# 2011 年度森泰吉郎記念研究振興基金「研究育成費」
# 研究成果報告書

研究課題名：「電気自動車の空力開発と GPU コンピューティング」

政策・メディア研究科　修士２年　GE 所属
久野敦礼

## ***研究概要***

近年、グラフィック用の計算機である GPU を用いた GPGPU が、有限要素法全般における新たな計算手法として注目されている。本来 GPU は単純な処理を繰り返す描画処理に特化した計算機として開発された。コンピューターのコアとしての CPU が複雑な処理に対応しているのに比べ、単純な処理速度では圧倒的に GPU が勝る。そのため、有限要素法等の単純処理式に応用することにより、通常よりも速いスピードで解析を進めることが出来る。従来の方法による CFD では、かなりの時間を解析に割かなければならず、研究効率という点においてほかの分野に比べると劣っている。ここで、GPGPU を用いた CFD について研究をすることにより、今後の研究効率の向上につなげることが今回の研究目的である。

## ***背景・目的***

−GPGPU

グラフィック用の計算機であるGPUを新たな計算処理手法として使用する手法である。本来GPUは単純な処理を繰り返す描画処理に特化した計算機として開発された。コンピューターのコアとしてのCPUが複雑な処理に対応しているのに比べ、単純な処理速度では圧倒的にGPUが勝る。そのため、有限要素法等の単純処理式に応用することにより、通常よりも速いスピードで解析を進めることが出来る。

−openFoam

本研究においては、CFDの基礎実験と解析手法についての研究を行うため、openFoamという無料のソフトウェアを利用した。世界的に学術分野で用いられているCFDソフトであり、公開されている情報やプラグインが多いのが特徴である。

-計算環境

CPU:Intel(R) Xeon(R) X5355　@ 2.66GHz

RAM:16GB

GPU:Geforce GTX590

OS:Ubuntu 10.10

| | |
|---|---|
| **流入条件** | 流速5m/s一様流<br>圧力勾配なし |
| **流出条件** | 速度圧力共に勾配なし |
| **境界条件** | 円柱上で滑りなし条件<br>圧力勾配なし |
| **モデル** | 標準smagorinsky model |
| **アルゴリズム** | PisoFoam |
| **時間** | 0～0.1s<br>Δt = 1E-05<br>10000steps |
| **差分法** | 風上差分 |

-Large Eddy Simulation(LES)

計算格子を用意し、その大きさ以上の渦を直接計算、それ以下の渦をモデル化する手法である。大きなスケールの運動は、一般的に小さなスケールの運動よりも遥かに多くのエネルギーを保有しており、その大きさや強さは流体流の保存量の輸送に大きく影響する。しかし、小さなスケールの運動は通常、大きなスケールの運動に比べてはるかに弱く、物理量の輸送にはほとんど寄与しない。そこで、大きな渦をより正確に取り扱うシミュレーションを行うことは妥当である。LESでは、3次元、非定常であるため、計算負荷の大きな計算を必要とするが、自動車の車体周りの空気の流れ等の複雑な形状の周りの流体解析にはこちらが適している。

*計算結果*

19caseシミュレーションを行い成功したのが4caseなので、以下4caseの共通条件と相違条件を挙げる。

*case1*

nCorrectors = 2、初期条件流れなし

| 項 | p | pFinal | u | k |
|---|---|---|---|---|
| solver | PCGgpu | PCGgpu | PBiCGgpu | PBiCGgpu |
| preconditioner | diagonal | diagonal | diagonal | diagonal |
| tolerance | 1E-06 | 1E-06 | 1E-05 | 1E-05 |
| relTol | 0.05 | 0 | 0 | 0 |

*case2*

nCorrectors = 20、初期条件流れなし

| 項 | p | pFinal | u | k |
|---|---|---|---|---|
| solver | PCGgpu | PCGgpu | PBiCGgpu | PBiCGgpu |
| preconditioner | diagonal | diagonal | diagonal | diagonal |
| tolerance | 5E-07 | 5E-07 | 1E-05 | 1E-05 |
| relTol | 0.05 | 0.01 | 0 | 0 |

*case3*

nCorrectors = 2、初期条件流れあり

| 項 | p | pFinal | u | k |
|---|---|---|---|---|
| solver | PCGgpu | PCGgpu | PBiCGgpu | PBiCGgpu |
| preconditioner | diagonal | diagonal | diagonal | diagonal |
| tolerance | 1E-06 | 1E-06 | 1E-05 | 1E-05 |
| relTol | 0.05 | 0 | 0 | 0 |

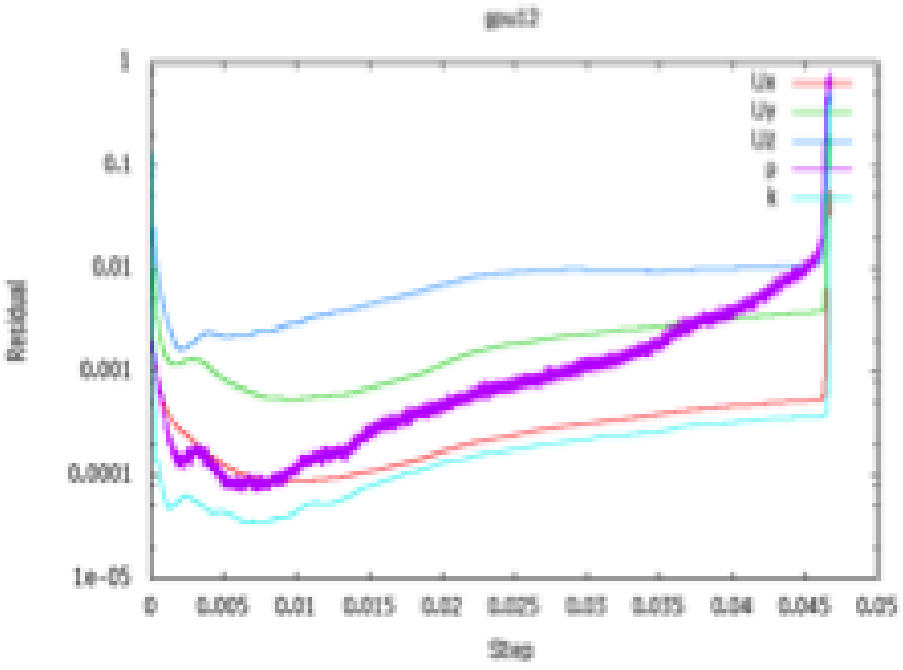
## *case4*

nCorrectors = 20、初期条件流れあり

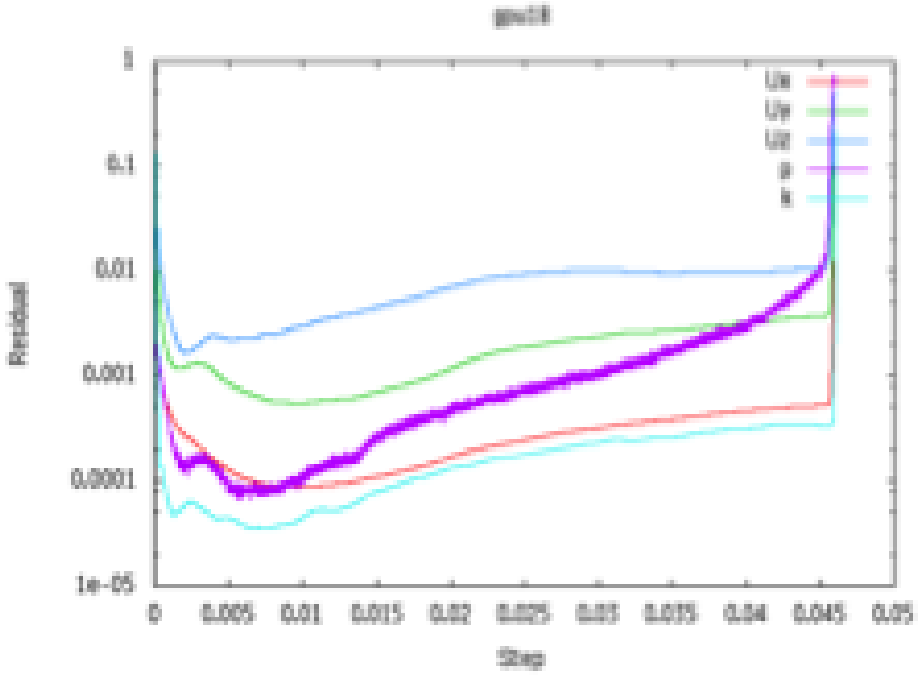
| 項 | p | pFinal | u | k |
|---|---|---|---|---|
| solver | PCGgpu | PCGgpu | PBiCGgpu | PBiCGgpu |
| preconditioner | diagonal | diagonal | diagonal | diagonal |
| tolerance | 5E-07 | 5E-07 | 1E-05 | 1E-05 |
| relTol | 0.05 | 0.01 | 0 | 0 |

## *case別相違条件*

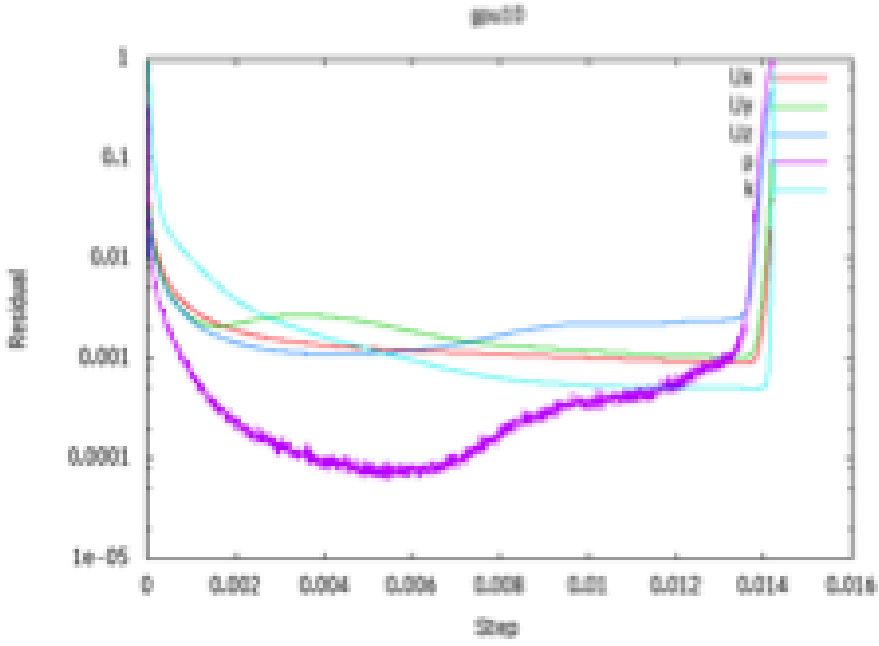
| case | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| nCorrectors | 2 | 20 | 2 | 20 |
| 初期条件流れ | なし | なし | あり | あり |
| ptolerance | 1E-06 | 5E-07 | 1E-06 | 5E-07 |
| pFinaltolerance | 1E-06 | 5E-07 | 1E-06 | 5E-07 |
| pFinalrelTol | 0 | 0.01 | 0 | 0.01 |



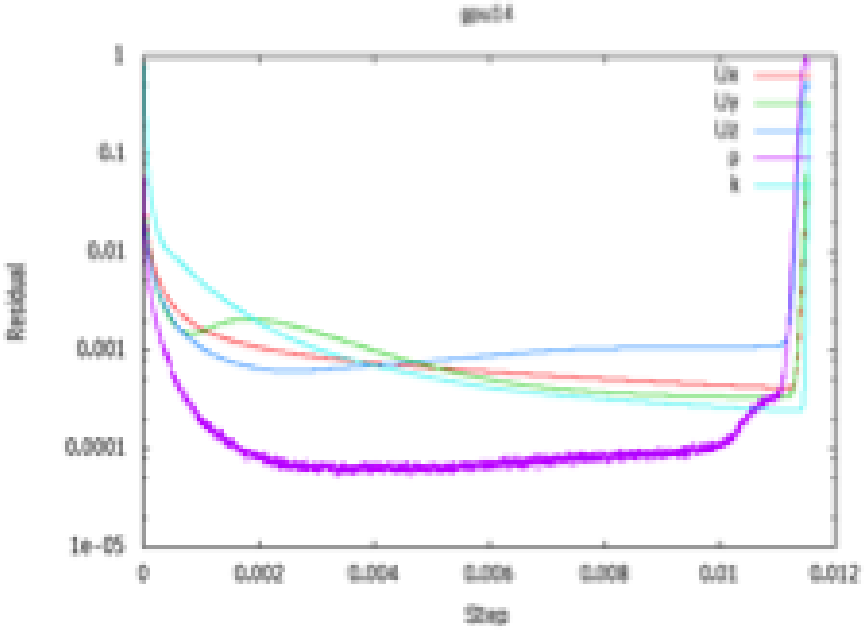
case1 グラフ



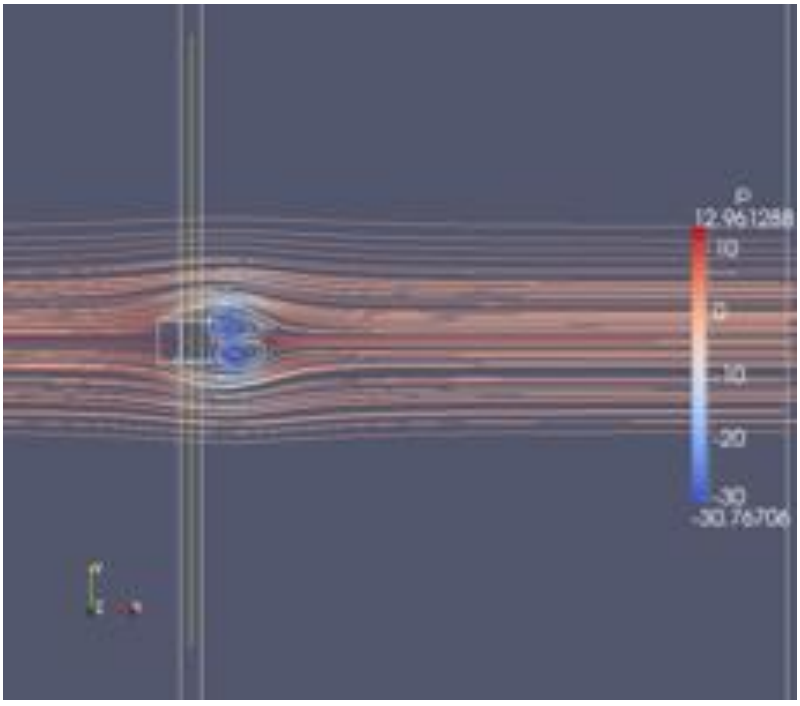

case2 グラフ

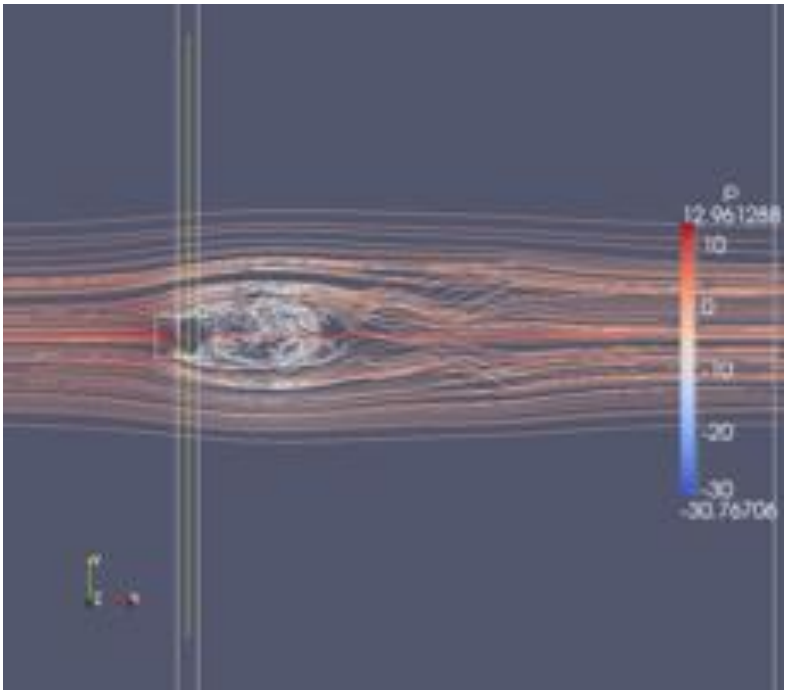


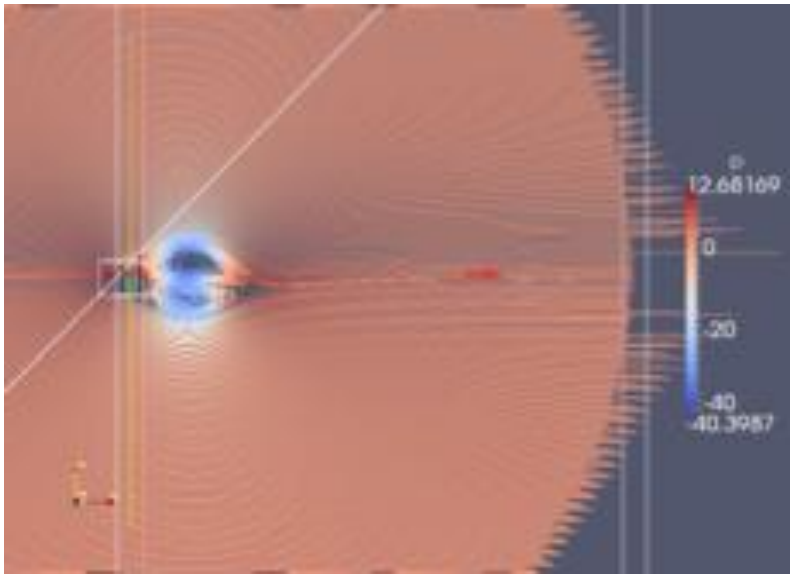

case3 グラフ

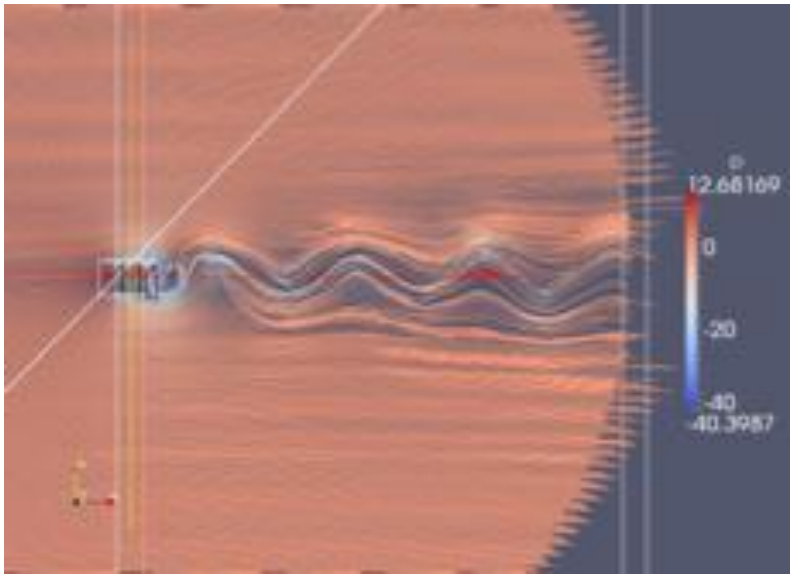

gnuplot
Residual
Ux
Uy
Uz
p
k
1
0.1
0.01
0.001
0.0001
1e-05
0
0.002
0.004
0.006
0.008
0.01
0.012
Step

case4 グラフ


p
12.961288
10
0
-10
-20
-30
-30.76706


p
12.96128
10
0
-10
-20
-30
-30.76706


p
12.68169
0
-20
-40
-40.3987




結果を見た場合、19case中4caseだけ解析が回っており、それも途中で発散しているため、gpuによる解析はできていないことになる。しかし、残差やtimestepsを見た場合、個々の解析条件の差異で内容が変わっている事もわかる。

まず、解析条件の初期条件流れを与えた場合、CPUでは与えても与えなくても計算ができたが、GPUの場合かなり早い段階で発散してしまっているので、GPUにおける解析で初期条件流れを与えない方が良いと考える。これは、初期条件流れを与えた場合に急激な変化が起こるため、現状のライブラリを使ったGPU解析では変化の許容範囲を超えてしまうのではないかと考える。

また、グラフ5-1、2を見た場合に、case12のほうが値のブレが大きくなっているが、これはcase18に比べてtolerance、relTolの値が大きいためであることがわかる。計算としては二つの値を小さくした方が早く収束し、安定する事がわかるが、一般的にこれらの値と計算時間は反比例しているため、case12で計算が回っている以上、計算時間を短くする事を優先して二つの値は1E-05で回すべきである。

前期解析したCPUの結果を見る限り、GPUによるcase12とcase18でもCPU解析による双子渦形成の段階までしか届いていない。カルマン渦の段階を飛ばしてそのまま渦が崩壊しているため、結果が発散してしまっている。

計算時間については、CPU 解析が約 2 日ほどかかるのに対し、case12 ではその半分の step しか回っていないにも関わらず、同じ 2 日かかっている。case18 では値を細かくしているため、5 日程かかっている。これは、計算を安定化さ

せるために、case12 においても値を CPU の時よりも細かく設定しているからである。そして、1step の 1 計算辺りの時間は GPU のほうが速かった。

***まとめ***

現状のライブラリを使った GPU 解析では、CPU 解析に比べて 1Step の 1 計算辺りの速度では上である。しかし、GPU 解析には計算の不安定さが目立ち、安定させるために値を細かく設定する必要があるため、結果的に時間がかかってしまう。また、そこまで設定しても発散する可能性が CPU 解析に比べて高いため、GPU 解析の有効性については現状のライブラリでは確立する事が難しいと結論づける。