GPUによる境界要素法の高速化

○群馬大学　学生会員　増村佳大
群馬大学大学院　正会員　斎藤隆泰

1. はじめに

近年, 計算力学の分野において計算速度の向上を目的とした GPU コンピューティングが注目を集めている. GPU コンピューティングとは, 画像処理に用いられる GPU の機能を画像処理とは異なる数値計算の目的に転用するもので, 通常, 大規模計算を実行するために用いられる. GPU コンピューティングによる計算の高速化は製造分野, 科学技術分野, 医療分野など各種分野ではじまっている. しかし, それらの多くは, 差分法や有限要素法を扱ったものであり, 境界要素法の高速計算に GPU を用いた研究は数少ない. 本研究では CUDA Fortran を使用した GPU コンピューティングを用いて, 境界要素法の計算の高速化を行う.

2. 2 次元 Laplace 問題に対する境界要素法

境界要素法は, 境界上に与えられた節点に関する支配微分方程式を境界積分方程式に変換し, それを離散化した後に得られる代数方程式を解くことで近似解を求める数値解法 [1] である. 本研究では, 簡単のため, 2 次元 Laplace 方程式の境界値問題を対象とし, GPU を用いた数値計算の高速化を図る.

図 1 のような滑らかな境界 $\Gamma(\Gamma = \Gamma_u \cup \Gamma_q)$ をもつ 2 次元の閉鎖領域 Ω 内で, Laplace 方程式 :

$$\nabla^2 u(\mathbf{x}) = 0, \qquad \mathbf{x} \in \Omega \tag{1}$$

を満足し, 境界条件

$$u(\mathbf{x}) = \bar{u}, \qquad \mathbf{x} \in \Gamma_u \tag{2-a}$$

$$q(\mathbf{x}) = \frac{\partial u}{\partial n}(\mathbf{x}) = \bar{q}, \qquad \mathbf{x} \in \Gamma_q \tag{2-b}$$

を満たすようなポテンシャル u を求める問題を考える. ここで ∇^2 はラプラシアンであり, $\partial/\partial n$ は境界における外向き法線方向微分を表す. この問題の解は, 次の境界積分方程式を解くことで求まる.

$$c(\mathbf{x})u(\mathbf{x}) = \int_\Gamma u^*(\mathbf{x},\mathbf{y})q(\mathbf{y})d\Gamma(\mathbf{y}) - \int_\Gamma q^*(\mathbf{x},\mathbf{y})u(\mathbf{y})d\Gamma(\mathbf{y}) \tag{3}$$

ここで c は境界の形状に依存する自由項である. また, $u^*(\mathbf{x},\mathbf{y})$ は 2 次元 Laplace 方程式に対する基本解で,

$$u^*(\mathbf{x},\mathbf{y}) = \frac{1}{2\pi}\ln\left(\frac{1}{r}\right) \tag{4}$$

で与えられる. ただし $\mathbf{x}$, $\mathbf{y}$ はそれぞれ観測点とソース点を表し, r は $r = |\mathbf{x}-\mathbf{y}|$ で定義される. 一方 $q^*(\mathbf{x},\mathbf{y})$ は $q^*(\mathbf{x},\mathbf{y}) = \partial u^*/\partial n$ で表される二重層核である.

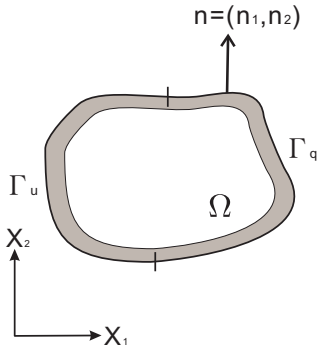

図 1 2 次元 Laplace 問題

式 (3) を近似的に評価するために区分一定要素で離散化を行うと, 次の式を得る.

$$c(\mathbf{x}^i)u(\mathbf{x}^i) + \sum_{j=1}^{N}\int_{\Gamma_j} q^*(\mathbf{x}^i,\mathbf{y}^j)u(\mathbf{y}^j)d\Gamma(\mathbf{y}) = \sum_{j=1}^{N}\int_{\Gamma_j} u^*(\mathbf{x}^i,\mathbf{y}^j)q(\mathbf{y}^j)d\Gamma(\mathbf{y}) \qquad (i = 1,\cdots N) \tag{5}$$

境界上では通常, ポテンシャルまたは流速のどちらかがあらかじめ境界条件として与えられるため, 式 (5) の境界未知量は全部で N 個となる. 最終的に式 (5) は, 境界未知量を左辺にまとめた次の代数方程式に帰着される.

$$A\mathbf{x} = \mathbf{f} \tag{6}$$

ここで A は基本解 $u^*(\mathbf{x},\mathbf{y})$ や $q^*(\mathbf{x},\mathbf{y})$ により計算される係数行列, $\mathbf{x}$ は境界未知ベクトル, $\mathbf{f}$ は境界既知量により計算されるベクトルである. 最終的に式 (6) を計算すれば N 個の境界未知量 $\mathbf{x}$ を求めることができる.

3. GPU による境界要素法の高速化

GPU が高速計算に利用されつつある訳は, 高い並列処理能力にある [2]. 図 2 (a) のように通常の数値計算に幅広く用いられている CPU の計算処理は逐次処理であり, 一つの演算コアにより連続に計算を行う. それに対して, GPU を用いた場合は図 2 (b) のように複数の演算コアにより並列計算を行うことができる. そのため GPU を効果的に利用すれば, CPU を用いた場合と比べ, 大規模計算を高速に解くことが可能であると思われる.

本研究では式 (6) の係数行列作成部分, および最終的に解くこととなる式 (6) の求解に対して, GPU を用いて高速計算を行う. 境界要素法の計算は, 式 (6) の係数行列の作成の

Key Words: 境界要素法, GPU, 2 次元 Laplace 方程式
〒 376-8515　群馬県桐生市天神町 1-5-1

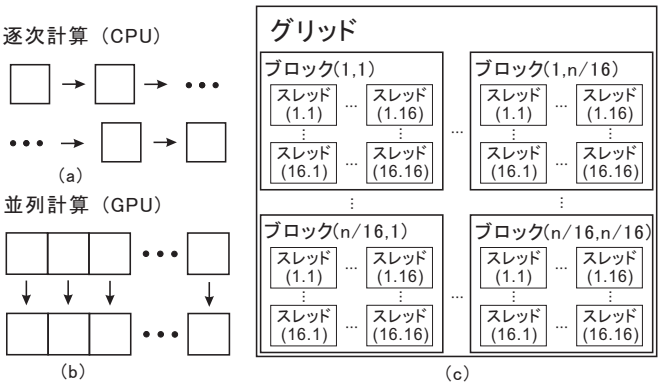

図 2 CPU と GPU (a) CPU (b) GPU 計算 (c) GPU の構造

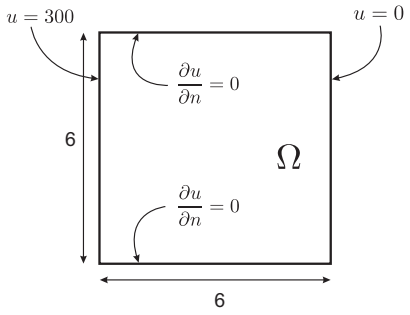

図 3 解析モデル

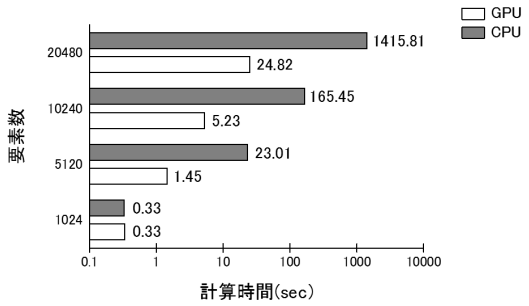

図 4 CPU または, GPU を用いた場合の計算時間

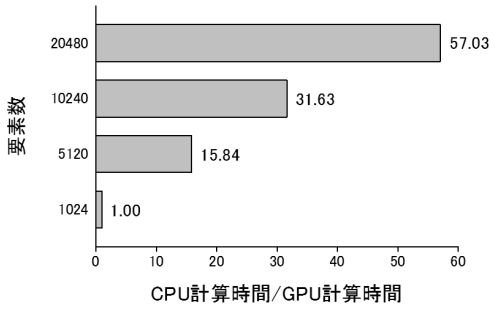

図 5 CPU 計算時間 / GPU 計算時間

部分と, その代数方程式の計算に多くの時間を要する. 実際, 境界要素数 N が増加すれば計算量は膨大となり, 係数行列作成時間の計算時間は $O(N^2)$, 方程式の求解は $O(N^3)$ 程度となる. そのため, 本研究では, このように境界要素法でボトルネックとなる, この二ヵ所を, GPU により計算を行うことで境界要素法を高速化する.

4. 計算効率の確認

以下, 本研究で実装した GPU を用いた境界要素法の計算効率について確認する. ここでは CPU を用いて, 通常通り計算を行った場合と, GPU を用いて計算を行った場合の 2 通りの計算時間の比較を行うことにより, GPU を適用した場合の有効性を検討する. 解析モデルは図 3 のようなポテンシャル問題とする. 要素数は一辺に $N/4$ となるように等分割配置する. 境界条件は, 上辺, 底辺が $\partial u/\partial n = 0$, 左辺が $u = 300$, 右辺が $u = 0$ である. なお, この問題は文献 [3] で例題として紹介されており, CPU のみを用いた標準的な Fortran ソースコードも付属している. そのためベンチマーク問題として用いるのにも適していると思われる.

なお, GPU の計算には 16 × 16 スレッドの 2 次元配列のスレッドを作成して計算を行った. スレッドは 2 次元で作成する場合, 図 2 (c) のように作成される. なおスレッドは GPU で計算を行う部分の単位であり, スレッドの集合をブロック, ブロックの集合をグリッドと呼ぶ.

また, CPU を用いた場合の式 (6) の方程式の求解には数学ライブラリ LAPACK を, GPU を用いた場合は CULAPACK を用いた. CULAPACK は数学ライブラリ LAPACK の, GPU による並列処理対応型である. なお, プログラミング言語には Fortran を用い, 使用した計算機の性能はメインメモリが 16GB, CPU が XEON E5-1620, GPU が TESLA C2070 であり, GPU の搭載メモリは 6GB である.

図 4 に CPU または GPU を用いた場合の計算時間を, 図 5 に (CPU 計算時間) / (GPU 計算時間) を示す. ただし, 図 4 は両対数で表示されていることに注意されたい. 図 4 より, 要素数の増加に伴い計算時間は $O(N^2)$ で増加することがわかる. 今回の計算では要素数 N はメインメモリの都合上, 最大で 20,480 までしか計算することができなかった. また, 図 5 より, 要素数の増加とともに, GPU を用いた場合の計算効率は増加していることがわかる. 計算時間は要素数 $N = 20480$ のときに, 最大で 57.03 倍に短縮された. よって, GPU による計算を効率的に実行できたといえる.

5. まとめ

本研究では GPU を用いて, 境界要素法の計算の高速化を行った. 2 次元の簡単な Laplace 方程式の境界値問題を解くことにより, GPU を用いた場合の高速化効率を確認した. 今後は, MPI をも適用したさらなる高速化や, 演算子積分時間領域高速多重極境界要素法に対する GPU 高速化を行う予定である.

6. 参考文献

参考文献

1) 田中正隆・松本敏郎・村中正行 共著 : 境界要素法, 培風館, 1991.
2) J. Sanders, E. Kandrot 共著 (株式会社クイープ 訳) : CUDA BY EXAMPLE, 株式会社インプレスジャパン, 2011.
3) C.A. ブレビア (神谷紀生・田中正隆・田中喜久昭 共訳) : 境界要素法入門, 培風館, 1980.