEEE Trans. Electromagnetic Compat.*, EMC-23(4), 377–382, 1981.

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

[4] http://www.baic.jp/product/product_artic.html