經營科學
第21卷 第2號
2004年 11月

자동조절자 내부점 방법을 위한 선형방정식 해법*

설 동 렬**

Computational Experience of Linear Equation Solvers for Self-Regular Interior-Point Methods*

Tongryeol Seol**

■ Abstract ■

Every iteration of interior-point methods of large scale optimization requires computing at least one orthogonal projection. In the practice, symmetric variants of the Gaussian elimination such as Cholesky factorization are accepted as the most efficient and sufficiently stable method. In this paper several specific implementation issues of the symmetric factorization that can be applied for solving such equations are discussed. The code called McSML being the result of this work is shown to produce comparably sparse factors as another implementations in the MATLAB*** environment. It has been used for computing projections in an efficient implementation of self-regular based interior-point methods, McIPM. Although primary aim of developing McSML was to embed it into an interior-point methods optimizer, the code may equally well be used to solve general large sparse systems arising in different applications.

Keyword : Self-regular Interior-point Methods, Orthogonal Projections, Sparse Systems, Symmetric Factorization

1. 개 요

내부점방법(interior-point methods)은 대형선형계획법(large-scale linear programming) 문제에 대한 매우 강력한 도구로 인정받고 있으며, 그 적용범위를 이차계획법(quadratic programming)이나

논문접수일 : 2004년 4월 25일 논문게재확정일 : 2004년 8월 16일
* 이 논문은 한국과학재단의 해외 Post-Doc. 연수지원에 의하여 연구되었음.
** LG CNS
*** MATLAB은 The Math Works, Inc.의 등록된 상표임.

준정치계획법(semi-definite programming)과 같은 응용분야가 많은 비선형계획법 문제에까지 넓혀가고 있다[2]. 본 논문에서는 다음과 같은 선형계획법 문제 (LP)를 풀 수 있는 일반적인 내부점방법 알고리즘을 고려한다.

$$\begin{aligned} \min\ & c^T x \\ s.t.\ & Ax = b, \\ & 0 \le x_i \le u_i,\ i \in I, \\ & 0 \le x_j,\quad j \in J. \end{aligned} \tag{LP}$$

단, $A \in R^{m \times n}$, $c, x \in R^n$, $b \in R^m$, $u_i \in R$ 그리고 I와 J는 $I \cup J = \{1,2,\cdots,n\}$이고 $I \cap J = \emptyset$인 지수집합이다. 식 (LP)는 다음과 같이 표현하여도 일반성을 잃지 않는다.

$$\begin{aligned} \min\ & c^T x \\ s.t.\ & Ax = b, \\ & Fx + s = u, \\ & x,\ s \ge 0. \end{aligned} \tag{P}$$

단, $F \in R^{m_f \times n}$, $s \in R^{m_f}$, $u = (u_1, u_2, \cdots, u_{m_f})^T$이고 $m_f = |I|$이다. 여기에서 F는 각 행이 단위벡터이다. 또한, 식 (P)의 쌍대문제 (D)는 다음과 같다.

$$\begin{aligned} \max\ & b^T y - u^T w \\ s.t.\ & A^T y - F^T w + z = c \\ & w, z \ge 0 \end{aligned} \tag{D}$$

단, $y \in R^m$, $w \in R^{m_f}$ 그리고 $z \in R^n$이다.

내부점방법은 초기해에서 시작하여 매회 현재의 해를 개선할 수 있는 개선방향을 구하여 이를 따라 해를 수정하는 과정을 최적조건에 이를 때까지 반복하는 방법이다[2]. 따라서, 내부점방법에 기반을 둔 선형계획법 해법 코드는 다음과 같은 형태의 선형방정식을 풀어야 한다는 공통적인 특징을 가진다[6].

$$\begin{pmatrix} -\theta^{-1} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} f \\ h \end{pmatrix} \tag{AS}$$

단, $\theta = (X^{-1}Z + F^T W S^{-1} F)^{-1}$, $\Delta y, h \in R^m$, $\Delta x, f \in R^n$이며, X, Z, W, S는 각각 x_j, z_j, w_i, s_i을 대각요소로 가지는 대각행렬이다. 여기에서 h, f는 내부점방법 알고리즘에 따라 다른 값을 가지며, 내부점방법의 매 회마다 θ의 값이 변화하고 행렬의 구조는 그대로 유지된다. 식 (AS)은 첨가방정식(augmented system)[17]이라고 불리는데 개선방향식을 얻기 위한 뉴턴 방정식으로부터 유도되며, 내부점방법으로 문제를 푸는 시간의 60~90%가 이 방정식을 푸는 데에 소요되기 때문에 선형방정식 해의 정확성뿐만 아니라 계산속도도 구현에 있어서 중요한 고려요소가 된다[15, 18]. 선형방정식을 푸는 방법으로는 식 (3)을 그대로 풀어내는 첨가방정식 방법과 함께 다음과 같이 Δx를 소거한 정규방정식(normal equation)을 푸는 방법이 사용되고 있다.

$$A\theta A^T \Delta y = A\theta f + h \tag{NE}$$

여기서 $A\theta A^T$가 대칭양정치(symmetric and positive definite)의 성질을 가지기 때문에 수치적 안정성에 대한 고려 없이 희소성만을 고려해서 계산할 수 있으므로 촐레스키 분해(Cholesky factorization)가 가장 널리 사용된다[4, 15, 18, 25]. 최소차수 순서화와 같이 분해 된 행렬의 비영요소 개수를 줄여주는 소거 순서를 찾는 기법을 미리 적용하여 적은 계산량을 가지도록 자료구조를 고정해 놓을 수 있어서 행렬의 희소성을 활용한 고속 계산이 가능하다[8, 13].

A 행렬에 밀집열(dense columns)이나 밀집행(dense rows)이 존재하면 $A\theta A^T$가 희소성을 잃기 때문에 정규방정식 방법을 사용하면 희소성을 활용한 고속 계산이 불가능하게 된다[3]. 이러한 경우에는 Sherman-Morrison 공식을 이용하여 A 행렬을 분할하여 암묵적으로 정규방정식 방법을 적용하거나, 정규방정식 방법 대신 첨가방정식 방법을 사용하는 등의 기법이 이용된다[18]. 첨가방정식은 정규방정식이 가지는 대칭양정치의 성질을 갖지 않기 때문에 첨가방정식 방법을 사용할 때에는 정규방정식과 같이 희소성을 활용하는 것뿐만 아

니라 수치적 안정성을 함께 고려해야 한다[11].

본 연구에서 구현된 선형방정식 해법 McSML (McMaster Sparse Matrix Library)은 MATLAB 기반에서 자동조절자(self-regular) 내부점방법[23]을 구현한 McIPM[29]에 적용되었다. McIPM은 선형계획법 문제를 위한 코드로 개발이 시작되었고, 이차계획법, 이차원추계획법(second-order conic programming)이 개발된 상태이며 준정치계획법으로 확장되고 있다. 선형계획법에서는 정규방정식 방법이 널리 사용되고 있으나 이차계획법이나 이차원추계획법에서는 첨가방정식 방법이 주로 사용되기 때문에 McSML에서는 정규방정식 방법과 첨가방정식 방법을 모두 구현하였다. 정규방정식 방법에서는 촐레스키 분해가 구현되었다. 행렬분해계산 중에 발생되는 추가요소(fill-in)의 개수를 최소화하여 계산에 고려되는 비영요소 전체 개수를 줄이는 것이 촐레스키 분해 구현의 핵심이다. McSML에서는 마코비츠 전략(Markowitz strategy)의 대칭형태인 최소차수순서화(minimum degree ordering)를 사용하였으며, 순서화 계산 속도와 품질 향상을 도모하기 위하여 외부차수(external degree), 다중소거(multiple elimination) 등의 기법이 적용되었다[16]. 첨가방정식 방법에서는 첨가방정식이 대칭행렬이기는 하지만 양정치가 아니기 때문에 조정(regularization) 기법[5, 19, 24]을 적용한 LDL^T 분해를 구현하였다. 정규방정식 방법의 구현에 사용된 대부분의 기법이 동일하게 적용되었으며, 방정식의 해가 높은 수치오차를 가지는 경우에 반복정제(iterative refinement)[5]를 적용하였다.

본 논문은 다음과 같이 구성되었다. 2장에서는 자동조절자 내부점방법에서 풀어야 하는 선형방정식의 특성에 대해서 논의하며, 3장에서는 내부점방법을 위한 선형방정식 해법을 구현하는 데에 있어서 고려할 이슈들과 구현된 기법들에 대하여 설명한다. 4장에서는 구현된 해법을 실제로 컴퓨터상에서 적용한 결과에 대해 토의하고 마지막으로 5장에서 본 연구를 통해 얻은 결론을 제시한다.

2. 자동조절자 내부점방법과 선형방정식

자동조절자 내부점방법은 이제까지 개발된 내부점방법 가운데 이론적으로 최악의 경우에 대한 복잡도가 가장 우수한 알고리즘이다[23]. 자동조절자 내부점방법은 기존의 원쌍대 내부점방법(primal-dual path-following IPMs)과 같은 구조로 되어 있지만, 새로운 proximity와 개선방향을 사용한다. 기존의 원쌍대내부점방법은 로그함수를 장벽함수(barrier function)로 사용하여 해가 가능해공간의 변두리에 가까이 오지 못하여 중심경로를 따라 효율적으로 최적해에 접근해 가도록 한다. 자동조절자 내부점방법은 기존의 장벽함수 개념을 확장하여 자동조절 근접(self-regular proximities)을 사용한다. 로그장벽함수도 자동조절 함수류에 속하지만, 자동조절자 내부점방법에서는 보다 더 중심경로에 해가 근접할 수 있게 하는 자동조절자 함수를 사용한다.

자동조절자 내부점방법에서 매 회 다음 뉴튼시스템을 풀어 개선방향을 얻는다.

$$\begin{pmatrix} -\theta^{-1} & A^T & F^TWS^{-1}u-c \\ A & 0 & -b \\ -u^TWS^{-1}F-c^T & b^T & x/\tau+u^TWS^{-1}u \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\tau \end{pmatrix} = \begin{pmatrix} f \\ h' \\ g' \end{pmatrix} \qquad \text{(SR)}$$

식 (SR)에 대한 정규방정식은 다음과 같이 유도된다.

$$[A\theta A^T+\bar{a}\hat{a}^T]\Delta y = \hat{r} \qquad \text{(SRNE)}$$

단,

$$\bar{a}=\frac{-b-A\theta(c-F^TWS^{-1}u)}{x/\tau+u^TWS^{-1}u+(c^T+u^TWS^{-1}F)\theta(c-F^TWS^{-1}u)},$$

$$\hat{a}=-b^T+(c^T+u^TWS^{-1}F)\theta A^T$$

이다.

식 (SRNE)의 좌변에 있는 행렬은 일차계수갱신(rank-one update)으로 인해 대칭이나 반대칭의 성질을 잃게 되었지만, Sherman-Morrison 공식을 사

용하여 계산량을 줄일 수 있다[29]. 즉, 식 (SRNE)은 $P=A\theta A^T$라고 할 때 다음과 같이 계산된다.

$$\Delta y = P^{-1}\hat{r} - P^{-1}\bar{a}(I+\hat{a}^T P^{-1}\bar{a})^{-1}\hat{a}^T P^{-1}\hat{r}.$$

Δy를 계산하기 위하여 다음 절차를 따른다.

단계 1. $(A\theta A^T)x_0 = \hat{r}$를 계산한다.
단계 2. $(A\theta A^T)x_1 = \bar{a}$를 계산한다.
단계 3. $\Delta y = \dfrac{x_0 + x_0\hat{a}^T x_1 - \hat{a}^T x_0 x_1}{1+\hat{a}^T x_1}$ 를 계산한다.

여기에서 θ가 대각행렬이기 때문에 θ의 값과 무관하게 매 회 $A\theta A^T$행렬의 비영요소구조는 동일하게 유지된다. 따라서, 내부점방법 알고리즘을 적용하기 이전에 순서화 및 상징적 분해를 통하여 수치분해를 위한 자료구조를 미리 결정해 놓을 수 있으며, 단계 1과 단계 2의 $A\theta A^T$행렬이 동일하기 때문에 매 회마다 수치분해는 한 번만 수행하고 치환연산을 서로 다른 우변상수에 대해 두 번씩 수행하면 된다.

첨가방정식 방법으로 식 (SR)을 풀기 위해서도 정규방정식 방법과 마찬가지로 일차계수갱신에 의해 잃어버린 대칭성을 해결하는 과정이 필요하다. 편의를 위하여 식 (SR)을 다음과 같이 표현하자.

$$\begin{pmatrix} M & \bar{b} \\ \hat{b}^T & d \end{pmatrix}\begin{pmatrix} \Delta\xi \\ \Delta\tau \end{pmatrix} = \begin{pmatrix} r_{xy} \\ g' \end{pmatrix}$$

단,

$$M=\begin{pmatrix} -\theta^{-1} & A^T \\ A & 0 \end{pmatrix},\quad \bar{b}=\begin{pmatrix} F^T S^{-1} W u - c \\ -b \end{pmatrix},$$

$$\hat{b}^T = (-u^T S^{-1} W F - c^T, b^T),$$

$$d = u^T S^{-1} W u + \frac{\chi}{\tau},\quad r_{xy}=\begin{pmatrix} f' \\ h' \end{pmatrix},\quad \Delta\xi=\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

이다.

위의 식에서 $\Delta\tau$를 제거하면

$$\left(M - \frac{1}{d}\bar{b}\hat{b}^T\right)\Delta\xi = r_{xy} - \frac{g'}{d}\bar{b}$$

을 얻는다. 정규방정식 방법에서와 같이 Sherman-Morrison 공식을 적용하여 해를 얻을 수 있다. 그러나, 첨가방정식은 정규방정식과 달리 양정치의 성질을 가지지 않기 때문에 촐레스키 분해를 적용할 수 없어서 대칭분해 방법인 LDL^T 분해를 대신 적용한다. 유사정치성(quasi-definiteness)을 이용하면 정규방정식 방법에서 촐레스키 분해를 적용하는 경우와 마찬가지로 순서화 및 상징적 분해를 이용하여 희소성을 활용한 고속계산이 가능하다[26, 27].

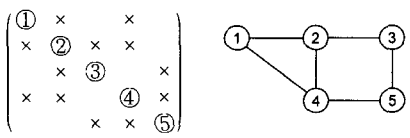
3. 내부점방법을 위한 선형방정식 해법의 구현 기법

3.1 정규방정식 해법의 구현

정규방정식 해법은 양정치성을 갖고 있기 때문에 수치적인 고려 없이 임의의 대각요소를 선회요소로 선택하더라도 수치적인 안정성이 유지된다. 따라서, 선회연산을 통하여 추가되는 비영요소 개수를 최소로 하는 선회순서를 찾는 데에 주력할 수 있다. 이와 같은 선회순서를 찾는 과정을 순서화라고 하며 일반적으로 마코비츠 전략의 대칭형태인 최소 차수 순서화가 널리 사용되고 있으며[8, 13] McSML에서도 최소 차수 순서화가 구현되었다. 내부점방법에서는 매 회 풀어야 하는 정규방정식의 좌변 행렬이 동일한 비영요소 구조를 가지기 때문에 순서화 계산은 내부점방법 초기에 한 번만 수행하게 된다[6].

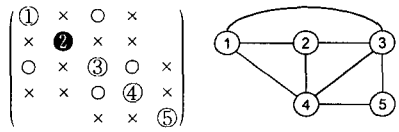
행렬의 분해 과정은 그래프를 통해서 표현할 수 있으며, 실제 순서화를 구현할 때에는 일반적으로 그래프 구조를 활용한다. 대각요소들은 그래프 상의 점으로, 비영요소들은 점과 점을 잇는 선으로 대응된다. 예를 들어, (1, 2) 요소가 비영이라면 그래프 상에 점 1과 점 2를 잇는 선이 대응된다. 이러한 그래프를 삭제그래프라고 한다[13]. 삭제그래프 상에서의 선회연산은 해당되는 선회점을 제거하고, 선회점에 이웃한 점들을 모두 연결시켜 주는 과정

으로 표현된다. 이 때에 원래 연결되지 않았던 점들이 연결되는 경우가 발생할 수 있는데 이것이 바로 비영요소가 추가되는 것이다.



〈그림 1〉 삭제그래프(× : 비영요소)

최소 차수 순서화의 기본 아이디어는 매 회 가장 비영요소를 적게 발생시킬 대각요소를 선회요소로 결정하는 것이다. 차수는 각 대각요소에 대해 해당 열(또는 행)의 비영요소 개수인데, 만일 어떤 대각요소가 선회요소로 선택되어지면 선회연산을 통해 해당되는 열의 비영요소가 있는 행(그리고 해당되는 행의 비영요소가 있는 열)에 대해서 계산이 이루어지므로 원래는 비영요소가 아니었어도 계산을 통해 비영의 값을 가지게 될 가능성이 생긴다. 따라서, 최소 차수를 가지는 대각요소를 선회요소로 선택하면 선회연산을 통해 가장 작은 수의 비영요소가 발생할 것을 기대할 수 있다. 차수는 삭제그래프 상에서 이웃한 점들의 개수와 같다. 따라서, 최소 차수 순서화는 점이 삭제될 때에 선을 잇게 되는 이웃점이 가장 적은 것으로 선회점을 삼는 것이다.



〈그림 2〉 ②가 선회요소가 되는 경우(× : 비영요소, ○ : 추가요소)

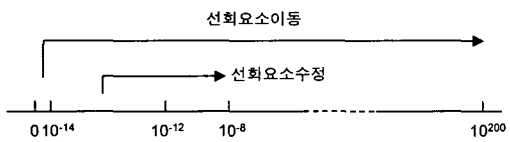
최소 차수 순서화는 비영요소에 대응되는 모든 지점에서 추가요소가 발생할 것으로 가정하기 때문에 <그림 2>에서처럼 이미 비영요소가 존재하는 (1, 4)와 같은 경우를 고려하고 있지 못하다. 이미 존재하는 비영요소까지 함께 고려하여 실제로 추가되는 요소의 개수로 선회순서를 결정하는 최소 부족수 순서화(minimum deficiency ordering 또는 minimum local fill-in ordering)를 사용하는 것이 가능하기는 하지만 이미 존재하는 비영요소 수를 고려하게 되면 순서화에 소요되는 계산량이 지나치게 늘어나 전체적인 효율성이 떨어지게 된다. 따라서, 정확한 부족수를 계산하는 대신 근사치를 사용하여 추가되는 비영요소 개수도 줄이면서 빠른 순서화 계산이 가능하도록 하는 방법이 필요하다. 외부 차수가 이러한 목적에서 사용된다. 순서화 계산의 효율성을 위하여 비영요소구조가 동일한 점들(구별불능점, indistinguishable nodes)을 하나의 점으로 묶어서 처리하는 기법이 사용되는데, 구별불능점 사이에는 이미 선이 연결되어 있다는 사실을 이용하면, 차수에서 구별불능점의 개수를 뺀 부족수의 근사치(외부 차수, external degree)를 쉽게 계산할 수 있다[16]. 최소 차수 순서화를 통해 얻어지는 선회순서가 발생시키는 비영요소수를 최소화하기 위하여 McSML에서는 외부 차수(external degree)를 선회요소 선택기준으로 사용하였다.

순서화를 효율적으로 구현하는 데에 있어서 차수 계산과 함께 고려해야 할 요소로 선회연산을 어떻게 처리할 것인가 하는 문제가 있다. 선회연산 처리는 삭제그래프를 관리할 자료구조에 좌우된다. 정규방정식 방법에서는 일반적으로 사용되는 쿼션트 그래프(quotient graph) 자료구조보다 A행렬의 각 열이 $A\theta A^T$에서의 비영요소를 나타내는 클릭(clique)이라는 성질을 이용한 클릭 자료구조가 효율적인 것으로 알려져 있다. 특히, 선회연산이 선회요소를 포함한 클릭들을 하나로 병합하는 것으로 처리되기 때문에 매우 신속한 계산이 가능하다[15, 22].

희소행렬은 비영요소들만 고려해서 계산하기 때문에 자료구조 상으로도 비영요소들의 값만을 보관하고 해당되는 행지수들은 따로 유지해서 필요할 때마다 지수배열을 통해 간접어드레싱(indirect addressing)으로 원하는 값을 찾는다. 이를 통해서

대형문제를 컴퓨터로 계산하는 것이 가능해지기는 하였으나 직접어드레싱(direct addressing)에서 활용할 수 있는 루프언롤링(loop-unrolling) 같은 기법을 사용할 수 없고 행렬분해 후반부에서 밀집도가 높아진 경우에는 오히려 지수배열들 때문에 계산효율이 떨어지는 경우가 발생하는 등의 단점을 갖게 된다. 희소행렬분해에서 직접어드레싱의 장점을 이용하기 위해 초마디(supernode) 기법을 사용한다[4]. 초마디 기법은 열 단위로 계산하는 열 촐레스키(column Cholesky) 방법이나 멀티프런탈(multifrontal)방법에서 사용되는데, 비영요소구조가 같은 연속된 열들을 하나의 초마디로 묶어서 초마디 내부의 계산은 직접어드레싱으로 처리하는 방법이다. McSML은 열촐레스키분해를 사용하며 초마디 기법을 구현하여 수치분해과정과 치환연산과정에 루프언롤링을 적용하였으며, 초마디에 속한 열들의 중복되는 행지수를 압축하여 보관하는 자료구조를 사용하였다.

이론적으로는 양정치의 행렬은 수치적인 안정성이 보장되지만, 컴퓨터상에서는 누적된 계산오차에 따른 수치적인 문제가 발생할 수 있으며[25], 원문제의 행렬 A가 정칙행렬이 아닐 경우도 있을 수 있다. 특히, 내부점방법의 알고리즘 후반부에서 상보조건에 의해 θ의 값들의 범위가 오차 허용수준을 넘을 수 있다[20]. 예를 들어, x_j가 0에 가까워지고 z_j가 0이 아닌 값을 가지게 되면 $\theta_j = x_j/z_j$이 0에 가까운 작은 값을 갖게 되어 행렬분해의 수치적 안정성에 영향을 주게 된다. McSML에서는 선회요소이동(pivot shifting) 및 선회요소수정(pivot boosting)을 적용하였다[29].



〈그림 3〉 선회요소이동 및 선회요소수정

선회요소의 값이 10^{-14} 보다 작거나 같으면 값을 10^{200}으로 두어 해당 열을 계산에서 제외시키고, 선회요소의 값이 10^{-12} 보다 작거나 같으면 값을 10^{-8}으로 상승시켜서 오차발생의 가능성을 줄이도록 하였다. 즉, 0에 매우 가까운 값을 가지는 경우에는 값을 0으로 가정하고 계산에서 제외시켜 지나치게 작은 값으로 선회연산을 수행했을 때에 발생할 수 있는 오차를 피하도록 하고, 0으로 간주할 만큼 작은 값은 아니지만 작은 값으로 인해 발생할 수 있는 오차가 클 것으로 판단될 경우에는 그 값을 일정수준 이상의 큰 값을 가지도록 수정하는 것이다.

3.2 첨가방정식 해법의 구현

첨가방정식은 좌변의 행렬이 대칭행렬이기는 하지만 양정치행렬은 아니기 때문에 원칙적으로는 정규방정식에서처럼 순서화 및 상징적 분해 과정을 수치분해 과정과 분리해서 수행할 수 없다. 첨가방정식과 같은 부정방정식(indefinite equation)에 대한 전통적인 해법은 Bunch-Parlett 전략과 같이 2×2 선회연산을 이용하는 것이다[10]. Bunch-Parlett 전략이나 이를 발전시킨 Bunch-Kaufman 전략 모두 희소행렬이 아닌 밀집행렬을 가정하고 개발된 방법이기는 하지만, 멀티프런탈(Multifrontal) 방법에 기반을 두어 희소성을 어느 정도 활용할 수 있는 코드들도 개발되어 있다[9, 11]. 그러나, 수치적인 고려를 통해 선회요소를 결정하고 또한 2×2 선회연산을 수행하는 것은 많이 비용이 소요되기 때문에 고속계산을 위하여 양정치행렬에 대해서 적용되었던 희소행렬기법을 첨가방정식에도 그대로 적용하고자 하는 연구가 내부점방법의 구현에 관련된 연구자들 사이에서 활발히 이루어졌다. 양정치의 성질을 가지지 않더라도 행렬이 다음과 같은 유사정치일 경우에는 양정치행렬과 마찬가지로 수치적 안정성에 대한 고려 없이 선회순서를 임의로 결정할 수 있는데, 이 성질을 이용하면 첨가방정식 해법의 효율적인 구현이 가능하다.

$$Q=\begin{pmatrix} -D & A^T \\ A & E \end{pmatrix}$$

단, D와 E는 대칭양정치 행렬이고 A는 완전계수(full rank)이다. 유사정치행렬은 대각요소가 양이 아니기 때문에 대각요소에 대한 제곱근 계산이 필요한 촐레스키 분해는 사용할 수 없고 대신 대각요소를 따로 보관하는 LDL^T분해를 사용할 수 있다[8, 17]. LDL^T분해는 구현의 관점에서는 촐레스키 분해에 적용가능한 대부분의 희소행렬기법들을 똑같이 사용할 수 있다. 첨가방정식은 Q 행렬에서 θ가 양정치의 대각행렬이고 E가 0인 경우이므로, E가 양의 값을 가지도록 섭동시켜 줌으로써 수치적인 문제에서 벗어날 수 있다. 이와 같은 기법을 조정(regularization)기법이라고 하는데, 다음과 같은 원쌍대 조정방법이 효과적인 것으로 알려져 있다[5].

$$\begin{pmatrix} -\theta^{-1} & A^T \\ A & 0 \end{pmatrix} \rightarrow \begin{pmatrix} -\theta^{-1} & A^T \\ A & 0 \end{pmatrix}+\begin{pmatrix} -R_p & 0 \\ 0 & R_d \end{pmatrix}$$

단, R_p와 R_d는 모든 대각요소가 비음인 대각행렬이다. R_p와 R_d가 가지는 값의 크기는 조정방법에 따라 서로 다르지만, 조정방법의 기본 아이디어는 첨가방정식의 2사분면은 보다 더 음정치(negative definite)의 성질을 갖게 하고 4사분면은 보다 더 양정치의 성질을 갖도록 섭동시켜서 결과적으로 첨가방정식이 유사정치가 되도록 하는 것이다. R_p와 R_d의 값이 커지면 유사정치의 성질은 더욱 분명해지지만 내부점방법의 개선방향에 그만큼 왜곡되기 때문에 조정의 정도는 작은 수준으로 유지한다. McSML에서는 R_p와 R_d을 일괄적으로 모든 요소에 대해 적용하지 않고 대각요소의 값이 0에 가까운 값을 가지는 경우에만 적용하도록 하였다. 조정을 적용하는 조건은 다음과 같다.

$$d_{jj} < d_{\max} \times (r_d)^2$$

단, d_{jj}는 j번째 선회요소, $d_{\max}$는 Q의 대각요소 가운데 절대값이 가장 큰 요소의 값 그리고 r_d는 R_d에 적용되는 조정값이며 McSML에서는 실험을 통하여 10^{-9}로 결정되었다.

첨가방정식에 조정방법을 적용하여 수치적 안정성에 대한 부담을 덜어버리고 희소성을 십분 활용하는 것이 가능하지만, 개선방향의 왜곡을 최소화하기 위해서는 조정수준을 줄일 수 있는 선회순서를 이용하는 것이 적절하다. 대각요소의 값이 비영인지 여부를 상징적으로 관리하면서 비영인 경우에 한해서만 선회요소로 선택하도록 하는 수정된 삭제그래프를 이용한 최소 차수 순서화를 적용하면 희소성을 유지하면서 동시에 조정이 필요한 선회요소를 줄일 수 있다[1]. 그러나, 상징적인 차원에서만 대각요소의 값을 관리하기 때문에 순서화 과정 중에 비영으로 판단했을지라도 실제로 가지는 값이 0 또는 0에 매우 가까운 작은 값을 가질 수 있는 가능성이 있다. McSML에서는 먼저 일방절단(one-way dissection) 순서화로 Q 행렬의 2사분면과 4사분면을 분리한 다음 각각의 블록에 대해 최소 차수 순서화를 적용하도록 하는 복합적인 순서화를 적용하였다. 선형계획법을 위한 내부점방법에서의 첨가방정식에서는 2사분면에 대한 최소 차수 순서화는 A 행렬의 열들을 비영요소개수가 작은 순서로 정렬하는 것으로 간단히 계산된다. 4사분면에 대한 최소 차수 순서화는 정규방정식에 대한 최소 차수 순서화와 동일하다. Q 행렬에 이와 같이 일방절단 순서화를 적용하는 것은 비영인 2사분면 블록을 먼저 계산함으로써 4사분면을 정규방정식의 행렬로 만들게 되어 4사분면의 대각요소들이 비영의 값을 갖도록 하는 결과를 가져오기 때문에 조정이 필요한 요소를 최소화할 수 있다. 다만, A 행렬에 밀집열이 존재하는 경우에는 4사분면의 블록을 밀집행렬로 만드는 효과가 있기 때문에 가장 마지막 선회순서가 되도록 하였다.

3.3 반복정제(Iterative Refinement)의 구현

이론적으로는 수치적 안정성이 보장되더라도 전산기에서 구현되었을 때에는 계산오차가 발생하는

경우가 있다. 특히, 내부점방법에서는 알고리즘의 마지막 단계에 가까와질수록 상보조건에 의하여 θ가 0에 가까운 값에서부터 매우 큰 값까지 유효자리수를 벗어나는 매우 넓은 범위의 값들을 가질 수 있기 때문에 행렬의 분해와 치환연산에 따른 계산오차가 발생한다. McSML에서는 정규방정식 방법에 대해서는 선조절공액경사법(preconditioned conjugate gradient method)을 통해 반복정제를 수행하였다. 선조절자로는 이미 계산되어 있는 촐레스키 분해요소를 사용하였다. 한편, 첨가방정식 방법에서는 양정치의 성질을 가지지 않기 때문에 공액경사법에 따른 해의 수렴을 보장할 수 없어서 별도의 반복정제 방법이 필요하다. 다음과 같은 선형방정식 $Mx=b$을 푼다고 가정하자. 계산오차를 Δx라고 하면, 선형방정식 해법에 의하여 얻은 해 x는 $M(x+\Delta x)=b$을 만족한다. 따라서, 다음과 같이 Δx를 계산하여 x를 정제할 수 있다.

$$\Delta x = M^{-1}(b - Mx)$$

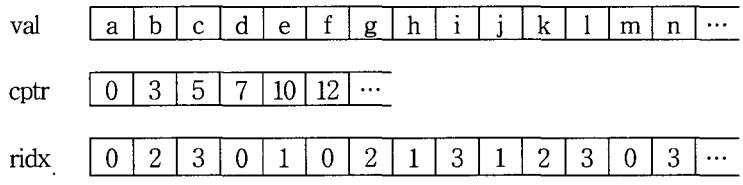
M^{-1}에 대한 부분은 원래 선형방정식 $Mx=b$을 풀 때에 계산된 분해요소($LDL^T=M$)를 이용하여 치환연산을 통해 계산할 수 있다. 이 과정을 계산오차가 허용 가능한 범위에 들어올 때까지 반복한다. 실제로 반복정제는 내부점방법의 마지막 2~3회에서 주로 실행되며, 반복정제의 반복회수도 1~2회 정도만으로 충분했다.

3.4 프로그램의 구조

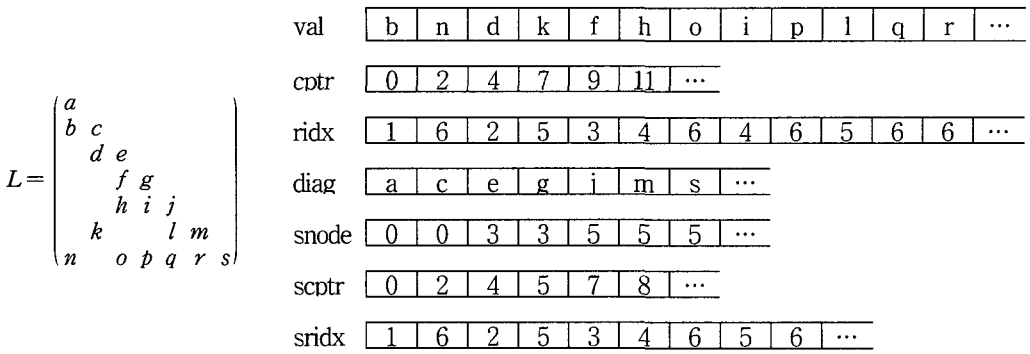
McSML은 정규방정식과 첨가방정식을 푸는 10개의 프로그램으로 구성되어 있다. C 언어로 작성되었기 때문에 유닉스나 윈도우즈 환경에서 대부분 사용가능하다. McSML은 McIPM의 선형방정식 해법으로 사용하기 위해 개발되었기 때문에 McIPM과 마찬가지로 MATLAB 환경에서 구동된다. McSML의 정규방정식 해법은 일반적인 대칭양정치 방정식의 해법으로 사용될 수 있으며, 첨가방정식 해법은 일반적인 대칭유사정치 및 부정방정식의 해법으로 사용가능하다.

McSML은 원칙적으로 MATLAB의 희소행렬 자료구조와의 호환성을 유지하도록 제약식 행렬 A나 분해요소 행렬 L의 자료구조를 정의하였다. 행렬 A의 자료구조는 전형적인 희소행렬 자료구조를 따르고 있는 MATLAB의 희소행렬 자료구조[28]와 동일하다. A의 자료구조는 1개의 실수 배열(val), 2개의 정수 배열(cptr, ridx) 그리고 3개의 정수 변수(nrow, ncol, nnzr)로 구성된다. nrow는 A행렬의 행 개수, ncol은 열 개수를 의미하며, nnzr은 A행렬에 있는 비영요소의 개수를 의미한다. val은 A행렬의 비영요소들이 갖는 값을 실제로 보관하는 배열로서 열 단위로 비영요소들이 보관된다. cptr는 각 열들이 val의 어느 위치에서부터 저장되는지 그 시작위치를 보관하는 정수 배열이고, ridx는 val의 각 값들에 대해서 행지수 값을 보관하는 정수 배열이다. McSML은 계산상의 편의를 위하여 행렬 A 뿐만 아니라 A^T를 함께 유지한다. 희소행렬이 열 단위로 저장되기 때문에 행단위의 접근이 필요한 계산에서의 효율성을 제고하기 위하여 행렬 A를 행단위로 저장하게 되는 행렬 A^T를 이용하는 것이다. 다음은 행렬 A가 McSML에서 저장되는 모습을 보여주는 예제이다.

$$A=\begin{pmatrix} a & d & f & & & m \\ & e & & h & j & \\ b & & g & & & k \\ c & & & i & l & n \end{pmatrix}$$



〈그림 4〉 행렬 A의 자료구조 예제



〈그림 5〉 행렬 L의 자료구조 예제

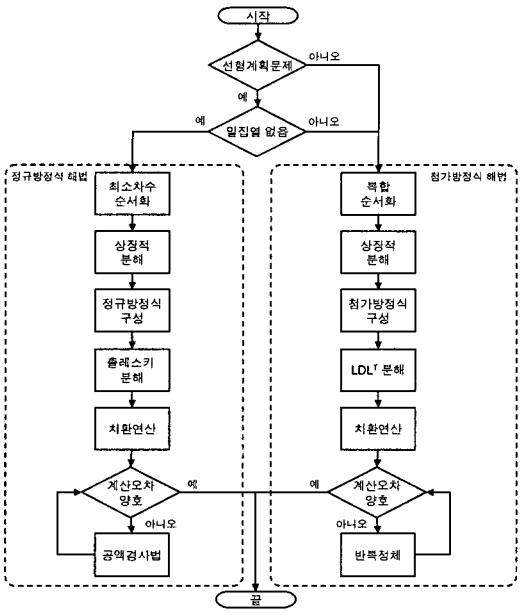
단, 예제에서 McSML이 C 언어로 작성되었기 때문에 배열의 시작주소는 0이며, MATLAB의 자료구조형을 따라 행지수 또한 0부터 시작한다. 따라서, ridx의 값이 0이면 행지수가 1이라는 의미이다.

행렬 L의 자료구조는 행렬 A와 거의 동일하며 다만 L이 정방행렬이기 때문에 ncol을 사용하지 않는다는 점과 L의 대각요소는 계산상의 편의를 위하여 val 배열이 아닌 diag라는 별도의 실수 배열에 저장한다는 점이 다르다. 또한, MATLAB에서 촐레스키 분해요소에 대한 자료형(data type)을 가지고 있지 않기 때문에 McSML에서는 초마디에 관련된 정보는 별도의 구조체를 정의하여 사용하였다. 초마디에 관한 구조체는 3개의 정수 배열(snode, sridx, scptr)로 구성된다. snode는 각 열이 초마디에 속하는지 그리고 어느 초마디에 속하는지 알기 위한 정수 배열이고, sridx와 scptr는 ridx와 cptr를 초마디를 이용한 행지수압축을 적용한 정수 배열들이다. 다음은 행렬 L이 McSML에서 저장되는 모습을 보여주는 예제이다.

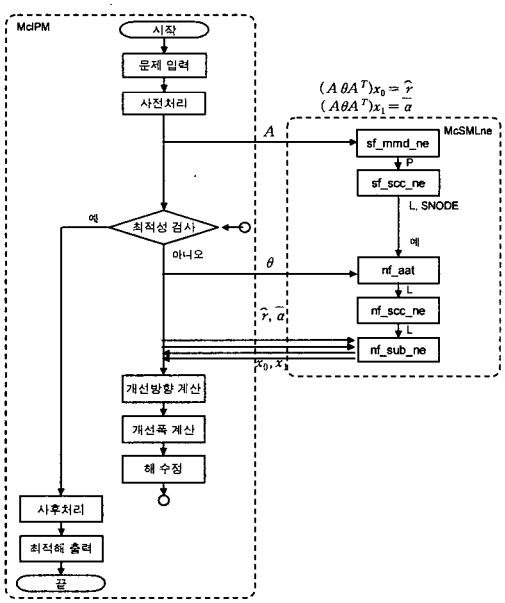
단, 위의 예제에서 보는 바와 같이 초마디를 이용하여 압축된 행지수 배열 scptr, sridx가 생성되기 때문에 ridx는 불필요하기 때문에 제거하는 것이 가능하다. 그러나, MATLAB에서 압축된 행지수 자료구조를 자료형으로 가지고 있지 않기 때문에 호환성을 유지하기 위해 중복을 허용하였다. 한편, 위와 같이 L의 대각요소와 비대각요소를 저장하는 자료구조는 촐레스키 분해를 사용하는 정규방정식 해법뿐만 아니라 LDL^T 분해의 경우에도 동일하게 적용되었다. LDL^T 분해에서 L의 대각요소는 모두 값이 1이므로 따로 보관할 필요가 없기 때문에 L의 대각요소를 저장할 공간에 대각행렬 D를 저장하였다.

McSML은 정규방정식과 첨가방정식을 구성하는 기능을 가지고 있기 때문에 따로 $A\theta A^T$나 첨가행렬을 계산할 필요 없이 A와 θ, 그리고 우변상수만을 입력하면 된다. 다만, A는 열 단위로 저장되어 있는 희소행렬로 가정하며 우변상수는 밀집벡터로 가정한다. <그림 4>에서 보는 바와 같이 선형계획법을 위한 내부점방법일 경우에 A행렬에 밀집열이 존재하지 않으면 정규방정식 해법을 사용하고 나머지의 경우에는 첨가방정식 해법을 사용한다. 비선형계획법을 위한 내부점방법일 경우에는 정규방정식이 희소성을 잃을 뿐만 아니라 정규방정식의 행렬을 구성하는 데에 역행렬 계산이 포함되어 계산량이 많아져서 정규방정식 방법을 선형방정식을 푸는 것이 효율적이지 않기 때문에 첨가방정식 해법을 사용한다.

McSML을 구성하는 프로그램들은 다음과 같다. sf로 시작되는 프로그램은 상징적 계산에 사용되는 프로그램들이고, nf로 시작되는 프로그램은 실제 수치를 이용한 계산에 관련된 프로그램들이다. 정규방정식 해법(McSMLne)에 사용되는 프로그램들은 이름이 ne로 끝나고, 첨가방정식 해법(McSMLas)에 사용되는 프로그램들은 이름이 as로 끝난다.



〈그림 6〉 McSML의 구조



〈그림 7〉 McSML과 McIPM의 자료 교환
(정규방정식 해법의 경우)

- sf_mmd_ne : 정규방정식 해법에서 순서화를 수행한다. 외부차수를 이용한 최소 차수 순서화 방법을 사용하며 자료구조로는 클릭구조를 이용한. A행렬을 입력받아 순서화 계산에 따른 선회순서를 출력한다.
- sf_scc_ne : 정규방정식 해법에서 상징적 분해를 수행한다. A행렬과 선회순서를 입력받아 촐레스키 분해요소 L의 비영요소구조를 계산하여 자료구조를 미리 준비한다. 또한, L의 초마디 구조를 분석하여 수치적 분해에서 활용할 수 있도록 초마디에 대한 정보 및 초마디를 활용한 행지수압축 자료구조를 만든다.
- nf_aat : 정규방정식을 구성한다. A와 θ를 입력받아 $A\theta A^T$를 계산하여 L에 하삼각부분의 값을 저장하여 출력한다.
- nf_scc_ne : 정규방정식 행렬에 대하여 촐레스키 분해를 수행한다. 컴퓨터의 메모리를 경제적으로 사용하기 위하여 대칭행렬($A\theta A^T$)의 하삼각부분을 L의 자료구조로 입력받아 촐레스키 분해결과를 L에 덮어써서 출력한다.
- nf_sub_ne : 촐레스키 분해요소를 이용하여 치환연산으로 정규방정식의 해를 계산한다. 촐레스키 분해요소 L과 정규방정식의 우변상수가 입력되어 방정식의 해가 출력된다. 해의 정밀도가 나쁠 경우에는 공액경사법을 이용한 해의 정제를 수행한다.
- sf_mmd_as : 첨가방정식 해법에서 순서화를 수행한다. A행렬을 입력받아 순서화 계산에 따른 선회순서를 출력한다.
- sf_scc_as : 첨가방정식 해법에서 상징적 분해를 수행한다. A행렬과 선회순서를 입력받아 분해요소 L의 비영요소구조를 계산하여 자료구조를 미리 준비한다. 또한, L의 초마디 구조를 분석하여 수치적 분해에서 활용할 수 있도록 초마디에 대한 정보 및 초마디를 활용한 행지수압축 자료구조를 만든다.
- nf_aug : 첨가방정식을 구성한다. L에 첨가행렬의 하삼각부분을 저장하여 출력한다.
- nf_scc_as : 정규방정식 행렬에 대하여 LDL^T 분해를 수행한다. 첨가행렬의 하삼각부분을 L의 자료구조로 입력받아 LDL^T 분해결과를 L에 덮

어써서 출력한다.

- nf_sub_as : LDL^T 분해요소를 이용하여 치환연산으로 첨가방정식의 해를 계산한다. 해의 정밀도가 나쁠 경우에는 반복정제를 실시한다.

내부점방법에서 선형방정식의 매 회마다 한 차례 이상 계산되지만, 같은 회차 안에서는 선형방정식의 좌변은 동일하고 우변상수만 다르기 때문에 행렬분해 계산은 한번만 수행되고 나머지 경우에는 치환연산만으로 방정식을 풀 수 있다. 또한, 행렬의 비영요소 구조가 고정되어 있기 때문에 선회요소 선택을 수치를 고려하여 동적으로 결정하지 않으면 내부점방법 초기에 분해요소의 자료구조를 결정해 놓으면 이후에 수정 없이 그대로 사용할 수 있다. <그림 5>는 McSML이 자동조절자 내부점 방법 프로그램인 McIPM에서 어떻게 자료를 교환하는지 보여준다. 사전처리를 마친 후 A가 McSML에 넘겨지면 내부점 알고리즘이 시작하기 전에 sf_mmd_ne와 sf_scc_ne가 각각 순서화와 상징적 분해 계산을 수행한다. 매 회마다 값이 달라지는 θ에 따라 nf_aat가 $A\theta A^T$를 계산해서 nf_scc_ne에 넘겨주면 촐레스키 분해를 수행한다. 정규방정식 방법을 사용할 경우에 McIPM은 매 회마다 방정식 (SRNE)을 풀어야 하므로[29] nf_sub_ne를 2회 실행해야 하며, 만일 예측자-수정자(predictor-corrector) 기법을 적용하게 되면 nf_sub_ne는 매 회차마다 4번씩 실행하게 된다.

4. 실험 결과

McSML은 MATLAB 기반으로 구현하였다. 일차적인 이유는 McIPM을 위한 선형방정식 해법 프로그램으로 개발되었기 때문이지만, 한편 MATLAB 환경이 가지고 있는 장점 때문이기도 하다. MATLAB은 수리계산 및 시각화 그리고 공학계산을 위한 유연한 환경을 제공하는 강력한 언어를 통합한 환경을 제공하며, MATLAB의 개방구조(open architecture)는 다양한 응용 소프트웨어의 개발을 용이하게 해주기 때문에 다양하고 편리한 행렬 연산 기능은 수리계획법 알고리즘의 빠른 구현을 가능하게 하여 새로운 아이디어의 신속한 검증과 활용을 가능하게 한다[14]. 실제로 MATLAB은 C나 Fortran으로 작성된 외부 프로그램을 MEX라는 인터페이스를 통하여 내부 프로그램처럼 실행시킬 수 있다. 1990년대 초반 내부점방법에 대한 연구가 활발하게 시작될 시기에 LIPSOL[28]이 MATLAB 기반으로 구현되어 내부점방법을 통하여 대형선형계획문제들을 효율적으로 풀 수 있음을 보여준 바 있으며, 최근에 자동조절자 기반의 내부점방법인 McIPM[29]이 MATLAB 기반으로 개발되어 새로운 알고리즘의 실효성을 확인할 수 있었다.

McSML의 성능 평가를 위하여 MATLAB 기반에서 사용가능한 선형방정식 해법들과의 비교 실험을 수행하였다. 비교 평가에 사용된 프로그램들은 MATLAB에 내장되어 촐레스키 분해를 수행하는 LAPACK의 DPOTRF[7]와 MATLAB 기반의 내부점방법 코드인 LIPSOL에서 선형방정식 해법으로 사용하는 ORNL 희소 촐레스키 패키지[21]이다. ORNL 코드는 Esmond Ng과 Barry Peyton에 의하여 개발되었으며, 순서화는 Joseph Liu의 최소차수 순서화 코드를 이용한다. LAPACK은 MATLAB에 내장된 내부명령으로 실행시킬 수 있고, ORNL 코드는 Fortran으로 작성되어 MEX 인터페이스로 MATLAB에서 사용가능하다. 실험에 사용된 컴퓨터는 IBM RS/6000 44P Model 270 워크스테이션으로 AIX 4.3.3.0 환경에 설치된 MATLAB Release 13에서 실험하였다.

실험에는 선형계획법 코드의 표준 비교평가 문제인 Netlib의 문제들[12]을 사용하였다. 먼저 선형방정식을 1회 계산하는 경우에 대한 성능 평가를 하고 두 번째로 McIPM을 이용하여 내부점방법에서 사용했을 경우에 대한 성능 평가를 실시하였다. 선형방정식을 1회 계산하는 경우에 대한 성능 평가에 사용된 선형방정식은 다음과 같이 $\theta = I$인 경우로 구성했다.

$$(AA^T)\Delta y = Ac + b$$

LAPACK, ORNL 그리고 McSMLne는 위의 정규방정식을 계산하도록 하였으며, McSMLas는 다음의 첨가방정식을 계산하도록 하였다.

$$\begin{pmatrix} -I & A^T \\ A & 0 \end{pmatrix}\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} c \\ b \end{pmatrix}$$

엄밀히 말하자면, 정규방정식을 푸는 경우에서는 $\Delta x = A^T\Delta y - c$를 계산하는 데에 소요되는 시간까지 고려해야 첨가방정식을 푸는 경우와 정확한 비교가 되지만 계산량을 많이 소요하는 계산이 아니기 때문에 무시하도록 하였다. ORNL 코드의 경우에 Netlib 문제 가운데 woodlp, fit1p 그리고 fit2p는 프로그램 오류로 인해 문제를 푸는 데에 실패하였다.

<표 1>은 실험에 사용된 Netlib 문제들에 대한 기본적인 정보와 함께 선형방정식 해법들의 정밀도를 비교한 내용을 보여준다. 전반적으로 LAPACK 코드에 비하여 ORNL 코드와 McSML의 코드들이 작은 계산오차로 방정식의 해를 계산해내는 것을 볼 수 있다. McSMLne 코드가 ORNL 코드와 비슷한 수준의 정밀도를 가지고 있으며, McSMLas 코드는 전체적으로 가장 정밀한 해를 계산해내었다.

<표 2>는 선형방정식 해법들의 계산 속도에 대해 비교한 것이다. 희소행렬 방정식을 푸는 데에 있어서 계산 속도에 큰 영향을 주는 것 가운데 하나가 분해요소의 비영요소개수이다. McSMLne는 순서화 및 상징적 분해와 같은 상징적 계산에 있어서는 가장 빠른 계산 속도를 보여주었으나 ORNL 코드가 dfl001과 같은 대형문제에 대해서 가장 적은 수의 비영요소 개수를 가지는 선회순서를 계산해내었기 때문에 전체 계산속도 면에서는 McSMLne를 앞섰다. 참고로 ORNL은 SPARSPAK의 순서화 코드를 이용하고 있다. 그러나, 전체적으로는 McSMLne이가 ORNL 코드와 대등한 수준의 추가요소를 생성하였으며, 비영요소의 개수에 있어서는 LAPACK이 가장 낮은 성능을 보여주었다. McSMLas는 다른 해법들과 달리 첨가방정식을 풀기 때문에 직접적인 비영요소 개수에 대한 비교는 어렵지만 밀집열을 가지고 있는 fit1p나 fit2p와 같은 문제에 있어서는 정규방정식 해법들에 비해 탁월하게 빠른 계산 속도를 보여주었다. 예를 들어, fit2p의 경우에 McSMLas는 McSMLne에 비해 70배 이상 빠른 계산속도로 방정식을 풀어냈다. 한편, 밀집행렬인 fit2p의 정규방정식에 대해서 McSMLne는 LAPACK 보다 25% 정도 더 빠른 행렬분해를 수행하여 행렬 내부에 밀집한 부분이 존재하는 경우에 대한 계산효율도 우수한 것으로 판단되며 이는 초마디 기법이 효과적으로 구현되었기 때문으로 해석된다.

<표 3>은 선형방정식 해법들을 자동조절자 내부점방법 코드인 McIPM에 연결하여 McIPM을 수행한 결과를 비교한 것이다. 앞선 실험에서 ORNL 코드나 McSML에 비해 매우 열등한 것으로 나타난 LAPACK은 실험에서 제외하였다. McIPM은 예측자-수정자 기법을 사용하기 때문에 매 회마다 선형방정식으로 적어도 4번 풀게 된다. 즉, 1번의 행렬분해와 최소한 4번의 치환연산을 수행한다. 자동조절자 내부점방법에서는 현재의 개선방향이 효과적이지 않다고 판단될 경우에 보다 강력한 자동조절자를 사용하여 개선방향을 다시 계산하기 때문에 더 많은 치환연산이 발생할 수도 있다. 내부점방법에서는 순서화를 통해 정해진 선회순서로 알고리즘의 반복수만큼 행렬분해를 수행하기 때문에 순서화에 걸리는 시간이 조금 더 많더라도 적은 수의 비영요소가 발생하면 내부점방법 전체의 수행속도 면에서 오히려 더 유리하게 된다. McSMLne가 ORNL보다 많은 비영요소수를 가진 dfl001과 같은 경우에 대해서는 ORNL을 사용한 경우가 매우 빠른 내부점방법 계산 속도를 보여주었다. 밀집열을 가지는 fit1p나 fit2p의 경우에 대해서는 ORNL을 사용할 경우 Schur 상보법을 적용하였는데, 밀집열을 그대로 두고 계산한 McSMLne보다는 매우 빠른 계산이 가능했으나 첨가방정식 방법이 밀집열을 가진 경우에 대한 가장 좋은 선택임을 알 수

〈표 1〉 선형방정식 해법 프로그램들의 정밀도 비교

Netlib 문제	행렬 A			해 의 오 차			
	행 개수	열 개수	비영요소 개수	LAPACK	ORNL	McSMLas	McSMLne
fit1d	24	1,049	13,428	3.184571E-08	5.054267E-08	9.432113E-09	4.507237E-08
fit2d	25	10,524	129,043	5.843231E-07	5.763979E-07	1.153016E-07	5.879320E-07
afiro	27	51	103	7.317240E-13	2.842624E-13	1.290736E-13	3.430260E-13
kb2	43	68	314	2.494120E-11	2.955116E-11	3.057629E-11	1.794578E-11
sc50b	50	78	149	5.533623E-13	1.252069E-12	2.935206E-13	1.094883E-12
sc50a	50	78	161	5.012920E-13	3.498102E-13	2.669277E-13	3.616202E-13
blend	74	114	523	8.619528E-13	5.158601E-13	1.333002E-12	4.429136E-13
recipe	91	204	687	4.906382E-15	2.521942E-15	1.056425E-15	3.461333E-15
share2b	96	162	778	1.382152E-10	7.038387E-11	9.276266E-11	1.036606E-10
sc105	105	163	341	6.287460E-13	6.392205E-13	5.133046E-13	7.457141E-13
stocfor1	117	165	502	5.132145E-09	4.819848E-09	2.104373E-09	2.883844E-09
share1b	117	253	1,180	1.218369E-07	1.160649E-07	5.236190E-08	1.293785E-07
grow7	140	301	2,613	1.436684E-14	1.521788E-14	7.346957E-15	1.267235E-14
lotfi	153	366	1,137	3.223359E-08	3.851437E-08	1.795012E-08	3.080465E-08
israel	174	316	2,444	3.916295E-05	1.612127E-05	6.885252E-09	6.734392E-05
vtp_base	198	346	1,052	2.829310E-06	2.116289E-06	1.848006E-10	1.464261E-06
brandy	220	303	2,203	2.185584E-06	5.059976E-11	5.160220E-11	2.463765E-11
bore3d	233	334	1,449	9.435650E-06	1.962953E-10	1.684690E-10	5.187624E-10
wood1p	244	2,595	70,217	4.250157E-09	N/A	3.134721E-12	1.637114E-11
capri	271	482	1,897	1.844938E-09	1.208362E-09	7.083745E-10	8.327701E-10
grow15	300	645	5,621	2.420896E-14	2.128665E-14	1.403983E-14	2.045703E-14
bandm	305	472	2,495	1.198610E-11	7.069006E-12	6.505433E-12	8.125983E-12
tuff	333	628	4,562	5.380971E-11	3.857867E-16	3.449584E-16	3.039853E-16
stair	356	614	4,004	3.106727E-12	4.308582E-12	3.293056E-12	3.961556E-12
standata	359	1,274	3,231	1.372347E-08	4.086162E-08	6.807884E-09	4.921598E-08
standgub	361	1,383	3,340	6.215570E-06	4.086162E-08	9.890614E-09	3.450845E-08
etamacro	400	816	2,538	1.597607E-09	2.231869E-09	3.944692E-10	1.682721E-09
ship04s	402	1,506	4,401	8.193888E-04	1.996653E-09	1.047021E-09	1.820549E-09
ship04l	402	2,166	6,381	8.311419E-04	2.754792E-09	1.786967E-09	1.991518E-09
pilot4	410	1,123	5,265	4.351172E-10	5.687810E-10	3.151407E-10	9.095966E-10
grow22	440	946	8,253	3.095087E-14	2.297547E-14	2.021084E-14	2.893338E-14
standmps	467	1,274	3,879	5.513047E-08	3.135137E-08	1.534886E-08	3.594719E-08
ffff800	524	1,028	6,402	7.077134E-02	1.755874E-01	3.961936E-07	2.768761E-01
shell	536	1,777	3,559	3.908584E-04	6.258152E-11	4.035011E-11	4.690274E-11
gfrd_pnc	616	1,160	2,446	4.901769E-06	3.686878E-06	6.443340E-09	3.419191E-06
perold	625	1,506	6,149	8.000504E-10	1.157952E-09	8.200742E-10	2.563166E-09
fit1p	627	1,677	9,869	3.096884E-06	N/A	6.974918E-07	6.314434E-06
scfxm2	660	1,200	5,470	1.115090E-08	5.433161E-09	4.281439E-09	5.729802E-09
pilot_we	722	2,928	9,266	9.811038E-08	3.497631E-08	6.552713E-08	3.461573E-08
pilot_ja	940	2,267	14,978	5.934674E-07	3.471226E-07	4.471068E-07	4.791471E-07
pilotnov	975	2,446	13,332	5.762324E-07	7.153512E-07	5.409615E-07	4.099848E-07
scfxm3	990	1,800	8,207	1.301365E-08	5.902547E-09	4.726320E-09	6.876784E-09
woodw	1,098	8,418	37,488	1.711606E-10	1.790987E-10	1.122518E-10	1.709948E-10
sierra	1,227	2,735	8,002	2.618139E+00	1.013871E-01	1.899450E-07	7.442442E-02
ganges	1,309	1,706	6,938	4.578865E-09	5.279368E-09	2.714681E-09	5.211624E-09
cycle	1,903	3,371	21,235	8.540851E-08	2.808177E-12	1.509394E-12	2.078463E-12
pilot87	2,030	6,680	74,950	3.418996E-10	2.526719E-10	2.551463E-10	2.168695E-10
80bau3b	2,262	12,061	23,265	1.423232E-11	1.293002E-11	2.381950E-11	6.414922E-12
greenbea	2,392	5,598	31,071	3.163434E-05	3.707248E-11	3.333704E-11	3.483777E-11
greenbeb	2,392	5,598	31,071	3.163434E-05	3.707248E-11	3.333704E-11	3.483777E-11
fit2p	3,000	13,525	50,285	2.336572E-05	N/A	7.400772E-06	1.075874E-05
dfl001	6,071	12,230	35,633	4.781453E+00	7.685944E-06	1.073627E-07	6.735289E-06

〈표 2〉 선형방정식 해법들의 계산 속도 비교

Netlib 문제	L의 비영요소 개수				상징적 계산 (단위 : 초)				수치 계산 (단위 : 초)			
	LAPACK	ORNL	McSMLas	McSMLne	LAPACK	ORNL	McSMLas	McSMLne	LAPACK	ORNL	McSMLas	McSMLne
fit1d	298	296	14,752	300	0.00	0.00	0.01	0.01	0.01	0.02	0.00	0.01
fit2d	325	324	139,866	325	0.00	0.00	0.68	0.06	0.13	0.11	0.08	0.06
afiro	108	108	238	112	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
kb2	503	503	869	531	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sc50b	253	234	421	245	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sc50a	244	243	437	249	0.01	0.00	0.01	0.00	0.00	0.01	0.00	0.00
blend	1,032	1,013	1,585	1,023	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00
recipe	774	760	1,479	679	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
share2b	1,180	1,049	1,868	1,025	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00
sc105	593	568	983	585	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
stocfor1	1,088	966	1,488	939	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.00
share1b	1,310	1,464	2,598	1,283	0.01	0.00	0.01	0.01	0.01	0.01	0.00	0.00
grow7	2,808	2,730	5,545	2,772	0.01	0.00	0.00	0.00	0.02	0.01	0.01	0.00
lotfi	1,954	1,904	3,246	1,897	0.02	0.00	0.00	0.00	0.01	0.00	0.00	0.00
israel	11,503	11,488	14,024	11,439	0.41	0.01	0.01	0.00	0.06	0.03	0.00	0.01
vtp_base	3,185	2,932	4,121	2,922	0.02	0.01	0.00	0.00	0.02	0.00	0.00	0.01
brandy	3,457	3,454	5,757	3,472	0.05	0.00	0.00	0.00	0.04	0.01	0.01	0.00
bore3d	3,273	3,126	4,547	2,998	0.06	0.01	0.00	0.01	0.03	0.00	0.01	0.00
wood1p	18,386	N/A	90,893	18,326	0.19	N/A	0.08	0.06	0.38	N/A	0.06	0.07
capri	6,884	5,626	7,936	5,829	0.07	0.00	0.01	0.00	0.03	0.01	0.00	0.00
grow15	6,524	6,090	12,262	6,297	0.03	0.01	0.00	0.01	0.03	0.01	0.01	0.01
bandm	5,314	4,677	7,323	4,662	0.07	0.01	0.01	0.00	0.03	0.01	0.00	0.01
tuff	8,179	8,636	13,083	8,227	0.19	0.01	0.01	0.00	0.06	0.02	0.00	0.01
stair	18,786	17,469	19,388	15,127	0.22	0.01	0.00	0.01	0.06	0.02	0.00	0.00
standata	3,384	3,353	7,511	3,366	0.02	0.01	0.01	0.01	0.01	0.01	0.00	0.00
standgub	3,418	3,355	7,852	3,492	0.02	0.00	0.01	0.01	0.03	0.01	0.00	0.00
etamacro	15,645	14,709	19,002	16,049	0.05	0.00	0.02	0.00	0.04	0.02	0.01	0.01
ship04s	3,662	3,654	9,166	3,662	0.04	0.00	0.01	0.00	0.04	0.01	0.00	0.00
ship04l	4,830	4,830	12,974	4,830	0.05	0.01	0.01	0.00	0.05	0.02	0.00	0.00
pilot4	14,202	14,724	20,339	14,362	0.20	0.01	0.01	0.01	0.06	0.03	0.01	0.01
grow22	10,145	9,058	18,133	9,375	0.05	0.01	0.02	0.01	0.04	0.03	0.00	0.01
standmps	5,371	5,428	9,985	5,300	0.04	0.00	0.01	0.00	0.03	0.02	0.00	0.00
ffff800	19,417	18,783	25,669	18,764	0.41	0.02	0.02	0.01	0.08	0.03	0.02	0.00
shell	4,956	4,480	9,265	4,466	0.03	0.00	0.01	0.01	0.04	0.01	0.00	0.00
gfrd_pnc	2,311	2,170	5,260	2,271	0.01	0.00	0.00	0.01	0.01	0.01	0.00	0.00
perold	28,867	25,690	32,488	25,459	0.15	0.01	0.01	0.02	0.09	0.03	0.02	0.01
fit1p	196,878	N/A	11,817	196,878	1.16	N/A	0.11	0.07	1.47	N/A	0.09	0.38
scfxm2	11,510	9,579	15,507	9,498	0.14	0.00	0.01	0.01	0.05	0.03	0.01	0.01
pilot_we	19,713	17,660	29,185	17,714	0.09	0.01	0.01	0.02	0.06	0.03	0.02	0.01
pilot_ja	53,132	55,652	69,684	53,380	0.71	0.05	0.03	0.03	0.18	0.08	0.05	0.04
pilotnov	56,038	55,249	68,621	53,819	0.42	0.05	0.03	0.03	0.17	0.06	0.04	0.03
scfxm3	17,564	14,479	23,383	14,367	0.22	0.01	0.01	0.01	0.07	0.03	0.01	0.01
woodw	50,593	48,377	93,653	48,846	0.71	0.04	0.05	0.06	0.21	0.10	0.05	0.02
sierra	5,813	5,608	22,157	12,648	0.04	0.00	0.02	0.01	0.06	0.02	0.02	0.01
ganges	31,004	27,152	29,052	21,718	0.16	0.01	0.01	0.02	0.09	0.03	0.02	0.01
cycle	159,952	77,530	104,979	82,277	1.55	0.05	0.05	0.04	0.80	0.12	0.06	0.05
pilot87	484,065	410,687	503,984	424,385	36.67	0.79	0.26	0.24	3.12	0.97	0.88	0.88
80bau3b	47,755	44,517	77,476	44,413	0.26	0.06	0.06	0.04	0.15	0.06	0.03	0.03
greenbea	128,327	82,705	116,797	82,521	1.08	0.14	0.06	0.06	0.68	0.16	0.06	0.05
greenbeb	128,327	82,705	116,797	82,521	1.07	0.15	0.06	0.06	0.68	0.14	0.06	0.05
fit2p	4,501,500	N/A	64,109	4,501,500	40.83	N/A	2.15	1.80	99.91	N/A	0.94	68.44
dfl001	1,884,207	1,549,742	1,732,685	1,690,894	4.06	1.13	0.76	0.80	37.35	3.50	8.98	8.70

〈표 3〉 McIPM에서의 선형방정식 해법 성능 비교

Netlib 문제	McIPM + ORNL			McIPM + McSMLne			McIPM + McSMLas		
	반복수	쌍대간격	계산시간	반복수	쌍대간격	계산시간(초)	반복수	쌍대간격	계산시간(초)
fit1d	28	1.59E-09	4.14	28	1.59E-09	4.29	28	1.59E-09	3.06
fit2d	21	1.27E-09	27.58	21	1.27E-09	29.17	21	1.27E-09	21.07
afiro	10	8.94E-10	0.17	10	8.94E-10	0.15	10	2.14E-09	0.13
kb2	17	3.98E-09	0.44	17	3.98E-09	0.40	17	3.98E-09	0.32
sc50b	10	3.32E-09	0.18	10	3.32E-09	0.16	10	3.57E-09	0.15
sc50a	11	3.90E-09	0.20	11	3.90E-09	0.19	11	3.90E-09	0.15
blend	11	4.25E-09	0.26	11	4.25E-09	0.22	11	2.29E-09	0.18
recipe	12	2.86E-08	0.44	12	2.86E-08	0.43	12	2.86E-08	0.34
share2b	12	1.04E-08	0.30	12	1.04E-08	0.27	12	1.21E-08	0.22
sc105	12	1.81E-08	0.26	12	1.81E-08	0.23	12	1.81E-08	0.20
stocfor1	15	1.76E-09	0.34	15	1.76E-09	0.31	16	2.03E-05	0.28
share1b	27	3.88E-09	0.73	27	4.41E-09	0.67	27	2.89E-08	0.52
grow7	19	3.61E-09	1.09	19	3.61E-09	1.06	19	3.61E-09	0.76
lotfi	23	2.66E-09	0.69	23	2.67E-09	0.62	23	2.64E-08	0.48
israel	22	5.51E-09	1.63	22	5.51E-09	1.41	26	6.83E-06	0.92
vtp_base	17	7.46E-09	0.76	17	7.39E-09	0.72	17	7.41E-09	0.53
brandy	20	7.49E-10	0.73	20	7.49E-10	0.64	20	1.06E-09	0.44
bore3d	24	8.85E-10	1.14	24	8.85E-10	1.04	48	4.57E-08	1.45
woodlp	16	2.65E-09	5.82	16	2.69E-09	6.52	16	2.65E-09	3.81
~capri	19	8.66E-09	1.17	19	8.66E-09	1.10	19	8.66E-09	0.74
grow15	18	4.94E-08	1.77	18	4.94E-08	1.70	18	4.94E-08	1.16
bandm	18	5.14E-09	0.81	18	5.14E-09	0.72	18	1.10E-08	0.46
tuff	21	1.57E-08	1.52	21	1.57E-08	1.41	21	1.85E-08	0.94
stair	19	8.55E-09	1.48	19	8.54E-09	1.37	19	8.54E-09	0.86
standata	19	4.31E-10	1.64	19	4.31E-10	1.59	19	1.73E-08	1.19
standgub	19	4.06E-10	1.72	19	4.06E-10	1.66	19	6.87E-09	1.32
etamacro	24	2.35E-08	1.63	24	2.35E-08	1.55	24	2.33E-08	1.13
ship04s	17	5.22E-10	1.20	17	5.21E-10	1.11	17	1.52E-01	0.93
ship04l	15	1.58E-06	1.55	16	8.60E-09	1.44	16	3.59E-01	1.09
pilot4	37	1.12E-08	4.61	37	1.33E-08	4.46	37	1.84E-08	2.87
grow22	18	3.05E-08	2.42	18	3.06E-08	2.34	18	3.06E-08	1.57
standmps	21	1.01E-09	1.95	21	1.01E-09	1.85	22	1.92E-08	1.45
ffff800	28	1.66E-08	2.63	29	1.75E-08	2.58	27	8.87E-09	1.35
shell	27	1.75E-09	2.73	27	1.75E-09	2.63	27	1.75E-09	1.82
gfrd_pnc	17	1.42E-08	1.50	17	1.42E-08	1.43	17	1.42E-08	1.03
perold	45	1.20E-08	6.19	45	2.30E-08	6.22	44	1.59E-08	4.12
fit1p	16	1.21E-08	4.00	16	1.21E-08	16.55	16	1.96E-08	1.59
scfxm2	28	1.62E-08	2.35	28	1.62E-08	2.16	28	2.75E-08	1.43
pilot_we	45	1.12E-08	7.94	43	2.99E-08	7.50	44	1.08E-08	5.24
pilot_ja	44	3.58E-08	10.08	43	4.16E-08	9.80	45	2.37E-08	6.40
pilotnov	26	6.46E-09	5.45	26	1.05E-09	5.60	26	8.19E-09	3.69
scfxm3	28	6.49E-09	3.41	28	6.49E-09	3.12	28	2.53E-08	2.02
woodw	25	5.61E-09	10.59	25	5.61E-09	10.52	25	1.36E-08	7.64
sierra	19	5.40E-09	4.76	19	5.43E-09	4.52	19	1.25E-08	3.22
ganges	20	1.03E-08	2.98	20	1.03E-08	2.76	20	1.03E-08	1.74
cycle	38	3.75E-09	13.33	38	3.59E-08	12.28	40	2.52E-09	7.45
pilot87	79	4.29E-08	141.26	80	4.56E-08	167.26	70	3.29E-08	98.94
80bau3b	42	3.51E-09	30.86	42	3.51E-09	30.30	42	3.51E-09	22.34
greenbea	47	2.00E-08	24.82	47	2.05E-08	23.26	47	1.70E-08	14.05
greenbeb	46	5.92E-09	24.04	46	5.93E-09	22.46	46	5.80E-09	13.48
fit2p	20	6.13E-10	33.44	20	6.13E-10	1666.13	20	4.98E-10	19.15
dfl001	46	4.67E-06	229.80	47	2.87E-06	456.76	53	2.59E-07	511.53

있다. 전반적으로 McSMLas가 McSMLne와 대등한 수준의 성능을 보여주었으나, 해법 후반에 행렬이 수치적으로 불안정해지는 현상에 대해서 McSMLas가 McSMLne보다 민감하게 반응하여 McSMLas를 McIPM에서 사용하는 경우에 최적해의 정밀도가 정규방정식 방법을 사용하는 경우보다 나쁜 경우들(stocfor1, israel, ship04s, ship04l)이나 반복수가 매우 많이 증가하는 경우(bore3d)가 발생하였다.

5. 결 론

본 논문에서 내부점방법에서 사용되는 선형방정식 해법의 구현에 고려되어야 하는 사항들을 제시하였으며, 실제로 본 연구를 통하여 자동조절자 내부점방법을 위한 선형방정식 해법 코드인 McSML이 개발되었다. McSML은 정규방정식 방법을 사용하는 McSMLne와 첨가방정식 방법을 사용하는 McSMLas로 구성되어 있으며, McSMLas의 경우에는 첨가행렬의 2사분면 블록이 선형계획법에서처럼 대각행렬이 아니라 이차계획법에서와 같이 일반적인 대칭행렬로 구성되더라도 계산이 가능하도록 하였기 때문에 비선형계획법을 위한 내부점방법에도 적용할 수 있다. 물론, 내부점방법이 아니라 일반적인 대칭행렬로 이루어진 방정식을 푸는 데에도 활용할 수 있다.

McSML은 순서화 방법으로 외부차수를 이용한 최소 차수 순서화를 사용하였으며, 기존에 개발되어 있던 LAPACK이나 ORNL 코드의 순서화와 대등한 수준의 추가요소를 생성하였다. 순서화 계산속도는 McSMLne가 ORNL 코드보다 우수했으나, 몇 개의 대형문제에 대해서 ORNL 코드의 순서화에 비하여 많은 추가요소를 생성시키는 결과를 보여주었다. McSMLne가 순서화 계산속도의 향상을 위해 다중삭제(multiple elimination)의 허용수준을 높인 반면 ORNL 코드는 다중삭제의 허용수준을 0으로 두어 순서화 계산속도보다 추가요소 개수를 최소화하는 전략을 취하였기 때문으로 해석된다. 특히, 내부점방법에서는 동일한 구조의 선형방정식을 여러 차례 계산하기 때문에 순서화 계산속도가 빠른 것보다는 순서화 계산시간이 좀더 많이 소요되더라도 추가요소 개수를 줄이는 것이 더 중요하다는 것을 알 수 있다.

수치 계산에 있어서는 McSMLne와 McSMLas가 LAPACK이나 ORNL에 상응하는 계산속도와 정밀도를 보여주었다. 그러나, 내부점방법에서 풀어야 하는 선형방정식이 내부점방법 후반부에 가까워질수록 θ값이 넓은 범위를 가지게 되어 결과적으로는 선형방정식이 수치적으로 불안정해지게 되는데, 첨가방정식 방법을 사용하는 McSMLas가 정규방정식 방법을 사용한 McSMLne보다 더 민감하게 반응하여 최적해의 정밀도가 떨어지거나 반복수가 늘어나는 경우가 발생하였다. 선형방정식의 해가 큰 계산오차를 가지는 경우에는 반복정제를 통하여 계산오차를 줄이도록 하였으나 충분히 효과를 발휘하지 못하였던 것으로 판단된다.

참 고 문 헌

[1] 도승용, 성명기, 박순달, "내부점 방법에서 augmented system의 촐레스키 분해", 「한국경영과학회지」, 제28권, 제1호(2003), pp.51-61.

[2] 박순달, 김우제, 설동렬, 박찬규, 성명기, 임성묵, 「고등선형계획법」, 초판, 교우사, 2001.

[3] 설동렬, 도승용, 박순달, "내부점 방법에서 밀집열 처리에 관한 연구 - Schur 상보법의 효율적인 구현", 「한국경영과학회 '98 추계학술대회 논문집」(1998), pp.67-70.

[4] 설동렬, 정호원, 박순달, "내부점방법을 위한 초마디 열촐레스키 분해의 실험적 고찰", 「경영과학」, 제15권, 제1호(1998), pp.87-96.

[5] Altman, A. and J. Gondzio, "Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization," *Optimization Methods and Software*, Vol.11-12(1999), pp.275-302.

[6] Andersen, E.D., J. Gondzio, C. Mészáros

and X. Xu, "Implementation of interior point methods for large scale linear programming," *Interior Point Methods in Mathematical Programming*, T. Terlaky (ed.), Kluwer Academic Pub., 1996.

[7] Anderson, E., Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK User's Guide*(http://www.netlib.org/lapack/lug/lapack_lug.html), Third Edition, SIAM, Philadelphia, 1999.

[8] Duff, I.S., A. Erisman and J. Reid, *Direct Methods for Sparse Matrices*, Monographs on Numerical Analysis, Clarendon Press, Oxford, 1986.

[9] Duff, I.S. and J. Reid, "Exploiting zeros on the diagonal in the direct solution of indefinite sparse symmetric linear systems," *ACM TOMS*, Vol.22, Issue 2(1996), pp. 227-257.

[10] Duff, I.S., N. Gould, J. Reid, J. Scott, K. Turner, "The factorization of sparse symmetric indefinite matrices," *IMA J. Numer. Anal.*, Vol.11(1991), pp.181-204.

[11] Fourer, R. and S. Mehrotra, "Solving symmetric indefinite systems in an interior-point method for linear programming," *Math. Prog.*, Vol.62, No.1(1993), pp.15-39.

[12] Gay, D.M., "Electronic mail distribution of linear programming test problems," *Mathematical Programming Society Committee on Algorithms Newsletter*, No.13(1985), pp. 10-12.

[13] George, A. and J. Liu, *Computer Solution of Large Sparse Positive Definte Systems*, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1981.

[14] Gilbert, J., C. Moler and R. Schreiber, "Sparse Matrices in MATLAB : design and implementation," *SIAM Journal on Matrix Analysis and Applications*, Vol.13(1992), pp. 333-356.

[15] Gondzio, J., "Implementing Cholesky factorization for interior point methods of linear programming," *Optimization*, Vol.27(1993), pp.121-140.

[16] Liu, J., "Modification of the minimum-degree algorithm by multiple elimination," *ACM TOMS*, Vol.11, Issue 2(1985), pp.141-153.

[17] Maros, I. and C. Mészáros, "The role of the augmented system in interior point methods," *EJOR*, Vol.107(1998), pp.720-736.

[18] Mészáros, C., "Fast Cholesky factorization for interior point methods of linear programming," *Comp. Math. Appl.*, Vol.31, No.4 (1996), pp.49-51.

[19] Mészáros, C., "The augmented sywstem variant of IPMs in two-stage stochastic linear programming computation," *EJOR*, Vol.101, No.2(1997), pp.317-327.

[20] Mészáros, C., "On a property of the Cholesky factorization and its consequences in interior point methods," WP 98-7 Laboratory of Operations Research and Decision Systems Hungarian Academy of Sciences, 1998.

[21] Esmond, N. and B. Peyton, "Block sparse Cholesky algorithms on advanced uniprocessor computers," Technical Report ORNL/TM-11960, Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1991.

[22] Park, C.K., S. Doh, S. Park and W. Kim, "A minimum degree ordering algorithm using the lower and upper bounds of degrees," *Int. Journal of Management Science*, Vol.

8, No.1(2002), pp.1-19.

[23] Peng, J., C. Roos and T. Terlaky, *Self-Regularity : A New Paradigm for Primal-Dual Interior-Point Algorithms,* Princeton University Press, Princeton, New Jessy, 2002.

[24] Saunders, M. and J. Tomlin, "Solving regularized linear programs using barrier methods and KKT systems," Report SOL 96-4 Dept. of EESOR, Stanford Univ., 1996.

[25] Seol, T. and S. Park, "Solving linear systems in interior-point methods," *Computers & Operations Research*, Vol.28, No.4 (2002), pp.317-326.

[26] Vanderbei, R., "Symmetric Quasidefinite Matrices," *SIAM J. Opt.*, Vol.5, No.1(1995), pp.100-113.

[27] Vanderbei, R. and T. Carpenter, "Symmetric indefinite systems for interior-point methods," *Math. Prog.*, Vol.58, No.1(1993), pp.1-32.

[28] Zhang, Y., "Solving large-scale linear programs by interior-point methods under the MATLAB Environment," *Optimization Methods and Software*, Vol.10(1998), pp.1-31.

[29] Zhu, X., J. Peng, T. Terlaky and G. Zhang, "On implementing self-regular proximity based feasible IPMs," AdvOl-Report# 2004/3, 2004.