Geometry-based Adaptive Octree 방법에 대한 고찰

Analysis of Using Geometry-based Adaptive Octree Method

O박종렬[1], 사종엽[2]

Jong-Ryoul Park, Jong-Youb Sah

Automatic method for generation of mesh and three dimension natural convection flow result adapted by this method are presented in this paper. It take long time to meshing complex 3-D geometries, and It's difficult to clustering grid at surface boundary. Octree structure resolve this difficulty.

1. 서론

수치해석을 이용한 유동 현상을 해석하는 전산유체역학이라는 학문의 발전은 산업전반에 중요한 기술로 적용되어 왔다. 컴퓨터 성능의 향상으로 유동 현상을 고려한 설계가 활발히 이루어지고 있으며 많은 발전을 해왔다. 이러한 발전은 3 차원 유동해석에 대한 요구를 가져다 주었으며 형상 또한 복잡하여 격자생성이 어렵고 상당한 시간을 요구한다.

S.Chang[5] 은 유동을 해석하는 과정에서 Octree 형태의 적응격자를 사용하여 수렴속도를 비교하였고 그외 선행연구자들 또한 유동 영역에서 Octree 형태의 적응격자를 사용한 반면에 본 연구에서는 기하학적 형상이 복잡한 경우에 대한 격자 형성을 빠르게 하기 위해서 Octree 형태의 격자를 사용하였다. 복잡한 3 차원 형상에 대해 Octree 격자를 형성할 경우 많은 수의 격자가 필요하게 된다. 많은 수의 격자는 많은 메모리가 필요하다. 따라서 Cell 당 차지하는 메모리 사용량이 중요한 문제가 된다. 본 연구에서는 Octree 구조의 특성을 이용하여 메모리를 절감하는 방안를 생각하였고 메모리 절감으로 인한 계산속도에 미치는 영향을 분석하였다.

2. 자료 구조

2.1 Octree 구조

Octree 의 기본적인 구조는 Parent 와 Child 관계를 유지하는 tree 구조를 하고 있다. Fig.1 은 Parent 와 Child 의 관계를 나타낸 것이다. 초기 cell 은 모두 Level 0 인 Child cell 이다. Level 를 올릴 경우 Child cell 은 Parent cell 이 되고 이 Parent cell 은 8 개의 Child cell 를

1) 영남대학교 기계공학부 대학원 컴퓨터 응용유체 연구실(CAF Lab.)

2) 영남대학교 기계공학부 (712-749 경북 경산시 대동 214-1 Tel:053-810-2574)

가지게 된다. Parent cell 은 Child cell 를 관리하기 위한 cell 이므로 유동인자를 가지고 있지 않다. 단지, 격자를 형성하는 과정에서만 필요하게 된다.

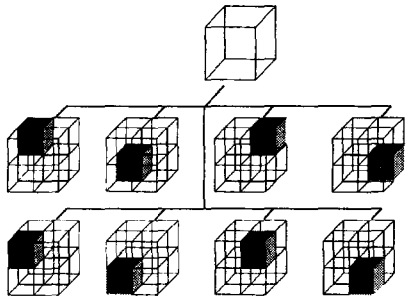


Fig1. Octree 구조

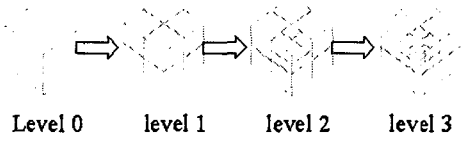


Fig2. Level 에 다른 격자형성

2.2 일반적 자료구조

각 cell 은 유동인자와 주변 cell 의 관계에 대한 정보를 가지고 있어야 한다. 이는 격자를 형성하고 수치계산에 필요하게 된다. 본 연구에서는 주변 cell 의 Level 차이가 1 이하가 되도록 하였기 때문에 각 수직 방향으로 인접하게 되는 cell 의 수는 4 개 이거나 1 개가 된다. 따라서 전체 주변 cell 의 개수는 4 개에서 24 개 사이에 있게 된다.

각 cell 은 최대 24 개의 주변 cell 에 대한 정보를 기억하고 있어야 한다.

Table 1 일반적인 구조의 cell 에 사용되는 Byte 수

자료형 (Byte 수)	변수명	메모리크기(Byte)	설 명
Double (8byte)	P,T,U,V,W	8 × 5 개 = 40	유체인자
Ccell* (4byte)	E[1],E[2],E[3],E[4], W[1],W[2],W[3],W[4], N[1],N[2],N[3],N[4], S[1],S[2],S[3],S[4], T[1],T[2],T[3],T[4], B[1],B[2],B[3],B[4]	4 × 24 개 = 96	주변 cell 에 대한 Point
전체크기(Byte)	40 + 96 = 136		

2.3 메모리를 절약하기 위한 방안

일반적인 구조에서는 각 cell 은 주변 cell 에 대한 정보를 가지고 있어야 한다. 주변 정보에 대해 상당한 메모리를 낭비하고 있다는 것을 알 수 있다. 그래서 각 Cell 의 Index 를 계산함으로써 주변 cell 를 알아낼 수 있다. Fig 3. 는 Level 에 따른 Cell 의 크기와 Index 를 나타낸다.

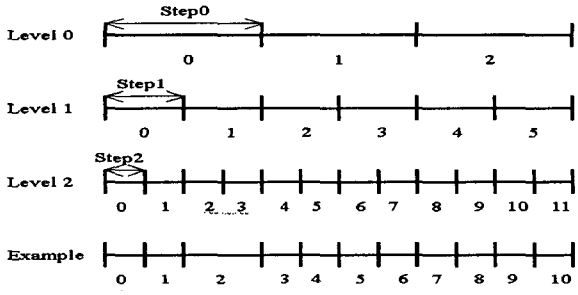


Fig 3. Level 에 따른 Index

$$Step_N = \frac{Step_0}{2^N}, \quad N = Level \quad (Eq.1)$$

$$Index_T = \mathrm{int}(\frac{Index_S}{2^N}), \quad N = Level_S - Level_T, \quad Level_S \geq Level_T \quad (Eq.2)$$

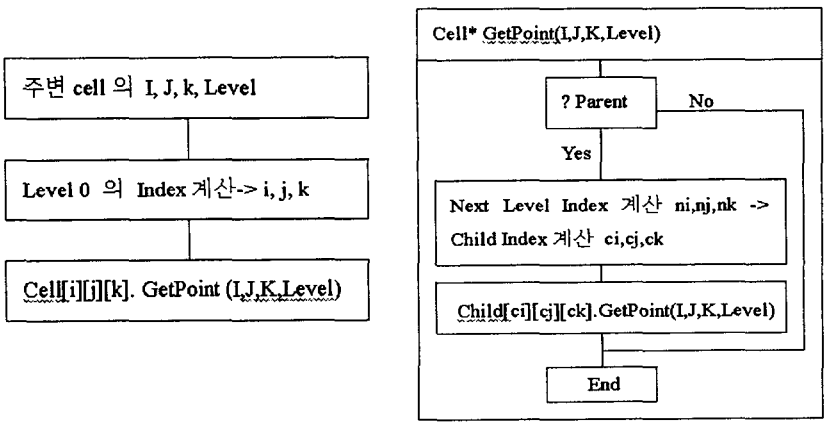


Fig 4. 주변 격자의 Index 계산 FlowChart

Eq.1 는 Level 에 따른 Cell 의 크기를 구할 수 있고 Eq.2 는 높은 Level 의 Index 에서 낮은 Level 의 Index 를 구할 수 있다. 즉 각 cell 은 Child-Parent 관계의 tree 구조를 형성할 수 있게 된다. Fig 3.의 Example 에서 3 번 cell 이 2 번 cell 를 찾기 위해서는 2 번 cell 의 Index 와 Level 를 Eq.2 에 대입하여 Level 0 인 0 번 cell 를 찾을 수 있다. Level 0 인 0 번 cell 이 Parent 이라면 다시 Level 1 의 child cell 들 중 Example 의 2 번 cell 이 속하는 cell 를 찾게 된다. 따라서 각 cell 이 자신의 Level 와 Index 를 가지고 있음으로써 주변 cell 에 대한 정보를 찾을 수 있게 된다.

Table 2 Index 에 의한 방법에서 cell 에 사용되는 Byte 수

자료형 (Byte 수)	변수명	메모리 크기 (Byte)	설 명
Double (8byte)	P,T,U,V,W	8 × 5 개 = 40	유체인자
Int (4byte)	I,J,K,Level	4 × 4 개 = 16	Cell 의 Level 및 Level 에 따른 Index
전체크기(Byte)	40 + 16 = 56		

2.4 자료구조의 비교

앞 Table 1 과 Table 2 를 비교하면 Index 에 의한 방법을 사용할 경우 메모리가 59%의 절감을 볼 수 있었다. Table 3 의 결과는 Fig 5 의 형상에 대한 격자 생성시간과 Fig 6 의 격자에 대한 수치해석 시간을 비교한 것이다. Table 3 의 결과에서 알 수 있듯이 Index 에 의한 방법이 메모리는 절감할 수 있는 반면 수치해석시간이 상당한 차이가 난다. 격자 형성에서는 Index 에 의한 방법이나 일반적인 방법이나 격자 형성시간이 비슷하다. 주변 cell 에 대한 정보를 가지고 있는 경우는 격자형성을 할 경우 각 cell 의 주변 정보가 정리되는 시간이 필요하고 Index 에 의한 방법은 주변 cell 에 대한 정보가 정리되어 가지고 있을 필요가 없으나 Index 를 계산하는 시간이 필요하게 된다. 따라서 Index 에 의한 방법이나 일반적인 방법에 의한 격자형성시간이 비슷해 진다. 그러나 메모리가 부족에 의한 가상메모리가 사용되는 시점에서는 격자형성시간이 급격하게 늘어나기 때문에 격자형성시 가상메모리가 사용되지 않게 하기 위해서는 메모리가 적게 사용되는 Index 를 이용하는 방법을 사용하는 것이 효율적일 것이다.

수치해석시간은 Index 를 이용한 방법를 사용할 경우 20 배 이상의 차이가 난다. 격자를 형성할 때는 Index 계산이 반복적으로 이루어지지 않지만 수치해석시 인접격자에 대한 참조를 반복적으로 이루어지기 때문에 계산속도에서 많은 차이가 생겼다. 이런점들을 고려할 때 격자형성에는 Index 를 이용한 방법을 사용하고 수치해석을 할 때는 일반적인 방법을 사용하는 것이 효율적임을 알 수 있다.

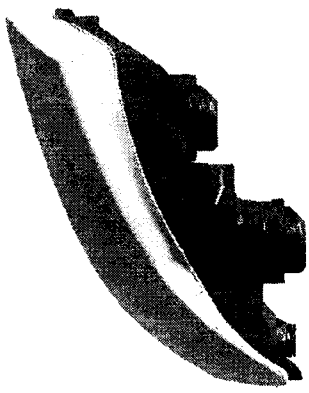


Fig 5. Head Lamp 형상

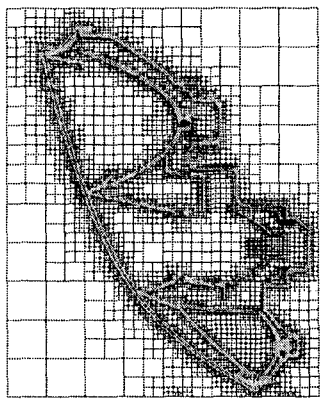


Fig 6. Head Lamp 격자

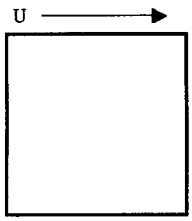
Table 3 방법에 따른 격자생성시간 및 수치해석시간 비교

방 법	일반적인 방법	Index 를 이용한 방법
격자생성시간(sec)	16	16.5
1 iteration 시간(sec)	28	1000

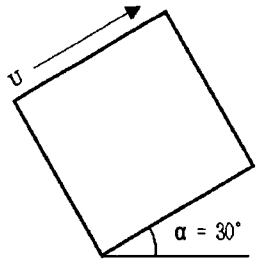
3. 수치해석 코드 검증

본 연구에서 Driven Cavity 문제에서 격자의 영향을 비교하기 위해 다음과 같이 30 도 회전 시켰을 경우와 회전 시키지 않았을 경우에 대해 각각 Level 를 다르게 하여 비교하였다.

3.1 경계 조건 및 결과

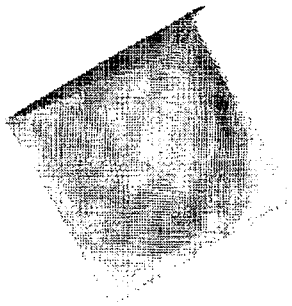


(a) driven-cavity with α = 0°

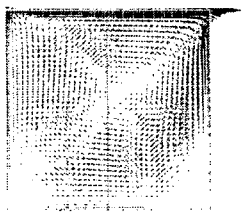


(d) driven-cavity with α = 30°

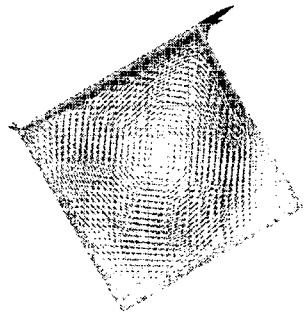
(b) at Level 0 , α = 0°



(e) at Level 0 , α = 30°



(c) at Level 3 , α = 0°



(f) at Level 3 , α = 30°

Fig 7. Driven-Cavity 문제 (Re = 1000)에 대한 코드 검증

4. 결론

정육면체의 3 차원 Octree 구조에서 Octree 구조의 특성을 사용하여 메모리를 절감할 수 있는 방법을 생각할 수 있었다. 각 cell 이 필요한 정보들 중에서 유동인자들을 제외한 인접격자들에 대한 정보를 Index 계산에 의해 구할 수 있었다. 격자형성에서는 주변격자에 대한 Index 계산이 반복해서 이루어지지 않기 때문에 메모리를 절감할 수 있는 Index 에 의한 방법이 효율적이었는 반면, 수치해석을 할 경우 주변격자에 대한 정보가 반복적으로 필요하기 때문에 일반적인 방법이 효율적임을 알 수 있었다. 수치해석의 검증은 Driven Cavity 문제를 격자형성시 격자가 불규칙하게 형성되도록 하기 위해서 기울어서 해석하였고 Level 를 높임에 따라 해석결과를 비교하였다. Level 를 높임에 따라 벽면 근처의 격자가 일직선상에 있게 되어 기울어진 효과의 영향이 작아짐을 알 수 있었다.

5. 참고문헌

[1] C.M.Rhie, W.L.Chow "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation", AIAA Journal, Vol.21, No.11(1983), pp1525-1532

[2] Majumdar "Role of underrelaxation in momentum interpolation for calculation of flow with nonstaggered grids", Numerical Heat Transfer, Vol.13(1988),pp.125-132

[3] Michael J. Laszlo "Computational Geometry and Computer Graphics in C++" (1996)

[4] Suhas V.Patankar "Numerical Heat transfer and Fluid Flow" 1980

[5] S.Chang, D.C.Haworth "Adaptive grid refinement using cell-level and global imbalances", Numerical methods in fluids, Vol.24(1997), pp375-392.

[6] R. Schreiber, H.B.Keller "Driven Cavity Flows by Efficient Numerical Techniques",Journal of Computational Physics 49(1983),pp310-333.