神戸大学都市安全研究センター
研究報告，第26号，令和４年３月

地盤応答解析コードの GPU 並列化

GPU parallelization of geotechnical response analysis code

堂ヶ原 健
Takeshi Dogahara
竹山 智英
Tomohide Takeyama
銭谷 誠司
Seiji Zenitani
橘 伸也
Shinya Tachibana
飯塚 敦
Atsushi Iizuka

概要：現在，動的土/水/空気連成有限要素解析コードである DACSAR-I（Deformation Analysis Considering Stress Anisotropy and Reorientation –Integration）が存在するが，現在の解析速度では大規模計算を一般に行うには速度不足である．本研究では，並列的な計算画像処理に特化した演算装置であり，連続的な計算を得意とする GPU（Graphics Processing Unit）を用いた DACSAR-I の並列化を行い，処理を高速化する．GPU 並列化のプログラミング環境として CUDA（Compute Unified Device Architecture）を用いる．

キーワード：有限要素解析，GPGPUs，データ構造

1. 研究背景及び目的

DACSAR-I（Deformation Analysis Considering Stress Anisotropy and Reorientation - Integration：動的土/水/空気連成有限要素解析コード）[1]によってある程度大規模な解析を実施したい場合，一般に利用する上では現在の計算速度では不十分である．

代表的な大規模解析手法としてスーパーコンピュータの利用が挙げられるが，一般的に利用コストが高いという問題点が存在する．また，非常に多くの CPU を積載して計算を行うスーパーコンピュータは演算と冷却に際して莫大な消費電気量を要する．社会的に許容される消費電力量の限界値が，同時に CPU で構成されたスパコンの高性能化の限界値を示している [2]．東京工業大学では世界で初めて GPU を搭載したスーパーコンピュータ”TSUBAME”が開発・運用されている．特に冷却に要する消費電力が小さく，極めて省エネなスーパーコンピュータであると言われている．また，GPU は比較的安価に並列化を行うことができる点でも有用である．よって，消費電力の観点からみて将来の拡張性が高いこと，比較的安価であることから，GPU を用いたコードの並列化を行い，計算速度を向上させる．

本研究では，計算速度の向上を目的として，CUDA（汎用並列コンピューティングプラットフォーム）を用いた GPU 並列化を行う．その際，今後の発展性を考慮し，可読性及び拡張性を重視した実装を行う．また，その成果の検証として液状化解析を行い，実行速度について考察する．

2. 手法

(1) DACSAR

DACSAR(Deformation Analysis Considering Stress Anisotropy and Reorientation)[3]は有限要素法を用いて地盤の解析を行うプログラムである. 本研究では DACSAR-I の GPU 並列化を行う. DACSAR-I は総合的な解析を行うことができ, 次元, 水・空気の連成, 動的・静的解析などを選択して実行できる.

(2) CUDA

本研究で利用した開発環境 CUDA について順を追って説明する. まず, CPU と GPU について説明し, GPGPU と CUDA について述べる.

CPU(Central Processing Unit)とは, コンピュータの中枢部分にあたる, 各種装置の制御やデータの処理を行うプロセッサである. 中央演算処理装置とも呼ばれる. コンピュータの基本性能を決めるパーツであり, 高度な演算機能が求められ, 連続的な計算処理に長ける[4].

GPU(Graphics Processing Unit)とは, CPU と同様に, プログラムに従ってデータの処理を行うプロセッサであり, 画像処理に特化したプロセッサである. 特に 3DCG のリアルタイム処理が重視されており, ピクセル単位, オブジェクト単位での画像処理といった非常に高い並列性を持ったグラフィックス処理を行うことが要求される. そのため, CPU と比較して並列的な計算処理に長けている. 並列数はハードウェアレベルで大きな差があり, CPU のコア数がメジャーなもので 1～16 コアなのに対し, GPU は数百~数千個のコアを擁している[5].

GPU は CPU と比較して非常に高い並列計算能力を持っているほか, グラフィックス処理には XYZW(座標計算)および ARGB(色計算)の 4 要素ベクトル演算が多用されるため, GPU はベクトル演算に強いという特徴がある. そのため, 画像処理に用いられていた GPU を汎用演算に利用することが考えられるようになった. GPU の演算資源を画像処理以外の目的に応用する技術のことを GPGPU(General-Purpose computing on GPUs)と呼ぶ.

CUDA(Compute Unified Device Architecture)は NVIDIA 社が提供している, 同社製 GPU 向けの GPGPU 開発環境である. CUDA は C++言語を利用した開発環境を提供でき, 現在の DACSAR-I が C++言語で記述されていること, 情報の入手が容易であることから, CUDA を利用した GPU 並列化を行った.

3. 高速化手法

CUDA によって並列化を行う際, その仕様や制約を考慮した高速化を行う必要がある. 後にコードに変更を加える可能性や利用者にとっての利便性を考慮し, 可読・拡張性を担保した高速化を行う. 本章では, 本研究で並列化を行った DACSAR-I の特徴となるデータ構造および具体的な並列化手法の例について述べる.

(1) レポジトリ構造

可読性を確保しつつ, CPU 用メモリから GPU 用メモリへのメモリコピー(データコピー)や, 一部変数を連続した要素で抜き出す際に生じる処理の削減を目的とした構造である. 上述した処理は, 各要素にあたるクラスのメンバ変数を指定して行われる. クラスとは, 変数や関数を持つまとまり(データ型)であり, クラスの持つ変数をメンバ変数という. メンバ変数はクラスを用いてデータ(オブジェクト)が作成された際, メモリ上に順番に格納される. クラス, オブジェクトの概要を図1に示す.

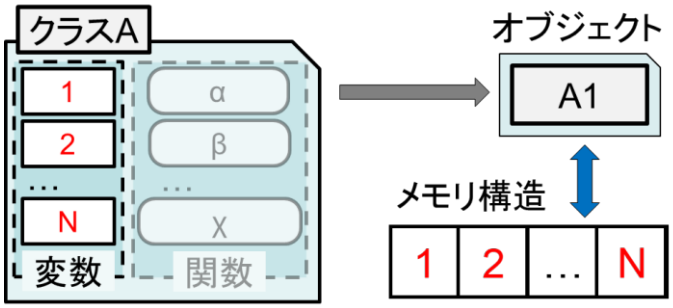

図 1 クラス，オブジェクトの概要

このとき，各要素に存在する同種のメンバ変数への一括アクセスを考える．仮に図1中に表されるメンバ変数 1 にアクセスする際，オブジェクト A1，A2，A3と要素（オブジェクト）を連続して確保しても，各要素の変数1はオブジェクトごとに存在するため，アクセスにはループ処理が必要になることがわかる．
よって，データ処理を高速化する手法として，条件分岐やループ処理を削減し，一括したアクセスを行える構造を作成する．方法としては，要素ごとに変数のデータを格納するのではなく，変数の種別ごとにデータを連続したメモリに確保することで行う．具体的には，

1．クラスメンバの実体を，連続したメモリで確保する役割を持つクラス（レポジトリクラス）を作成する．
2．クラス本体が持つメンバ変数には，それぞれの変数に適したクラスを用いて，レポジトリクラスの該当するメモリへのポインタを持たせる．

という方法を用いてデータを保存する．ポインタとは，メモリ上の位置（アドレス）を指す型である．ある程度の幅を持つメモリを指す場合，その先頭アドレスを指す．変数名の先頭に*（アスタリスク）を付けることでポインタとして宣言できる．また，通常の変数のアドレスを取得する場合，変数名の先頭に&（アンパサンド）を付けることで可能になる．この方法により，可読性を確保したまま，クラスメンバの実体を保持するメモリを連続して確保することができる．レポジトリ構造の概要，比較を図2に示す．

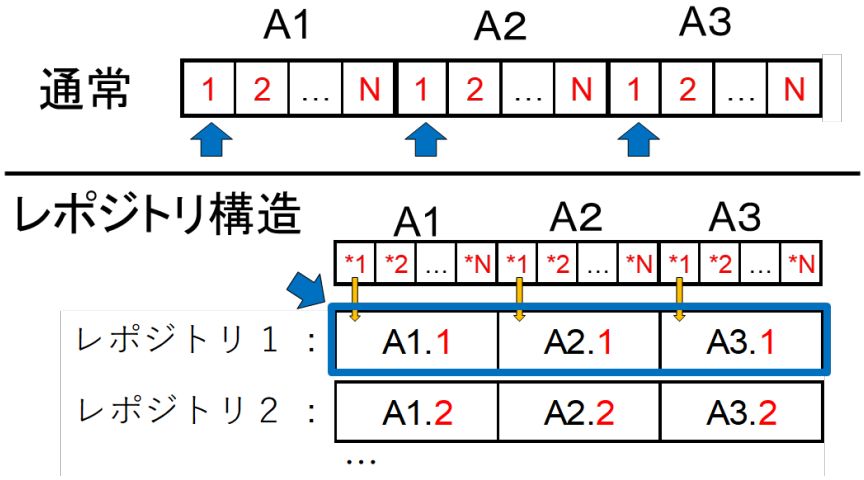

図2 レポジトリ構造の概要，比較

(2) コンテナ構造

DACSAR-I には要素の種類や次元，節点の自由度数などの要素の情報を持つ element クラスや，材料特性の情報を持つ material クラスなどのクラスが存在する．いくらかの要素の計算を行う際，要素に応じてこれらのクラスの情報を取得する必要がある．その際，要素数分だけそれぞれのクラスをまとまって保持するクラスを用意することで，”element の集合”の何番目，”material の集合”の何番目といった形で扱うことが

できるようになる. この”集合クラス”とでも言うべきクラスがコンテナクラスである. コンテナクラスの概要を図3に示す.

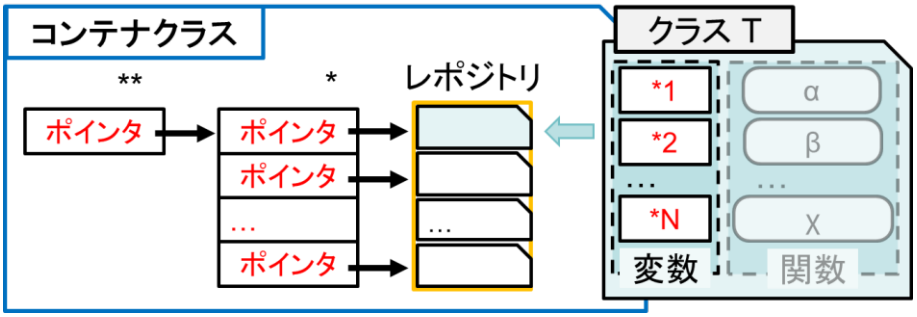

図3 コンテナ構造の概要

コンテナ構造はレポジトリ構造を多重ポインタによって示す. レポジトリ自身もクラスを通して変数にポインタでアクセスするため, コピーや代入を伴わずに素早く簡易的な記述でデータにアクセスできるようになる. また, コンテナクラスは上述のように様々なクラスに対して利用するため, クラステンプレートを用いて作成する. テンプレートとは, 関数やクラスを使用する際に, プログラマが一度コードを書くだけで, そのコードが型の形式に準拠して広範囲の型に適用できるようにするための仕様である.

(3) 疎行列ソルバーの並列化

DACSAR-I における並列化の例として疎行列ソルバーの並列化について説明する.

前提として, CUDA が並列処理を行う基本的な方法について説明する. GPU はスレッドと呼ばれる大量の処理単位を持っており, CUDA はそのスレッドを利用して並列計算を行う. 例として, 研究で使用した GPU (Geforce GTX 1060 6GB) において, CUDA は最大で同時に 2147483647*65535*65535*1024 スレッド並列での計算を行うことができる. このような膨大なスレッドを管理するために, グリッドとブロックと呼ばれる処理単位を用いる. 1024 スレッドが 1 ブロックを構成し, 2147483647*65535*65535 のブロックの 3 次元構造がグリッドを構成する. これらの処理単位に対して__global__関数識別子を用いて CPU から命令を行い, CUDA の並列処理は実行される. __global__関数識別子 (カーネル) は CPU から呼び出されて GPU 上で実行される関数である. 1カーネルが1スレッドとなる. 主な制約として, 以下の 5 点が存在する.

- CPU メモリにアクセスできない.
- 可変引数は利用できない.
- 戻り値は常に void 型 (戻り値を返せない) となる.
- static 宣言などによる静的変数は使用できない.
- 再帰処理を行えない.

このカーネル関数を並列数・引数を指定して呼び出すことで CUDA は並列化を行う.

改めて, 疎行列ソルバーの並列化について説明する. DACSAR-I では外積形式ガウス法による LU 分解法を用いる. LU 分解では第 0 列から順番に消去が行われる. 対角成分 a_{kk} を用いて第 k 列を更新し, k+1 以降の行・列の更新を行う. LU 分解の手順を図4に示す.

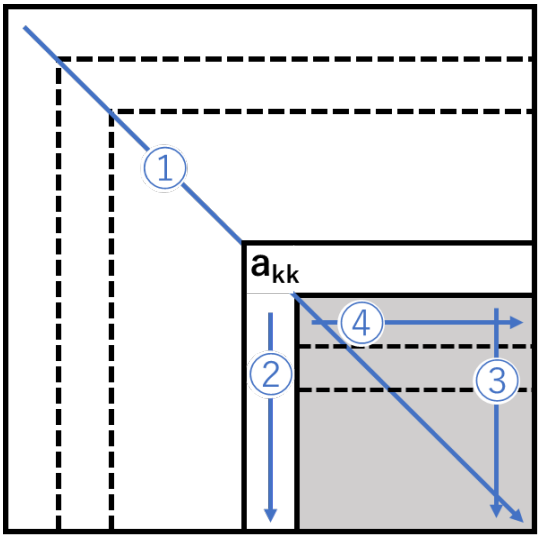

図4　LU 分解の概要

並列化を用いない計算では，コンピュータはループ処理を用いて LU 分解を行う．図4の①~④はそのループを示している．ループ①の内部でループ②~④が行われる．順番としては，ループ②で k 列を更新した後，ループ③により行ごとの計算が実行され，ループ③の内部でループ④によって成分一つ一つの計算が行われる．今回 CPU での計算はこの方法によって行われている．対して今回行った GPU による並列計算では，②~④はループ処理ではなくそれぞれ並列処理されている．正確には，②がスレッドで並列計算され，その後スレッド・ブロックによる二次元並列化によって③と④の計算を行う．②は a_{kk} での除算，③・④は k 行，k 列を用いた計算であり，内部で計算が独立しているため並列化が可能になっている．①は直前の計算の結果が影響するループ処理であるため，並列化を行うことはできない．

LU 分解後，前進代入と後退代入を行う．概要を図5に示す．

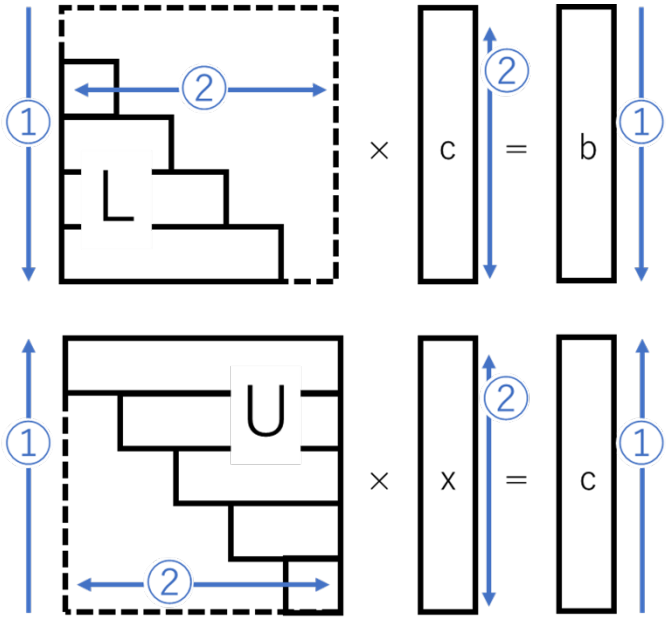

図5 前進代入・交代代入の概要

CPU の場合は L・U それぞれで①②のループ処理を行う．GPU でも，①で示される前進・後退は独立でないためループ処理で行うが，②で示される行ごとの乗算・和算は並列計算を行う．1つの乗算を1スレッドで行い，行ごとに並列化を行う．スレッド間の共有メモリに乗算の解を保存することで，リダクション処理と呼ばれ

る総和手法を用いて和算の並列化を可能にしている. リダクション処理の概要を図6に示す.

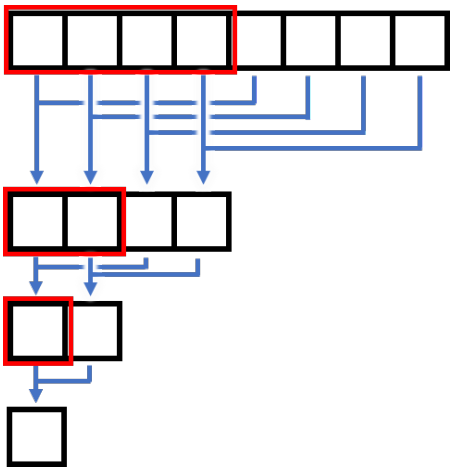

図6 リダクション処理

リダクション処理はデータを2の累乗個で扱う. データが存在しない範囲は自動的に0として扱われる. 並列起動したスレッドの内, 前半に位置するスレッドが後半のデータを自身の持つデータに加算する. これを総和が求められるまで(データが一つに収束するまで)繰り返す. このリダクション処理によって, 2^n 個のデータの総和をn回の処理で計算することができる. よって, ループ回数を大幅に削減することが可能となる.

疎行列ソルバーの並列化における問題点として, スレッド間の共有メモリ容量が小さく, 計算を行う行の非0要素数が多い場合, 不足する可能性がある点が挙げられる. 本研究では, 前進代入・交代代入の並列数の上限を設定し, 1つのスレッドに複数の計算を行わせることで対応している. 当然, 1スレッドが複数の処理を行うためにループ処理を必要とするが, それでもCPUでの処理と比較して非常に高速である.

4. 効果検証

(1) 検証環境

本章では, GPU並列化を行ったDACSAR-Iで液状化解析を行い, CPUコードとGPUコードの実行時間の比較を行う. 解析を行った環境は以下の通りである.

OS:Ubuntu 18.04.5 LTS
CUDA:Ver 11.1.74
CPU:Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz
メモリ:67.3GB
グラフィックボード(GPUを搭載する演算機器):GeForce GTX 1060 6GB
グローバルメモリ:6.07GB

(2) 解析方法

EC モデル [6]を用いて土水連成の解析を行った. 解析領域は 10m*10m の2次元解析領域とする. 左右端面は変位増分の指定なし, 下端面は固定し, 上端面を排水境界とする. 同様のパラメータに対する解析では実行時間は自由度数に依存するため, 解析領域の要素分割を, 4節点四角形要素を用いて 10*10, 50*50, 100*100, 200*200 要素数の4種類で行う. 以上の解析をCPU, GPUそれぞれで行い比較する. 図7に解析領域の概要を示す.

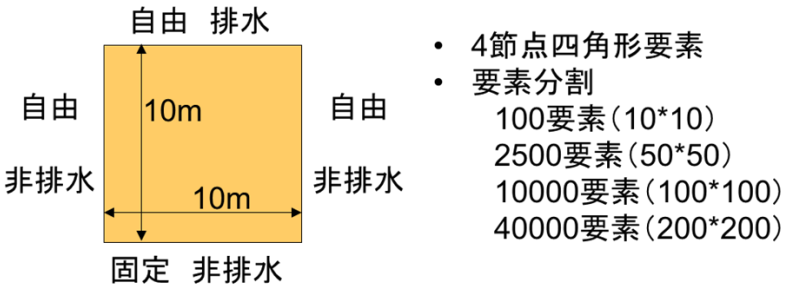

図7 解析領域の概要

入力パラメータを表1に示す.

表1 入力パラメータ

Analysis Control Values			
γw	A unit weight of pore water	kN/m3	9.8
Ωw	Reference height for specifying the head of water	-	1
β	Parameter for time descritization of water head	-	1
α0	Rayleigh damping mass-proportional coefficient	-	0
α1	Rayleigh damping stiffness-proportional coefficient	-	0
Kw	Bulk modulus of pore water	-	1.0E+15
βn	Parameter for Newmark-beta method	-	0.25
γn	Parameter for Newmark-beta method	-	0.5
Material Properties			
D	Coefficient of dilatancy proposed by Shibata (1963)	-	0.01
Λ	Irreversibiity ratio expressed by $\Lambda = 1 - \kappa / \lambda$	-	0.8
M	Critical state parameter	-	1.4
ν'	Effective Poisson's ratio	-	0.333
λ	Compression index in the e-lnp' relationship	-	0.3
e0	Void ratio corresponding to σ'v0	-	1
λk	Gradient of e plotted against ln(k)	-	0
Gs	Specific Gravity	-	2.65
nE	Fitting parameter to describe non-linear contractancy	-	1.2
u0	Constant parameter to control expansion rate of sub-loading surface	-	0.1

(3) 結果

以下に CPU と GPU における実行時間の比較を示す. 表2に比較表を, 図8にそのグラフを示す. これは計算の第1ステップに要する実行時間である.

表2 実行時間比較表

要素分割➡	10*10要素	50*50要素	100*100要素	200*200要素
CPU[s]	0.022	1.118	7.828	71.185
GPU[s]	0.026	0.317	1.622	10.658
GPU/CPU[比率]	1.187	0.283	0.207	0.150

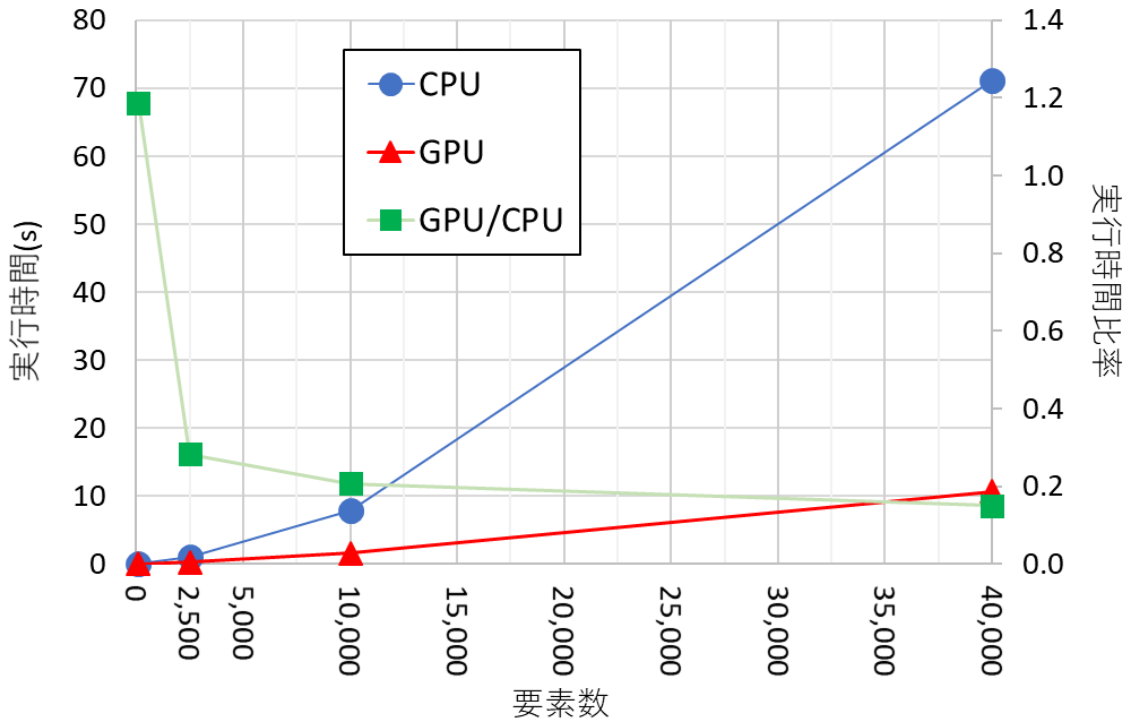

図8 実行時間比較グラフ

(4) 考察

100 要素においては GPU の方が遅れているが，要素数(自由度数)の増加に応じて CPU の実行時間が二次関数的に伸びていき，2500 要素時点ですでに3倍以上 GPU の方が高速になっている．その後も，より緩やかな伸びをする GPU との実行時間差が開いていくことがわかる．これは，2(2)で述べたように，CPU は連続的な処理に長けており，GPU は並列的な処理に長けているためである．少数の計算を高速に処理する点において CPU は GPU に勝るが，自由度数の増加に伴って並列化の恩恵が高まり，CPU と比較して GPU がより短時間で処理を行うことができるようになる．ホストコード上でしか行えず，並列化を行うことができない処理も存在すること，ホストメモリとデバイスメモリ間のメモリ移動が必要であることなどから GPU においても実行時間は徐々に増加している．

今回の並列化の効果は疎行列ソルバーの並列化の効果が非常に大きく現れている．疎行列ソルバーの実行時間の関係を表3に示す．

表3 疎行列ソルバーに関する実行時間比較表

要素数	10*10		50*50		100*100		200*200	
実行ユニット	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
ソルバー[s]	0.003	0.007	0.629	0.205	5.869	1.157	63.324	8.818
ソルバー/全体[比率]	0.144	0.275	0.563	0.647	0.750	0.713	0.890	0.827
ソルバー:G/C[比率]	2.272		0.325		0.197		0.139	

CPU と GPU の双方において，自由度数の増加に伴い DACSAR-I 全体の実行時間における疎行列ソルバーの計算が占める割合が大きくなっていくことが確認できる．4 万要素の計算においては CPU で 9 割弱，GPU で 8 割強の時間を占めている．また，表2で確認した全体の実行時間比率と同様に，疎行列ソルバーの計算でも，自由度数の増加に伴い GPU が CPU と比較して高速になっていることがわかる．以上から，今回の高速化において疎行列ソルバーの並列化が大きく寄与していることが確認できる．

5. 結論

本研究では，大規模計算に用いるには DACSAR-I の実行速度が不十分であるという点から，CUDA を用いた GPU 並列化による高速化を目的として研究を行った．その際，可読性と拡張性に留意した開発を行った．液状化解析による検証を通して，並列化による高速化を確認できた．課題点として，GPU のメモリ上限の存在が挙げられる．今回4万要素で行った解析で GPU の 6070MB のメモリの内，既に 5000MB 強のメモリを使用している．よって，これ以上の自由度を持つ解析を行うことは現状厳しい．これは大規模解析において大きな課題である．今後の展望として，メモリ不足の解決を行う．方法としては，コードの見直しとマルチ GPU 化を行う．コードの見直しによって，メモリを無駄に消費している部分を修正し，メモリ消費量を抑えられる可能性がある．また，マルチ GPU 化は GPGPU におけるさらなる高速化・大規模化の手法であり，単一のグラフィックボードを用いた並列化ではなく，複数のグラフィックボードを使用して GPGPU を行う手法である．理論上，台数分だけ並列数とメモリを増やすことが可能になり，メモリ不足による解析規模の限界を解消できるほか，より高速な処理を実現できる可能性がある．CUDA もマルチ GPU 化に対応しているが，専用の記述が必要であるため，追加のコーディングが必要となる．具体的には，stream 演算や OpenMP と呼ばれる規格を用い，複数のグラフィックボードにデータと演算を割り振る指示およびグラフィックボード間でのデータのやり取りの指示を行う必要がある．

参考文献

1) Takeyama, T., Tachibana, S., Fukukawa, A., A finite element method to describe the cyclic behavior of saturated soil, *International Journal of Material Science & Engineering*, Vol.2, Issue.1, 2015
2) 遠藤敏夫, 松岡 聡, 橋爪信明, 長坂真路：ヘテロ型スーパーコンピュータ TSUBAME の Linpack による性能評価, 情報処理学会論文誌コンピューティングシステム, Vol.48, No.SIG8, pp.62-70, 2007.
3) Iizuka, A., Ohta, H., A determination procedure of input parameters in elasto-viscoplastic finite element analysis, *Soils and Foundations*, Vol.27, No.3, pp.71-87, 1987.
4) CPU - ASCII.jp デジタル用語辞典,（最終閲覧日：2022 年1月 25 日）, https://yougo.ascii.jp/caltar/CPU
5) 大島聡史：これからの並列計算のための GPGPU 連載講座(I) GPU と GPGPU の歴史と特徴,（最終閲覧日：2022 年1月 25 日）, https://www.cc.u-tokyo.ac.jp/public/VOL12/No1/201001gpgpu.pdf
6) 大野進太郎, 飯塚敦, 太田秀樹：非線形コントラクタンシー表現式を用いた土の弾塑性構成モデル, 応用力学論文集, Vol.9, pp.407-414, 2006.

筆者：1）堂ヶ原健、神戸大学大学院工学研究科市民工学専攻、学生；2）竹山 智英、神戸大学大学院工学研究科、教授；3）銭谷 誠司、神戸大学都市安全研究センター、特命准教授；4）橘 伸也、神戸大学都市安全研究センター、准教授；5）飯塚 敦、神戸大学都市安全研究センター、教授

GPU parallelization of geotechnical response analysis code

Takeshi Dogahara
Tomohide Takeyama
Seiji Zenitani
Shinya Tachibana
Atsushi Iizuka

Abstract

DACSAR-I（Deformation Analysis Considering Stress Anisotropy and Reorientation －Integration）, a dynamic soil/water/air coupled finite element analysis code, is not fast enough at the current analysis speed to perform large-scale calculations in general. In this study, we parallelize DACSAR-I using a GPU（Graphics Processing Unit）, which is a computing device specialized for parallel computational image processing and excels in continuous computation, to accelerate the process, and use CUDA（Compute Unified Device Architecture）as the programming environment for GPU parallelization.