GPU를 이용하여 압축성 유동 해석을 위한 불연속 갤러킨 기법의 효율적인 계산

Efficient Computation of Discontinuous Galerkin Methods for Compressible Flow on GPU

장태규[1]*, 박진석[1], 김종암[1]
서울대학교[1]

초 록

본 연구에서는 GPU를 이용하여 불연속 갤러킨 기법 기반의 압축성 유동 해석 알고리즘을 효율적으로 계산하고자 한다. 불연속 갤러킨 기법은 고차 정확도를 얻기 위해 격자 내부를 여러 기저함수의 합으로 표현하고 이를 갱신하므로, 기존에 널리 사용되던 유한 체적법에 비해 격자당 계산량이 크게 증가한다. CPU와 달리 GPU의 경우 수백~수천 개의 계산 단위로 구성되어 있어, 이러한 불연속 갤러킨 기법 기반의 유동 해석 알고리즘을 효율적으로 병렬 처리할 수 있어, 계산 시간을 크게 단축시킬 수 있다. 충격파 부근에서 수치 진동을 제어하기 위해 다차원 공간 제한 기법을 적용하였으며, 이 기법 역시 GPU를 이용하여 효율적으로 구현되었다. NVIDIA에서 제공한 CUDA 라이브러리를 이용하여 GPU에서 계산을 수행하였으며, CPU기반의 순차 및 병렬 해석 프로그램과 계산 효율성을 비교하였다.

ABSTRACT

The present paper deals with the efficient implementation of higher-order discontinuous Galerkin(DG) methods to compute compressible flow using graphics processing unit(GPU). The solution of DG methods is the sum of base functions and each mode is updated, thus it requires tremendous computational costs. Graphics processing unit(GPU), consisting of hundreds or thousands small cores, is apt to massive parallel computations and can reduce computational time. To handle shock discontinuity robustly, multi-dimensional limiting process(MLP) is applied and implemented on GPU. The program is written in CUDA library by NVIDIA and the efficiency are compared with serial and parallel computations on CPU.

Key Words : Compressible Flow(압축성 유동), Discontinuous Galerkin Method(불연속 갤러킨 기법), Graphics Processing Unit(그래픽 처리 장치)

1. 서 론

최근 컴퓨팅 환경의 급격한 발달에 따라, 전산유체 역학을 기반으로 실제 비행체 주위 복잡한 유동 물리 현상을 분석할 수 있게 되었다. 최근에는 수백에서 수천 코어로 구성된 슈퍼컴퓨터를 이용하여 격자점이 수천만 개로 구성된 공간에서 비정상 유동을 해석하기도 한다. 그러나 이러한 계산을 수행하는데 있어서, 와류가 지배적이거나 비정상 특성이 강한 유동에 대해서는 현재의 컴퓨팅 환경과 정밀 유동 해석 알고리즘으로 신뢰성 있는 해를 얻는데 한계가 있다.

이를 극복하기 위해 최근 고차 정확도 수치 기법이 크게 각광을 받고 있다. 그 중에서 불연속 갤러킨(DG) 기법은 복잡한 형상에 쉽게 적용이 가능하며 고차 내삽시 최소의 정보만을 활용하는 등과 같은 다양한 장점이 있다. 이러한 고차 정확도 수치 기법을 이용하여 실제 압축성 유동을 해석하기 위해서는 추가적인 연구가 필요한데, 대표적으로 강건한 공간 제한 메커니즘 개발과 효율적인 계산 기법이 적용되어야 한다. 충격파에서 수치 진동을 억제하기 위해서 본 연구에서

는 다차원 공간 제한 기법(MLP)를 적용하였다. MLP는 유한 체적법(FVM)을 근간으로 개발되었는데,[1-3] 최근 이 기법은 DG framework으로 확장되었으며 다차원 충격파 주위에서 수치 안정성을 보장하면서도 복잡한 유동 구조를 정밀하게 포착하는 것을 확인할 수 있었다[4].

DG 기법은 격자 내부의 해를 여러 기저 함수의 합으로 나타내고 이를 갱신하게 되어, 기존의 해석 알고리즘에 비해 격자당 계산량이 크게 증가하였다. 이 계산을 효율적으로 하기 위해서는 많은 코어를 갖춘 환경에서의 병렬 컴퓨팅이 필요하다. GPU(Graphics Processing Unit)는 원래 컴퓨터 화면의 많은 점(픽셀)들을 계산, 표현하기 위한 장치로 빠른 계산을 위하여 수백 개 이상의 많은 계산 단위로 이루어져 있다. 최근에 이를 활용하여 과학 병렬 계산 연구가 활발하게 진행되고 있다. 특히 NVIDIA사에서는 자사 GPU에서 병렬 해석 프로그램을 쉽게 개발할 수 있도록 C 언어 기반의 CUDA(Compute Unified Device Architeture)라이브러리와 개발툴을 제공하고 있으며[5] 이를 이용한 과학 계산 및 유동 해석이 시도되고 있다. 본 연구에서는 고정밀 압축성 유동 해석을 위해서 MLP를 적용한 DG 기법 해석 프로그램을 GPU를 이용하여 계산 효율성을 향상시키고자 한다.

2. MLP 제한 기법이 적용된 비정렬 격자계 불연속 갤러킨 기법

2.1 불연속 갤러킨 기법

불연속 갤러킨 기법은 고차 정확도를 얻기 위해 격자 내부의 분포를 형상 함수의 선형 결합으로 나타낸다. 이 함수는 n차 다항 함수로 구성된 공간 V^n에서 정의된다.

$$\mathbf{Q}_i^h(\mathbf{x},t) = \sum_{l=1}^{n} \mathbf{Q}_i^{(l)}(t) b_i^{(l)}(\mathbf{x}) \tag{1}$$

$\mathbf{Q}_i^h$는 격자 T_i에서 근사화된 물성치 벡터이고, $b_i^{(l)}$는 형상 함수이다. 형상 함수는 다항 함수 공간 V^n의 기저 함수들은 모두 가능하나, 본 연구에서는 계산 효율성을 위해 직교 형상 함수를 사용하였다.

다차원 비정렬 격자계에서 쌍곡 편미분 방정식을 DG 기법에 따라 표현하면 다음과 같다.

$$\int_{T_i} \frac{\partial \mathbf{Q}}{\partial t} \mathbf{W} dV + \int_{\partial T_i} \mathbf{W} \mathbf{F}_{\mathrm{com}} \cdot \mathbf{n} dS - \int_{T_i} \nabla \mathbf{W} \cdot \mathbf{F}(\mathbf{Q}) \mathrm{dV} = 0, \tag{2}$$

여기서 n은 격자 바깥 방향 벡터이고, W는 실험함수 벡터로 다항함수 공간 V^n에서 표시된다. F는 flux 벡터, $\mathbf{F}_{\mathrm{com}}$은 수치 flux 벡터이다. 적절한 수치 flux를 사용하게 되면 다음과 같은 근사화된 지배방정식을 얻을 수 있다.

$$\int_{T_i} \frac{\partial \mathbf{Q}_i^h}{\partial t} \mathbf{B}_i dV + \int_{\partial T_i} \mathbf{H}(\mathbf{Q}_{ij}^h, \mathbf{Q}_{ji}^h) \cdot \mathbf{n} \mathbf{B}_i dS - \int_{T_i} \nabla \mathbf{B}_i \cdot \mathbf{F}(\mathbf{Q}_{ij}^h) dV = 0 \tag{3}$$

$\mathbf{Q}_{ij}^h$는 격자 T_i에서 주변 격자 T_j방향으로의 경계면에서의 물성치이다. $\mathbf{H}(\mathbf{Q}_{ij}^h, \mathbf{Q}_{ji}^h)$은 수치 flux 함수 벡터이고, $\mathbf{B}_i$는 다항 함수 공간 V^n의 기저 함수 벡터이다. 경계 및 영역에서의 적분은 Gauss Quadrature 방법을 적용하여 각각 $2n$, $2n+1$까지의 정확도를 가지도록 하였다.

불연속 갤러킨 기법의 경우 유한체적법과 다르게, 인접 격자 정보만으로도 매우 높은 정확도를 유지할 수 있는 장점이 있다. 그러나, 불연속 갤러킨 기법만으로는 쌍곡 편미분 방정식을 해석하는데 있어서 안정성 측면에서 문제가 있다. 이를 보완하기 위해 본 연구에서는 강건한 공간 제한 기법인 MLP 제한 기법을 이용하였다. 시간 적분 기법으로는 3차 정확도의 TVD Runge-Kutta 기법을 적용하였다.

2.2 DG-MLP 기법

다차원 공간에서 단조성을 보장하기 위해서, MLP 기법에서는 1차원 단조 조건을 확장한 다차원 공간 제한 기준을 제안하였다. 각 꼭지점에서의 물성치까지 공간 제한하여, 물성치 평균값이 단조성을 보장하도록 하는 것이다. 이러한 MLP 조건은 다음과 같이 정리할 수 있다.

$$\bar{q}_{v_i neighbor}^{\min} \leqq q_{v_i} \leqq \bar{q}_{v_i,neighbor}^{\max}, \tag{4}$$

여기서 q는 물성치이고 q_{v_i}는 꼭지점 v_i에서 값이다. $\left(\bar{q}_{v_i,neighbor}^{\min}, \bar{q}_{v_i,neighbor}^{\max}\right)$는 꼭지점 v_i를 공유하는 셀의 평균값의 최대 최소이다. 이 MLP 조건은 근본적으로 격자 위상 정보와 관계없이 적용할 수 있다. 효율성과 정확성을 고려하여 q_{v_i}를 근사

화하는 방법에 따라 정렬 및 비정렬 격자계에 적용할 수 있다.

기존의 유한체적법에서는 격자 내부를 선형 분포로 가정하여 MLP 조건을 이용해 maxium principle을 위배하는 셀을 찾고 이 셀에 대해 공간제한 기법을 적용할 수 있었다. 그러나 셀 내부를 선형 이상의 고차 정확도로 내삽하는 경우 이러한 기준을 바로 적용하는데 한계가 있다. 고차 다항식 근사화된 셀에 대해 MLP 조건만으로는 troubled-cell을 모두 포착하는데 한계가 있다. 이를 보완하기 위해서 아래와 같이 강화된 조건을 제안하였다.[4]

$$\bar{q}_{v_i}^{\min} \leqq q_{v_i}^{h,\min} \leqq q_{v_i}^{h}, q_{v_i}^{h} \leqq q_{v_i}^{h,\max} \leqq \bar{q}_{v_i}^{\max}, \quad (5)$$

여기서 $q_{v_i}^{h}$는 꼭지점 v_i에서 내삽된 값이며 $\left(\bar{q}_{v_i}^{\min}, \bar{q}_{v_i}^{\max}\right)$는 꼭지점 v_i를 공유하는 셀의 최소/최대 값이다.

강화된 조건을 적용시 극점 주위에서 물성치가 과도하게 제한되어서 정확성이 저해될 수가 있다. 이를 방지하기 위하여 MLP Stencil을 고려하여 극점을 구별할 수 있는 다음의 간단한 형태의 Extrema detector를 적용하였다.

$$\Delta\bar{q}_{v_i} = \bar{q}_{v_i}^{\max} - \bar{q}_{v_i}^{\min} \leqq (K\Delta x)^2 \quad (6)$$

여기서 K는 유동 현상에 따라 결정되는 계수이며, 본 계산에서는 K를 1에서 100사이의 값을 사용하였다.

3. GPU를 이용한 병렬 계산

3.1 GPU를 이용하는 CUDA 프로그램 구조

CUDA 라이브러리는 C 언어를 기반으로 하면서 원하는 병렬 해석할 코드를 GPU의 많은 코어에서 병렬로 계산하는 것을 가능하게 한다. CUDA 프로그램은 CPU영역인 호스트(host)와 GPU영역인 디바이스(device)를 모두 이용하여 수행 가능하도록 되어 있다. 데이터 병렬성이 적은 부분은 호스트 코드로 구현되고, 데이터 병렬성이 큰 부분을 디바이스 코드로 구현하게 된다. 전체적인 코드는 순차적인 ANSI C 코드로 되어 있으며, 디바이스 코드는 여기에 추가로 커널(kernel) 함수 또는 커널이라고 불리는 데이터 병렬 함수가 포함된다. NVIDIA C 컴파일러(nvcc)는 이 둘을 분리하여 컴파일하게 된다.

커널은 일반적으로 데이터를 병렬로 처리하기 위해 디바이스에 많은 수의 쓰레드(thread)를 생성한다. 쓰레드는 하나의 GPU 코어가 할당된 것이며, 커널 코드의 같은 부분을 동시에 실행하게 된다. 따라서 일반적으로 루프문의 하나의 루프를 하나의 쓰레드에서 수행하는 방식으로 커널 코드를 작성한다. 조건문의 경우는 특정 조건에 따라 그 코드를 쓰레드가 수행하느냐 아니냐의 분기가 일어나서 계산의 병렬성을 떨어뜨리기 때문에 커널에는 가능하면 피하는 것이 좋다.

GPU메모리는 CPU메모리와는 물리적으로 다른 칩에 존재하기 때문에 GPU에서 계산하기 위해서는 CPU메모리에 저장된 데이터를 GPU메모리에 복사하는 것이 필요하다. CUDA 라이브러리는 이런 메모리 할당, 복사, 해제 등을 함수의 형태로 제공한다. 따라서 기본적인 CUDA 프로그램 구조는 호스트와 디바이스에 메모리를 할당하고 디바이스에 처리할 데이터를 복사, GPU에서 병렬 계산 수행, 수행 결과를 호스트로 복사, 호스트와 디바이스 메모리 해제의 순서로 이루어진다.

3.2 DG-MLP 기법의 CUDA 프로그램 작성

DG-MLP 기법을 비정렬 격자계에서 적용하여 압축성 유동을 해석할 때, 계산 시간을 가장 많이 차지하는 핫스팟(hot-spot)은 공간 차분을 수행하는 부분이다. 격자 내부와 경계면에서의 구분구적법 계산을 포함하며, 모든 경계면에 대해 수치 flux 계산을 해야 하기 때문에 그 계산 시간이 길 수밖에 없다. 그래서 이 부분을 계산하는데 있어, 데이터를 병렬성이 보장되도록 하여 GPU의 쓰레드에 분배, 병렬 계산할 수 있는 디바이스 코드를 작성하였다.

DG-MLP 기법의 지배방정식인 식 (3)에서 시간 적분 항과 세 번째 항은 각 격자에서 계산되지만, 수치 flux는 격자의 경계면에서 계산되어야 한다. 격자 내부에서 계산되는 항들은 하나의 쓰레드에 하나의 격자를 대응시켜 격자가 가지고 있는 물성치, 부피 등의 데이터를 읽어서 각 격자에 대해 병렬로 계산을 수행할 수 있다. 수치 flux항을 위와 같이 병렬 계산하려면 격자의 경계면 각각에 대해 인접 격자의 정보를 읽어서 수직 방향 벡터와 flux를 계산하고, 이를 합하여 수치 flux항을 구하여야 한다. 결국 이는 한 격자 경계

면에 대하여 양쪽의 격자에서 중복된 계산을 하는 결과를 가져온다. 본 연구에서는 이런 중복된 계산을 피하기 위해, 하나의 경계면을 하나의 쓰레드에 할당하여 수치 flux항을 병렬 계산하고, 다시 시간 적분을 병렬 계산하는 과정에서 격자의 각 경계면의 수치 flux를 공간 차분항에 합하는 형태로 계산을 수행하였다.

4. 수치 실험 결과

대표적인 압축성 유동 예제인 마하 10의 이동 충격파가 30도 경사각을 갖는 쐐기와 부딪히면서 생기는 복잡한 물리 현상을 해석하였다. RoeM 수치 flux 기법을 사용하였고 $t = 0.2$까지 계산을 수행하였다. Figure 1은 격자 특성 길이가 $h = 1/300$인 경우 해석한 결과이다. CPU 코드와 해석 결과는 거의 같으며 충격파에서 수치 진동을 억제하면서도 미끄럼 유선 주위의 복잡한 와류 구조를 정밀하게 포착할 수 있었다.

Table 1은 순차 및 병렬 CPU 코드와 GPU 코드의 계산 시간(Wall clock)을 격자 크기에 따라 비교한 결과이다. CPU 순차 해석은 2.40GHz Intel Xeon CPU를 사용하였으며, CPU 병렬 해석은 OpenMP 병렬 프로그래밍을 기반으로 동일 Xeon CPU 2개를 SMP(Symmetric Multiprocessing)로 구성 총 8개의 코어를 사용하였다. GPU 계산은 512개의 코어를 가진 NVIDIA Geforce GTX580을 사용하여 해석하였다. GPU로 계산시 CPU보다 약 2배 빠른 것을 확인할 수 있다.

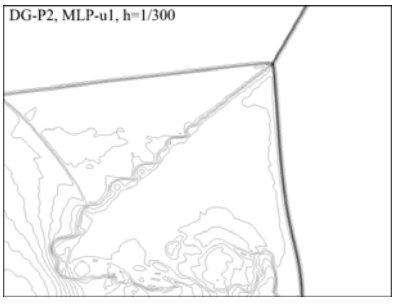

Fig 1. GPU로 계산한 압축성 유동 해석 결과

Table 1. 계산 환경에 따른 계산 시간 비교 (단위 : 시간)

격자 크기	*h=1/100*	*h=1/200*	*h=1/300*
CPU 순차 계산	1.98	16.21	57.32
CPU 병렬 계산	0.40	3.32	11.22
GPU 병렬 계산	0.20	1.62	5.56

5. 결 론

본 연구에서는 DG-MLP 기법을 이용하여 고정밀 압축성 유동을 해석하는데 있어서 CUDA 라이브러리를 이용한 GPU 병렬 계산을 수행하였다. 큰 계산시간이 요구되는 격자 내부와 경계면에서의 공간 차분항을 데이터 병렬성이 보장되는 디바이스 코드로 작성함으로써 GPU의 수백~수천 개의 코어에서 병렬 계산할 수 있다. 기존의 단일 및 병렬 CPU 해석 코드와 비교했을 때 GPU 계산이 효율성을 크게 증가시킬 수 있음을 확인할 수 있었다.

후 기

본 연구는 한국연구재단을 통해 교육과학기술부의 우주기초원천기술개발 사업(NSL, National Space Lab, 과제번호 2012-0009099), 교육과학기술부 첨단사이언스 교육허브개발사업(EDISON, 과제 번호 2012-0006661)으로부터 지원받아 수행되었습니다.

참고문헌

(1) Kim, K. H. and Kim, C., 2005, "Accurate, Efficient and Monotonic Numerical Methods for Multi-dimensional Compressible Flow Part II: Multi-dimensional Limiting Process," *Journal of Computational Physics*, Vol. 227, pp. 570~615.

(2) Park, J. S., Yoon, S. H. and Kim, C., 2010, "Multi-dimensional Limiting Process for Hyperbolic Conservation Laws on Unstructured Grids," *Journal of Computational Physics*, Vol. 229, pp. 788~812.

(3) Park, J. S. and Kim, C., 2012, "Multi-dimensional Limiting Process for Finite Volume Methods on Unstructured Grids," *Computers & Fluids*, Vol. 65, pp. 8~24.

(4) Park, J. S. and Kim, C., submitted, "Higher-order Multi-dimensional Limiting Strategy for Discontinuous Galerkin Methods in Compressible Inviscid and Viscous Flows," *Computers & Fluids*.

(5) NVIDIA, 2012, “CUDA C Programming Guide,”