グラフィックスハードウェアを使った数値流体解析の高速化

佐藤 清志　　藤澤 誠　　三浦 憲二郎
静岡大学工学部機械工学科

1. はじめに

近年，グラフィックス分野で用いられているGPU(Graphics Processing Unit)は演算能力の大幅な向上に加えて，プログラムにより様々な処理を可能とするプログラマビリティを備えるようになった．例えば，nVIDIA 社 GeForce FX のような高性能 GPU はリアルタイムに 3D 描画を行うために，強力な浮動小数点計算能力と並列演算能力を備えている．またシェーダ言語のような，GPU を効率良く活用するためのプログラミングインターフェースも登場した．

GPU の性能向上とそのプログラム開発環境の整備により，GPU はグラフィックス処理のみでなく，これまで CPU で行われていたような数値計算(例えば物理現象や布のシミュレーション，行列の計算など)への応用も模索されている．

そこで，本研究では GPU を数値流体解析に応用し，基礎方程式を解く過程の反復計算に GPU を使い高速化し，さらにリアルタイムに可視化を行うことで現象の理解を深めることを目的とする．本研究の数値解析はスーパーコンピュータを用いた大規模解析ではなく，パーソナルコンピュータを用いた対話性を重視した数値解析を想定している．

2. 基礎方程式

流れとは，流体に外部から力が加えられた場合，あるいは流体内部に温度差や濃度差がある場合に熱移動と物質移動が生じることである．2 次元における流れの基礎方程式は以下のように表される．

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{1}$$

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) \tag{2}$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right) \tag{3}$$

$$\rho C_P\left(\frac{\partial \theta}{\partial t}+u\frac{\partial \theta}{\partial x}+v\frac{\partial \theta}{\partial y}\right)=\kappa\left(\frac{\partial^2 \theta}{\partial x^2}+\frac{\partial^2 \theta}{\partial y^2}\right) \tag{4}$$

ここで u，v は x，y 各方向の速度，ρ は密度，p は圧力，Cp は定圧熱容量，θ は温度，κ は熱伝導率を示している．式(1)は連続の式，式(2)，(3)は x，y 軸各方向のナビエ・ストークス式，式(4)はエネルギー保存則である．

これら基礎方程式を以下の順に前処理をする．

① 各変数を各代表値で割ることで無次元変数を定義し，それらを用いて基礎方程式を無次元化する．

② 各基礎方程式を有限差分法で離散化する．ここで，対流項に 1 次精度風上差分，拡散項に 2 次精度中心差分，そして非定常項はオイラー陽解法(1 次精度前進差分)で離散化する．

離散化した各基礎方程式を解くアルゴリズムには大きく分けて，Patankar らによる SIMPLE 法に基づいた解法と，Halow らによる MAC 法に基づいた解法がある[3].

SIMPLE 法は半陰的な解法であり，連立一次方程式を解くのに共役勾配法を用いるのが一般的である．共役勾配法は Bolz らによって GPU による疎行列の解法として実装された[1]．Bolz らの結果では CPU より 60%から 80%ほど計算スピードが向上することが述べられている．

MAC 法は陽的な解法であり，特に MAC 法に基づいた Hirt らによる HSMAC 法は連立方程式を解く必要がないのでメモリを大量に消費することがなく，よりコンピュータでの解析に向いているといえる手法である．また HSMAC 法には，境界における速度が既知であれば圧力の境界条件を必要としないため，処理が高速化する利点がある．しかしながら，GPU に実装した研究例はないため，本研究ではこの HSMAC 法を GPU に実装することで更なる高速化を目指す．HSMAC 法のフローチャートを図 1 に示す．

3. GPU への実装方法

現在 GPU においてプログラム可能となっているは，Vertex Shader と Pixel Shader の 2 つである．Vertex Shader はジオメトリ処理を担当する演算ユニットで，Pixel Shader はピクセル処理を担当する演算ユニットである．これらは実際には複数

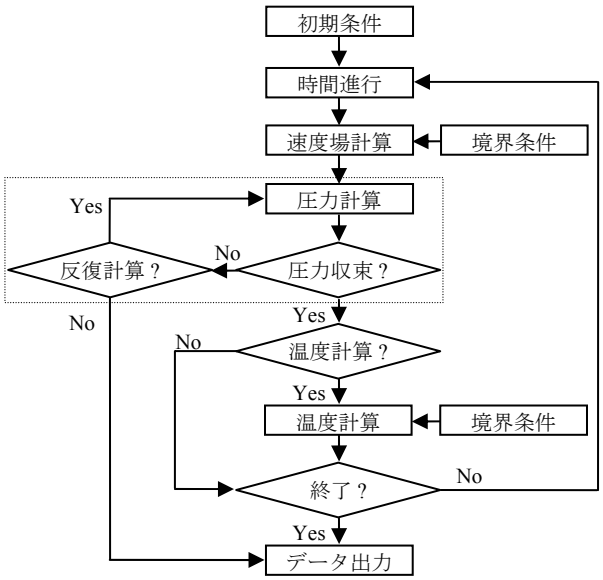

図 1. フローチャート(点線内が GPU による計算)

の浮動小数点 SIMD 演算ユニットで構成されている．したがって，演算ユニットの数だけ一度に SIMD 演算が可能である．GeForce FX ではこのユニット数が 4 つであるから，4 つのデータを一度に浮動小数点計算することができる．

実装にはテクスチャマッピングを応用した．GPU で計算する際には，CPU からレイノルズ数などの定数と，格子数分の速度と圧力のデータを GPU に渡す必要がある．このとき，大量のデータをパラメータとして GPU に渡すことはできないので，大量のデータを効率よく扱うことができるテクスチャを使用した．また，Pixel Shader がテクスチャ演算をサポートしているため，計算はすべて Pixel Shader で行った．

以下が計算の流れである．

① CPU で計算した速度と，1 つ前のタイムステップで計算した圧力をテクスチャに格納して GPU に渡す．
② Pixel Shader にて圧力を求める反復計算と，速度の修正値を求める．
③ Pixel Shader の出力はカラー値のみなので，結果の速度と圧力をカラー値としてフレームバッファに書き込む．
④ フレームバッファからカラー値を読み取り，CPU で速度と圧力に変換する
⑤ CPU で速度から温度の計算を行う．

この際のカラー値はフレームバッファ内に RGBA の 4 チャンネル，各 8 ビット幅で表される．数値計算において 8 ビットによる表現では精度不足である．実装では 4 チャンネル計 32 ビットに 1 つの値を格納するようにした．これにより必要な精度を確保できた．

図 2 は CPU と GPU とでの処理の流れである．

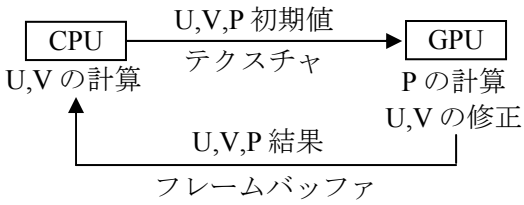

図 2. CPU と GPU による処理の流れ

4. 結果

計算モデルとしてキャビティ流れを考え，計算格子にはスタッガード格子を用いた．境界条件は上辺のみ u=1，v=0，他の辺では u=0，v=0．初期条件は境界を除く各グリッドにおいてすべての u，v，p は 0 である．

フレームバッファから値を読み取るときに誤差が生じた．誤差を少なくするために CPU と GPU 間のデータの受け渡しの回数を少なくする必要がある．

数値解析では一般に使われる演算子や関数が，Cg 言語では未対応である場合があり(例えば比較演算子など)，これらを回避しつつ，プログラムの長さの制限にかからないように実装する必要がある．

可視化の例を図 3 に示す．

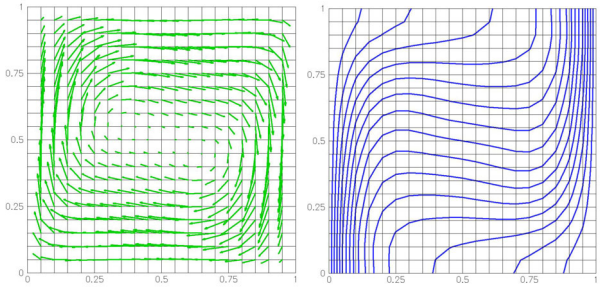

図 3. 可視化の例(左図が速度，右図が温度)

実行環境は，CPU は intel Pentium 4 2.53GHz，メモリは 512MB，グラフィックカードには nVIDIA GeForce FX 5600 を使用し，シェーダ言語として nVIDIA 社の Cg 言語を使用している．

参考文献

1) Jeff Bolz, Ian Farmer, Eitan Grinspun, Peter Schroder : “Sparse Matrix Solvers on the GPU” , Conjugate Gradients and Multigrid , ACM Transactions on Graphics , pp.917-924 , 2003.
2) 平野博之：流れの数値計算と可視可 Tecplot で視る流体力学，丸善株式会社，2001.
3) 塚越誠一：数値流体力学，培風館，1997.
4) Randima Fernando, Mark J. Kilgard : The Cg Tutorial 日本語版，ボーンデジタル，2003.