상하분해 단체법에서 수정 Forrest-Tomlin방법의 효율적인 구현†

김우제*, 임성묵**, 박순달**

*대진대학교 산업공학과, **서울대학교 산업공학과

Abstract

In the implementation of the simplex method program, the representation and the maintenance of basis matrix is very important, In the experimental study, we investigates Suhl's idea in the LU factorization and LU update of basis matrix. First, the triangularization of basis matrix is implemented and its efficiency is shown. Second, various technique in the dynamic Markowitz's ordering and threshold pivoting are presented. Third, modified Forrest-Tomlin LU update method exploiting sparsity is presented. Fourth, as a storage scheme of LU factors, Gustavson data structure is explained. Fifth, efficient timing of reinversion is developed. Finally, we show that modified Forrest-Tomlin method with Gustavson data structure is superior more than 30% to the Reid method with linked list data structure.

1. 서론

단체법은 선형계획법의 해법들 중 하나로서 내부점 기법과 더불어 가장 효과적인 방법이 되어왔다[1]. 현대로 오면서 선형계획법으로 해결해야할 문제의 크기가 기하급수적으로 커지고 있고, 또한 그 문제의 형태가 수치적으로 불안정한 경우가 많아지고 있다. 따라서, 대형문제를 고속으로, 안정적으로 풀어낼 수 있는 단체법 프로그램의 개발은 아주 중요하다고 할 수 있다.

단체법의 계산 과정을 살펴볼 때, 기저행렬이 관계된 $Bx = b$형태의 선형방정식을 매번 풀어야 한다[1]. 따라서 위 형태의 선형방정식을 효과적으로 풀어낼 수 있는 기저행렬의 보관, 수정방법이 중요하게 된다. 실제적으로 기저행렬 관련 연산이 단체법 계산 시간의 60-90%를 차지하므로 기저행렬 보관, 수정방법이 단체법 프로그램의 효율성이 미치는 영향은 아주 크다고 할 수 있다. 기저행렬을 보관하는 방법으로 최근에는 기저행렬을 상하분해하여 보관하는 상하분해법이 많이 사용되고 있다. 상하분해 방법은 행렬의 희소성을 잘 이용할 수 있어 전체 계산량을 줄일 수 있고 수치 안정성을 이루기 쉽다는 장점을 가진다. 기저역행렬을 상하분해형으로 유지하기 위해서는 크게 상하분해방법과 기저수정방법이 필요하게 된다.

상하분해에서는 희소도를 이용하기 위해 행과 열의 순서화가 요구되는데 순서화 방법으로는 마코비츠순서화 방법이 널리 사용된다. 그리고 수치적 안정성을 확보하기 위한 기법으로 현재까지 효과적이라고 알려진 것으로 threshold pivoting이 있다. 상하분해 형태로 보관된 기저의 수정 방법에 관한 연구는 Bartels, Golub[4], Forest, Tomlin[6], Reid[7], Saunders[8] 등에 의해 이루어 졌다.

상하분해 형태의 기저 수정 방법의 실제 구현에서는 L과 U를 보관하는 효과적인 자료구조와 적절한 재역산 시점의 선정이 요구된다. 상삼각행렬 U를 보관하는 자료구조로 Gustavson 구조[3]와 Reid 등에 의해 사용된 연결리스트 구조[2]가 알려져 있다.

재역산은 수치적 안정성 뿐만 아니라 상하분해 행렬의 희소성을 유지하기 위해서 수행되는데 적절한 재역산 시점의 선정은 단체법의 수행 속도와 수치적 안정성에 매우 큰 영향을 미친다. 재역산 기준에 관한 연구는 Reid등에 의해 이루어 졌으나 아직까지 일반적인 기준은 없는 상황이다.

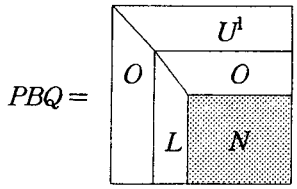
본 연구는 단체법의 기저행렬을 상하분해 형태로 보관하는 경우에 L과 U의 자료구조로 Gustavson 구조를 사용할 때의 효과적인 구현방법을 제시하고, 또한 많은 단체법 프로그램에서 사용되고 있는 수정된 Forrest-Tomlin방법의 효율적인 방법을 제시한다.

2. 기저행렬의 상하분해방법

2.1 기저행렬의 삼각화(Triangularization)

Bixby에 의해면 현실세계의 선형계획법 문제에서 기저행렬은 삼각행렬에 가까운 형태를 지니고 있다고 한다. 따라서 주어진 기저행렬에 대해 적절한 순서화를 수행하여 상삼각행렬에 가까운 형태를 만들게 되면 기저행렬의 상하분해시 소요되는 계산량을 많이 줄일 수 있다. 기저행렬의 삼각화는 Smith에 의해 고안되었고, Orchard-Hays에 의해 자세히 설명되었다. 기저행렬 B에 대해 행치환 P와 열치환 Q로 삼각화를 수행하게 되면 다음 [그림 1]과 같은 형태로 변하게 된다.

† 본 연구는 정보통신부의 97년도 대학기초연구지원사업(97-G-0683)의 지원을 받았음



[그림 1] 기저행렬의 삼각화

위 그림에서 삼각화가 되지 못한 부분인 N을 Nucleus라고 부르며 상하분해의 대상이 된다. 실제적으로 기저행렬의 삼각화를 수행하게 되면 Nucleus의 크기는 원 기저행렬의 크기에 비해 40-50%정도가 되므로 그 효과는 크다고 할 수 있다. 다음 [표 1]은 Netlib.문제의 최적기저에 대한 삼각화 수행에 따른 Nucleus의 크기를 나타내는 실험결과이다.

[표 1] 기저행렬의 삼각화에 따른 Nucleus의 크기

문제이름	행	열	기저크기	Nucleus
25fv47	822	1571	744	408
80bua3b	2263	9799	2021	146
bnl2	2325	3489	1672	450
cycle	1904	2857	1354	528
czprob	930	3523	492	5
d2q06c	2172	5167	2007	1038
d6cube	416	6184	403	303
degen3	1504	1818	1456	759
greenbea	2393	5405	1773	595
truss	1001	8806	1000	953

2.2 동적 마코비츠 순서화의 구현

마코비츠 순서화는 비대칭 행렬의 가우스 소거시 비영요소의 도입(fill-in)의 수를 최소화하고자 하는 경험적 방법이다. 마코비츠순서화는 도입 비영요소수의 예측치로 사용되는 마코비츠수의 계산방법에 따라 정적 마코비츠 순서화와 동적 마코비츠 순서화로 나누어진다. 정적 마코비츠순서화는 처음 주어진 기저행렬의 각 요소에 대한 마코비츠수를 모두 구하여 놓고 그것을 가우스소거 과정동안 계속해서 사용하는 방법으로 마코비츠수를 유지하는데 비용이 크게 들지 않는다. 반면 동적 마코비츠 순서화는 가우스소거 과정 중 매번 변하는 마코비츠수를 계속 갱신시켜나가는 방법으로 계산 비용은 크지만 효율적인 구현을 통해 효과적인 희소도를 얻어낼 수 있다. 동적 마코비츠순서화의 구현은 다음과 같이 두가지 측면에서 효율적으로 이루어져야 한다.

- 마코비츠수의 보관

마코비츠순서화는 마코비츠수를 계산하기 위해 부분행렬의 각 행과 열의 비영요소수를 알아야 한다. 부분행렬의 모든 요소에 대해 마코비츠수를 계산하여 최소의 마코비츠수를 갖는 비영요소를 검색하는 데는 많은 시간이 소요되므로 검색시간을 보다 줄이는 방법이 요구된다. 본 연구에서는 마코비츠순서화에서 검색시간을 줄이기 위해 행의 열의 비영요소 수를 두개의 양방향 연결리스트 자료구조를 이용하여 보관한다.

- 선회요소 탐색전략

위의 연결 리스트에는 부분행렬의 각 행·열이 비영요소 갯수에 의해 정렬된 순서로 나타나게 된다. 선회요소의 탐색을 위해서 비영요소가 작은 행과 열을 순서대로 마코비츠수를 계산한다. 이 때 이후에 발견될 수 있는 최소의 마코비츠수 값의 하한을 예측할 수 있는데 탐색과정에서 이 하한값보다 작거나 같은 최소의 마코비츠수를 갖는 비영요소를 찾으면 탐색을 멈추게 된다.

하한을 이용한 방법에서는 비영요소가 작은 순서대로 열과 행을 교대로 탐색하는데 비영요소가 k개의 열을 탐색할 때 이후에 발생될 수 있는 마코비츠수의 하한은 $(k-1)(k-1)$이다. 또한 비영요소가 k개인 행을 탐색할 때 이후에 발생될 수 있는 마코비츠수의 하한은 $k(k-1)$이다. 검색과정에서는 아직 검색하지 않은 비영요소들의 마코비츠수의 하한값을 예측하여, 선회연산 후보자의 마코비츠수가 이 하한값 이하가 되는 순간 검색이 끝내게 된다([12]). 선회연산이 일어날 때마다 이를 수정함으로써 항상 정렬된 순서를 유지할 수 있는데 연결리스트 수정에는 상수 시간이 소요된다.

2.3 수치적 안정성의 고려

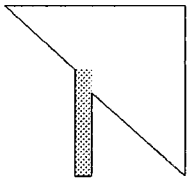
마코비츠순서화와 더불어 수치적 안정성을 고려한 선회요소선택 전략을 사용하게 되는데 널리 사용되고 있는 것이 threshold pivoting이다. 이 방법은 행렬의 한 요소가 속해 있는 행이나 열의 최대 절대치보다 그 요소의 절대치가 $\alpha(0<\alpha<1)$배 이상될 때에만 선회요소로 선택될 수 있도록 하는 방법으로 그 효과가 이미 검증되어 있다.

Threshold pivotin을 구현하기 위해서는 매회 선회요소 후보자가 속해 있는 행의 최대절대값 요소를 알아야 한다. 따라서 이에 대한 계산량이 상당히 많다. 이를 개선하기 위해서는 매회 선회연산이 일어날 때마다 최대절대값요소를 자료구조의 맨 앞으로 위치시켜 탐색시간을 줄여야 한다[10].

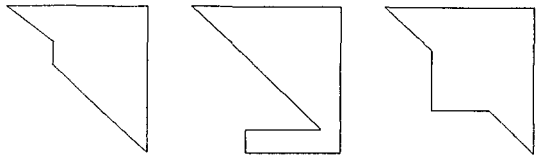
3. 상하분해요소의 수정

3.1 상하분해요소 수정 방법

단체법에서는 매회 기저가 한 열씩 수정되므로 이에 따른 기저행렬의 상하분해요소의 적절한 수정작업이 필요하게 된다. 일반적으로 기저행렬의 한 열이 바뀌게 되면 U는 다음과 같이 스파이크(spike)가 도입된 형태로 변하게 된다.



위와 같이 spike가 생긴 상삼각 요소를 상삼각행렬 형태로 복원시켜 주기 위한 방법으로는 전통적인 Bartels-Golub방법, Forrest-Tomlin방법, Reid방법 등이 있는데 각각의 방법은 다음과 같은 형태로 위 행렬을 치환변환시켜 대각요소 아래부분을 선회연산으로 소거한다.



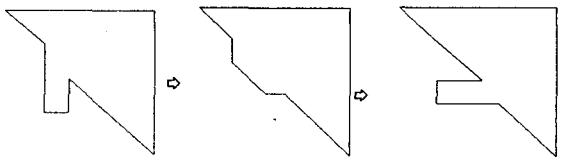
[Bartels-Golub] [Forrest-Tomlin] [Reid]

Bartels-Golub방법의 경우 대각선 아래의 요소들을 소거할 때 수치적 안정성을 고려한 행 치환연산을 수행할 수 있다는 장점이 있는 반면 U에 도입되는 비영요소수가 많다는 점이 단점이다. Forrest-Tomlin방법의 경우에는 U에 도입되는 비영요소수가 거의 없다는 점이 장점인 반면 수치적 안정성을 고려할 수 있는 행 치환연산을 하지 못한다는 단점이 있다. Reid방법의 경우에는 Bartels-Golub방법과 Forrest-Tomlin방법의 장점을 모두 취하고 있어 희소도와 수치적 안정성면에서 뛰어난 방법이지만 상하삼각요소 수정연산의 복잡도가 너무 크다는 단점이 있다.

현재 대부분의 상용 단체법 프로그램이나 공개 단체법 프로그램에서는 수정된 Forrest-Tomlin방법을 사용하고 있다. 이 방법은 아주 고속의 수정연산의 구현을 가능케하는 장점뿐만 아니라, 수치적 안정성 면에서도 Reid방법이나 Bartels-Golub방법보다 크게 떨어지지 않는다고 알려져 있다. 따라서 본 연구에서는 이 방법의 효율적인 구현방법에 대해 알아 본다.

3.2 수정된 Forrest-Tomlin방법

수정된 Forrest-Tomlin방법은 스파이크 열(spike column)의 희소도를 활용하는 방법으로 기존의 방법이 무조건 스파이크 열을 맨 뒤로 보내는 것이 아니라 스파이크 열의 비영요소중 가장 아래에 있는 요소에 해당되는 열로 치환하는 방법이다[9]. 즉 다음 [그림 2]와 같은 열 치환연산을 수행한다.



[그림 2] 수정된 Forrest-Tomlin방법

수정된 Forrest-Tomlin방법을 효율적으로 구현하는 방법으로 Suhl이 제시한 방법 두가지 이다. 첫째는 스파이크 열의 비영요소 위치를 스택에 미리 저장하여 스파이크 열을 상삼각 요소에 삽입할 때 스파이크 열 전체를 탐색하는 것을 방지하는 방법이다. 두번째 방법은 소거대상행의 열지수를 스택에 보관하는 방법이다. 이것은 소거대상행을 가우스 소거할 때 scatter/gather기법을 사용하므로 소거후 gather 과정을 유리하게 하기 위한 방법이다. 두가지 방법 모두 본 연구의 실험결과 우수한 성능을 보였다.

4. 상하분해요소의 자료구조

본 연구에서는 기저행렬 B를 L^{-1}와 U로 보관하는 것으로 가정한다. 기저행렬 관련 연산이 고속으로 이루어지기 위해서는 L^{-1}와 U를 효율적인 자료구조를 사용하여 보관하여야 한다.

4.1 L^{-1}의 보관

상하분해 요소중 L^{-1}는 기저행렬을 가우스소거할 때 행해지는 모든 선회연산의 정보를 가지고 있다. 기존의 L^{-1}보관방식은 선회연산의 정보 즉, 선회행, 소거대상행, 승수(multiplier)들을 각각 일차원배열에 보관하였다. 기저행렬의 상하분해시 일어나는 선회연산의 특징은 하나의 선회행에 관련된 일련의 선회연산이 연속적으로 일어난다는 것이고, Forrest-Tomlin방법을 사용하여 상하분해 요소를 수정하는 연산에서 일어나는 선회연산의 특징은 하나의 소거대상행에 관련된 일련의 선회연산이 연속적으로 일어난다는 것이다. 따라서, 두가지 선회연산에 대해 다른 L^{-1}보관 방식이 필요하게 된다. 즉, 행위주로 선회연산 정보를 보관하되 상하분해시의 선회연산과 상하분해요소 수정시의 선회연산에 대한 의미를 다르게 해석하여 L^{-1}를 사용하여야 한다. 또한, 상하분해시의 선회연산 정보는 행위주 뿐만 아니라 열위주로도 보관가능하므로 BTRAN, FTRAN에서 이를 활용할 수 있도록 하는 것이 훨씬 유리하다.

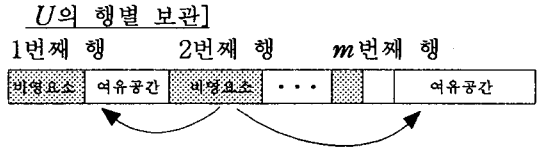
기존의 방법대로 선회연산 정보를 일차원배열에 일렬로 보관하게 되면 L^{-1}를 곱하는 연산시 진입열이나, 목적함수계수 벡터의 희소성을 활용할 수 없다는 단점이 있게 되는데 L^{-1}를 행위주나 열위주로 보관하게 되면 입력자료의 희소성을 충분히 활용할 수 있다는 장점이 생기게 된다. 즉, 예를 들어 진입열의 특정요소가 영일때 그 요소의 행에 관련된 선회연산을 모두 하지 않아도 되는 것이다. L^{-1}를 열위주나 행위주로 보관하려면 각 행이나 열의 시작주소를 나타내는 배열이 추가적으로 필요하게 된다. 또한 상하분해에 관련된 선회연산과 수정연산에 관련된 선회연산을 구분짓는 지시자 변수도 하나 필요하게 된다.

4.2 U의 보관

U를 보관하는 방법으로 Reid에 의해 사용된 연결리스트 구조와 Gustavson구조가 대표적이다. 연결리스트 구조가 비영요소의 삽입, 삭제가 비교적 자유로운 반면 행이나 열단위로 비영요소가 인접하여 보관되지 못하므로 기저행렬 관련 연산시 페이지 부재 현상이 자주 일어나게 되는 단점이 있다. 반면 Gustavson구조는 행과 열단위로 비영요소를 인접하게 보관하여 페이지 부재 현상을 극소화하는 방법으로 비영요소의 삽입, 삭제가 편리하지 못한 단점이 있다. 그러나, 본 연구의 실험 결과 효율적으로 구현된 Gustavson구조가 연결리스트 구조에 비해 단체법의 수행속도를 높이는데 훨씬 효과적이라는 것을 알 수 있었다.

Gustavson 구조는 U의 비영요소를 행과 열별로 모두 보관한다. 그래야만 BTRAN, FTRAN시 가장 효과적인 보관구조를 활용할 수 있기 때문이다. Gustavson 구조는 충분한 크기의 큰 배열속에 각 행과 열별로 일정한 크기의 저장공간을 먼저 확보하고 삽입, 삭제를 수행한다. 만일 비영요소의 삽입

이 많이 일어나게 되어 초기에 설정된 저장공간이 모자라게 되면 그 공간을 인접한 열이나 행의 저장공간으로 넘겨주고 큰 배열 뒤쪽에 여분으로 남아있는 공간에 저장공간 확보한다. 여분으로 남아 있는 저장공간이 없을 때에는 낭비되고 있는 각 행과 열별 여유공간을 압축하는 과정을 거치게 된다. 다음 그림은 두번째 행의 비영요소수가 저장공간의 크기를 넘었을 때 일어나는 연산을 설명한다.



[그림 3] Gustavson 구조

Gustavson구조를 구현하기 위해서는 각 비영요소를 행별, 열별로 보관하는 배열과 이 비영요소의 행, 열 지수를 보관하는 배열이 있고 각 행, 열의 시작 주소와 각 행, 열별 저장공간, 실제 보관된 비영요소수를 보관하는 배열이 있어야 한다.

한편, 한 행에 새로운 비영요소가 생길 경우에 그 행에 여유 공간이 없을 경우에는 맨 뒤로 그 행 전체를 옮기는 연산이 필요하므로, 해법의 수행 도중에 자료 내부에 사용하지 않는 공간이 많이 발생한다. 따라서 더 이상 옮길 공간이 없을 경우에는 비어있는 공간을 제거하는 압축 과정이 필요하다.

5. 재역산 기준

기저행렬을 상하분해 후 계속 상하분해 요소를 수정하다 보면 수치오차가 누적된다. 또한, 상하분해 요소의 비영요소수가 계속해서 증가하게 된다. 따라서 적절한 시점에서 재역산, 즉 재 상하분해가 필요하게 된다. Forrest-Tomlin방법은 상하분해 요소 수정시 수치적 안정성을 고려하지 못하므로 사후에 수치적 안정성을 제어하는 방법을 효과적으로 구현하여야 한다. 또한 기저행렬의 삼각화 수행으로 재역산 시간이 상하분해 요소 수정연산에 비해 그리 크지 않으므로 희소도가 어느 정도 이상 나빠졌을 경우에는 적절히 재역산을 수행하여야 단체법의 수행속도를 향상시킬 수 있다. 본 연구에서는 비영요소수가 30%이상 증가하거나 상삼각행렬의 대각요소의 절대값이 10^{-6}이하로 떨어졌을 경우에 재역산을 수행하였다.

6. 실험 결과

본 연구에서는 Gustavson 자료구조를 기반으로 수정 Forrest-Tomlin방법과 앞에서 제시한 여러가지 기법들을 구현하여 [3]에서 제시하였던 Reid법과 연결리스트 구조 기반의 상하분해단체법과의 비교실험을 수행하였다. 실험 대상 문제로는 Netlib. LP문제이며, 그 결과는 [표 2]와 같이 본 연구의 기법을 기반으로한 상하분해 단체법의 수행시간이 30%이상 우수한 것으로 판명되었다.

[표 2] 수정 Forrest-Tomlin방법 실험 결과

문제이름	제약식	변수	Reid	수정 Forrest-Tomlin
25fv47	822	1571	13.16	9.43
80bau3b	2263	9799	69.84	52.63
bnl2	2325	3489	19.87	13.49
d2q06c	2172	5167	148.42	86.67
dfl001	6072	12230	4591.31	2212.22
fit2p	3001	13525	391.42	209.83
greenbea	2393	5405	62.48	39.82
pilot87	2031	4883	1055.52	279.67
stocfor3	16676	15695	538.53	288.70
pilotja	941	1988	20.55	12.28

7. 결론

본 연구에서는 Suhl 등이 제안한 수정 Forrest-Tomlin방법의 효율적인 구현 방법에 대해 알아보았다. 그리고 이를 단체법 프로그램에 구현하여 Reid방법과 연결리스트 구조를 사용하는 것에 비해 30%이상의 단체법 프로그램 성능향상을 이루었다. 본 연구를 바탕으로 단체법 프로그램에서 보다 효율적으로 이용될 수 있는 상하분해 요소저장 방식의 개발과 이의 효과적인 구현기법들에 대한 연구가 필요하겠다.

8. 참고 문헌

[1] 박순달, 「선형계획법(3정판)」, 민영사, 1992

[2] 김우제, 안재근, 서용원, 성명기, 박순달, "단체법에서 기저행렬과 입력자료의 보관방법과 자료구조", 한국경영과학회/대한산업공학회 '95 춘계 공동학술대회 논문집, 1995

[3] 박찬규, 임성묵, 김우제, 박순달, "상하분해를 이용한 단체법의 구현에 관한 연구", 대한산업공학회 '97추계학술대회 논문집

[4] Bartels. R. H., G. H. Golub., "The Simplex method of linear programming using LU decomposition." Communication of ACM, 12, pp 266-268, 1969

[5] Duff. I. S., A. M. Erisman, J. K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford 1986

[6] Forrest. J. J. H., J. A. Tomlin, "Updating triangular factors of the basis to maintain sparsity in the product-form simplex method." *Math. Prog.* 2, pp 263-278, 1972

[7] Reid J. K., "A Sparsity-Exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases", *Computer Science and Systems Devision*, A.E.R.E., Harwell, CSS20 pp 1-23, 1975

[8] Saunders, M. A., "A fast, stable implementation of the Simplex method using Bartels-Golub updating", 1975

[9] Suhl, L. M., U. H. Suhl, "A fast LU update for linear programming", *Annals of Operations Research 43*, pp.33-47, 1993

[10] Suhl U. H., L. M. Suhl, "Computing Sparse LU Factorizations for Large-Scale Linear Programming Bases", *ORSA Journal on Computing*, Vol.2 No. 4, 1990