문제 2 (학부/대학원 공통): 격자 기반 PDE 연산 고속화를 위한 멀티 그리드 계산 병렬화 (배점: 학부 30 점/대학원 20 점)

문제 개요

멀티그리드(Multi-grid) 방법은 격자 기반 PDE 문제를 풀기 위해 2개 이상의 계층적인 격자 구조를 이용하는 방법으로, 격자 기반 PDE 계산을 위한 선형 솔버와 같이 사용되어 계산의 효율을 높일 수 있는 수치해석 기법 중 하나이다.

멀티그리드 방법의 핵심적인 개념은 격자가 조밀한 fine grid에서의 해를 보다 빨리 수렴시키기 위해, 격자가 성기지만 계산량이 작은 coarse grid에서의 해를 이용하는 것이다. 이를 위해 coarse grid의 해를 fine grid의 해를 구하기 위한 초기조건으로 이용하거나, fine grid에서의 수렴오차에 대한 보정항(correction term)으로 이용하는 방법이 있다. 본 문제에서는 coarse grid의 해를 초기조건으로 이용하는 방법을 사용한다. 아래 그림 1은 서로 다른 두 개의 그리드를 이용하는 멀티그리드 방법의 활용 flow을 보여준다.

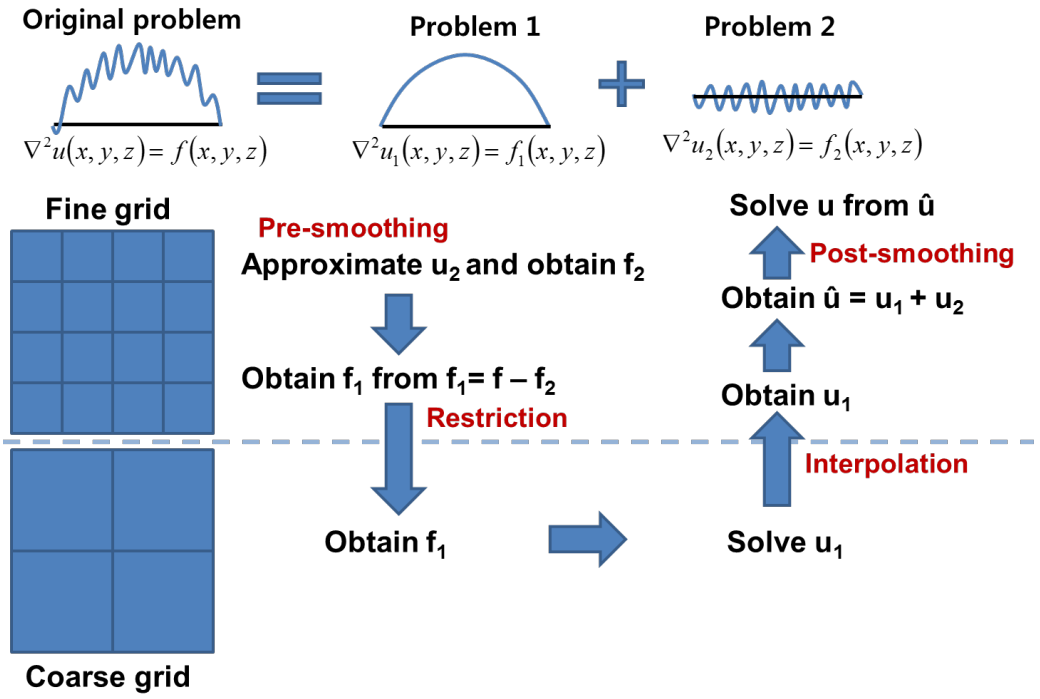

그림 1. Basic flow of two-grid multigrid method

멀티그리드 방법의 핵심 계산은 3가지로 구성되어 있으며, 해당 그리드에서 해를 구하기 위한 **solving** (또는 smoothing라고도 함), find grid에서 coarse grid로 정보를 보내는 **restriction**, coarse grid에서 fine grid로 정보를 보내는 **interpolation** (또는 prolongation 이라고도 함)이 이에 해당한다. 계산 알고리즘은 다음과 같다.

1. Fine grid에서 매우 제한된 횟수만큼만 반복계산을 하여 u_2와 f_1, f_2를 계산
2. Fine grid의 f_1을 restriction하여 coarse grid에서의 새로운 RHS인 f_1을 계산
3. Coarse grid에서의 해인 u_1를 계산한 후, interpolation하여 fine grid의 해로 변환
4. u_1과 u_2로부터 해를 계산

이 과정은 아래 그림과 같이 fine grid와 coarse grid의 계산을 하기 때문에 V-cycle라고 불리며, 그리드 수를 늘려 여러 단계의 multi-grid를 이용할 수 있다.

그림 2. V-cycle multi-grid

문제 설명

본 문제에서는 4번 문제로 이어지는 2차원 Poisson 방정식의 풀이를 위한 멀티그리드 방법 중 restriction 과정과 interpolation 과정을 다룬다. 문제를 단순화하기 위해 아래와 같이 변형된 멀티그리드 방법을 이용한다.

1. Pre-smoothing과 post-smoothing 등 모든 smoothing 과정을 제거
2. Coarsest level에서의 해는 다음과 같이 격자점이 하나만 존재하는 2차원 Poission 방정식의 해로 주어짐

$$\mathrm{u} = \frac{RHS}{4}$$

3. 그림 2에 설명된 V-Cycle을 20회 반복하며, 매 V-cycle에서 얻어진 해를 다음 V-Cycle의 입력으로 이용한다.

본 문제의 격자는 아래와 같이 셀의 절점(vertex) 대신 셀의 중간(center)에 위치하고 있으며, 이 경우 멀티그리드 방법의 핵심인 restriction과 interpolation은 다음과 같이 주어진다.

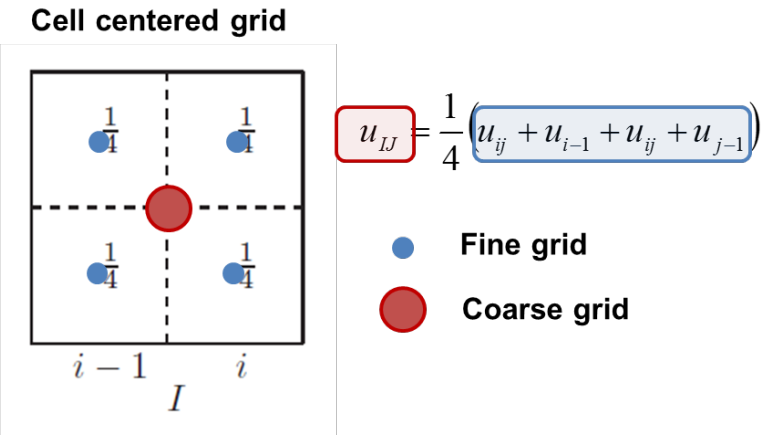

그림 3. Restriction

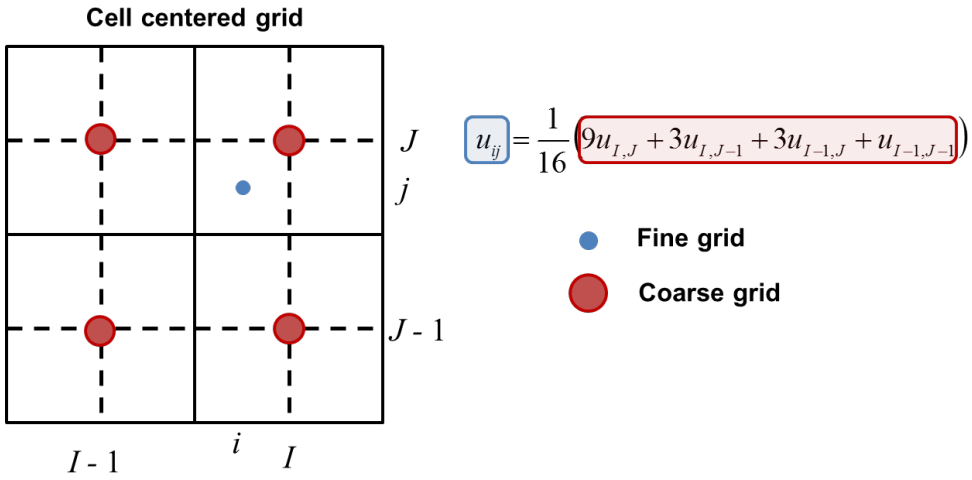

그림 4. Interpolation

본 문제의 병렬화 과정에서 가장 중요한 부분은 크게 두 가지이다. Interpolation시에 필수적으로 필요한 이웃한 영역의 coarse grid에서의 값과 경계 부분을 어떻게 처리해야 하는지가 첫 번째 해결 사항이고, restriction을 반복하다 보면 grid의 숫자가 MPI 프로세스 수보다 작은 상황이 발생하는데 이를 어떻게 처리해야 할지가 두 번째 해결해야 할 사항이다.

참고 및 유의 사항

1. 주어진 순차코드는 V-cycle 멀티그리드 방법을 주어진 개수의 계층적 격자에 대해 적용하여 푸는 문제이다. 제시된 멀티그리드 알고리즘의 실행 Flow 및 주어진 순차코드의 subroutine 호출구조를 유지하는 범위에서의 코드변경은 허용하나, 주어진 restriction 및 interpolation 외 다른 알고리즘을 사용하거나 다른 공개코드를 사용해 주어진 subroutine 의 호출 구조를 변경하는 행위는 일체 금지한다.
2. 주어진 순차코드는 main.c (f90) 에서 multigrid.c (f90) 에 정의된 subroutine 들을 호출해 사용하는 방식으로 수행된다. Compile 및 Link를 위한 Makefile 이 함께 주어져 있으며 이를 변경하는 것은 금지한다.
3. 동적메모리의 할당/삭제는 반드시 시간측정 구간 안에서 수행한다. 순차코드에 정의된 그리드 레벨 수, 그리드 크기 (gridsize)값의 변경은 금지한다.
4. 본 문제는 멀티그리드에 대한 사전지식이 없이 코드 flow만 이해하면 풀 수 있다는 점을 참고한다. 주어진 문제에 대해 핵심적인 연산은 restriction과 interpolation에 국한되며 이 연산을 위한 인접 그리드와의 관계만 파악하면 해석 가능한 알고리즘이다. **알고리즘의 이해를 위해 필요한 경우 Staff에게 질문할 수 있다.**

평가 방법

1. 병렬코드가 32 core 에서 수행되는 시간을 time command 로 측정한다.
2. 순차코드 실행이 끝나면 result 폴더 밑에 solution.dat 파일이 생성된다. 병렬코드의 해는 순차코드의 해와 오차범위내(1e-6)에서 일치해야 하며 조건이 만족되지 않는 경우 코드의 정확성 검증 실패로 결격 처리.