구분별 갱신 방식을 이용한 유한차분시간영역의 객체 지향적 구현

전경원, 김희운, 정영주*
광주과학기술원 정보통신공학과
e-mail: *ychung@gist.ac.kr

Implementation of the Finite-Difference Time-Domain Method in Object-Oriented Programing Using Piecewise Update Scheme

Kyungwon Chun, Huioon Kim, and Youngjoo Chung
Dept. of Information and Communications, Gwangju Institute of Science and Technology

요 약

GMES 는 유한차분시간영역 방법의 객체지향적 구현이다. 기존의 구현에선 점별(piecewise) 갱신 방식을 사용해서 물성 표현 능력은 우수하지만, 계산 시간과 메모리 요구량에서 큰 약점이 있었다. 구분별(piecewise) 갱신 방식을 반영해 설계를 변경한 결과 물성 표현 능력은 기존과 같은 수준으로 유지하면서 70~90%에 이르는 계산 속도 개선과 사용 메모리 감소의 효과를 얻을 수 있었다.

1. 서론

유한차분시간영역(Finite-Difference Time-Domain, FDTD) 방법은 미분방정식 형태의 맥스웰 방정식을 차분방정식의 형태로 변형한 뒤, 시공간 영역에서 해를 구하는 수치해석 알고리듬으로, 최소한의 제한조건만으로도 해를 구할 수 있어서 전자기파 관련 분야에서 폭넓게 사용되고 있다[1]. 전산모사를 하려는 대상을 FDTD 구현 안에서 정확하게 표현하려면 대상을 이루는 전파 매질의 전자기적 성질을 정확하게 표현할 수 있는 수치모형과 대상의 기하학적 형태를 세밀하게 표현할 수 있는 도구와 FDTD 알고리듬이 필요하다. FDTD 구현에서 전파 매질을 구현하는 방식은 다시, 모든 종류의 전파 매질을 단일 수치모형으로 표현하는 방식과 각 전파 매질에 적합한 여러 수치모형을 사용하는 방식으로 나눌 수 있다.

단일 수치모형으로 전체 전파 매질을 표현하는 방식은 구현이 단순해지는 장점이 있지만, 필요 이상으로 지나치게 복잡한 수치모형을 전체 계산 구간에서 사용해야 하므로 메모리 사용량이 급격히 늘어나는 문제와 수치모형을 변경하면 전체 FDTD 알고리듬이 영향을 받게 되어 모든 부분을 다시 검증해야 하는 문제가 있다. 전산모사 대상을 구성하는 전파 매질의 종류에 따라 다른 수치모형을 사용하는 구현 방식은 각 매질의 전자기적 특성에 최적화된 수치모형을 사용할 수 있지만, 자칫 구현이 복잡해지고 적절한 계산속도와 메모리 요구량을 유지하기 어렵다는 단점이 있다.

2. FDTD 알고리듬 분석

FDTD 알고리듬의 기본구조는 Δt/2 시간단계마다 전기장과 자기장을 번갈아 가며 갱신하게 되어 있다. 각 공간지점의 갱신에 사용하는 수식과 해당 자룟값은 다른 공간지점과 무관하며, 같은 시간단계에서 각 공간지점은 임의의 순서로 갱신할 수 있다. 각 공간지점의 공간적 독립성은 각 공간지점의 갱신 수식과 자룟값으로 이루어진 객체를 사용할 수 있음을 의미한다. 각 공간지점에 각각의 갱신 수식과 이 수식에 필요한 자룟값으로 이루어진 객체를 사용하면 임의의 매질 수치모형을 임의의 공간지점에 사용할 수 있으므로 매질 표현의 자유도를 극대화할 수 있다. 하지만, 공간지점마다 객체를 생성하여 사용하게 되므로 매질을 표현하는 클래스가 동적 바인딩을 사용하거나 [2] 클래스를 스크립트 언어로 래핑해서 사용하면 메모리 사용량이 터무니없이 커지게 되는 문제가 발생한다.

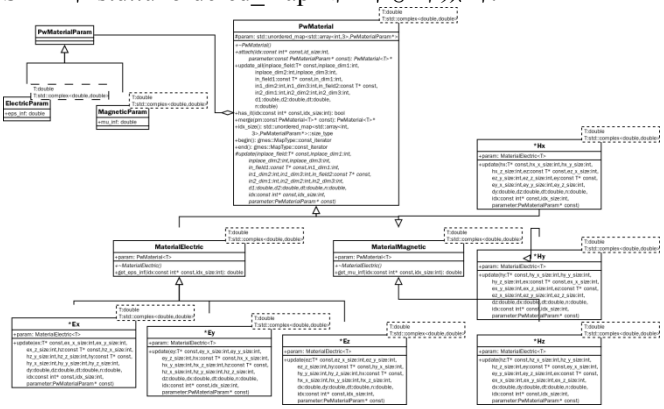
앞서 언급한 갱신순서의 무관함을 이용하면 매질 표현의 자유도를 위의 설계와 같은 수준으로 유지하면서도 메모리 사용량의 문제를 상당 부분 개선할 수 있다. 동일 시간단계에선 각 공간지점의 전기장 혹은 자기장 갱신순서가 무관하므로 같은 수치모형으로 표현되는 공간지점을 모두 모아서 하나의 객체로 표현한 후, 각 수치모형 별로(piecewise) 갱신 할 수 있다. 또한, 각 공간지점에 해당하는 자룟값은 동일한 자료구조로 표현하여 객체의 속성으로 저장할 수 있다.

FDTD 알고리듬은 여러 단계의 병렬화가 가능하다. 각 공간지점의 갱신 알고리듬은 해당 위치의 물질 파라미터와 바로 옆의 전자기장의 값만을 필요로 하므로 MPI 를 이용한 병렬화에 적합하다. 또한, 동일한 시간 단계에선 전기장의 각 컴포넌트 혹은 자기장의

각 컴포넌트는 이전 시간 단계의 자기장 혹은 전기장 값이 요구하며, 동시에 갱신할 수 있으므로, 쓰레드를 이용한 병렬화에 적합하다. 또한, 더 나아가 위의 설계처럼, 같은 수치모형으로 표현되는 공간지점을 모두 모아 한꺼번에 전기장 혹은 자기장을 갱신하면 범용 GPU(General-Purpose Graphics Processing Unit, GPGPU)를 이용한 Single-Process Multiple-Data (SPMD) 병렬화를 적용할 수도 있다. 아래에서 다룰 구현은 모든 병렬화를 대비한 설계는 되어 있지만, MPI 를 이용한 병렬화만 적용하기로 한다.

3. 구분별 갱신 방식의 구현

구분별(piecewise) 갱신 알고리듬을 반영하여 (그림 **1**)과 같이 설계한 FDTD 갱신 알고리듬은 계산속도와 메모리 효율성을 고려하여 컴파일 언어로 구현하는 방안을 선택했다. 또한, 객체를 효율적으로 표현할 수 있어야 하므로, 일반적으로 FDTD 구현에서 사용하는 포트란이나 C 가 아닌 C++로 구현하였다. 갱신 수식에서 사용할 자룟값은 표준 템플릿 라이브러리(Standard Template Library, STL)의 std::array, std::vector 등을 속성으로 갖는 구조체로 구현하였다. 이들 구조체는 순서가 중요하지 않으며 빠른 조회가 가능해야 하므로, 각 공간지점의 배열 색인을 킷값으로 하여 STL 의 std::unordered_map 에 저장하였다.



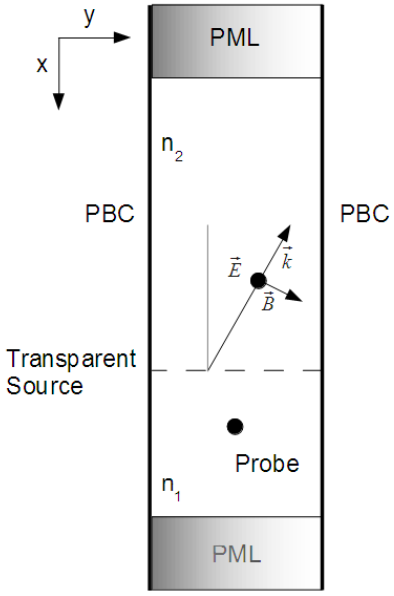
(그림 1) 구분별 갱신 알고리듬을 반영한 전파 매질 객체의 클래스 다이어그램

(그림 **1**)의 구현은 비분산 등방성 유전체 모형과 드바이, 드루드, 로렌츠, 드루드-임계점(Drude Critical-Points) 등의 분산 모형은 물론이고, UPML(Uniaxial Perfectly Matched Layer), CPML(Convolutional Perfectly Matched Layer) 등의 흡수경계조건(Absorbing Boundary Condition, ABC) 등을 모두 구현할 수 있고 실제로 이들을 구현하였다. 비선형 매질 모형이나, 능동 매질 모형 등은 아직 구현하지 않았지만, 구현에 문제가 없을 것으로 추정한다.

이들 전파 매질 클래스(PwMatrial)를 제외한 거의 모든 부분은 개발 및 유지보수 비용 절약과 사용 편의성을 고려하여 파이썬으로 구현하였다. 전체 FDTD 알고리듬 및 입력 광원, 파일 입출력, 실시간 가시화 등 모든 부분도 역시 객체지향 설계를 기반으로 파이썬으로 작성했다. 전체 구현은 GIST Maxwell's Equations Solver(GMES) 프로젝트[3]의 pw_redesign 브랜치에서 찾을 수 있다.

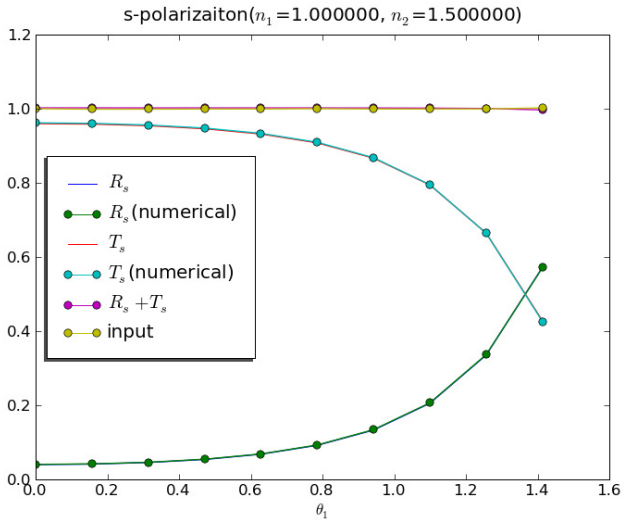
4. 성능측정

기존의 GMES 구현은 공간지점마다 FDTD 갱신 객체를 생성하는 방식으로, 이러한 점별(piecewise) 갱신 방식은 메모리 사용량이 크고 계산 속도도 느린 단점이 있었다. 기존의 GMES 구현과 새로운 GMES 구현의 메모리 사용량과 속도를 비교해 보았다. 사용한 전산모사 모형은 매질의 반사율과 투과율을 분석하는 용도로 사용하는 (그림 1)과 같은 2 차원 구조이다. 이 모형의 전파 매질은 비분산 등방성 유전체와 Perfectly Matched Layer(PML)로 이루어져 있으며, 무한 평면을 표현하기 위해 양쪽 경계에 주기적 경계 조건(Periodic Boundary Condition, PBC)을 설정했다. 입사 광원은 평면파로 큰 입사각에서, 정확한 파형을 입사하려면 광원의 세기를 천천히 올려야 하므로 많은 시간 단계를 계산해야 한다.



(그림 2) 구현의 검증과 성능 측정에 사용한 시뮬레이션 모형

입사각은 0 부터 $0.9\pi/2$ 까지 $0.1\pi/2$ 단위로 증가하면서 계산했고, 각 입사각에 대한 계산은 공간을 반으로 나누어 병렬 계산했다. 아래는 이 계산 결과로 실선으로 표시한 이론적 예측값과 점으로 표시한 시뮬레이션 결과가 일치하는 것을 확인할 수 있다.



(그림 3) 평면파가 1 의 굴절율을 갖는 공기에서 1.5 의 굴절율을 갖는 유전체로 입사할 때의 투과율과 반사율 그리고 이들의 합

(그림 3)의 결과를 얻으려면 공기 중에서 입사파의 세기와 유전체 경계면이 존재할 때의 전자기장 세기를 측정해야 하므로 각 입사각에 대해서 두 번의 시뮬레이션이 필요하다. <표 1>에서 공기만 존재하는 공간에 평면파를 입사하는 설정의 실행시간은 Ref*로 유전체 경계면이 존재하는 설정의 실행시간은 Exp*로 표시했다. 실행시간을 조사해 보면, 초기화 시간은 약 76.62%와 73.59 %가 개선된 것을 알 수 있고, 28570 번째 시간단계까지 계산에 걸린 시간은 90.78%와 84.78%가 개선된 것을 알 수 있다.

<표 1> 예전 설계와 새로운 설계의 실행시간 비교. Ref Init 과 Exp Init 은 공기만으로 이루어진 공간에서의 초기화 시간과 유전체를 위치시켰을 때의 실행시간, Ref Init 과 Exp Init 은 28570 번째 시간단계까지 계산에 걸린 시간. 모든 시간은 벽시간(walltime)으로 측정함.

	Ref Init (초)	Exp Init (초)	Ref Step (초)	Exp Step (초)
구설계	283.39	271.09	6622.06	4129.76
신설계	66.27	71.60	610.52	628.68

또한, 메모리 사용량의 최댓값을 조사해 보면, 이전 구현은 각 계산 노드당 약 128.71MB 지만, 새로운 구현은 해시표를 사용하면서도 약 114.81MB 로 10.80% 개선된 것을 확인할 수 있다.

예전 설계에선 파이썬 계층에서 복잡한 물질 클래스를 메모리에 할당해야 했지만, 새로운 설계에선 C++ 계층에서 최소한의 속성만 갖는 클래스를 메모리에 할당하므로 실행시간이 크게게 단축된 것을 알 수 있다. 시간단계를 밟아나가며 FDTD 알고리듬에 따라 전자기장을 계산하는 루프가 파이썬 계층에서 numpy.ndarray 에 저장된 물질 클래스를 순회하는 것보다, C++ 계층에서 std::unordered_map 을 순회하는 것이 훨씬 빠른 것을 확인할 수 있다.

5. 결론

FDTD 의 객체지향적 설계를 구현한 GMES 의 알고리듬을 깊이 있게 분석해 기존의 점별(pointwise) 갱신 방식을 구분별 (piecewise) 갱신 방식으로 변경한 결과 각 물질 모형별로 C++ 계층에서 갱신 순회가 가능하여 상당한 계산 속도의 개선과 메모리 사용량의 감소를 이룰 수 있었다.

6. 감사의 글

This work was partially supported by BK-21 Information Technology Project and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (R15-2008-006-03001-0), Republic of Korea.

참고문헌

[1] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method, Third Edition*, 3rd ed. 685 Canton Street, Norwood, MA 02062, USA: Artech House Publishers, 2005.

[2] A. Koenig and B. E. Moo, *Accelerated C++: Practical Programming by Example*. Addison-Wesley Professional, 2000.

[3] "GMES". [Online]. Available: http://sourceforge.net/projects/gmes/. [Accessed: 09-9-2011].