經營科學
第19卷 第2號
2002年 11月

초마디 멀티프런탈 방법의 효율적인 구현*

박찬규** · 박순달***

An Efficient Implementation of the Supernodal Multifrontal Method*

Chan-Kyoo Park** · Soon-dal Park***

■ Abstract ■

In this paper, some efficient implementation techniques for the multifrontal method, which can be used to compute the Cholesky factor of a symmetric positive definite matrix, are presented. In order to use the cache effect in the cache-based computer architecture, a hybrid method for factorizing a frontal matrix is considered. This hybrid method uses the column Cholesky method and the submatrix Cholesky method alternatively.

Experiments show that the hybrid method speeds up the performance of the supernodal multifrontal method by 5%~10%, and it is superior to the Cholesky method in some problems with dense columns or large frontal matrices.

Keyword : Multifrontal Method, Cholesky Factor, Interior Point Method, Factorization

1. 서 론

내부점법(interior point method)은 개선방향을 구하기 위해 반복적으로 대칭양정치(symmetric positive definite)행렬로 이루어진 선형방정식을 풀어야 하는데 이 과정이 전체 수행시간의 대부분을 차지한다[5, 11]. 따라서, 내부점법의 수행속도는 이 선형방정식을 얼마나 효율적으로 푸는가에 달려

논문접수일 : 1999년 7월 13일 논문게재확정일 : 2002년 8월 16일
* 본 연구는 한국과학재단 특정기초연구사업(R01-2002-000-00168-0)의 지원을 받았음.
** 한국전산원 지식정보기술단 정보기술감리부
*** 서울대학교 산업공학과

있다. 대칭양정치행렬로 구성되는 선형방정식을 푸는 방법으로 촐레스키 분해(Cholesky factorization)가 흔히 사용되는데[2], 촐레스키 분해는 하삼각행렬(lower triangular matrix)인 촐레스키인자(Cholesky factor)를 구하여 전 · 후방 치환(forward · backward substitution)에 의해 선형방정식의 해를 구한다[2]. 촐레스키인자를 구하는 방법으로는 열촐레스키(clolumn Cholesky) 방법, 행촐레스키(row Cholesky) 방법, 부분행렬 촐레스키(submatrix Cholesky) 방법과 본 연구에서 다루게 될 멀티프런탈 방법(multifrontal method) 등이 있다[9].

멀티프런탈 방법은 대형희소행렬(large sparse matrix)의 상하분해를 수행하는 방법의 하나로서 밀집된 작은 행렬들의 분해를 통해서 희소행렬의 상하분해를 구하는 방법이다[1, 8, 12]. 따라서, 내부점법 수행 과정에 발생하는 대칭양정치 행렬의 촐레스키인자를 구하는데 멀티프런탈 방법이 사용될 수 있다. 멀티프런탈 방법은 Duff & Reid에 의해 개발되었는데[7], 밀집행렬을 사용하기 때문에 직접주소 참조(direct addressing)가 가능하여 특히 벡터컴퓨터(vector computer)에 적합하다. 또한, 분해하는 동안 행렬 전체 요소들을 접근할 필요 없이 소거에 참여하는 열들에 대한 정보만을 가지고 있으면 분해가 가능하여 가상메모리(virtual memory)시스템에 효과적으로 사용될 수 있는 방법이다[8, 12].

멀티프런탈 방법에서 순서화(ordering) 과정은 두 단계로 이루어진다. 첫 번째 단계에서는 열촐레스키 방법에서 사용되는 것과 동일한 순서화 과정을 통해 촐레스키인자의 비영요소(nonzero) 개수를 줄이고, 두 번째 단계에서는 멀티프런탈 방법의 수행도중에 나타나는 수정행렬(update matrix)의 보관이 용이하도록 추가적인 후재순서화(post reordering)를 수행한다(순서화와 후순서화 방법은 [3, 4, 5, 10, 13] 등을 참조하기 바란다). 본 연구에서는 행렬의 비영요소 형태만을 고려하는 두 단계의 순서화가 이미 수행되었다고 가정하고 그 후에 촐레스키인자를 구하는 과정을 다룬다.

멀티프런탈 방법에 초마디(supernode)의 개념을 도입한 초마디 멀티프런탈(supernonal multifrontal) 방법이 제안되면서 멀티프런탈 방법의 수행속도가 크게 향상되었다[5, 14, 11]. 또한 멀티프런탈 방법의 구현시 필요한 기억공간을 최소화하는 순서화(ordering) 방법을 Liu[12]가 제시하였고, Jung[11]은 loop-unrolling기법을, Ashcraft[6]는 지역인덱스(local index)사용과 초마디 완화 등을 통해 멀티프런탈 방법의 수행 속도를 개선한 연구결과를 발표하였다.

한편, 멀티프런탈 방법은 중간 단계에서 수정행렬(update matrix)을 생성하고 프런탈 행렬(frontal matrix)의 분해를 통해 촐레스키인자를 구한다. 대형 대칭양정치 행렬이 밀집열을 갖거나 희소도가 낮으면 프런탈 행렬의 크기가 매우 커지는데, 이 때 수정행렬의 분해 속도 제고가 멀티프런탈 방법의 전체 속도 향상에 큰 영향을 미친다. 따라서, 본 연구에서는 멀티프런탈 방법을 수행하는 중간과정에서 생성되는 프런탈 행렬을 분해할 때 캐쉬(cache) 효과를 높임으로써 프런탈 행렬의 분해시간을 단축할 수 있는 혼합 초마디 멀티프런탈 방법(hybrid supernodal multifrontal method)을 제시한다. 제시된 혼합 초마디 멀티프런탈 방법은 프런탈행렬의 분해시에 부분행렬촐레스키(submatrix cholesky)와 열촐레스키를 혼합하여 적용하는 형태의 방법으로서, 실험을 통해 제시된 방법의 효율성을 보인다. 제시할 방법과 멀티프런탈 방법, 열촐레스키 및 부분행렬촐레스키 등과의 개념적 비교는 2장에서 살펴보도록 한다.

2. 멀티프런탈 방법과 열 · 부분행렬 촐레스키 방법

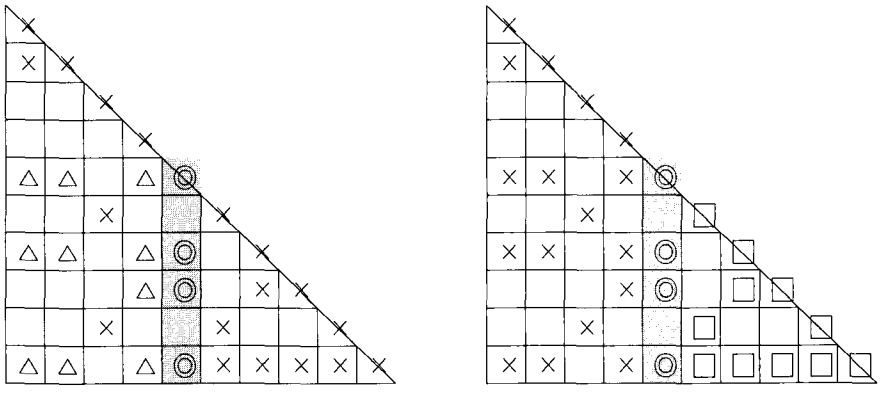
촐레스키인자를 구하는 방법으로 본 연구에서

다루게 될 열촐레스키와 부분행렬 촐레스키 방법을 먼저 비교해 본 후 부분행렬 촐레스키 방법과 비교하여 멀티프런탈 방법의 개념을 설명하도록 한다.

대칭양정치 행렬 A이 $m \times m$이고, A의 촐레스키인자 L(즉, $A = LL^T$)을 구한다고 하자. 열촐레스키나 부분행렬촐레스키 방법을 사용하여 L을 단계적으로 구하기 위해 먼저 하삼각행렬 M의 대각이하의 요소를 A의 요소들로 초기화한다. 즉, 다음과 같이 하삼각행렬 M을 초기화하였다고 하자. 최종단계에서 M은 촐레스키인자 L이 된다 (임의의 차원을 갖는 행렬 G의 (i,j)번째 요소를 G_{ij}로 나타내기로 한다.)

$$M_{ij} \leftarrow \begin{cases} A_{ij} & , \text{ if } i \geq j \\ 0 & , \text{ otherwise} \end{cases}$$

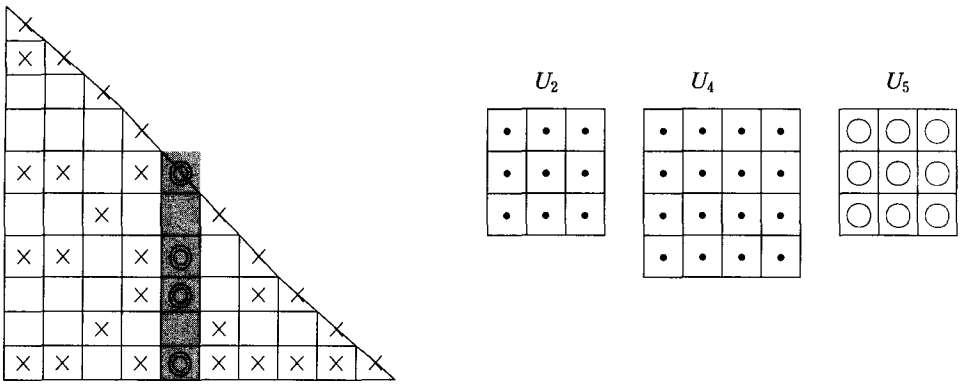
열촐레스키과 부분행렬 촐레스키 방법의 차이는 <그림 1>에 나타나 있다. 열촐레스키와 부분행렬 촐레스키는 촐레스키인자 L를 구할 때 열순서로 구한다는 점에서 공통점이 있지만, L의 각 열을 구할 때 참조하거나 수정되는 요소가 서로 다르다. <그림 1>에서 열촐레스키는 이미 계산된 L의 열들을 참조하여 새로운 촐레스키인자의 열을 계산한다. 반면 부분행렬 촐레스키는 L의 새로운 열 값을 계산할 때 이미 계산된 L의 열들을 참조하지 않는다. 대신 부분행렬 촐레스키는 아직 계산되지 않은 L의 열들에 해당하는 M의 열들의 값을 수정한다. 열촐레스키는 L의 이미 계산된 열의 값을 참조하여 L의 새로운 열을 계산하고, 부분행렬촐레스키는 L의 새로운 열을 계산한 후 아직 계산되지 않은 L의 열들을 위해 수정 연산을 수행한다. 따라서, 열촐레스키에서는 초기에 참조하는 열들이 적지만 후반부로 갈수록 참조해야 하는 열의 개수가 많아지게 된다. 반면, 부분행렬촐레스키는 초기에는 수정해야 하는 M의 요소들이 많지만 후반부로 갈수록 그 개수가 감소하게 된다. 이러한 열촐레스키와 부분행렬 촐레스키의 대비되는 특성은 캐쉬효과를 극대화하고자 할 때 중요하게 고려되어야 할 사항으로 본 연구의 4장에서 이용된다.



열촐레스키 부분행렬촐레스키

주) × : 참조되지도 수정되지도 않는 요소
△ : 참조되지만 수정되지 않는 요소
◎ : 현단계에서 계산되는 촐레스키인자의 열 요소
□ : 값이 수정되는 요소

〈그림 1〉 열촐레스키와 부분행렬촐레스키의 비교



주) × : 참조되지도 수정되지도 않는 요소
◎ : 현단계에서 새로 계산되는 촐레스키인자의 열 요소
• : 참조된 후 삭제되는 요소
○ : 새로 생성되어 저장되는 요소

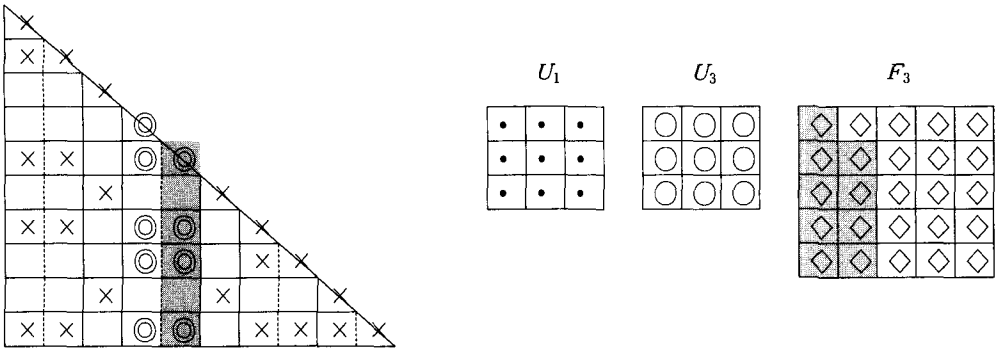
〈그림 2〉 멀티프런탈 방법의 개념

그러면, 멀티프런탈 방법과 부분행렬촐레스키와의 차이점을 살펴보자. 멀티프런탈 방법의 한 단계를 그림으로 표현하면 <그림 2>와 같다. 멀티프런탈 방법은 촐레스키인자의 다섯 번째 열을 계산하기 위해 미리 저장되어 있던 수정행렬 U_2, U_4와 M의 다섯 번째 열을 참조한다. 이 때, U_2와 U_4은 참조된 후 삭제되고 다섯 번째 열의 계산이 끝나면 새로운 수정행렬 U_5을 생성하여 저장해 둔다. U_5는 이 후의 일곱 번째 열을 계산할 때 참조된다.

부분행렬 촐레스키는 촐레스키인자의 새로운 열의 계산이 끝나면 이 후의 열들을 수정하는데, 멀티프런탈 방법은 이 후의 열들을 수정하지 않는 대신에 수정행렬을 만들어 나중에 참조할 수 있게 한다. 멀티프런탈 방법은 수정행렬만 저장해 두면 이미 계산된 촐레스키인자의 열들을 참조하지 않아도 되고, 또한 아직 계산되지 않은 열들도 수정할 필요가 없다. 따라서, 멀티프런탈 방법은 대형희소행렬을 메모리 상에 모두 올려놓지 않고 프런탈 행렬과 계산할 열만을 가지고 촐레스키인자를 계산할 수 있는 장점이 있다.

지금까지 열 · 부분행렬 촐레스키, 멀티프런탈 방법은 모두 촐레스키인자를 계산할 때 한 열씩 계산해나갔는데, 이를 발전시켜 비영요소가 나타나는 위치가 유사한 여러 개의 열들을 묶어 같이 계산하기 위해 초마디(supernode)라는 개념이 사용된다. 초마디의 개념을 멀티프런탈 방법에 도입하면 초마디 멀티프런탈 방법이 되는데, 초마디 멀티프런탈 방법에서는 하나의 초마디 내에 있는 여러 개의 L의 열을 같이 계산한다. <그림 2>에 초마디 개념을 도입한 초마디 멀티프런탈 방법의 개념을 그림으로 나타내면 <그림 3>과 같다. <그림 3>에서 1열 · 2열, 4열 · 5열은 동일한 초마디에 속하므로 4열과 5열은 같은 단계에서 계산된다.

<그림 3>에는 <그림 2>에서 나타나지 않았던 F_3 행렬이 있는데, F_3는 프런탈 행렬로 U_1과 M의 4열 · 5열을 참조하여 만들어진다. F_3은 촐레스키인자의 4열과 5열을 계산하는데 사용되고 또한 F_3으로부터 U_3가 계산된다. <그림 2>에서도 동일하게 프런탈 행렬이 계산과정에 발생하지만 <그림 2>에서는 생략하여 표시했을 뿐이다. F_3의 1열과 2열의 대각이하 요소는 촐레스키인자 L



주) × : 참조되지도 수정되지도 않는 요소
◎ : 현단계에서 새로 계산되는 촐레스키인자의 열 요소
• : 참조된 후 삭제되는 요소
○ : 새로 생성되어 저장되는 요소
◇ : 현단계에서 새로 생성되어 삭제되는 임시 비영요소

〈그림 3〉 초마디 멀티프런탈 방법의 개념

의 4열과 5열이 된다. F_3의 1열과 2열을 계산할 때 열촐레스키 방법과 부분행렬 촐레스키 방법을 모두 사용할 수 있다. 그러나, 대형선형계획법문제에서 한 초마디에 속하는 열의 개수가 백 개 이상이 되는 경우가 흔히 발생한다. 따라서, 프런탈 행렬 F_3에서 캐쉬효과를 최대한 이용하기 위해서는 열촐레스키와 부분행렬 촐레스키의 장점을 혼합한 방법이 필요하다. 열촐레스키는 초기단계일수록 참조하는 비영요소의 수가 작고, 부분행렬 촐레스키는 후반부로 갈수록 참조하는 비용요소의 수가 작으므로 이를 적절히 혼합하여 사용하는 방법을 제시하는 것이 본 연구의 핵심 목적이다.

3. 초마디 멀티프런탈 방법

혼합 초마디 멀티프런탈 방법을 제시하기 전에 초마디 멀티프런탈 방법을 자세히 알아볼 필요가 있다.먼저 $m \times m$ 차원의 희소한 대칭양정치 행렬 A의 촐레스키인자를 $L=(l_{ij})$이라 하자. 행렬 A의 각 행은 촐레스키인자의 비영요소를 줄이는 순서화 방법에 의해 이미 행과 열의 순서가 매겨진 행렬이라고 가정한다.

행렬 A의 삭제나무(elimination tree)는 하삼각행렬 L의 비영요소 구조에 의해 다음과 같이 정의된다. 여기서 $Parent[j]$는 삭제나무에서 점 j의 부모점(parent node)을 의미한다.

[정의 1] 삭제나무

$$Parent[j] = \min\{i \mid l_{ij} \neq 0, i > j\}$$

즉, 점 j의 부모점은 j열의 대각요소 아래에 첫 번째로 나타나는 비영요소의 행번호에 해당하는 점이 된다. 이렇게 생성되는 그래프(graph)는 일반적으로 여러개의 나무(tree)로 이루어질 수 있으나 본 연구에서는 하나의 나무만 존재한다고 가정하기로 한다. 여러 개의 나무가 있는 경우 각각의 나무에 본 연구결과를 적용하면 되므로 일반성을 잃지는 않는다.

멀티프런탈 방법에서 삭제나무는 두 가지 용도로 사용되는데 첫 번째는 다음의 정리에 의해 알 수 있듯이 L의 각 열간의 계산 의존성에 대한 정

보를 제공한다. 임의의 행렬 M에 대해, $M_{\cdot i}$과 $M_{i\cdot}$는 각각 i번째 열과 i번째 행을 나타낸다.

[정리 1] 열 사이의 수치적 종속성[13]

$L_{\cdot i}$는 $A_{\cdot i}$와 $\{L_{\cdot k} \mid k \in T[i],\ k \neq i\}$에 의해 결정된다.

단, $T[j]$는 삭제나무에서 점 j를 루트(root)로 하는 부분나무(subtree)에 속하는 점의 집합

즉, 촐레스키인자의 i번째 열의 값은 주어진 행렬 A와 삭제나무에서 점 i의 후손점들에 해당하는 L의 열들에 의해 구해지게 된다. 이는 $T[i] \cap T[j] = \emptyset$ 인 $L_{\cdot i}$와 $L_{\cdot j}$는 독립적으로 계산될 수 있음을 의미한다.

두 번째로 삭제나무는 멀티프런탈 방법에서 재순서화(reordering)를 위한 도구로 사용된다. 멀티프런탈 방법에서는 계산 특성상 행렬 A의 열들이 삭제나무에서의 후순서(postorder)에 따라 배열되는 것이 좋다. 후순서에 의한 행과 열의 재배열은 촐레스키 비영요소 개수에는 영향을 미치지 않고 다만 행과 열의 위치만을 바꾼다. 후순서화가 필요한 이유는 이 후의 초마디 멀티프런탈 방법의 흐름을 보면 쉽게 알 수 있다. 본 연구에서는 행렬 A는 이미 삭제나무의 후순서에 의해 배열되어 있다고 가정한다.

초마디는 촐레스키 분해의 속도를 향상시키고자 제안된 개념으로서 L의 인접한 열들을 하나로 묶어서 한꺼번에 계산하고자 하는 것이 기본 개념이다. 점 i에 대해, $Adj(i)$는 $L_{\cdot i}$에서 대각요소를 제외한 비영요소의 행지수 집합을 나타낸다. 즉, $Adj(i) = \{j \mid l_{ji} \neq 0,\ j > i\}$이다. 또한, 편의상 $Adj(\{\cdot\})$는 집합 $\{\cdot\}$에 속하는 점들의 $Adj(\cdot)$의 합집합을 나타낸다고 하자. 초마디의 정의는 다음과 같다.

[정의 2] 초마디(supernode)

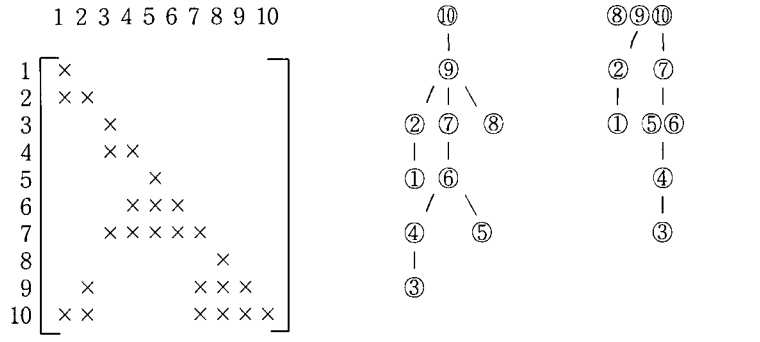
$Adj(T[j]) = \{j+1, \cdots, j+t\} \cup Adj(T[j+t])$을 만족하는 연속된 점의 최대 집합(maximal set) $\{j, j+1, \cdots, j+t\}$을 초마디라 한다 (단, $t = 1, 2, l \cdots$ 이고, $j+t \leq m$이다).

같은 초마디에 속한 열들은 L에서 연속된 열들이고, 각 열들의 하삼각행렬 아래 부분이 모두 같은 열구조(off-block-diagonal column structure)를 가지게 된다.

초마디를 사용하면 압축 행지수(compressed row index scheme)를 통해 L의 각 비영요소의 행지수를 보관하는데 드는 기억 장소를 줄일 수 있다. 또한 초마디에 속하는 열들에 대해서는 비영요소의 행지수가 같으므로 메모리 참조(memory reference) 회수를 줄일 수 있고 데이터의 지역성(data locality)으로 캐쉬 효과를 활용하여 계산시간을 줄일 수 있다.

삭제나무에서 같은 초마디에 속하는 점들을 묶어 하나의 초마디로 표현하여 만들어진 삭제나무를 초마디 삭제나무(supernodal elimination tree)라고 한다. 초마디 삭제나무는 삭제나무에서 각 열간의 계산 종속성을 그대로 가지게 된다. 예를 들어 아래와 같은 촐레스키 인자 L이 있을 때 삭제나무와 초마디 삭제나무는 <그림 4>과 같다. <그림 4>에서 행렬 L의 1열의 대각요소를 제외한 첫 번째 비영요소는 2행에 있으므로 점 ①의 부모점은 점 ②가 된다. 마찬가지로 4열에서 대각요소를 제외한 첫 번째 비영요소는 6행에 있으므로 점 ④의 비영요소는 점 ⑥이 된다. 점 ⑤와 ⑥은 정의 2에 의해 초마디가 되고, 점 ⑧, ⑨, ⑩도 정의 2에 의해 초마디가 된다. 따라서, 삭제나무에서 같은 초마디에 속하는 점들을 하나의 초마디로 묶어 표현한 것이 초마디 삭제나무이다.

초마디의 개념은 열촐레스키 방법뿐만 아니라 멀티프런탈 방법에도 그대로 적용할 수 있으므로, 멀티프런탈 방법에 초마디의 개념을 도입한 것이 초마디 멀티프런탈 방법이다.



L 행렬 삭제나무 초마디 삭제나무

〈그림 4〉 삭제나무와 초마디 삭제나무

멀티프런탈 방법은 부분행렬 촐레스키 방법을 일반적인 희소행렬에 이용할 수 있도록 변형 또는 개선한 방법으로 볼 수 있다. 멀티프런탈 방법은 기본적으로 삭제나무의 부모-자식(parent-child) 관계를 이용하여 계산을 수행한다. 멀티프런탈 방법에서는 수정행렬 U_j를 사용하는데 개념적으로 U_j는 $T[j]$에 속하는 열들이 j의 조상점들에게 미치는 영향을 모두 나타내고 있는 행렬이다. 따라서, $Parent[j]$를 계산할 때는 $Parent[j]$의 자식점들의 수정행렬을 참조하게 되고 $Parent[j]$의 계산이 끝나면 $T[Parent[j]]$에 속하는 점들이 $Parent[Parent[j]]$에 미치는 영향을 나타내는 수정행렬 $U_{Parent[j]}$를 만들어 스택(stack)이라는 임시 기억 장소에 저장하게 된다.

부분행렬 촐레스키 방법에서는 $L_{\cdot j}$가 구해지면 $L_{\cdot j}$의 값이 $(j+1) \sim m$열에 미치는 영향을 수정연산을 통해 계산해 준다. 그러나, 멀티프런탈 방법에서는 $L_{\cdot j}$가 구해지면 $(j+1) \sim m$열에 대한 수정연산을 곧바로 수행하는 것이 아니라 $L_{\cdot j}$의 값이 $(j+1) \sim m$열에 미치는 영향을 U_j라는 행렬에 저장하여 이를 스택에 임시 보관한다. 이렇게 스택에 저장된 정보는 $L_{\cdot Parent[j]}$를 계산하기 시작할 때 스택에서 제거되어 수정연산에 이용된다.

먼저 $S = \{j, j+1, \ldots, j+t\}$을 초마디라고 하고, $i_1, i_2, \ldots, i_r$을 열 $L_{\cdot j+t}$의 대각 아래의 비영요소들의 행번호라고 하면 부분나무 수정행렬(subtree update matrix)과 프런탈 행렬(frontal matrix)은 다음과 같이 정의된다.

[정의 3] 부분나무 수정행렬 $\overline{U_j}$과 프런탈 행렬 F_j

$$\overline{U_j} = -\sum_{k \in T[j+t] - \{j, \cdots, j+t\}} \begin{pmatrix} l_{j,k} \\ l_{j+1,k} \\ \cdots \\ l_{j+t,k} \\ l_{i_1,k} \\ \cdots \\ l_{i_r,k} \end{pmatrix} \begin{pmatrix} l_{j,k} \\ l_{j+1,k} \\ \vdots \\ l_{j+t,k} \\ l_{i_1,k} \\ \cdots \\ l_{i_r,k} \end{pmatrix}^T,$$

$$F_j = \begin{pmatrix} a_{j,j} & \cdots & a_{j,j+t} & a_{j,i_1} & \cdots & a_{j,i_r} \\ a_{j+1,j} & \cdots & a_{j+1,j+t} & a_{j+1,i_1} & \cdots & a_{j+1,i_r} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ a_{j+t,j} & \cdots & a_{j+t,j+t} & a_{j+t,i_1} & \cdots & a_{j+t,i_r} \\ a_{i_1,j} & \cdots & a_{i_1,j+t} & & & \\ \vdots & \cdots & \vdots & & \mathbf{0} & \\ a_{i_r,j} & \cdots & a_{i_r,j+t} & & & \end{pmatrix} + \overline{U_j}$$

이렇게 정의된 $\overline{U_j}$는 $(t+r+1)\times(t+r+1)$ 행렬이 되고 삭제나무에서 점 j의 자손점들이 S에 속하는 열들에 미치는 영향을 나타내게 된다. F_j은 마찬가지로 $(t+r+1)\times(t+r+1)$ 행렬이고, F_j는 S에 속하는 A행렬 요소의 값과 L의 j번째 열 이전의 요소들이 S에 속하는 열들에 미치는 영향을 모두 더한 결과를 나타낸다. 초마디 멀티프런탈 방법은 프런탈 행렬 F_j를 구하여 S에 속하는 L의 열들의 값을 차례로 구해 나간다. 여기서 F_j는 밀집행렬 형태로 보관된다. S에 속하는 L의 모든 열들의 값이 구해지면 다음과 같은 관계가 성립하는 행렬 U_j가 구해지게 되는데 U_j를 수정행렬(update matrix)이라 한다.

$$F_j = V_j \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & & & \\ \vdots & \vdots & \cdots & \vdots & & U_j & \\ 0 & 0 & \cdots & 0 & & & \end{pmatrix} V_j^T, \quad \text{단,}$$

$$V_j = \begin{pmatrix} l_{j,j} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ l_{j+1,j} & l_{j+1,j+1} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ l_{j+t,j} & l_{j+t,j+1} & \cdots & l_{j+t,j+t} & 0 & \cdots & 0 \\ l_{i_1,j} & l_{i_1,j+1} & \cdots & l_{i_1,j+t} & & & \\ \vdots & \vdots & \cdots & \vdots & & I & \\ l_{i_n,j} & l_{i_n,j+1} & \cdots & l_{i_n,j+t} & & & \end{pmatrix}$$

U_j는 출레스키인자 L에서 $T[j+t]$를 자손점으로 포함하고 있는 열들에 대해 $T[j+t]$에 속한 점들이 미치는 영향을 모두 합산한 것이다. 프런탈 행렬 F_j를 구하기 위해 먼저 계산되어야 하는 $\overline{U_j}$는 다음과 같은 관계식에 의해 구할 수 있게 된다. 여기서 $c_1, c_2, \ldots, c_y$을 초마디 삭제나무에서 초마디 j의 자식 초마디들이고 0은 $(t+r+1)\times(t+r+1)$ 차원 행렬을 나타낸다. 여기서, $\oplus$는 확장 덧셈(extend-add) 연산자로 지수와 차원이 다른 행렬을 $\overline{U_j}$의 차원인 $(t+r+1)\times(t+r+1)$로 맞추어 해당 행과 열에 덧셈을 수행한다.

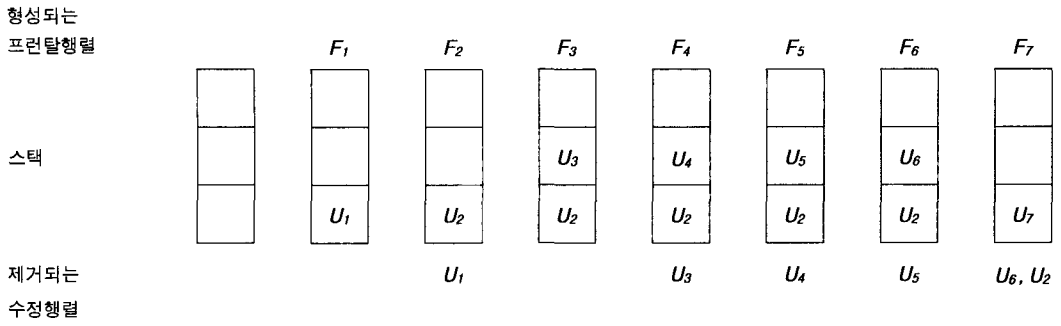
$$\overline{U_j} = \mathbf{0} \oplus U_{c_1} \oplus U_{c_2} \oplus \cdots \oplus U_{c_y}$$

위의 식으로부터 알 수 있듯이 프런탈 행렬 F_j을 구하기 위해 자식 초마디들의 수정행렬을 보관해야 한다. 수정행렬은 부모 초마디가 계산되는 시점에 읽혀지고 이 후에 그 수정행렬은 없어지게 된다. 따라서 보관 구조로 스택(stack)을 사용할 수 있는데, 이는 행렬 A를 후순서에 따라 배열하면 항상 LIFO(Last In First Out)형태로 수정행렬들이 사용되기 때문이다. 이것이 바로 멀티프런탈 방법에서 행(열)을 삭제나무의 후순서에 따라 배열하는 이유이다. 따라서, $\overline{U_j}$는 스택의 맨 위에 저장된 자식 초마디의 개수의 수정행렬들을 가져와서 확장 덧셈을 수행함으로써 구할 수 있다. 예를 들어 <그림 4>에서는 초마디가 7개 존재한다. 초마디 멀티프런탈 방법을 사용할 때 각 단계에서 스택의 저장되는 수정행렬과 읽혀지는 수정행렬은 다음 <그림 5>와 같다.

프런탈 행렬과 수정행렬를 사용하여 L의 각 열을 구해 나가는 초마디 멀티프런탈 방법은 다음과 같다.

4. 혼합 초마디 멀티프런탈방법

초마디 멀티프런탈 방법에서는 <그림 6>의 8



〈그림 5〉 스택에 보관되는 예

1 : **for** each S **do**
2 : $S = \{j, j+1, \ldots, j+t\}$ 이고 $i_1, i_2, \ldots, i_r$ 을 $L_{\cdot j+t}$의 대각 아래 비영요소의 행지수라 하자.
3 : 아래와 같이 F_j를 형성한다.

$$F_j = \begin{pmatrix} a_{j,j} & \cdots & a_{j,j+t} & a_{j,i_1} & \cdots & a_{j,i_r} \\ a_{j+1,j} & \cdots & a_{j+1,j+t} & a_{j+1,i_1} & \cdots & a_{j+1,i_r} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ a_{j+t,j} & \cdots & a_{j+t,j+t} & a_{j+t,i_1} & \cdots & a_{j+t,i_r} \\ a_{i_1,j} & \cdots & a_{i_1,j+t} & & & \\ \vdots & \cdots & \vdots & & \mathbf{0} & \\ a_{i_r,j} & \cdots & a_{i_r,j+t} & & & \end{pmatrix}$$

4 : nchildren을 초마디 삭제나무에서 S의 자식의 개수라 하고, 자식점을 $c_1, \cdots, c_{nchild}$ 라 하자.
5 : **for** k : = 1 **to** nchildren **do**
6 : update matrix U_{c_k}을 스택에서 꺼낸 다음 $F_j := F_j + U_{c_k}$을 만든다.
7 : **endfor ;**
8 : $L_{\cdot j}, \cdots, L_{\cdot j+t}, U_{j+t}$ 을 구한다.
9 : U_{j+t}을 스택에 넣는다.
10 : **endfor ;**

〈그림 6〉 초마디 멀티프런탈 방법

행과 같이 프런탈 행렬내에서 촐레스키인자의 값을 구하게 된다. 이 때 어떤 촐레스키 분해 방법을 사용하느냐에 따라 초마디 멀티프런탈 방법의 효율이 달라질 수 있다.

다음은 열촐레스키 방법과 부분행렬촐레스키 방법을 각각 사용하여 $(s \times s)$ 행렬 A의 촐레스키인자 L의 t번째 열의 값을 구하는 방법을 기술한 것이다. L의 각 요소의 값은 처음에 분해하고자 하는 A의 각 요소 값으로 초기화되어 있고 계산결과는 L에 씌어짐에 주의하자. 또한 cmod (j, k)는

$$\begin{pmatrix} l_{kk} \\ l_{k+1,k} \\ \cdots \\ l_{sk} \end{pmatrix} \leftarrow \begin{pmatrix} l_{kk} \\ l_{k+1,k} \\ \cdots \\ l_{sk} \end{pmatrix} - l_{kj} \begin{pmatrix} l_{kj} \\ l_{k+1,j} \\ \cdots \\ l_{sj} \end{pmatrix}$$

```
procedure col_cholesky ( t )
    for j : = 1 to t do
        for k : = 1 to j-1 do
            cmod ( k, j ) ;
        endfor
        cdiv ( j ) ;
    endfor
end
```

열출레스키 방법

```
procedure sub_cholesky ( t )
    for j : = 1 to t do
        cdiv ( j ) ;
        for i : = j + 1 to s do
            cmod ( i, j ) ;
        endfor
    endfor
end
```

부분행렬 출레스키 방법

〈그림 7〉 열출레스키와 부분행렬 출레스키

를 의미하고 cdiv (j)는

$$\begin{pmatrix} l_{kk} \\ l_{k+1,k} \\ \cdots \\ l_{sk} \end{pmatrix} \leftarrow \frac{1}{l_{kk}} \begin{pmatrix} l_{kk} \\ l_{k+1,k} \\ \cdots \\ l_{sk} \end{pmatrix}$$

를 나타내는 연산이다. 여기서 cmod( )를 수정연산이라 부르기로 한다.

col_cholesky (t)는 t개의 열의 출레스키인자의 값을 열출레스키로 구하는 방법을 나타낸 것이고 sub_cholesky (t)는 t개의 열의 출레스키인자의 값을 부분행렬 출레스키 방법을 써서 구하는 방법을 나타낸 것이다. 열출레스키 방법은 앞서 계산된 L의 값들을 이용하여 다음 열의 값을 구하는 반면에 부분행렬 출레스키는 L의 한 열이 계산되면 그 열의 값을 이용하여 이 후의 열들에 대한 수정연산을 수행한다.

만약 분해하고자 하는 행렬이 크고 밀집도가 높다면 부분행렬 출레스키 방법은 수정연산을 수행하기 위해 초기에는 많은 L의 요소를 접근하게 되므로 캐쉬효과를 살리기 어렵다. 반면 열출레스키 방법은 초기에는 접근하는 이전 열의 수가 작으므로 비교적 메모리 참조 회수가 작지만 뒤로 갈수록 앞서 계산된 비영요소를 모두 참조해야 하므로 메모리 참조 회수가 점점 많아지게 된다. 열출레스키 방법을 사용하면 후반부에 가서는 캐쉬효과를 살릴 수 없게 된다.

```
procedure cs_cholesky (t)
    for p : = 1 to t do
        for j : = p to min ( t, p + NSWITCH ) do
            for k : = p to j-1 do
                cmod ( k, j ) ;
            endfor
            cdiv ( j ) ;
        endfor
        for j : = p to min ( t, p + NSWITCH ) do
            for k=min ( t, p + NSWITCH ) + 1 to t do
                cmod ( j, k ) ;
            end for
            p = p + NSWITCH ;
        endfor
    endfor
```

〈그림 8〉 열출레스키와 부분행렬 출레스키의 혼합

이상의 논의로부터 촐레스키 분해의 전반부에서는 열촐레스키 방법이 캐쉬효과를 살리는데 효과적이고 후반부에 갈수록 부분행렬촐레스키 방법을 사용하는 것이 캐쉬효과를 살리는데 유리하다.

열촐레스키 방법이 후반부에 갈수록 L의 한 열을 구하기 위해 필요한 메모리 참조회수가 많아지는 것을 방지하고 항상 일정한 개수내의 비영요소만을 참조하여 L의 한 열을 구하는 방법은 다음 <그림 8>과 같다. NSWITCH는 $1 \sim t$ 사이의 정수 값을 갖는 상수라 하자.

위의 수정된 열촐레스키 방법은 항상 L의 한 열을 구하기 위해 참조해야 하는 이전 열의 개수를 NSWITCH 이내로 유지하는 방법이다. 이를 위해서 NSWITCH개의 열이 새로 계산되면 이들 열들로 이후의 열들에 대한 수정연산을 해 줘야 한다. NSWITCH를 스위치 파라미터라 부르기로 한다.

멀티프런탈 행렬 내에서의 L의 각 요소를 구하는 방법으로 수정된 열촐레스키 방법 cs_cholesky ( ) 를 사용하는 혼합 초마디 멀티프런탈 방법은 구체적으로 다음과 같다.

5. 실험 결과

5.1 스위치 파라미터 값 설정

cs_cholesky( )에서 스위치 파라미터 NSWITCH 값을 변화시킴에 따라 전체 수행속도가 달라지게 된다. 스위치 파라미터는 캐쉬효과를 극대화하기 위해 참조하는 비영요소의 개수가 캐쉬메모리의 크기보다 작도록 유지하는 것이 좋다.

스위치 파라미터를 상수값으로 설정하면 일정한 개수의 L의 열이 계산되면 수정연산을 수행하게 된다. 그러나 이 경우는 프런탈 행렬의 크기가 각 초마디마다 달라지므로 참조하는 비영요소의 개수가 너무 많거나 적은 경우가 발생한다. 따라서 스위치 파라미터의 값은 프런탈 행렬의 크기를 고려하여 그 값이 결정되어야만 참조되는 비영요소의 수를 어느 정도 일정한 개수이내로 유지할 수 있을 것이다. 따라서 다음과 같이 NSWITCH를 설정하는 방법을 생각해 볼 수 있다.

$$NSWITCH = \begin{cases} \delta_1, & \text{if } \eta \le \sigma \\ \delta_2, & \text{if } \eta > \sigma \end{cases}$$

단, η는 초마디에 속하는 점의 수를 나타낸다.

1 : **for** each S **do**
2 : $S = \{j, j+1, \cdots, j+t\}$ 이고 $i_1, i_2, \cdots, i_r$ 을 $L_{:j} + t$ 의 대각 아래 비영요소의 행지수라 하자.
3 : 아래와 같이 F_j를 형성한다.

$$F_j = \begin{pmatrix} a_{j,j} & \cdots & a_{j,j+t} & a_{j,i_1} & \cdots & a_{j,i_r} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ a_{j+t,j} & \cdots & a_{j+t,j+t} & a_{j+t,i_1} & \cdots & a_{j+t,i_r} \\ a_{i_1,j} & \cdots & a_{i_1,j+t} & & & \\ \vdots & \cdots & \vdots & & \mathbf{0} & \\ a_{i_n,j} & \cdots & a_{i_n,j+t} & & & \end{pmatrix}$$

4 : nchildren을 초마디 삭제나무에서 S의 자식의 개수라 하고, 자식점을 $c_1, \cdots, c_{nchild}$ 라 하자.
5 : **for** k : = 1 **to** nchildren **do**
6 : update matrix U_{c_k}을 스택에서 꺼낸 다음 F_j : = $F_j + U_{c_k}$을 만든다.
7 : **end for** ;
8 : cs_cholesky (t+1) ;
9 : $U_j + t$ 을 스택에 넣는다.
10 : **end for**

〈그림 9〉 혼합 초마디 멀티프런탈 방법

여기서 NSWITCH를 항상 1로 두면 프런탈 행렬 내에서는 부분행렬 촐레스키 방법을 사용함을 의미한다. 반면 NSWITCH를 항상 각 초마디에 속하는 점의 수와 같게 두면 프런탈 행렬 내에서는 열촐레스키를 사용하고 수정행렬을 구하기 위해 맨 나중에 수정연산을 수행하는 방식을 의미한다. 일반적으로 $\delta_1 \geq \delta_2$로 두는 것이 합리적임을 알 수 있다.

5.2 실험 결과

<표 1>은 NETLIB에 있는 문제들을 내부점 선형계획법 방법에 의해 풀 때 수치적분해(numerical factorization)에 소요되는 시간만을 측정한 것이다. 두 번째는 가장 큰 초마디의 크기를 의미하고 세 번째 열은 열촐레스키 방법을 사용할 때의 시간이고 네 번째에서 여덟 번째는 아래와 같이 스위치 파리미터의 값을 설정한 경우이다. 열촐레스키 방법에 대한 시간은 LPABO ver 4.6f [1] 을 사용한 것이다(LPABO에 대한 상세한 설명은 [15]을 참조하기 바란다).

4번째 열(방법 1) : $NSWITCH = 3$
5번째 열(방법 2) : $NSWITCH = 5$
6번째 열(방법 3) :

$$NSWITCH = \begin{cases} 3, & \text{if } t \leq 10 \\ 5, & \text{if } t > 10 \end{cases}$$

7번째 열(방법 4) :

〈표 1〉 실험 결과

NAME	IT	maxf	Chol.	방법 1	방법 2	방법 3	방법 4	방법 5
bandm	23	36	0.29	0.30	0.33	0.31	0.30	0.32
etamacro	29	86	1.00	1.03	1.07	1.04	1.02	1.03
israel	31	141	1.61	1.51	1.61	1.54	1.49	1.50
agg3	30	102	2.18	2.12	2.12	2.10	2.10	2.10
bnl1	38	76	1.68	1.68	1.77	1.69	1.67	1.69
degen2	18	121	1.31	1.26	1.27	1.27	1.25	1.27
fffff800	44	79	2.78	2.90	2.96	2.91	2.90	2.91
perold	49	139	4.97	4.89	5.20	4.93	4.90	4.91
seba	19	327	11.12	9.54	9.25	9.38	9.66	9.09
shell	26	36	0.48	0.51	0.52	0.51	0.51	0.51
ship04l	35	29	0.99	1.06	1.06	1.05	1.07	1.04
stair	20	112	1.13	1.19	1.19	1.17	1.19	1.20
stocfor2	24	28	2.08	2.28	2.34	2.22	2.36	2.30
tuff	31	67	0.95	0.97	0.99	0.97	1.00	1.02
25fv47	32	160	5.00	4.99	5.08	4.89	4.96	4.92
bnl2	44	336	28.18	28.30	27.10	28.14	26.72	26.59
cycle	53	101	11.03	10.36	10.29	10.26	10.33	10.23
d2q06c	39	343	41.81	39.01	37.61	38.98	37.71	39.09
d6cube	35	310	20.34	19.77	19.33	19.26	18.81	18.66
fit1p	19	626	67.41	55.88	54.16	55.61	54.65	54.29
fit2d	29	24	15.23	15.18	15.09	15.10	15.21	15.31
pilot	51	428	111.51	111.26	107.06	110.40	105.86	113.20
pilot87	48	804	440.05	411.93	408.82	410.32	408.33	414.93
truss	25	121	8.08	8.18	8.13	8.24	8.25	8.28
woodlp	33	155	8.83	8.69	8.62	8.68	8.62	8.74
합 계			790.04	744.79	732.97	740.97	730.87	745.13

주) 실험기종 : Sun Sparc Ultra-170 , RAM : 64MB, 캐쉬메모리 : 32K

$$NSWITCH = \begin{cases} 7, & \text{if } t \le 10 \\ 5, & \text{if } t > 10 \end{cases}$$

8번째 열(방법 5) :

$$NSWITCH = \begin{cases} 3, & \text{if } t \le 50 \\ 5, & \text{if } t > 50 \end{cases}$$

열출레스키와 혼합 초마디 멀티프런탈 방법(방법 4)을 비교해 보면 총시간 면에서 혼합 초마디 멀티프런탈 방법이 우수한 것으로 나타났다. 총 25개의 문제중 16문제에서 혼합 초마디 멀티프런탈 방법(방법 4)이 열출레스키 방법보다 약 5~10% 정도 수행시간이 작은 것으로 나타났다. 그러나, 대체로 문제 크기가 작고 희소한 행렬 또는 프런탈 행렬의 크기가 비교적 작은 문제에서는 열출레스키 방법이 보다 우수한 것으로 나타났고, 문제 크기가 크고 밀집도가 높은 문제 또는 프런탈 행렬의 크기가 비교적 큰 문제에서는 혼합 초마디 멀티프런탈 방법이 우수한 것으로 나타났다.

6. 결 론

멀티프런탈 방법은 기존의 열·부분행렬출레스키 방법 등과 더불어 출레스키인자의 L를 구하는데 사용될 수 있는 방법으로서 본 연구에서는 초마디를 이용한 초마디 멀티프런탈 방법의 효율적 구현 방법을 제안하였다.

초마디 멀티프런탈 방법에서 삭제나무의 보관과 초마디 보관 방법을 알아보았고, 또한 초마디 멀티프런탈 방법을 구현할 때 수행속도에 큰 영향을 주는 캐쉬효과를 최대화하기 위해 혼합 초마디 멀티프런탈 방법을 제안하였다.

실험에 의하면 혼합 멀티프런탈 방법은 기존의 열출레스키 방법에 대등하거나 문제 크기가 크고 밀집도가 높은 문제 또는 프런탈 행렬의 크기가 비교적 큰 문제에서는 열출레스키 방법보다 약 5~10%정도 수행시간이 개선되었다.

1) http://orlab.snu.ac.kr/software/.

<표 1>에서 NSWITCH에 대한 설정은 대체로 방법 2는 항상 NSWITCH = 5로 설정하는 것이고 방법 4는 방법 2와 같으나 프런탈 행렬의 크기가 10 이하인 경우에는 NSWITCH = 7로 설정하는 방법이다. 따라서 NSWITCH를 대체로 5~7정도로 설정하는 것이 가장 좋은 것으로 나타났다.

참 고 문 헌

[1] 김병규, 박순달, "내부점선형계획법에서의 멀티프런탈방법에 대하여", 「경영과학회」, '95 추계학술대회논문집, 1995, pp.370-380.

[2] 박순달, 「선형계획법」, 민영사, 제3판, 1992.

[3] 박찬규, 박순달, "하한을 이용한 효율적인 최소차수순서화", 「한국경영과학회지」, 제23권 제4호(1998), pp.21-31.

[4] 설동렬, 박찬규, 박순달, "클릭저장구조에서 최소부족수 순서화의 효율화", 「대한산업공학회지」, 제24권 제3호(1998. 9), pp.407-416.

[5] 설동렬, 정호원, 박순달, "내부점 방법을 위한 열출레스키 및 멀티프런탈 방법의 초마디 완화", 한국경영과학회/산업공학회 「'97 춘계 공동학술대회 논문집」, 1997, pp.414-417.

[6] Ashcraft, C., and Roger Grimes, "The influence of relaxed supernode partitions on the multifrontal method," *ACM Transactions on Mathematical Software*, Vol.15, No.4(1989), pp.291-309.

[7] Duff, I.S., and J. K., Reid, "The multifrontal solution of indefinite sparse symmetric linear equations," *ACM Transactions on Mathematical Software*, Vol.9, No.3(1983), pp.302-325.

[8] Duff, I.S., A.M., Erisman, J.K., Reid, *Direct methods for sparse matrices*, Clarendon Press, Oxford, 1986.

[9] George, A., Joseph W.H., Liu, *Computer solution of large sparse positive definite sys-*

tems, Prentice-Hall, 1981.

[10] George, Alan, Joseph W.H., Liu, "The evolution of the minimum degree ordering algorithm," *SIAM Review*, Vol.31, No.1(1989), pp.1-19.

[11] Jung, Ho-Won, R.E., Marsten, M.J., Saltzman, "Numerical factorization methods for interior point algorithms," *ORSA Journal on Computing*, Vol.6, No.1(1994), pp.94-104.

[12] Liu, Joseph W.H., "Multifrontal method and paging in sparse Cholesky factorization," *ACM Transactions on Mathematical Software*, Vol.15, No.4(1989), pp.310-325.

[13] Liu, Joseph W.H., "The role of elimination trees in sparse factorization," *SIAM Journal on Matrix Analysis and Applications*, Vol.11, No.1(1990), pp.134-172.

[14] Liu, Joseph W.H., "The multifrontal method for sparse matrix solution : Theory and practice," *SIAM Review*, Vol.34, No.1(1992), pp. 82-109.

[15] Park, Soondal, Woo-Je Kim, Myeongki Seong, Chan-Kyoo Park, "LPABO : A program for interior point methods for linear programming," *Asia-Pacific Journal of Operational Research*, 17(2000), pp.81-100.