한국소성가공학회 2001년도 추계학술대회 논문집 pp.113~116

직접 행렬해법에 대한 일반적 비교

안병기* · 박용복** · 김정호*** · 양동열****

The General Comparison between Direct Matrix Solvers

B. K. An, Y. B. Park, J. H. Kim, and D. Y. Yang

Abstract

Finite element analysis programs have been for metal forming process design They will become more and more important in understanding forming process For large-scale forging analysis problems, the performance of a linear equation solver is very important for the overall efficiency of the analysis code. With problem size increased, the computation time needs to be reduced, which is spent on solving the system of algebraic equations associated with finite element model Many matrix solvers have been developed and used usefully in finite element program for this purpose.

Key Words : Fill-in, Band, Profile, Skyline, Sparse, Frontal, Multifrontal

1. 서 론

단조공정의 개선 및 소재의 결함 방지 등을 위해서는 금형의 설계 및 가공에 대한 연구가 필수적이며, 최근에는 컴퓨터의 발전과 더불어 CAD/CAM/CAE를 이용하여 금형의 설계 및 가공을 효율적으로 행할 수 있는 연구가 진행중이다. 본 연구에서는 고가의 금형을 직접 제작하기 이전에 컴퓨터에서 금형을 임으로 설계변경을 하여 시뮬레이션을 해본 후에 원하는 제품의 성형이 가능할 때 실제로 금형을 제작함으로서 금형의 설계 및 가공의 시행착오를 최대한 줄이도록 하기 위한 3차원 단조공정 전용 유한요소 해석프로그램 개발에 있어서 계산시간의 효율화를 위한 다양한 행렬해법(matrix solver)의 적용에 초점을 맞추고 있다.

효율적인 행렬해법의 적용은 해석에 있어서 계산시간과 저장공간의 최소화를 가져오고 이를 통하여 복잡한 형상에 대한 3차원 단조해석을 가능하게 함으로서 단조실험을 통한 시행착오를 최소화하여 개발시간 단축 및 비용절감 등의 효과를 볼 수 있다. 또한 효율적인 행렬해법은 소성가공공정 뿐 아니라 다른 공정 및 설계에서도 그대로 적용될 수 있는 공통 기술이기 때문에 그 활용이 여타 분야에도 확산될 수 있을 것으로 기대된다.

2. 희소행렬

희소행렬(Sparse Matrix)은 많은 zero 요소들을 포함하고 있는 행렬을 말하며, 행렬의 이러한 특성은 행렬해법에 있어서 계산과정이 효율적으로 수행되도록 행렬의 데이터 구조를 어떻게 정의할 것인가에 관심이 모아진다. 또한, 이런 행렬의 성질은 기본적인 선형 대수학 문

* 공주대학교 기계공학과 대학원
** 공주대학교 기계공학부
*** KISTI 슈퍼컴퓨팅 센터
**** 한국과학기술원 기계공학과

제에 적용시키는 과정에서 저장공간과 계산시간을 감소시키는데 유용하게 적용되어진다.

행렬해법에는 크게 직접해법과 반복해법이 있다. 직접해법은 계수행렬의 소거나 분해를 통해 삼각행렬화 하는 과정에서 얻어진 역행렬을 이용해 해를 구하고 간접해법에 있어서는 계산 초기에 임으로 해를 가정하고, 이때 가정한 해를 여러 가지 반복계산알고리즘에 의해 단계적으로 개선해 나감으로써 주어진 방정식의 근사해를 구하는 방법이다. 어떤 방법을 쓰는가는 행렬의 크기나 형태에 따라 그 효율성을 달리 할 수 있다.

직접해법에서 저장공간과 계산시간을 감소시키는 일은 행렬을 구성하는 zero 요소들에 대해서는 계산을 수행하지 않도록 시도되고 있다. 행렬의 nonzero 요소의 희소성(sparsity)를 이용하는 고전적 방법들은 다양한 방법으로 접근이 이루어지고 있으며 매우 효율적이다. 이 방법에서는 fill-in이 중요하게 다루어지고 있다. 행렬의 계산과정에서 처음에는 zero 요소인 부분이 nonzero 요소의 분해나 소거의 계산과정을 통해서 nonzero 요소로 채워지게 되고, 이렇게 새로 생겨나는 nonzero 요소들을 fill-in 이라 한다. 많은 시도들을 통해서 이 fill-in 이 적게 생겨나도록 연구되어 왔으며 이러한 시도들은 행렬내에서 zero 요소들이 계산후에도 zero 요소로 남을 것인지 아니면 nonzero 요소로 될 것인지에 관한 측면에서 이루어져 왔다[1].

3. 직접 행렬해법

3.1 General Sparse Method

직접해법에서 문제가 되는 Fill-in은 소거 과정중에 생겨나고 그 수를 줄이는 방법으로 소거될 경우 다음 소거될 요소에 포함되도록 방정식을 재배치하게 되면 계산시간을 감소시키게 된다. 그래프 이론에 기초한 여러 Ordering 방법들은 이러한 목적으로 수행되며 minimum degree ordering도 fill-in을 감소시키는데 사용되어진다. ALAN GEORGE과 JOSEPH W. LIU는 위의 Ordering 방법을 통해 일반적인 sparse matrix 해법을 제시하였으며 그 외의 Quotient Tree Method, One-Way Dissection, Nested Dissection 등의 Ordering 방법들을 연구하였다[2].

3.2 Band Solver

구조 해석시의 선형 방정식의 계수행렬(강성행렬)은 항상 대칭이고 띠(band)모양으로 되어 있다. 이러한 밴드폭은 작을수록 계산시간과 저장공간을 감소시키게 되며 절점 번호의 재배치를 통해서 이루어진다.

전체 방정식의 nonzero 요소들은 절점에 부여된 번호에 의해 그 위치가 정해지며 밴드폭의 크기를 결정하게 된다. 또한 같은 수의 절점을 가진 문제라도 그 형상에 따라서 밴드폭을 달리 할 수 있다. 3차원의 경우 절점의 자유도를 고려하지 않은 상태에서 1차적으로 절점 번호의 재배치를 통해 밴드폭을 감소시킬 수 있다. 또한 유한요소 수식화 과정의 penalty method에 의해 경계조건이 주어진 절점의 자유도는 전체 방정식에서 zero 요소로 나타나는데 이 경우 추가적인 방정식 재배치를 통해서 밴드폭을 감소시킬 수 있다.

절점 번호의 재배치 방법에는 Cuthill-McKee Ordering, Gibbs-Pool-StockMayer Ordering 등의 방법이 있으며 방정식 교환에 있어서도 같은 형태로 이용될 수 있다. Band Solver는 band 폭을 감소시킴으로 해서 fill-in을 줄이고 계산효율을 높이게 된다.

3.3 Skyline Solver

공학적으로 의미가 있는 해석은 일반적으로 변수가 많기 때문에 강성행렬을 압축하여 저장하는 것이 컴퓨터 기억장치의 활용과 계산 효율의 관점에서 바람직하다.

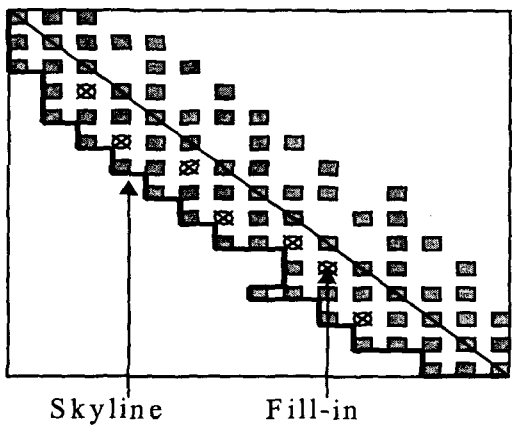

Fig. 1 Profile Matrix

강성행렬의 skyline 개념에 기초한 방법은 방정식을 푸는데 있어서 효율을 향상시키기 위해 자주 사용된다. Skyline은 강성행렬의 각 열에서 처음 0이 아닌 성분을 잇는 구획선이다. Skyline 방법(Fig. 1)에서는 주대각선과 skyline 사이의 성분만을 일렬로 이어 1차원 배열로 저장된다. 일반적으로, 이 과정은 컴퓨터에서 훨씬 작은 저장공간을 차지하고 일반적인 띠모양 형식보다 방정식을 푸는 관점에서 더 효율적이다[3].

Skyline 방법은 direct solver에서 주로 이용되고 있다. 구조화된 matrix로서 fill-ins에 관련된 문제를 해결하는데 있어서 효율적이고 대칭구조를 가진 matrix의 경우에 보다 효율적으로 적용시킬수 있다.

3.4 Multifrontal Solver

연립방정식을 푸는데 있어서 다른 방법들이 요소강성행렬로부터 전역 강성행렬을 완전히 구성한 다음 그것을 이용해서 문제를 푸는 반면, frontal solver는 Fig. 2에서처럼 요소 강성행렬을 조립해나가는 과정에서 조립이 완료된 자유도를 그 즉시 소거해나가는 방법이다. 특히 이 방법은 유한 요소 해석 과정에서 나타나는 행렬의 구조적 특성을 충분히 살려 계산량을 크게 줄일 수 있을 뿐만 아니라 전역강성행렬을 조립하지 않고 또 소거된 자유도를 보조기억장치(하드디스크)에 저장함으로써 필요로 하는 주기억장치의 용량을 크게 줄일 수도 있다.

Multifrontal solver는 여러 개의 프론트를 각각 다른 영역에서 퍼져나가도록 해서 이들 프론트를 다시 합쳐나가는 방법을 사용한다. 이 방법은 프론트를 여러 개로 분할함으로써 프론트의 크기가 작은 상태로 유지될 수 있도록 함으로써 프론트 해법보다도 계산량을 훨씬 더 줄일 수가 있는 매우 효율적인 방법이다[4].

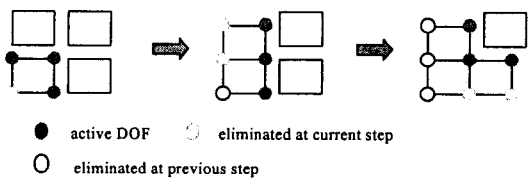

Fig. 2 Frontal Method

4. 결 론

이미 많은 상용 S/W들이 sparse solver를 사용하고 있고 저장공간과 계산시간 감소라는 측면에서 다양한 방법들이 시도되고 있으며, 일부 방법들은 그 코드가 공개되어 있다.

원통형상의 3차원 업셋팅 문제에서 각 행렬해법의 계산시간과 저장공간을 테스트하여 Table 1에 나타내었다. Multifrontal Solver(J.H. Kim[4])의 경우 가장 빠른 계산시간을 보여 주었으며, Band Solver에 비해 문제 크기가 커질수록 수십배에서 수백배 빠른 계산을 수행하였다. 저장공간의 사용량은 Profile Matrix를 사용하는 Skyline Solver(Noboru Kikuchi[5]) 가 Band Solver보다 1/3정도의 메모리를 사용하였으며 Multifrontal Solver의 경우는 임시파일 크기가 11.2~112MB 이었다. 전역강성행렬을 사용하는 Band Solver와 Skyline Solver의 강성행렬 크기는 Fig. 4에 나타내었다.

행렬해법에 따라 계산시간에 큰 차이를 나타냄으로써 적절한 해법을 적용할 경우 해석 시간을 단축시키고 단조해석 프로그램을 효율화 시킬 수 있을 것으로 기대된다.

Table 1 Matrix Calculate Time and Required Memory Size (at CPU PIII 700Mz, RAM 327MB)

절점수	계산시간(sec)			저장공간(MB)	
	MFS	Skyline Solver	Band Solver	Band Matrix	Profile Matrix
1960	2.3	6.8	118.2	52.2	16.3
3018	5.2	19.7	271.4	96.6	34.2
4992	11.2	66.2	2375.5	247.1	78.2
6942	23.1	147.8	6298.6	433.3	141.9
1001	31.9	338.4		985.0	271.7

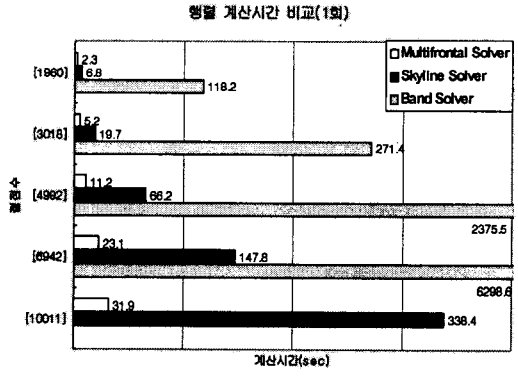

Fig. 3 The Comparison of Calculate Time between Matrix Solvers

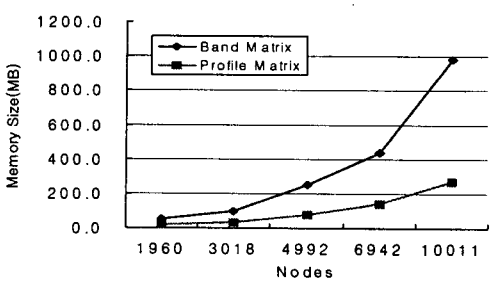

Fig. 4 Matrix Size

참 고 문 헌

(1) Zahari Zlatev, Computational Methods for General Spars Matrices, KLUVER ACADEMIC PUBLISHE -RS, 1991, pp.6~8

(2) ALAN GEORGE, JOSEPH W. LIU, Computer Solution of Large Sparse Positive Definite Systems, PRENTICE-HALL, 1981, pp.49~158

(3) Daryl L.Logan 원저, 신종계,이용신,조성욱 공역, Logan의 유한요소법 첫걸음, 시그마프레스, 1998, pp.145~152

(4) J.H.Kim and S.J.Kim, Multifrontal Solver Combined with Graph Partitioners AAIA Journal vol. 37, no.8, pp.964~970

(5) Noboru Kikuchi, Finite element methods in mechanics, CAMBRIDGE UNIVERSITY PRESS, pp.206~209