

내부점 방법에서 밀집열 처리에 관한 연구  
(Schur 상보법의 효율적인 구현)  
A study on handling dense columns in interior point methods for linear programming  
(An efficient implementation of Schur complement method)

설동렬, 도승용, 박순달  
서울대학교 산업공학과

**Abstract**

The computational speed of interior point method of linear programming depends on the speed of Cholesky factorization to solve  $A\Theta A^T \Delta y = \bar{b}$ . If the coefficient matrix  $A$  has dense columns then the matrix  $A\Theta A^T$  becomes a dense matrix. This causes Cholesky factorization to be slow. The Schur complement method is applied to treat dense columns in many implementations but suffers from its numerical instability. We study efficient implementation of Schur complement method. We achieve improvements in computational speed and numerical stability.

**1. 내부점 방법과 밀집열**

내부점 방법은 대칭양정치 행렬의 연립 선형방정식을 푸는 과정이 필요하다. 내부점 방법에서는 전체 해법 소요 시간의 많은 부분이 연립 선형방정식을 푸는 데에 소요되어, 효율적으로 연립 선형방정식을 풀어내는 것이 해법의 성능을 크게 좌우한다. 내부점 방법에서 나타나는 연립 선형방정식은 다음과 같다.

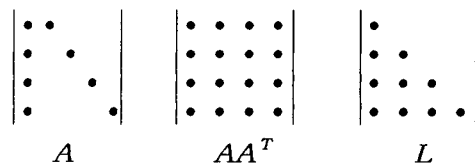
$$A\Theta A^T \Delta y = \bar{b} \quad (1)$$

단, 행렬  $A$ 는  $m \times n$  행렬이고,  $\Theta$ 는  $n \times n$ 의 대각행렬이고 모든 대각요소의 값은 양이다.  $\Delta y$ 와  $\bar{b}$ 는  $m \times 1$  벡터이다. 여기서 행렬  $A\Theta A^T$ 는  $m \times m$  대칭양정치(symmetric and positive definite) 행렬이다.

대형수리계획법 문제들에서 나타나는 행렬  $A$ 는 희소행렬(sparse matrix)의 특성을 가지기 때문에, 비영요소만을 보관하여 계산하도록 하는 희소

행렬 기법을 적용한다. 또한, 내부점 방법에서 풀어야 하는 선형방정식도 출레스키 분해를 적용하여 계산하므로, 출레스키 요소  $L$ 이 희소성을 유지하도록 하기 위해 최소 차수 순서화 또는 최소 부족수 순서화와 같은 순서화 기법을 적용한다.

그러나, 행렬  $A$ 가 희소행렬이라 할지라도 행렬  $A\Theta A^T$ 가 행렬  $A$ 의 희소성을 물려받지 못하고 밀집행렬(dense matrix)로 나타나는 경우가 발생한다. 행렬  $A\Theta A^T$ 의 밀집도가 높으면 선형방정식을 푸는 데에 많은 시간이 소요된다. 이것은 내부점 선형계획법의 수행속도를 저하시키는 결과를 가져오게 된다. 예를 들어, [그림 1]과 같이 행렬  $A$ 에 밀집열이 존재하면 행렬  $A\Theta A^T$ 의 밀집도가 매우 높아지는 성질을 가진다.



[그림 1] 밀집열을 가진 행렬의 형태

행렬  $A$ 의  $k$ 번째 열에서 비영요소를 가진 행들의 쌍들은 모두  $AA^T$ 에서 비영요소를 가지게 된다. 따라서, 행렬  $A$ 의 전체적인 밀집도는 낮다고 할지라도 특정한 열의 밀집도가 높은 경우에는 행렬  $A\Theta A^T$ 는 밀집행렬이 된다.

행렬  $A$ 에 존재하는 밀집열을 처리하는 기법으로, 입력된 문제를 쌍대문제로 전환하여 밀집열을 제거하는 쌍대문제 전환방법[1,5], 밀집열을 여러개로 쪼개는 밀집열 분할 방법[2,4,8], 밀집열을 제외한 나머지 부분에 대해서만 출레스키 분해를 적용하는 Schur 상보법[6] 등이 있다.

한편 출레스키 분해를 사용하지 않고 반복적

\* 본 논문은 1997년 한국학술진흥재단의 자유공모과제 1997-001-E00152 연구비에 의하여 연구되었음.

인 방법을 사용하여 선형방정식을 푸는 공액경사법 (Conjugate Gradient Method)[3]이 있다. 이 방법은 네트워크 문제와 같은 특수한 형태의 행렬을 가지는 문제에 대해 내부점 방법을 적용할 때에 주로 사용된다.

이 가운데 계산속도가 빠른 Schur 상보법이 널리 사용되고 있다. 본 논문에서는 Schur 상보법을 효율적으로 구현하기 위한 방법들을 다룬다.

## 2. Schur 상보법

Schur 상보법은 밀집열을 효과적으로 처리해 주기는 하지만, 수치적으로 안정적이 못한 방법으로 알려져 있다[6]. Schur 상보법은 밀집열을 제외한 나머지 부분 행렬에 대해서만 희소행렬 출레스키 분해를 직접 적용한다. 그런데 최적해에 가까이 가면서 행렬이 매우 계수가 부족해지는 현상이 발생하므로 수치적으로 불안정해진다. 이와같은 현상은 제거된 밀집열의 개수가 많을수록 심하게 나타나며, 반복정제(Iterative Refinement)를 통해 보정해 주어야 한다.

Schur 상보법은 행렬  $A$ 에 밀집열이 있는 경우에는 희소한 열들로 구성되어지는 부분과 밀집열로 구성되어지는 부분으로 행렬  $A$ 를 다음과 같이 나눈다.

$$A = \begin{pmatrix} A_s & A_d \end{pmatrix} \quad (2)$$

여기서 첨자  $s$ 와  $d$ 는 각각 희소한 열들과 밀집열들에 해당되는 부분을 가리킨다. 행렬  $A$ 를 이와같이 분할하면 식 (1)은 다음과 같이 된다.

$$A_s \theta_s A_s^T \Delta y + A_d \theta_d A_d^T \Delta y = \bar{b} \quad (3)$$

밀집도가 높은  $A_d$ 에 의해서  $A \theta A^T$ 이 밀집행렬이 되므로, Schur 상보법에서는  $A \theta A^T$  대신 희소한  $A_s \theta_s A_s^T$ 에만 희소행렬 출레스키 분해를 적용하여 주어진 선형방정식을 푸는 방법을 사용한다. 밀집열에 해당하는 부분에 대해서는 작은 행렬의 밀집행렬 출레스키 분해만 수행하면 되기 때문에 계산속도가 매우 빨라진다.

식 (1)은 다음식과 같이 쓰여질 수 있다.

$$\begin{pmatrix} A_s \theta_s A_s^T & A_d \\ A_d^T & -\theta_d^{-1} \end{pmatrix} \begin{pmatrix} \Delta y \\ r \end{pmatrix} = \begin{pmatrix} \bar{b} \\ 0 \end{pmatrix} \quad (4)$$

단,  $r$ 은  $NDCOL \times 1$  벡터이다. 여기서  $NDCOL$ 은 밀집열의 개수이다.

Schur 상보법은 다음과 같은 단계를 거쳐서 식 (4)를 푼다.

$$\text{단계 1 } L_s L_s^T = A_s \theta_s A_s^T$$

$$\text{단계 2 } L_s V = A_d$$

$$\text{단계 3 } L_s p = b$$

$$\text{단계 4 } (V^T V + \theta_d^{-1}) r = V^T p$$

$$\text{단계 5 } L_s^T \Delta y = p - V r$$

이 때  $r$ 은 쉽게 구해질 수가 있다. 왜냐하면  $V^T V + \theta_d^{-1}$ 가 밀집행렬이기는 하지만 차수는 작기 때문이다.  $r$ 을 구하기 위해서 행렬  $V^T V + \theta_d^{-1}$ 에 대해 밀집행렬 형태로 출레스키 분해를 수행한다.

Schur 상보법은  $A_s$ 가 완전계수가 아닌 경우에 출레스키 분해를 제대로 수행할 수 없는 문제를 가지고 있다. 즉, 출레스키 분해를 수행하는 도중에 대각요소가 0 요소가 될 수 있는 것이다. 이를 해결하기 위해서 제안된 방법이 수정된 Schur 상보법이다. 기본적인 아이디어는 대각요소에 발생하는 영인 요소를 제거하기 위해서 다음과 같이 적당한 값을 더하여 변형된 행렬을 분해하는 것이다.

$$L_s L_s^T = A_s \theta_s A_s^T + \rho I \quad (5)$$

단,  $\rho$ 는 출레스키 분해가 잘 수행되도록 하기 위하여 더하여준 양의 상수이다. 그러나, 위와 같이 구한  $L_s L_s^T$ 를 직접 사용할 경우에는 원래의 방정식과 다른 문제이므로 반복정제가 필요하다. 따라서 식 (5)를 사용하는 대신 다음과 같이 수정된 시스템 식 (6)을 풀도록 한다.

$$\begin{bmatrix} A_s \theta_s A_s^T + FF^T & A_d & F \\ A_d^T & -\theta_d^{-1} & 0 \\ F^T & 0 & I \end{bmatrix} \begin{bmatrix} \Delta y \\ r \\ s \end{bmatrix} = \begin{bmatrix} \bar{b} \\ 0 \\ 0 \end{bmatrix} \quad (6)$$

단,  $F$ 행렬은 각 열에 오직 하나의 비영요소만을 가지는  $m \times k$  행렬이다.  $s$ 는  $k \times 1$ 인 벡터이다.  $k$ 는 대각요소가 0요소인 열의 개수이다.  $F$ 행렬은 오직 대각요소의 값을 변화시키기 때문에 비영요소 구조에는 변화를 주지 않는다. 따라서,  $A \theta A^T \Delta y = \bar{b}$ 를 푸는 데에 다음과 같이 수정된 Schur 상보법을 적용할 수 있다.

$$\text{단계 1 } L_s L_s^T = A_s \theta_s A_s^T + FF^T$$

$$\text{단계 2 } L_s V = A_d \\ L_s W = F$$

$$\text{단계 3 } L_s p = \bar{b}$$

$$\text{단계 4 } \left( \begin{pmatrix} V^T V & V^T W \\ W^T V & W^T W \end{pmatrix} + \begin{pmatrix} \theta_d^{-1} & 0 \\ 0 & -I \end{pmatrix} \right) \begin{pmatrix} r \\ s \end{pmatrix} = \begin{pmatrix} V^T p \\ W^T p \end{pmatrix}$$

$$\text{단계 5 } L_s^T \Delta y = p - V r - W s$$

## 3. 구현방법

본 논문에서는 두 가지 구현 방법을 고려한다.

### • 구현방법 1

구현방법 1은 위에서 제시한 Schur 상보법을 푸는 단

계를 구현한 방법이다. 구현방법 1에서는 희소한 행렬  $A_s \Theta_s A_s^T + FF^T$ 에 순서화를 적용하여 출레스키 팩터  $L_s$ 에는 비영요소만을 저장한다.  $\begin{pmatrix} V^T V & V^T W \\ W^T V & W^T W \end{pmatrix} + \begin{pmatrix} \Theta_d^{-1} & 0 \\ 0 & -I \end{pmatrix}$ 는 밀집열의 개수가 희소한 경우에는 차수가 작기 때문에  $(NDCOL + k) \times (NDCOL + k)$ 의 밀집행렬로 저장한다.

구현방법 1은 다음과 같은 절차로 수행된다.

- 단계 1  $A_s \Theta_s A_s^T + FF^T$ 를 출레스키 분해를 한 다음  $L_s$ 을 구한다.
- 단계 2  $L_s [V \ W] = [A_d \ F]$ 에서 전방치환연산을 통해서  $V$ 와  $W$ 를 구한다.
- 단계 3  $L_s \mathbf{p} = \bar{\mathbf{b}}$ 에서 전방치환 연산을 통해  $\mathbf{p}$ 를 구한다.
- 단계 4  $\begin{pmatrix} V^T V & V^T W \\ W^T V & W^T W \end{pmatrix} + \begin{pmatrix} \Theta_d^{-1} & 0 \\ 0 & -I \end{pmatrix}$ 를 출레스키 분해를 한 다음 전후방 치환연산을 통해  $\mathbf{r}$ ,  $\mathbf{s}$ 를 구한다.
- 단계 5  $L_s^T \Delta \mathbf{y} = \mathbf{p} - V \mathbf{r} - W \mathbf{s}$ 에서 후방치환 연산을 통해 개선방향  $\Delta \mathbf{y}$ 를 구한다.

구현방법 1에서는 단계 2에서  $(NDCOL \times k)$ 번의 전방치환 연산이 필요하고 단계 3에서는 1번의 전방치환 연산이 필요하다. 단계 4에서는 밀집행렬에 대한 출레스키 분해를 수행한 다음 전방치환과 후방치환 연산을 수행한다. 그리고 단계 5에서는 1번의 후방치환 연산을 수행한다. 따라서 구현방법 1에서는 개선방향  $\Delta \mathbf{y}$ 를 구하기 위해 매단계마다 치환연산을 수행해야만 한다. 이는 구현방법 1의 속도를 저하시키는 주된 원인이다. 따라서 전·후방 치환연산을 한 번만 수행하는 구현방법 2를 제시하고자 한다.

#### • 구현방법 2

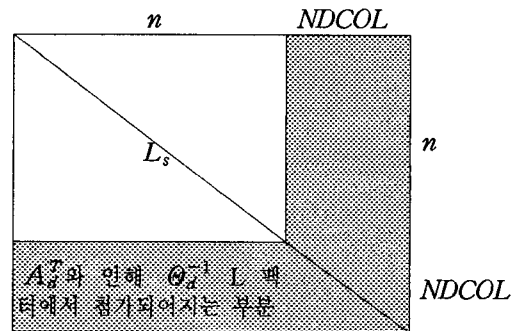
식 (4)를 직접푸는 방법을 고려해보자. 즉 식 (4)에 직접 출레스키 분해를 하면 다음식과 같이 된다.

$$LDL^T \begin{pmatrix} \Delta \mathbf{y} \\ \mathbf{r} \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{b}} \\ \mathbf{0} \end{pmatrix}$$

여기서  $D$ 는  $(n + NDCOL) \times (n + NDCOL)$ 인 대각행렬이다.

구현방법 2에서는 밀집열을 고려한 출레스키 팩터  $L$ 의 구조를 결정해 주어야 한다. 이는 기존의  $L_s$ 의 자료구조에 변화를 주지 않으면서 다음과 같이 쉽게 결정할 수가 있다. 희소행렬  $A_s \Theta_s A_s^T$ 에 대해서 순서화와 상징적 분해를 통해 출레스키 팩터  $L_s$ 의 구조를 정한다. 다만  $L_s$ 의 구조를 정할 때 각 열에 대해 밀집열의 개수  $NDCOL$ 개 만큼만 저장공간을 확보해 주면 된다.

그러면 밀집열을 고려한 출레스키 팩터  $L$ 의 구조는 다음과 같이 된다. 희소행렬  $A_s \Theta_s A_s^T$ 의 출레스키 팩터  $L_s$ 은 비영요소만을 저장하고 밀집열로 인해 첨가되는 부분은  $L_s$  뒤에 밀집열의 개수  $NDCOL$ 만큼 첨가한다. 그리고 나서  $\Theta_d^{-1}$ 에 의해 첨가되어지는 부분의 대각요소 아래의 부분을 고려하여 아래의 [그림2]과 같이 팩터  $L$ 의 구조를 정한다.



[그림 2] 밀집열을 고려한 출레스키 팩터  $L$ 의 구조

구현방법 2에서는 다음과 같은 절차에 의해 수행된다.

- 단계 1 희소행렬  $A_s \Theta_s A_s^T$ 에 순서화와 상징적 분해를 통해 생성된  $L_s$ 의 구조로부터 밀집열을 고려한 출레스키 팩터  $L$ 을 만든다.
- 단계 2 식 (4)에 출레스키 분해를 하여  $LDL^T [\Delta \mathbf{y} \ \mathbf{r}]^T = [\bar{\mathbf{b}} \ \mathbf{0}]^T$ 와 같이 되게 한다.
- 단계 3 전방치환연산과 후방치환 연산을 수행하여  $\Delta \mathbf{y}$ 를 구한다.

구현방법 2에서는 식 (4)에 대해 직접 출레스키 분해를 수행하므로 구현방법 1에서와 같이 매 단계마다 치환연산을 수행할 필요가 없다. 단지 단계 3에서 전방치환 연산과 후방치환연산이 단 한 번만 필요하다. 따라서 구현방법 1보다 속도면에 더 효율적이다. 구현방법 2에서는 수치적 정밀도를 증가시키기위해 대각요소가 0에 가까워질 경우 선회순서를 뒤로 미루어서 계산을 할 수가 있다.

#### 4. 실험결과

앞에서 제시한 Schur 상보법의 두 가지 구현방법을 내부점 방법에 구현하여 netlib data중에서 밀집열이 존재하는 israel, seba, fit1p, fit2p에 대해 적용하여 보았다. [표 1]을 보면 밀집열을 처리할 경우와 처리하지 않을 경우에 수행시간에는 상당한 차이가 있음을 알 수 있다. 구현방법 2가 구현방법 1보다 반복수당 수행시간이 15-30%정도 단축되어짐을 알 수 있다. 이는 구현방법 2가 치환연산을 한 번 수행하는 대신에 구현방법 1은 매 단계마다 치환연산을 하기 때문이다. 그리고 구현방법 2가 구현방법 1보다 반복수가 줄어들었다. 이는 구현방법

2의 수치적 정확성이 구현방법 1보다 더 정밀함을 보여 준다.

[표 1] 밀집열 처리 실험결과

문제 이름	수행시간		
	처리 않음	구현방법 1	구현방법 2
israel	1.19(0.06/20)	0.69(0.04/18)	0.45(0.03/18)
seba	9.82(0.54/18)	1.07(0.06/18)	0.66(0.04/15)
fit1p	52,40(3.74/14)	2.27(0.13/17)	1.55(0.10/15)
fit2p	*	22.21(0.91/24)	15.60(0.79/20)

\*: 메모리 부족으로 계산하지 못했음, (반복수당 수행시간/ 반복수)

## 5. 결론

내부점 방법에서의 수행시간은  $AQA^T \Delta y = \bar{b}$ 를 푸는 시간에 많은 영향을 받는다. 특히 행렬  $A$ 에 밀집열이 존재하는 경우에는  $AQA^T$ 가 밀집행렬이 된다. 이는 출레스키 분해의 수행속도를 느리게 한다. 밀집열을 처리하기 위한 기법인 Schur 상보법은 개선방향을 구하기 위해 매 단계마다 치환연산을 한다. 이는 Schur상보법의 효율을 저하시킨다. 따라서 본 논문에서는 Schur상보법에서 치환연산의 회수를 줄이는 구현방법을 통해 수행시간과 반복수를 개선하였다. 전체적으로 수행시간이 약 35%정도 향상되었다.

## 6. 참고문헌

[1] 설동렬, 정호원, 박순달, "내부점 선형계획법의 쌍대 문제 전환에 대하여", 한국경영과학회 '96추계학술대회논문집, pp. 289-292, 1996. 10.

[2] 설동렬, 정호원, 박순달, "내부점 선형계획법의 밀집열 분할에 대하여", 경영과학, 제14권 2호, pp. 69-80, 1997. 11.

[3] 설동렬, 조은영, 박순달, "네트워크 문제에서 내부점 방법의 활용(내부점 선형계획법에서 효율적인 공액경사법)", 군사운영분석학회지, 24권 1호(예정), 1998

[4] Vanderbei, Robert J., "Splitting dense columns in sparse linear systems", Linear Algebra Appl., No.152(1991), pp.107-117

[5] Vanderbei, Robert J., "ALPO: Another Linear Program Optimizer", ORSA Journal on Computing, vol.5, no.2, Spring 1993

[6] Andersen, Knud D., "A Modified Schur Complement Method for Handling Dense Columns in Interior Point Methods for Linear Programming", Technical Report, Department of Mathematics and Computer Science, Odense University, Oct7 1995,

[7] Choi, I. C., C. L. Monma and D. F. Shanno, "Further development of a primal-dual interior point method", ORSA Journal on Computing, No. 2, pp.304-311, 1990

[8] Gondzio Jacek, "Splitting dense columns of constraint matrix in interior point methods for

large scale linear programming", Optimization, Vol.24(1992), pp.285-297

[9] Andersen, Erling D., Jacek Gondzio, Csaba Meszaros, Xiaojie Xu, "Implementation of Interior Point Methods for Large Scale Linear Programming", Interior Point Methods in Mathematical Programming, T. Terlaky (ed.), Kluwer Academic Publishers, 1996

[10] Duff, I. S., A. M. Erisman and J. K. Reid, Direct Methods for Sparse Matrices, Oxford Univ. Press, 1986

[11] George, Alan and J. W. H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice Hall, 1981

[12] Golub, G. and C. Van Loan, Matrix Computation, 2nd Ed., The Johns Hopkins University Press, Baltimore, 1989

[13] Kojima, M., N. Meggido and S. Mizuno, "A Primal-Dual Infeasible-Interior-Point Algorithm for Linear Programming", Mathematical Programming, Vol.61, pp.263-280, 1993

[14] Mehrotra, S., "On the Implementation of a Primal-Dual Interior-Point Method", SIAM Journal on Optimization, Vol.2, pp575-601, 1992

[15] Meszaros, Csaba, The Efficient Implementation of Interior Point Methods for Linear Programming and their Applications, Ph. D. Thesis, 1996

[16] Maros, I., and Csaba Meszaros, "The Role of the Augmented System in Interior Point Methods", Technical Report TR/06/95, Brunel University, Department of Mathematics and Statistics, London, 1995

[17] Strang, Gilbert, Linear Algebra and its Applications, 3rd Ed., Harcourt Brace Jovanovich, Inc., 1986'