한국전산구조공학회
2004년도 가을 학술발표회 논문집

전산구조해석을 위한 고성능 직접적 연립방정식 해법의 개발 및 비교 연구

Development and comparative study of high-performance direct solvers for computational structural mechanics

우 성 운* 김 정 호**
Woo, Sung-Woon Kim, Jeong-Ho

ABSTRACT

In the structural analysis procedure using finite element technique, the performance of a linear equation solver is critical because the linear equation solving part spends most of the computing time for finite element analysis codes. However, most of researchers are still using inefficient profile-based direct solvers such as the band solver or the skyline solver. In this research, we introduce the multifrontal solution method as an efficient direct solution method for structural analysis, and show the efficiency and performance of the multifrontal solution method by comparing the performance of our own implementation of the multifrontal method with the band solver or the skyline solver. In addition, we also compare the performance of our solver with other implementations of the multifrontal method such as WSMP and MUMPS as well as commercial structural analysis packages such as ABAQUS and NASTRAN. Through the performance test results, the usefulness and efficiency of our domain-wise multifrontal solver for structural analysis is shown.

1. 서 론

유한요소법을 이용한 전산구조해석 과정에 있어서 가장 많은 시간을 차지하는 부분은 연립방정식을 푸는 부분으로서 문제의 규모가 커지게 되면 대부분의 계산 시간이 이 부분에서 소요된다. 컴퓨터의 발달로 인한 컴퓨터 시뮬레이션 기술의 발달로 구조해석을 위한 모델링도 점점 상세해 지고 이에 따라 구조해석을 위한 계산의 규모도 점점 커지게 됨에 따라, 효율적인 구조해석을 위해서는 효율적인 선형 연립방정식 해법이 필수적이다. 유한요소해석 과정에서 나타나는 행렬은 일반적으로 매우 0을 많이 가지고 있는 산재행렬(sparse matrix)이므로 효율적인 선형 연립방정식 해석을 위해서는 이러한 특성을 잘 살려서 계산량을 최소로 줄이고 또한 하드웨어 구조에 최적화되어 최대의 성능을 낼 수 있어야 한다. 산재행렬을 위한 연립방정식 해법은 크게 반복적인 방법(iterative method) 와 직접적 방법(direct method)으로 나눌 수 있다.

* 한국과학기술정보연구원 위촉연구원
** 한국과학기술정보연구원 선임연구원

반복적 방법은 사실상 구조해석 문제의 해결에는 잘 사용되지 않아왔으나, 근래에 병렬형 컴퓨터의 발달과 함께 병렬화에서의 이점으로 인하여 많은 주목을 받고 있다. 따라서 반복적 방법들은 병렬화를 전제로 한 영역분할기법을 기반으로 한 접근이 많이 시도되어 왔다. 이러한 기법으로는 C. Farhat 등이 제안한 FETI (Finite Element Tearing and Interconnecting) 기법이 가장 성공적인 것으로 평가되고 있다. 그러나 근본적으로 영역분할기법에 기반으로 한 반복적 방법 들은 수치적인 안정성과 이로 인한 범용성에 문제가 있으며 특히 구조해석 분야의 문제는 안정적으로 다루기 힘들다. 또한, 계산량의 예측이 불가능할 뿐만 아니라 구조해석 문제에서 흔히 나타나는 다중 우변항(Multiple Right-Hand Side) 문제를 효과적으로 처리할 수 없다. 이런 이유로 인해 ABAQUS, NASTRAN, ANSYS 등 범용 구조해석을 위한 대부분의 상용 유한요소해석 프로그램들은 대부분 가우스 소거법을 기반으로 해서 선형 연립방정식을 푸는 직접적 방법(direct method)을 주로 사용하고 있다.

직접적 방법에서는 많은 메모리 요구량과 계산 시간이 주요한 문제점이며 이것을 해결하기 위해 행렬의 구조적 특성을 이용한다. 그 중 가장 간단한 형태가 구조해석 시의 선형 연립방정식의 계수행렬(강성행렬)을 대각원소를 중심으로 띠(band) 모양으로 0이 아닌 항이 존재하는 형태로 가정하고 푸는 밴드 해법 또는 스카이라인 해법이다. 그러나 실제 구조해석 문제를 푸는 과정에서 나타나는 일반적인 산재행렬(sparse matrix)의 경우 이런 띠 내부에도 많은 0인 원소를 포함하므로 처음에는 0이었던 부분이 계산 과정(factorization) 중에 0이 아닌 값으로 채워지는 'fill-in' 또한 많이 발생하게 된다. 이러한 'fill-in'을 줄이는 것이 직접적 방법에 있어서 계산량 및 기억용량을 줄이기 위해 가장 중요한 문제인데 이러한 관점에서 볼 때 이러한 직접적 방법들보다 더욱 'fill-in'을 줄일 수 있는 여지가 충분히 있다고 볼 수 있다.

본 연구에서 소개하고자 하는 다중프론트 해법은 이와 같이 구조해석에서 나타나는 산재행렬의 구조적 특성을 잘 이용하여 'fill-in'을 최소화할 수 있도록 개발된 직접적 방법으로서 구조해석에 적용 시 매우 우수한 성능 향상 효과를 볼 수 있다. 그러나 이러한 다중프론트 해법은 소수의 상용 구조해석 소프트웨어 외에는 구조해석 분야에서 널리 활용되고 있지 못하며 기존의 비효율적인 스카이라인 해법 정도가 주로 많이 활용되고 있는 실정이다. 따라서 본 연구에서는 다중프론트 해법을 소개하고 이를 구조해석 문제에 적용 시의 효율성을 보이고, 특히 자체 개발한 다중프론트 해법인 영역기반 다중프론트 해법의 우수성 및 유용성을 보이고자 한다.

2. 다중프론트 해법

2.1. 프론트 해법

프론트 해법(Frontal solution method)는 제한된 주기억 용량으로 유한요소해석 문제의 해를 효율적으로 구하기 위해 Irons[1]에 의해 처음으로 소개되었다. Irons가 제안한 프론트 해법은 기존의 해법(밴드 및 스카이라인 해법 등)들이 전역강성행렬을 완전히 조립한 후에 가우스 소거법 등을 이용하여 연립방정식을 푸는 것과는 달리 전역강성행렬을 조립하지 않고 이웃하는 요소들에 대해서 요소강성행렬을 하나씩 조립해나가는 과정에서 조립이 완료된 부분을 부구조화(substructuring) 과정을 통하여 그 즉시 소거해 나가는 방법이다. 부구조화 과정은 요소강성행렬을 조립해 나가는 과정에서 식 (1)과 같이 연립방정식이 구성되었다면, 여기서 조립이 완료된 자유도 $\mathbf{u}_1$을 소거하고 조립이 완료되지 않은 자유도 $\mathbf{u}_2$만으로 구성된 식 (2)와 같은 새로운 연립방정식을 구성하는 과정이다.

$$\begin{bmatrix} \mathbb{K}_{11} & \mathbb{K}_{12} \\ \mathbb{K}_{21} & \mathbb{K}_{22} \end{bmatrix} \begin{Bmatrix} \mathbb{u}_1 \\ \mathbb{u}_2 \end{Bmatrix} = \begin{Bmatrix} \mathbb{f}_1 \\ \mathbb{f}_2 \end{Bmatrix}, \text{ where } \mathbf{u}_1 \text{ has fully assembled DOFs} \quad (1)$$

$$\begin{aligned} \overline{\mathbb{K}}_{22}\mathbb{u}_2 &= (\mathbb{K}_{22} - \mathbb{K}_{21}\mathbb{K}_{11}^{-1}\mathbb{K}_{12})\mathbb{u}_2 \\ &= \mathbb{f}_2 - \mathbb{K}_{21}\mathbb{K}_{11}^{-1}\mathbb{f}_1 = \overline{\mathbb{f}}_2 \end{aligned} \quad (2)$$

이 과정에서 소거된 자유도 관련 데이터는 보조기억장치로 옮길 수 있기 때문에 계산에 요구되는 주기억장치의 용량을 크게 줄일 수 있다. 그런데 프론트 해법은 적은 주기억장치를 사용하기 때문에 보다 큰 문제를 해석할 수 있다는 장점이 있기는 하지만 계산량은 기존의 스카이라인 해법 등에 비하여 거의 차이가 없기 때문에 계산 시간에 대한 성능 향상은 기대하기 힘들다.

2.2. 다중프론트 해법

다중프론트 해법은 유한요소해석을 위해 고안된 프론트 해법의 일반화 과정에서 그 개념의 확장을 통해 Duff 등[2]에 의해 제안되었으며 Liu[3]에 개념이 정립되었다. 다중프론트 해법은 산재행렬로 구성된 연립방정식을 풀기 위한 범용 연립방정식 해법으로써 이 알고리즘을 기반으로 하는 여러 가지 소프트웨어가 개발되었으나[4] 현재 가장 많이 사용되고 효율성이 우수한 것으로 널리 알려진 소프트웨어로는 MUMPS[5] 와 WSMP[6]를 들 수 있다.

다중프론트 해법은 계수 행렬(유한요소해석에서 전역강성행렬)의 산재패턴(sparse pattern)을 이용한 방법이다. 그림 1과 같은 행렬이 있다고 하자.

	1	2	3	4	5	6	7	8	9
1	x		x				x	x	
2		x	x					x	x
3	x	x	x				x	x	x
4				x		x	x	x	
5					x	x		x	x
6				x	x	x	x	x	x
7	x		x	x		x	x	x	
8	x	x	x	x	x	x	x	x	x
9		x	x		x	x		x	x

그림 1 Example matrix

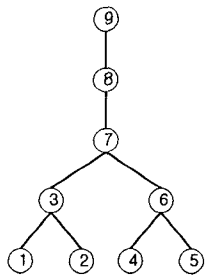

그림 2 Elimination tree

행 ④ ⑤ ⑥을 factorization 할 때 행 ① ② ③ 은 필요하지가 않으며 그 역으로 행 ① ② ③ 을 factorization 할 때 행 ④ ⑤ ⑥은 필요하지가 않다. 즉 행 ① ② ③ 과 행 ④ ⑤ ⑥ 은 factorization 시에 서로 독립적인 연산이 가능하며 이 과정에서 'fill-in'을 크게 줄일 수 있게 되며 이런 개념을 그림2와 같이 트리 구조 형태로 표현할 수 있다.[7] 이렇게 서로 독립적인 일반화된 프론트(generalized front)가 여러 개 형성되므로 다중프론트 해법이라 한다.

2.3. MUMPS

MUMPS는 P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, J. Koster 에 의해 개발되었으며 다중프론트 해법을 이용한 범용 산재행렬 연립방정식 해석용 소프트웨어 패키지이다. Fortran90으로 쓰여졌으나 C 에서도

호출이 가능하며 순차(serial) 버전과 MPI를 이용한 병렬 버전을 사용할 수 있다.[8] 풀 수 있는 연립방정식의 행렬 종류로는 대칭양정치(symmetric positive definite) 행렬, 대칭(symmetric) 및 비대칭(unsymmetric) 행렬이 있으며 행렬의 수치적 특성에 따라 LU, LDL^T 알고리즘을 이용할 수 있다. Fill in을 최소화하기 위해 사용하는 행렬재배열(matrix ordering) 기법은 AMD(Approximate Minimum Degree), AMF(Approximate Minimum Fill), QAMD(Approximate Minimum Degree with automatic quasi-dense row detection) 알고리즘을 이용한 내장형 라이브러리와 PORD, METIS[9] 등과 같은 외부 라이브러리를 이용할 수 있다.

2.4. WSMP

WSMP 는 A. Gupta 에 의해 개발되었으며 MUMPS와 마찬가지로 다중프론트 해법을 이용한 범용 산재행렬 연립방정식 해석용 소프트웨어 패키지이다. Fortran으로 만들어 졌으나 C 에서도 호출이 가능하며 현재 순차 버전 및 공유메모리 병렬화, 분산메모리 병렬화 버전을 이용할 수 있다.[10,11] 풀 수 있는 연립방정식의 행렬 종류로는 역시 MUMPS와 마찬가지로 대칭양정치(symmetric positive definite) 행렬, 대칭(symmetric) 및 비대칭(unsymmetric) 행렬이 있으며 행렬의 수치적 특성에 따라 LU, LDL^T 및 Cholesky 알고리즘을 이용할 수 있다. Fill-in을 최소화하기 위한 행렬재배열(matrix ordering) 기법은 A. Gupta가 개발한 nested dissection 알고리즘을 기반으로 하는 WGPP[12]란 루틴을 사용하여 구현하고 있다. WSMP는 현재 개발된 다중프론트 해법 중 전반적으로 가장 우수한 성능을 보이는 것으로 알려져 있다.

2.5. 영역기반 다중프런트 해법(Domain-wise Multifrontal method)

본 연구에서는 기존의 다중프론트 해법을 유한요소해석 과정에 보다 효율적으로 적용시키기 위해 모든 계산 과정을 요소 또는 영역 단위로 수행하도록 한 영역기반 다중프론트 해법을 제안하였다.[13] 이 방법은 유한요소해석 과정에서 필요한 기억용량 및 연산회수를 최소로 줄여줄 뿐만 아니라 모든 연산이 요소 및 영역기반으로 진행되므로 병렬화에 있어서도 매우 큰 이점을 가지고 있다.

영역기반 다중프런트 해법은 그림 3과 같이 문제의 영역을 반복적으로 반으로 분할한 다음, 분할한 역순으로 이웃하는 영역을 둘씩 합쳐나가면서 합쳐진 영역 내부의 자유도를 소거해 나가는 것으로 이 과정은 그림 4와 같이 재귀적 부구조화(Recursive substructuring) 과정으로 설명할 수도 있다.

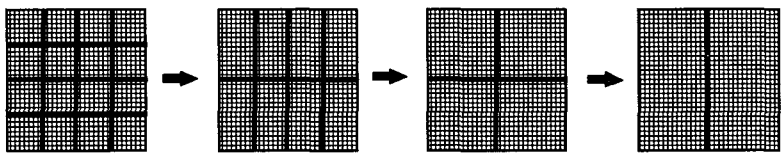

그림 3 Elimination(factorization and forward substitution) and back substitution

직접적 연립방정식 해법을 이용한 유한요소해석 문제에 있어서 또 하나 중요한 문제가 되는 것이 기억용량의 문제이다. 여기서 개발된 영역기반 다중프론트 해법은 소거가 된 자유도에 해당하는 행렬을 고전적인 프론트 해법과 마찬가지로 디스크에 임시로 저장(out-of-core)할 수 있도록 함으로써 성능에는 큰 영향을 미치지 않고 계산에 요구되는 주기억장치의 용량을 크게 줄일 수 있다. 뿐만 아니라 전영강성행렬을 조립하지 않고 메모리를 최대한 효율적으로 활용하도록 프로그래밍되어 있어 대형 문제를 매우 효율적으로 적은 메모리만으로 풀 수 있다.

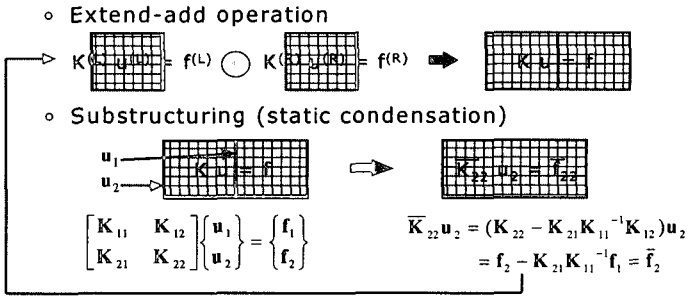

그림 4 Recursive substructuring

3. 성능 시험 및 분석

3.1. 성능 시험 모델

영역기반 다중프론트 해법의 성능 시험을 위해 표 1 및 그림 5와 같은 다양한 유한요소 모델을 이용하였다. 여기서 N은 시스템 전체의 자유도를 의미하여 NNZ (number of none zeros)는 행렬의 상대각 혹은 하대각 부분의 0이 아닌 원소의 총 개수를 의미한다. 여기서 S48는 S32과 형태는 같으나 전체 자유도의 수가 다른 유한요소 모델이다.

Name	N	NNZ	Name	N	NNZ
Dynax	6.34e4	2.21e6	Huge	3.01e4	1.67e7
Comph8	8.40e4	30.61e5	S32	10.78e4	41.61e5
Hole	55.87e4	42.56e6	Rocket	12.01e5	37.55e6
Gear	74.49e4	27.89e6	S48	10.59e5	13.89e6

표 1 성능 시험 모델 정보

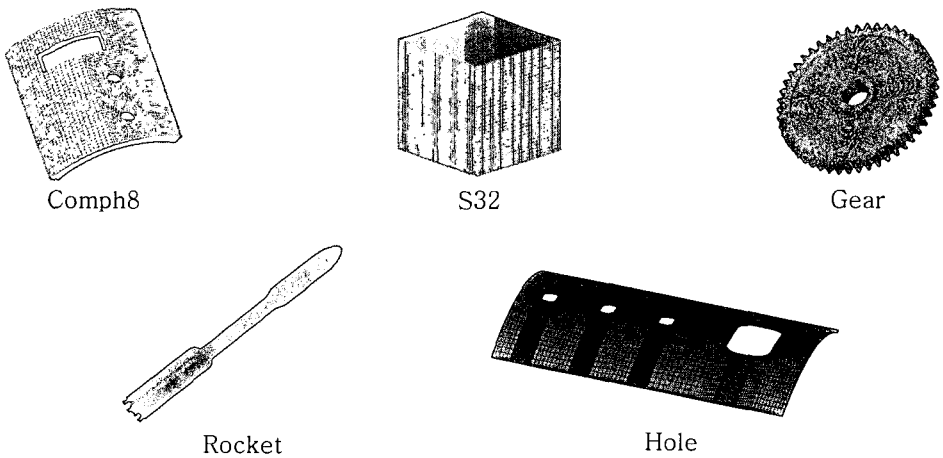

그림 5. 성능 시험을 위한 유한요소 모델

3.2. 상용 구조해석 소프트웨어와의 성능 비교

다중프론트 해법의 알고리즘의 우수성을 확인하고자 대표적인 상용 구조해석 프로그램인 NASTRAN 및 ABAQUS와 다양한 시스템 상에서 성능을 비교하였다. 32×32×32개의 3차원 고체요소로 구성된 규칙적인 유한요소 모델을 이용해 성능 비교를 수행하였다. 그 결과 그림 6에서 보는 바와 같이 다중프론트 해법을 쓰지 않는 NASTRAN에 비해서는 영역기반 다중프론트 해법(D-MFS)이 월등히 우수한 성능을 보이고 있고, 다중프론트 해법을 쓰는 ABAQUS에 비해서도 우수한 성능을 보이며 특히 Intel Xeon 시스템에서는 2배 이상의 우수한 성능을 보이고 있다.

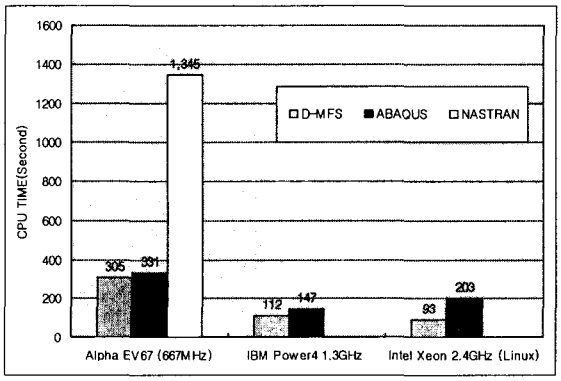

그림 6 상용 구조해석 소프트웨어와의 성능 비교

3.3. 밴드 및 스카이라인 해법과의 비교 시험

다중프론트 해법의 성능을 확인하기 위해 밴드 해법 및 스카이라인 해법과 성능을 비교하였다. 먼저 앞에서와 같은 32×32×32개의 3차원 고체요소로 구성된 규칙적인 유한요소 모델을 이용해 IBM POWER4 1.7GHz 시스템 상에서 시스템에 최적화된 밴드 해법과 개발된 영역기반 다중프론트 해법의 성능을 비교하였다. 그 결과 밴드 해법을 이용한 경우 321.13초가 걸렸으나 영역기반 다중프론트 해법은 58.45초가 소요되어 약 5.5배의 우수한 성능을 보였다. 만약 시스템에 최적화되지 않은 일반적인 밴드 해법을 사용한다면 성능차는 이보다 훨씬 더 크게 날 수가 있다.

다음으로 역시 시스템에 최적화된 스카이라인 해법을 이용해서 불규칙한 연결 구조를 가지는 유한요소 모델에 대한 성능 비교를 수행했다. 스카이라인 해법의 연산량 최적화를 위한 행렬재배열 기법으로는 Reverse Cuthilll Mckee 알고리즘을 사용하였으며 표 2의 시간은 행렬재배열에 걸린 시간을 제외한 것이다.

	Dynax	Huge	Comph8	Hole
D-MFS	5.9 sec	19.5 sec	11.1 sec	202.7 sec
Skyline	40.7 sec	46.1 sec	101.2 sec	2283 sec

표 2 스카이라인 해법과의 성능 비교

모델에 따라 차이가 있으나 행렬재배열 시간을 제외하고도 스카이라인 해법이 영역기반 다중프론트 해법에 비하여 2.3~11.3배의 계산시간이 소요됨을 확인할 수 있다.

3.4. 다중프론트 해법 성능 비교

앞에서 다중프론트 해법이 다른 연립방정식 해법에 비하여 월등히 우수한 것을 확인하였다. 여기서는 본 연구에서 개발한 영역기반 다중프론트 해법(D-MFS)과 앞에서 언급한 MUMPS 및 WSMP 등 다른 다중프론트 해법과의 성능을 역시 IBM POWER4 1.7GHz 시스템 상에서 비교하였다. 위의 6가지 모델에 대해 각각 성능 시험을 한 후 표 3과 같은 결과를 확인할 수 있었다.

Name	계산시간 (sec)			메모리 (Mbytes)		
	D-MFS	MUMPS	WSMP	D-MFS	MUMPS	WSMP
Comph8	12.4	16.9	14.5	345	536	389
S32	58.6	80.0	70.4	941	1484	1088
Hole	214.8	274.5	275.0	3591	5537	4812
Gear	377.7	436.9	453.6	4703	6869	5990
S48	614.2	833.9	856.0	4876	7463	6516
Rocket	270.4	288.1	272.3	4231	6108	5034

표 3 계산시간과 메모리 사용량

그림 7 은 표 3의 3가지 연립방정식 해법을 이용한 계산시간을 나타낸 것이며 그림 8은 영역기반 다중프론트 해법의 계산시간을 기준으로 WSMP, MUMPS의 계산시간의 상대적 비율을 표시한 것이다.

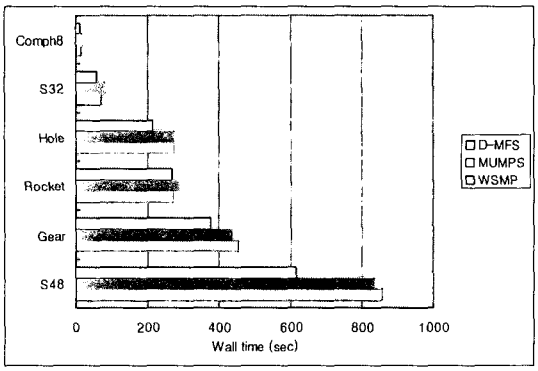

그림 7 계산 시간

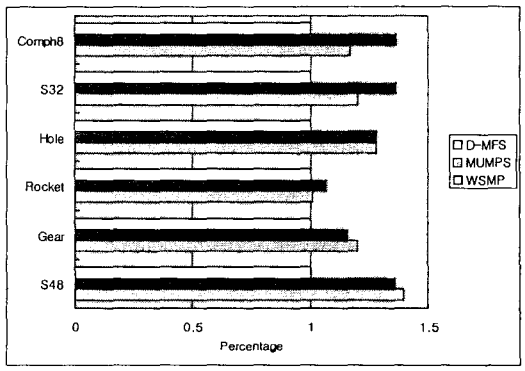

그림 8 계산 시간 비율

위의 성능 시험 결과로부터 본 연구에서 개발한 영역기반 다중프론트 해법이 가장 우수한 성능을 보이는 것을 알 수 있다. 뿐만 아니라 표 3으로부터 영역기반 다중프론트 해법(D-MFS)이 MUMPS나 WSMP에 비하여 차지하는 메모리 용량도 적은 것을 알 수 있다. 그런데 이러한 메모리 용량은 영역기반 다중프론트 해법에서 보조기억장치를 전혀 활용하지 않을 경우이므로 보조기억장치까지 활용하는 경우 제한된 주 기억용량 내에서 MUMPS나 WSMP에 비하여 훨씬 더 큰 규모의 문제를 풀 수도 있다.

4. 결 론

본 연구에서 다중프론트 해법의 개념에 대해 소개하였으며 구조해석에 주로 활용되던 기존의 해법, 즉 밴드 해법이나 스카이라인 해법과의 성능비교를 통해 구조해석 문제에 적용 시 다중프론트 해법의 우수성을 보였다. 또한 자체 개발한 다중프론트 해법인 영역기반 다중프론트 해법(Domain-wise MultiFrontal method)을 소개하고 기존의 다중프론트 해법 및 고성능 상용 구조해석 소프트웨어와의 비교를 통해 그 우수성을 입증하였다. 본 연구에서 소개한 다중프론트 해법을 적극적으로 활용한다면 구조해석 소프트웨어의 성능을 크게 향상시킬 수 있고 이를 통한 연구 효율의 향상과 연구 능력의 확대에 기여할 수 있을 것으로 기대된다.

참고 문헌

1. B. M. Irons. A frontal solution program for finite-element analysis. *Int J. Numerical Methods in Engineering*, 2:5-32, 1970.
2. S. Duff and J. K. Reid, The multifrontal solution of indefinite sparse symmetric linear equations, *ACM Trans. Math Software*, 9, 302-325 (1983)
3. J. W.-H. Liu, The multifrontal method for sparse matrix solution: Theory and practice, *SIAM Review, 23:82-109*, 1992
4. http://www.netlib.org/utk/people/JackDongarra/la-sw.html
5. P.R. Amesoy, I.S. Duff, J.-Y. L'Excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods Appl. Mech. Eng*, 2000.
6. A. Gupta, G. Karypis and V. Kumar, Highly Scalable Parallel Algorithms for Sparse Matrix Factorization, *IEEE Transactions on Parallel and Distributed systems*, 1995
7. V. Kumar, A. Grama, A, Gupta, G, Karypis. Introduction to parallel computing Design and analysis of algorithms, *The Benjamin/Cummings Publishing Company, Inc.* 1994.
8. P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, J.Koster. MUltifrontal Massively Parallel Solver Users' guide(MUMPS Version 4.3), 2003
9. G. Karypis, V. Kumar, METIS, A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, Version4.0, 1998
10. A. Gupta, WSMP: Watson Sparse Matrix Package. Part I-direct solutions of symmetric sparse systems, *IBM Research Report*, 2000
11. A. Gupta, WSMP: Watson Sparse Matrix Package. Part II-direct solutions of general sparse systems, *IBM Research Report*, 2000
12. A. Gupta, Fast and effective algorithms for graph partitioning and sparse matrix ordering, *IBM Research report*, 1996.
13. J. H. Kim and S. J. Kim, A Multifrontal Solver Combined with Graph Partitioners, *AIAA Journal, Vol 38, No.8*, 1999