

## 부구조법에 의한 영역 분할 및 강소성 유한요소해석의 병렬 계산

박 근, 양 동열 (한국과학기술원 기계공학과)

### Domain Decomposition using Substructuring Method and Parallel Computation of the Rigid-Plastic Finite Element Analysis

Keun Park and Dong-Yol Yang

#### Abstract

In the present study, domain decomposition using the substructuring method is developed for the computational efficiency of the finite element analysis of metal forming processes. In order to avoid calculation of an inverse matrix during the substructuring procedure, the modified Cholesky decomposition method is implemented. As obtaining the data independence by the substructuring method, the program is easily parallelized using the Parallel Virtual Machine(PVM) library on a workstation cluster connected on networks. A numerical example for a simple upsetting is calculated and the speed-up ratio with respect to various domain decompositions and number of processors. Comparing the results, it is concluded that the improvement of performance is obtained through the proposed method.

**Key Words** : Rigid-Plastic Finite Element Analysis(강소성 유한요소해석), Substructuring Method(부구조법), Domain Decomposition(영역 분할), Parallel Computation(병렬 계산)

#### 1. 서 론

많은 계산량이 요구되는 대형 구조물의 유한요소해석의 경우 계산 시간이 많이 소요되고, 혹은 계산량이 컴퓨터의 용량을 초과하게 되어 계산 자체가 불가능하게 되는 경우도 있다. 이러한 경우 전체 영역을 몇 개의 부영역(subdomain)으로 분할하여 부영역 내의 자유도를 독립적으로 응축시킴으로써 계산량을 감소시킬 수 있는데, 이러한 기법이 부구조법(substructuring method)이다.

부구조법은 1960년대에 들어 당시의 컴퓨터 용량으로는 계산이 불가능한 대형 구조물의 해석을 위해 제안되었고,<sup>(1)</sup> 1970년대에 들어 실제 문제의 해석에 본격적으로 적용되었다.<sup>(2)</sup> 또한 부구조법의 적용에 있어서 필요한 정적응축(static condensation)을 효과적으로 수행하기 위해 Cholesky 분할의 적용에 관한 연구가 진행되었고,<sup>(3,4)</sup> 부구조법에 의해 분할된 영역에 대한 강성행렬을 재분할함으로써 계산량을 더욱더 감소시키

는 반복적(recursive) 부구조법이 제안되었다.<sup>(5,6)</sup> 한편 1980년대 중반에 들어 병렬처리 컴퓨터의 개발과 더불어 유한요소해석에서의 병렬처리(parallel processing) 기법에 관하여 연구가 활발히 진행되었는데, 이를 위한 영역 분할(domain decomposition)의 일환으로서 부구조법이 적용된바 있다.<sup>(7)</sup>

강소성 유한요소해석은 비선형 해석이므로 반복계산(iteration)을 수행해야 한다. 따라서 계산 시간이 많이 소요되게 되는데, 3차원 성형해석의 경우 엄청난 계산량으로 인해 일반 워크스테이션으로는 해석이 불가능하여 슈퍼컴퓨터를 사용하는 경우가 많아 적용의 범위가 넓지 못한 실정이다. 본 논문에서는 이러한 문제점을 해소하기 위해 강소성 유한요소해석에 부구조법을 적용함으로써 계산 시간을 감소시키고, 또 계산용량을 줄일 수 있는 방안에 대해 검토해보고자 한다. 또한 이를 바탕으로 해석 소프트웨어의 병렬화를 구축함으로써 계산효율의 극대화를 추구하고자 한다.

## 2. 부구조법에 의한 영역 분할

### 2.1 부구조법의 기본 수식화

강소성 유한요소법에서 변분식은 다음과 같다.<sup>(8)</sup>

$$\int_V \bar{\sigma} \delta \bar{\epsilon} dV + K \int_V \dot{\epsilon}_v \delta \dot{\epsilon}_v dV - \int_{S_f} f_i \delta v_i dS = 0 \quad (1)$$

여기서  $K$ 는 비압축성조건을 부과하기 위한 벌칙상수(penalty constant),  $\bar{\sigma}$ ,  $\bar{\epsilon}$ 는 각각 유효응력과 유효변형을 속도를 의미한다. (1) 식을 유한요소수식화하여 얻어지는 행렬방정식( $\mathbf{K}\mathbf{u} = \mathbf{f}$ )을 푸는 과정에 있어서 각 부영역에 대해 내부의 자유도와 경계상의 자유도에 관련된 방정식으로 구분하여 정리하면 다음과 같다.

$$\begin{bmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ib} \\ \mathbf{K}_{bi} & \mathbf{K}_{bb} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_i \\ \mathbf{f}_b \end{Bmatrix} \quad (2)$$

여기서 첨자  $i$ 는 부영역의 내부를, 첨자  $b$ 는 부영역간의 경계를 의미한다. 위 식을 전개하여 정적응축 기법에 의하여 내부의 변위를 소거하고 경계상의 변위만으로 표시하면 다음과 같이 표현될 수 있다.

$$\mathbf{K}^* \mathbf{u}_b = \mathbf{f}^* \quad (3)$$

여기서

$$\begin{aligned} \mathbf{K}^* &= \mathbf{K}_{bb} - \mathbf{K}_{bi} \mathbf{K}_{ii}^{-1} \mathbf{K}_{ib} \\ \mathbf{f}^* &= \mathbf{f}_b - \mathbf{K}_{bi} \mathbf{K}_{ii}^{-1} \mathbf{f}_i \end{aligned} \quad (4)$$

이다. 한편 이때 내부의 변위는 다음과 같이 구해진다.

$$\mathbf{u}_i = -\mathbf{K}_{ii}^{-1} (\mathbf{K}_{ib} \mathbf{u}_b - \mathbf{f}_i) \quad (5)$$

즉 (3)식을 사용하여 경계상의 변위  $\mathbf{u}_b$ 를 구한 후 이를 (5)식에 대입하여 내부절점의 변위  $\mathbf{u}_i$  또한 구할 수 있다. 여기서 강성행렬  $\mathbf{K}$ 의 크기가 전체 영역의 자유도에 해당하는 반면  $\mathbf{K}^*$ 의 크기는 부구조간의 경계상의 자유도만큼으로 줄어들었음을 알 수 있다. 그러나 (4), (5) 식에서 볼 수 있듯이  $\mathbf{K}_{ii}^{-1}$ 의 계산과정이 필요하게 되는데, 이는  $\mathbf{K}_{ii}$ 의 크기가 클 때 많은 계산량이 필요하게

되어 계산의 효율이 저하된다. 따라서 본 논문에서는 역행렬의 계산과정을 피해주기 위해 다음 식과 같이 수정된 Cholesky 분할방법을 사용하였다.<sup>(3)</sup>

$$\begin{bmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ib} \\ \mathbf{K}_{bi} & \mathbf{K}_{bb} \end{bmatrix} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{R} & \mathbf{K}^* \end{bmatrix} \begin{bmatrix} \mathbf{L}^T & \mathbf{R}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (6)$$

$\mathbf{K}_{ii}$ 와  $\mathbf{K}_{bi}$ 는 다음과 같이 나타내어질 수 있다. 여기서  $\mathbf{L}$ 은 하부삼각행렬을 의미한다.

$$\begin{aligned} \mathbf{K}_{ii} &= \mathbf{L}\mathbf{L}^T \\ \mathbf{K}_{bi} &= \mathbf{R}\mathbf{L}^T \end{aligned} \quad (7)$$

이때 (4), (5) 식은 다음과 같이 표현될 수 있어  $\mathbf{K}_{ii}$ 의 역행렬을 구할 필요가 없어진다. 대신 상대적으로 계산시간이 조금 걸리는  $\mathbf{L}$ 의 역행렬을 계산하므로 계산시간의 절감이 가능해진다.

$$\begin{aligned} \mathbf{K}^* &= \mathbf{K}_{bb} - \mathbf{R}\mathbf{R}^T \\ \mathbf{f}^* &= \mathbf{f}_b - \mathbf{R}\mathbf{L}^{-1}\mathbf{f} \\ \mathbf{u}_i &= \mathbf{L}^{-T}(\mathbf{L}^{-1}\mathbf{f}_i - \mathbf{R}^T\mathbf{u}_b) \end{aligned} \quad (8)$$

### 2.2 부구조법을 사용한 해석 및 고찰

앞절에서 소개한 부구조법의 기본 수식화를 강소성 유한요소해석 프로그램에 적용하였고 이를 검증하기 위해 축대칭 업셋팅 공정을 해석하였다. 이때 영역 분할에 따른 계산시간의 변화를 고찰하기 위해 부영역의 개수를 변화시켜가며 입력데이터를 준비하였고, 각각의 경우에 대해 해석을 수행하여 결과를 비교하였다. Fig. 1에 각 경우에 대한 격자구조를 나타내었는데, 실선으로 표시한 부분이 부영역간의 경계를 의미한다. 여기서 실선상의 절점이 경계절점에 해당하고, 점선상의 절점이 내부절점에 해당한다.

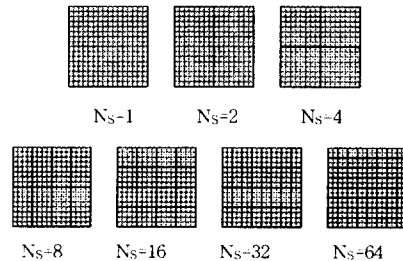


Fig. 1 Mesh structure of each case

각 격자구조에 대해 해석에 소요되는 시간은 부영역의 개수에 따라 많은 변화를 보였는데, Table 1에 이를 비교하였다. 여기서  $N_s$ 는 부영역의 개수,  $N_i$ 와  $N_b$ 는 각각 내부절점과 경계절점에 해당하는 경계절점의 자유도의 개수를 의미한다. 이 중  $N_s$ 가 1일 때의 계산시간을 기준으로 해석시간의 비(比)를 사용하여 속도향상율(speed-up ratio)을 정의함으로써, 다른 경우에 대해 얼마만큼의 계산시간이 감소되었는지를 비교해보았다. Table 1을 보면  $N_s$ 가 16일 때 속도향상율이 최대임을 알 수 있다. 이는 부영역의 수가 증가할수록 내부절점의 자유도가 감소하여 정적응축 과정에서의 계산시간이 감소하는 반면, 상대적으로 경계절점의 자유도가 증가함으로써 응축된 방정식을 푸는 과정에서의 계산시간이 증가하기 때문이다. 따라서 계산시간을 최소화할 수 있도록 부영역의 개수를 적절히 선정해야 하는데, 본 예제의 결과로 미루어볼 때 내부절점의 자유도와 경계절점의 자유도의 개수가 비슷한 수준일 때 속도향상율이 가장 높음을 알 수 있다.

$N_s$	$N_i$	$N_b$	Time(sec)	Speed-up
1	450	128	1836.1	1.00
2	420	158	590.1	3.27
4	392	186	308.5	5.95
8	336	242	237.0	7.75
16	288	290	193.5	9.49
32	192	386	263.3	6.97
64	128	450	325.8	5.64

Table 1 Elapsed time w.r.t. no. of subdomains

### 3. 유한요소해석의 병렬 계산

#### 3.1 PVM을 사용한 병렬 해석 프로그램 개발

유한요소법은 구조적으로 요소강성행렬(elementary stiffness matrix)의 생성, 강성행렬의 조합, 평형방정식의 계산 등 상대적으로 계산시간이 많이 걸리고 정보의 종속성(dependency)이 적은 부분에서 병렬처리 기능을 활용하기가 유리하다. 유한요소해석에서의 병렬계산은 사용 컴퓨터의 종류와 병렬처리 기법에 따라 많은 연구 결과가 발표되었는데 주로 구조 및 충돌해석, 유동해석 등 많은 계산시간이 소요되는 문제에 대

해 적용되었으며,<sup>(9)</sup> 최근에는 소성가공공정 유한요소해석에 병렬처리를 적용한 사례도 발표되었다.<sup>(7)</sup>

본 연구에서는 2차원 강소성 유한요소해석 프로그램에 부구조법을 적용하여 영역 분할 및 각 영역의 강성행렬 계산, 각 내부 자유도의 정적 응축 과정 등 영역에 따른 정보의 독립성을 확보하고, 이를 병렬적으로 계산할 수 있도록 프로그램을 병렬화시켰다. CPU간의 정보교환(message passing)을 위해 PVM(Parallel Virtual Machine)을 사용하였다.

PVM은 서로 다른 컴퓨터들을 네트워크로 연결하여 하나의 병렬 컴퓨터처럼 동작하게 해주는 정보교환(message passing) 라이브러리이다. PVM은 호환성이 좋고 별도의 병렬컴퓨터 없이도 기존의 워크스테이션을 활용하여 병렬처리 환경을 구축할 수 있다는 장점이 있는 반면, 네트워크를 사용하여 정보를 전달하므로 네트워크 상태에 따라 정보전달 효율이 저하될 가능성이 있다.

#### 3.2 소성가공 공정의 병렬 계산

기존에 순차적으로 진행되던 유한요소해석 과정의 병렬 계산을 위해서는 2단계의 수정작업이 필요하다. 첫 번째는 데이터의 병렬화, 즉 데이터 상호간의 독립성을 확보하는 과정인데, 이는 부구조법에 의한 영역 분할을 도입하여 각 부영역내의 계산을 독립적으로 계산할 수 있도록 하였다. 두 번째로는 독립성이 확보된 계산과정을 병렬적으로 계산할 수 있는 알고리즘을 구현하는 과정으로 이는 PVM에서 제공하는 각종 라이브러리 함수를 사용하여 구현하였다. 계산에 사용된 장비는 Silicon Graphics사의 INDIGO 기종 4대를 사용하였다.

본 연구에서 4대의 컴퓨터를 사용하여 PVM 환경을 구축하였으므로, Fig. 1에서 도시된 7개의 격자구조중 부영역의 개수를 1개, 2개, 4개로 나누었을 경우에 대해서 해석을 수행해보았다. 또한 각각의 경우 1개의 컴퓨터만을 사용하여 해석하였을 경우, 즉 병렬처리를 하지 않았을 경우와 부영역의 개수만큼 프로세서의 수를 늘려 사용하였을 경우에 대해 계산을 수행하여 소요 시간을 비교해보았다(Table 2). 여기서 NS는 부영

역의 개수를, NP는 프로세서의 수를 의미한다. 계산시간은 순수한 계산을 수행하는데 소요된 시간(analysis time)과 컴퓨터간에 정보를 주고받는데 소요되는 시간(message passing time)으로 구분하여 비교하였다. 이때 속도향상율은 NS1NP1의 계산시간을 기준으로 하여 다른 경우의 계산시간을 비(比)로서 나타내었다.

Elapsed time(sec)	NS1 NP1	NS2 NP1	NS2 NP2	NS4 NP1	NS4 NP4
Analysis	1172.5	379.8	182.3	217.9	55.8
Message passing	0.0	0.0	104.7	0.0	60.1
Total	1172.5	379.8	287.0	217.9	115.9
Speed-up	1.00	3.09	4.09	5.38	10.12

Table 2 Elapsed time for each case

이중 NS4NP4가 NS1NP1의 경우보다 무려 10배의 속도향상을 보여 가장 효율적인 계산이 이루어졌음을 알 수 있다. 이는 우선적으로 부구조법에 의한 영역분할에 따른 계산시간 절감의 효과가 있는데다가 병렬처리를 구현함으로써 추가적인 성능 향상이 이루어졌기 때문이다.

#### 4. 결 론

이상으로 본 연구에서는 소성가공 공정의 유한요소해석시 계산시간을 효과적으로 절감하기 위해 부구조법을 사용한 영역분할 기법을 적용하였다. 이때 정적응축 과정에서 발생하는 역행렬의 계산과정에서 수정된 Cholesky 분할법을 도입함으로써 계산의 효율을 증대시켰다. 또한 영역을 분할함으로써 각 영역간의 정보의 독립성을 확보하였고, 해석 프로그램에서 해당 영역의 독립적인 계산과정을 PVM 라이브러리를 사용하여 병렬화시켰다. 개발된 프로그램을 사용하여 실제 예제의 해석을 통해 영역의 분할에 따른 계산시간 절감 및 병렬 계산에 의한 성능 향상 효과를 확인하였다. 현재 국내에도 병렬컴퓨터가 점차로 널리 보급되는 점을 감안할 때, 차후에는 보다 개선된 병렬처리 환경에서 계산의 효율을 제고할 수 있을 것으로 전망된다.

#### 참고문헌

- (1) Przemieniecki, J. S., 1963, "Matrix Structural Analysis of Substructures", AIAA Journal, Vol. 1, pp. 138-147
- (2) Noor, A. K., Kamel, H. A. and Fulton, R. E., 1977, "Substructuring Techniques - Status and Projections", Computers & Structures, Vol. 8, pp. 621-632
- (3) Han, T. Y. and Abel, J. F., 1984, "Substructure Condensation Using Modified Decomposition", International Journal for Numerical Methods in Engineering, Vol. 20, pp. 1959-1964
- (4) Liu, X. L. and Lam, Y. C., 1995, "Condensation Algorithms for the Regular Mesh Substructuring", International Journal for Numerical Methods in Engineering, Vol. 38, pp. 469-488
- (5) Jonsson, J., Krenk, S. and Damkilde, L., 1995, "Recursive Substructuring of Finite Elements", Computers & Structures, Vol. 54, pp. 395-404
- (6) Farhat, C., Crivelli, L. and Rouz, F. X., 1994, "Extending Substructure Based Iterative Solvers to Multiple Load and Repeated Analyses", Computer Methods in Applied Mechanics and Engineering, Vol. 117, pp. 195-209
- (7) Doltsinis, I. St. and Nölting, S., 1991, "Generation and Decomposition of Finite Element Models for Parallel Computations", Computing Systems in Engineering, Vol. 2, pp. 427-449
- (8) Lee, C. H. and Kobayashi, S., 1973, "New Solution to Rigid Plastic Deformation using a Matrix Method", ASME, Journal of Engineering for Industry, Vol. 95, pp.865-873
- (9) Farhat, C. and Roux, F. X., 1994, "Implicit Parallel Processing in Structural Mechanics", Computational Mechanics Advances, Vol. 2, pp. 1-124