□ 硏 究 論 文 □

韓國國防經營分析學會誌

제 29 권, 제 1 호, 2003. 6. 30.

단체법에서 여러가지 상하 분해요소 수정방법들의 비교†
(A comparative study between various LU update methods in the simplex method)

임성묵*, 김기태**, 박순달***

Abstract

The simplex method requires basis update in each iteration, which is the most time consuming process. Several methods have been developed for the update of basis which is represented in LU factorized form, such as Bartels-Golub's method, Forrest-Tomlin's method, Reid's method, Saunders's method, etc. In this research, we compare between the updating methods in terms of sparsity, data structure and computing time issues. The analysis is mainly based on the computational experience.

† 본 연구는 한국과학재단의 특정기초연구과제(과제번호 98-0200-07-01-2)의 지원을 받았음

* 한국전산원

** 삼성 SDS

*** 서울대학교 산업공학과

1. 서론

선형계획법(Linear programming)은 수리계획법 모형 중에서 가장 널리 사용되고 있는 것 중에 하나로서, 지난 50년간 그 유용성이 잘 검증되었다. 특히, 수송계획, 포트폴리오 구성, 생산계획 등에 널리 사용되어 왔으며, 수리계획 모형의 기본 모형으로서 그 중요성 및 활용성이 한층 높다[1].

단체법(Simplex method)은 1947년에 G. B. Dantzig에 의해 선형계획법을 푸는 해법으로 개발되었고, 그 성능 개선을 위한 많은 연구가 이루어져 왔으며, 그 결과 실생활의 선형계획법 문제를 효율적으로 풀 수 있는 해법 중의 하나로 자리매김하고 있다. 반면, 단체법은 다항식의 복잡도를 가지지 못하여, 이론적인 면에서는 그 효율성을 인정받지 못한다[17].

다항식의 복잡도를 가지는 선형계획법의 해법은 1979년에 Khachian이 개발한 타원체법(Ellipsoid algorithm)이 최초인데, 그 성능은 단체법에 비해 크게 열등하여 현재 잘 사용되지 않고 있다. 한편, 1984년 Karmarkar는 가능해 공간의 내부로부터 최적해를 찾아가는 새로운 다항식 복잡도의 내부점 방법을 개발하였고[10], 이를 시초로 내부점 방법에 대한 많은 연구들이 진행되어 왔고, 그 결과 대형의 선형계획법 문제의 경우 단체법을 능가하는 성능을 보여주고 있다[11].

대형의 선형계획법 문제에 대해 단체법이 내부점 방법에 비해 열등한 반면, 중소형 문제에 대해서는 더 우수한 성능을 보이는 경우가 많고, 특히 네트워크 최적화와 같은 형태의 문제에 있어서는 내부점 방법에 비해 아주 우수하다. 또한, 정수계획법의 해법으로서의 분지한계법에서와 같이, 서로 유사하게 연관된 선형계획법 문제를 연속적으로 풀어야 하는 경우에는 단체법이 내부점 방법에 비해 절대 유리하다. 이는 내부점 방법의 경우, 주어진 문제와 유사한 문제의 최적해를 이용하여 현재의 문제를 푸는 방법이 용이하지 않기 때문이다.

선형계획법의 많은 응용분야에서는 최적해로서 기저해가 더 유용성이 높다. 단체법은 기저인 최적해를 출력하는데, 이는 최적면의 해를 출력하는 내부점 방법에 비해 장점이 된다. 내부점 방법에서는 이러한 단점을 극복하기 위해, 최적면의 해로부터 기저인 최적해를 구하는 기저추출 방법을 사용하게 되는데, Megiddo의 알고리즘이 대표적이다[12]. 한편, 이러한 기저추출 방법은 단체법의 한 변형으로 볼 수 있다.

이러한 이유로, 단체법에 대한 효과적인 구현방법에 대한 연구는 아주 중요하며, 실용적인 타당성이 있다. 그리고, 최근까지 단체법 프로그램의 구현방법에 대한 연구에는 많은 진전이 있었고, 대형의 선형계획법 문제를 고속으로 양질의 해를 출력하는 우수한 프로그램들이 많이 개발되었다. 예를 들어, ILOG의 CPLEX, IBM의 OSL 등과 같은 상용 프로그램에서는 내부점 방법과 함께 단체법을 구현하고 있으며, 대형의 선형계획법 문제를 효과적으로 풀어내고 있다.

단체법의 계산과정을 살펴보면 기저역행렬과 관련된 연산이 거의 70-90%정도의 비중을 차지한다[16]. 따라서 고속 단체법 프로그램의 개발에 있어서 기저역행렬의 보관·유지 방법의 효율화는 필수적인 요소가 된다.

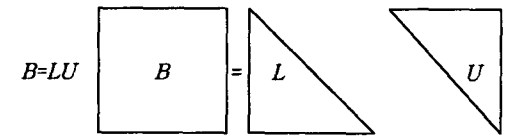
기저역행렬의 보관 형태는 많은 변화가 있어왔

는데, 처음에는 기저행렬의 역을 명시적으로 보관하는 명시형(Explicit form)에서 출발하였고, 다음에는 기저역행렬을 기본행렬들의 곱으로 표현하는 적산형(Product form)이 개발되어 많이 사용되었었다. 그 이후, Bartels, Golub등에 의해 상하분해형으로 기저행렬을 보관하고, 갱신하는 기법이 개발되어 현재까지 많이 애용되고 있다. 상하분해형은 계산 효율성면에서 우수할 뿐만 아니라 기저행렬의 희소도를 충분히 활용할 수 있다는 장점이 있어 현재 대부분의 단체법 프로그램에서 사용되고 있다[3][4]. 한편, 기저행렬을 QR분해하여 유지하는 기법[7]도 개발되어 수치적 안정성면에서 우수성을 보였으나 계산 복잡도가 커서 잘 쓰이지 않고 있다.

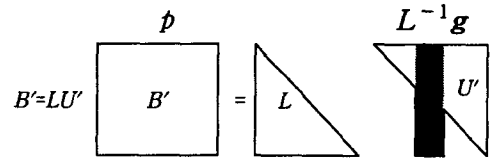
기저행렬을 상하분해하여 유지하는 상하분해 단체법에서는 기저행렬을 상하분해하는 기법의 효율화 뿐만아니라 상하분해 요소를 효과적으로 갱신하여 유지하는 기법의 개발이 필수적인데, 이러한 갱신방법으로는 현재까지 1960년대 말의 Bartels-Golub 방법[5]에서부터 시작하여, Forrest- Tomlin 방법[8], Reid 방법[13], Saunders 방법[14], 그리고 1993년 Suhl에 의한 수정 Forrest-Tomlin 방법[15]까지 다양한 방법이 개발되었다. Suhl의 연구 이후에는 상하분해 요소 갱신방법에 대한 연구는 이루어지지 않았다. 상하분해를 이용하여 기저역행렬을 유지하는 대부분의 최근 단체법 프로그램들은 위에서 언급한 상하분해 요소 갱신방법들 중의 하나를 사용한다. 본 연구에서는 이러한 여러 가지 상하분해 요소 갱신방법들을 서로 비교 고찰하여 그 특성을 파악한다.

2. 상하분해 요소 수정방법들

단체법에서는 기저에 속하는 열들이 매회 하나씩 바뀌게 되므로 기저행렬을 상하분해하여 보관할 때 상하분해 요소를 매회 수정하는 방법이 필요하게 된다. 즉, 다음과 같이 기저행렬 B가 상하분해 되었다고 할 때,



B의 p번째 열이 임의의 열 g와 교환되었다면 상하분해 요소는 다음과 같이 상삼각행렬에 도입열(spike) $L^{-1}g$가 발생한 형태가 된다.

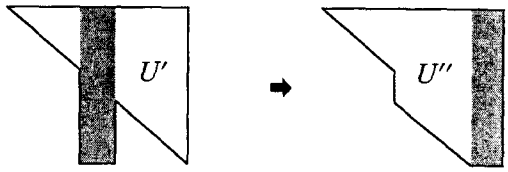


위와 같이 상삼각행렬에 도입열이 발생한 경우에 이를 제거하여 상삼각행렬의 형태를 복원하는 연산이 필요하게 되는데 이 연산의 형태에 따라 여러 가지 상하분해 요소 갱신방법의 변형들이 존재하게 된다. 본 절에는 이러한 여러 가지 갱신방법들에 대해 알아본다.

2.1 Bartels-Golub 방법

Bartels-Golub 방법은 상삼각행렬에서 발생한 도입열을 상삼각행렬의 맨 뒤로 치환하는 방법을 사용한다. 도입열을 맨 뒤로 치환하게 되면 다음과

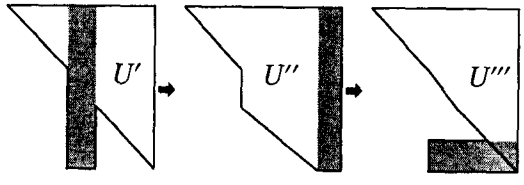
같은 형태가 되는데,



이 때, 대각 이하의 원소가 생겨나게 된다. Bartels-Golub 방법에서는 대각요소를 이용하여 대각이하의 원소를 소거하여 상삼각행렬 형태를 복원한다. 소거과정 중 수치적 안정성을 고려하여 행치환(Row permutation)이 허용되므로 Bartels-Golub 방법에서는 일반적으로 수치적 안정성이 어느 정도 보장된다.

2.2 Forrest-Tomlin 방법

Forrest-Tomlin 방법에서는 Bartels-Golub 방법보다 한번의 행 치환연산을 더 수행한다. 즉, 도입열에 대응되는 행을 상삼각행렬의 맨 아래로 치환한다. 그러면 다음과 같은 형태가 만들어지게 된다.

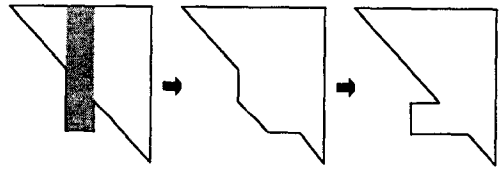


그러면, U'''에서 음영이 있는 부분을 소거하여 상삼각행렬 형태를 복원할 수 있다. 그러나, Forrest-Tomlin 방법에서는 소거연산 시 행 치환이 허용되지 않으므로 수치적 안정성을 고려하기가 힘들다. 하지만 비영요소의 도입(Fill-in)은 Bartels- Golub에 비해 작다고 할 수 있어 희소도 면에서는 우수성을 보인다.

2.3 수정 Forrest-Tomlin 방법

수정 Forrest-Tomlin 방법[15]은 Forrest-Tomlin 방법에서 도입열의 희소성을 활용한 방법이다. 일반적으로 대형의 선형계획법 문제를 단체법으로 풀 때, 도입열의 희소도는 상당히 높아서 이에 대한 활용이 필요하다.

수정 Forrest-Tomlin 방법에서는 Forrest-Tomlin 방법에서 처럼 도입열을 상삼각행렬의 맨 뒤로 치환하지 않고, 도입열의 비영요소 패턴이 전체 행에 걸쳐 존재하지 않는다는 것을 이용하여 적절한 열로 치환한다. 즉, 다음 그림과 같은 치환연산이 이루어진다.

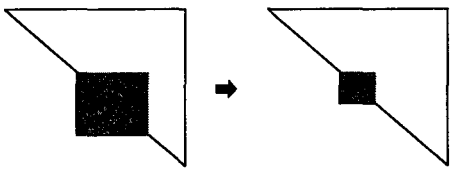


위와 같이 치환 연산을 수행하게 되면 실제로 소거해야 할 대상이 줄어들게 되므로 상하분해 요소 갱신 속도가 빨라지게 되는 이점이 있다.

2.4 Reid 방법

Reid는 Bartels-Golub 방법이 수치적 안정성 면에서는 우수함을 보이지만 희소도면에서 열등함을 발견하고 희소도를 활용할 수 있도록 Bartels-Golub 방법을 변형하였다. Reid 방법은 다소 복잡한 행·열 치환 과정을 거쳐서 소거해야할 대상을 줄이는 것으로 희소도를 활용하였다. 소거대상이 되는 상삼각행렬의 부분행렬을 범프(Bump)라고 정의하였을 때, 실제로 가우스 소거연산의 양은 범프의 크기에 비례한다. 따라서, Reid 방법은 이 범프

의 크기를 줄이기 위한 해·열 치환 과정을 주된 내용으로 한다. 이러한 개념을 그림으로 표시하면 다음과 같다. 아래의 그림에서 음영이 있는 부분이 범프이다.



Reid 방법에서 사용하는 행·열 치환 과정은 다음과 같이 요약될 수 있다.

단계 1

현재의 bump 내에서 singleton 열(column)들을 현재 bump의 가장 왼쪽으로 옮기는 행과 열의 대칭치환(symmetric permutation)을 수행한다. 단, singleton 열이란 비영요소의 수가 하나인 열을 의미한다.

더이상 singleton 열(column)이 없을 때까지 반복한다.

단계 2

현재의 bump 내에서 singleton 행(row)들을 현재 bump의 가장 아래쪽으로 옮기는 행과 열의 대칭치환(symmetric permutation)을 수행한다.

더이상 singleton 행(row)가 없을 때까지 반복한다.

단계 3

현재의 bump 내에서 singleton 열(column)이 있는지 조사한다. 없으면 단계 4로 간다. 있으면 해당 singleton 요소를 bump의 가장 왼쪽 위로 옮기는 행과 열의 비대칭치환 (unsymmetric permutation)을 수행한다.

더이상 singleton 열(column)이 없을 때까지 반복한다.

단계 4

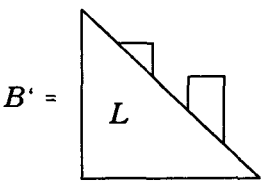
가우스 소거법(Gaussian elimination)을 통하여 상삼각행렬로 만든다.

한편, Reid 방법이 수치적 안정성 면이나 희소도 면에서 우수함을 보이지만 위의 계산 과정을 살펴볼 때 갱신 방법의 계산 복잡도는 타 방법들에 비해 상당히 높은 편이라고 할 수 있다.

2.5 Saunders 방법

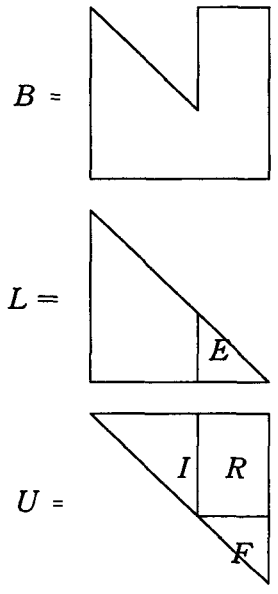
Reid 방법이 Bartels-Golub 방법의 수치적 안정성을 살리고, 상삼각행렬의 희소도를 유지하는 우수한 방법이지만, 상삼각행렬 전체를 주기억장치에 보관해야 하므로 대형 문제를 다루는 경우 기억장소를 많이 필요로 하게 된다. Saunders 방법에서는 Reid 방법의 장점인 수치적 안정성과 희소도를 유지하면서 상삼각행렬의 보관에 보조기억장치를 사용하여 기억장소를 절약하려는 목적을 가지고 있다.

Saunders 방법은 초기에 기저행렬을 상하분해할 때 P3, P4 순서화를 사용한다. 즉, 기저행렬에 대해 다음과 같은 형태가 되도록 치환연산을 수행한다.



즉, 하삼각행렬에서 도입열(spike)가 몇 개 존재하는 형태로 기저행렬을 순서화 한다. 그런 다음 도

입열 부분을 오른쪽으로 모두 이동한 다음 상하분해를 수행하는데 그러면 다음과 같은 상하분해 요소가 생성된다.



위의 하삼각행렬에서 E부분이 B와 다른 부분이고, 상삼각행렬에서 I는 단위 행렬을 표현한다. 이와 같이 상하분해되었을 때, 도입열이 발생하게 되면 Forrest-Tomlin 방법과 동일하게 치환연산을 수행하면 상삼각행렬의 맨 아래부분에 튀어나온 행이 발생하게 되는데 이를 소거하기 위해 Reid 방법을 사용한다.

3. 희소도

기저역행렬 관련 연산을 고속화하려면 기저행렬의 희소성을 효과적으로 활용하여야 한다. 일반적으로 대형의 선형계획법 문제의 기저행렬은 상당히 희소도가 높고, 또한 Bixby의 관찰에 의하면 삼각행렬의 형태를 지닌다고 한다[6]. 본 절에서는 앞에서 설명한 상하분해 요소 갱신 방법들의 희소도에 관계된 특성을 알아본다.

상하분해 요소의 희소도는 상삼각행렬과 하삼각행렬로 나누어 생각해 볼 수 있다. 본 연구에서는 기저행렬을 L^{-1}와 U로 분해하는 경우를 가정한다.

Bartels-Golub 방법의 경우 희소도를 고려하는 측면이 약하다. 즉, 대각원소들을 이용하여 대각이하의 원소를 소거할 때 비영요소의 도입이 아무 통제 없이 이루어 진다. 그러므로 Bartels-Golub 방법에서 갱신과정을 더해 갈수록 상삼각행렬의 희소도는 상당히 나빠질 우려가 있다.

Forrest-Tomlin 방법이나 수정 Forrest-Tomlin 방법의 경우는 비영요소가 도입되는 부분이 일부분으로 한정되어 희소도의 유지가 쉽게 이루어진다. 이러한 특성은 상삼각행렬을 보조기억장치에 보관하던 때에는 상당히 큰 이점으로 작용하였다. 반면, 도입열의 비영요소들이 하삼각행렬로 흡수되어 하삼각행렬의 비영요소가 증가할 우려가 있다.

Reid 방법의 경우 비영요소의 도입은 범프내에서 이루어진다. Reid 방법은 하삼각행렬의 희소도를 높이는 방법이라고 할 수 있는데 이는 범프의 크기를 줄이는 치환연산으로 도입열의 비영요소를 최대한 상삼각행렬내로 흡수하여 타 방법에 비해 가우스 소거연산의 수가 줄어들기 때문이다. 그러나, 소거연산으로 발생하는 비영요소의 도입보다는 치환연산으로 발생하는 비영요소의 도입이 훨씬 작다고 할 수 있어 Reid 방법은 희소도의 유지에 유리한 방법이라고 할 수 있다. Reid 방법을 통해 범프의 크기를 얼마나 줄일 수 있느냐에 따라 희소도

에 대한 효율성이 달라지게 된다. 그러나 일반적으로 기저행렬은 상당히 희소하므로 Reid 방법은 행·열 치환을 통해 범프의 크기를 많이 줄일 수 있어 희소도 유지가 효과적으로 이루어진다고 할 수 있다.

Sauders 방법은 Reid 방법이 희소도의 측면에서 우수하나, 상삼각행렬 전체를 주기억장치에 보관해야 하는 문제에 대한 대안으로 제시된 방법이다. 그리고 상삼각행렬 일부(I, R)의 보관에 보조 기억 장치를 사용할 수 있도록 하였다. 즉, 갱신과정의 선회연산으로 비영요소의 도입이 일어나는 부분은 F로 한정되어 있어 I와 R에는 비영요소의 도입이 거의 일어나지 않는다. 그러나 Reid 방법에 비해 초기 상하분해 요소의 희소도는 떨어진다.

상하분해 요소의 각 수정방법들이 상하분해 요소의 희소도에 실제로 어떠한 영향을 미치는지를 알아보기 위하여, 단체법의 매회에서 상하분해 요소내의 비영요소 증가율을 실험적으로 살펴보았다. 상하분해 요소내의 비영요소 수는 U의 비영요소 수와 L^{-1}의 비영요소 수를 합한 값을 의미한다. 다음 [표 2]은 각 수정방법 간의 평균 비영요소 증가율을 서로 비교 정리한 것이다.

한편, 실험을 위해 사용한 프로그램은 LPAKO ver 4.8[2]이며, ALPHA(Memory: 128M, CPU: 533MHz, OS: Linux)에서 GCC('-O3' 옵션)로 컴파일되어 구동되었다. 실험 대상으로 삼은 문제는 NETLIB 문제[9] 중 대형 문제들이고, 그 특성들은 [표 1]과 같다. 이러한 실험 환경은 이후 모든 요소에 대한 비교실험에서 동일하게 적용되었다.

[표 1] 실험 대상 문제의 특성

문제	제약식 개수	변수 개수	비영요소 개수
25fv47	821	1571	10400
bnl2	2324	3489	13999
80bau3b	2262	9799	21002
cycle	1903	2857	20720
degen3	1503	1818	24646
maros	846	1443	9614
nesm	662	2923	13288
piloja	940	1988	14698
woodlp	244	2594	70215

[표 2] 각 수정방법 간의 평균 비영요소 증가율 비교

문제	Bartels-Golub	Forrest-Tomlin	수정 Forrest-Tomlin	Reid
25fv47	71.48	37.41	35.17	35.26
bnl2	48.46	41.12	40.87	31.41
80bau3b	91.79	51.84	50.11	41.83
cycle	44.29	22.89	21.05	23.11
degen3	113.26	69.33	66.21	78.62
maros	95.41	48.12	48.03	33.12
nesm	60.55	34.81	33.41	31.18
piloja	75.24	49.12	50.07	36.16
woodlp	81.75	17.82	16.10	15.14
평균	75.80	41.38	40.11	36.20

(단위 : %)

이 실험 결과를 바탕으로, 각 방법 간의 평균 비영요소 증가율이 서로 차이가 나는지 알아보기 위해 t-검정법(쌍체비교)을 실시하였다. 유의수준 5%로 검정을 실시한 결과, Bartels-Golub 방법이 가장 높은 평균 비영요소 증가율을 보였고, Reid 방법과 수정 Forrest-Tomlin 방법이 가장 작은 증가율을 나타내었다. 단, Reid 방법과 수정 Forrest-Tomlin 방법 간에는 유의한 차이가 있다

고 볼 수 없었다. 그리고, Forrest-Tomlin 방법의 평균 비영요소 증가율은 Bartels-Golub 방법에 비하면 작고, 수정 Forrest-Tomlin 방법에 비하면 크다는 결론을 얻었다. 결론적으로, Bartels-Golub 방법이 희소도 유지 측면에서 가장 열등한 반면, 수정 Forrest-Tomlin 방법과 Reid 방법이 가장 우수하였다. 한편, t-검정(쌍체비교)는 Microsoft사의 Excel 프로그램을 이용하여 수행하였고, 자세한 결과는 부록에 수록하였다.

4. 수치적 안정성

단체법에서 기저역행렬이 관련된 연산으로는 진입열의 수정, 단체승수의 계산이 주를 이룬다. 만일 진입열의 수정에서 수치적 오차가 심하게 발생할 경우에는 탈락변수의 선정에 문제가 생겨 비가역 기저행렬을 생성할 가능성이 생기게 된다. 또한, 단체승수의 계산에 오차가 발생할 경우에는 최적판정이나 진입변수의 선정에 문제가 발생할 우려가 있다. 따라서, 기저역행렬 관련 연산의 정확성은 단체법 프로그램의 성능에 막대한 영향을 끼치게 된다. 지금까지 효과적으로 개발되어온 상하분해 방법으로 기저행렬을 분해할 경우 기저역행렬 관련 연산이 큰 오차없이 수행될 수 있다. 그러나, 계속적으로 상하분해 요소를 수정하다보면 수치적 오차가 누적되어 관련 연산의 정확도가 떨어지게 된다. 따라서, 상하분해 요소 갱신방법에서 수치적 안정성을 고려하는 것은 상당히 중요하다.

Bartels-Golub 방법의 경우 갱신 과정에서 수치적 안정성을 기할 수 있다는 장점이 있다. 즉, 대각요소와 대각이하요소 중 절대값이 큰 요소를 선회요소로 선택할 수 있어 수치오차의 범위를 미리 알 수 있다.

반면, Forrest-Tomlin 방법이나 수정 Forrest-Tomlin 방법의 경우에는 상하분해 요소를 갱신하는 과정에서 수치적 안정을 위한 치환연산을 수행할 수 없다. 따라서, 갱신 후에 심한 수치적 불안정이 초래될 수도 있다. 따라서, 이 방법들에서는 갱신연산 후에 수치적 안정성을 검사하여 재역산 과정을 수행하여야 한다.

Reid 방법의 경우, 범프를 소거하는 과정에서 여러 가지 선회요소 선택전략을 취할 수 있다. 예를 들어 paritial pivoting, full pivoting, threshold pivoting 등이 사용될 수 있다. 또한, 희소도를 고려한 선회요소 선택방법도 사용될 수 있다. Reid 방법은 도입열의 비영요소를 최대한 상삼각행렬에 흡수시켜 가우스 소거연산을 줄이는 특성이 있는데, 이는 수치적 불안정을 초래하는 가우스 연산의 양을 줄여 수치적 안정성을 더욱 높이고 있다.

Saunders 방법은 상삼각행렬 형태로 복원하기 위해 Reid 방법을 최종적으로 사용하므로 수치적 안정성을 고려하여 가우스 소거연산을 수행할 수 있다.

가우스 소거연산의 결과로 수치적 안정성이 얼마나 저해되었는지를 살펴보기 위한 기준으로 많이 사용되는 것은 growth factor이다. Growth factor는 가우스 소거연산이 이루어지기 전 행렬의 최대절대값 요소와 가우스 소거연산 후 행렬의 최대절대값 요소의 비율로 구해진다. 수치적으로 불안정한 가우스 소거연산이 수행된다면 growth factor의 값은 크게 나타나는데, 본 연구에서는 이 값을 통해 각 수정방법의 수치적 안정성 면을 비교 평가하

기로 한다. 다음 [표 3]는 각 수정방법을 사용하는 단체법의 매회에서 상삼각행렬 U에 대한 growth factor를 구하여 그 평균값을 나타낸 것이다. 실험 환경은 3절의 희소도에 대한 실험에서와 동일하다.

[표 3] 각 수정방법 간의 평균 growth factor 비교

문제	Bartels-Golub	Forrest-Tomlin	수정 Forrest-Tomlin	Reid
25fv47	100.7	198.3	154.2	57.9
bnl2	87.4	291.4	117.5	81.6
80bau3b	191.4	315.2	268.4	85.4
cycle	14.8	25.4	11.7	9.5
degen3	58.1	98.5	79.5	47.1
maros	91.4	108.6	121.6	15.7
nesm	38.9	67.2	59.4	25.4
piloja	725.5	1054.1	798.5	615.4
woodlp	62.8	118.4	107.6	12.7
평균	152.3	253.0	190.9	105.6

이 실험 결과를 바탕으로, 각 방법 간의 평균 growth factor가 서로 차이 나는지 알아보기 위해 t-검정법(쌍체비교)을 실시하였다. 유의수준 5%로 검정을 실시한 결과, Reid 방법이 가장 작은 평균 growth factor 값을 나타내었고, 다음으로 수정 Forrest-Tomlin 방법, Bartels-Golub 방법, Forrest-Tomlin 방법의 순으로 그 값이 커졌다. 결론적으로, Reid법이 수치적 안정성 면에서 타 방법을 능가하였고, Forrest-Tomlin 방법이 가장 불안정한 면을 보였다. 이에 대한 보다 자세한 검정 결과는 부록에 수록하였다.

5. 자료구조와 관련 연산

상하분해 요소를 위한 자료구조는 기저역행렬 관련 연산의 고속화에 결정적인 영향을 미친다. 그러므로 자료구조의 효율적인 설계는 아주 중요하다. 그리고 단체법에서 상하분해요소가 쓰이는 계산과정은 진입열의 수정(FTRAN)과 단체승수의 계산(BTRAN)인데 다음과 같다.

(FTRAN) $\overline{A}_{.j} = B^{-1}A_{.j} = U^{-1}L^{-1}A_{.j}$

(BTRAN) $\pi = B^{-T}c_B = L^{-T}U^{-T}c_B$

상하분해 요소 갱신 연산은 선회연산에 대한 정보인 L^{-1} eta의 연속적인 생성과정으로 볼 수 있다. 그리고, Saunders 방법을 제외한 나머지 상하분해 요소 갱신 방법에서는 기저행렬을 상하분해할 때 가우스 소거법을 사용할 수 있다. 가우스 소거방법을 사용할 때, 기저행렬은 하삼각행렬의 역인 L^{-1}와 상삼각행렬 U로 분해하는 방법이다. 따라서, Saunders 방법이외의 방법에서는 기저행렬을 L^{-1}와 U로 유지할 수 있다. 그러나, Saunders 방법에서는 기저행렬을 상하분해하여 L과 U로 표현하여야 하므로 가우스 소거법을 사용할 수 없고 Doolittle 법이나 Crout 법을 사용하여야 한다. 따라서, Saunders 방법에서는 기저행렬을 초기 하삼각행렬인 L_0, 기저갱신과정의 L^{-1}, 상삼각행렬 U로 유지하여야 한다.

상삼각행렬을 보관하는 자료구조로는 연결리스트구조와 Gustavson 구조가 많이 쓰인다. 연결리스트구조는 자료의 동적인 삽입·삭제에 유리하고, Gustavson 구조는 자료의 연속으로 인한 장점이 있다. 모든 상하분해 요소 갱신방법은 이러한 자료구조를 활용할 수 있다. 그러나, Saunders 방법의 경우 비영요소의 도입이 일어나지 않는 I, R

을 저장하는데 열압축구조를 사용하여 효율성을 높일 수도 있다. 상삼각행렬을 보관할 때, 행·열위주로 동시에 보관하여 기저역행렬이 곱해지는 입력벡터의 희소도를 충분히 활용할 수 있어야 한다.

기저갱신과정의 하삼각행렬은 L^{-1} eta 형태로 보관되는데, 이는 갱신과정 중 일어나는 선회연산의 정보를 보관하고 있다. 즉, 소거대상행, 선회행, 승수 등을 보관한다. 따라서, 각 갱신방법의 특성에 따라 서로 다른 형태를 띠게 된다. Bartels-Golub 방법의 경우 선회연산이 대각의 원소로 대각 바로 아래의 원소를 소거하는 과정이므로 서로 연속된 선회연산 간에 공통적인 정보가 없다. 반면, Forrest-Tomlin 방법이나 수정 Forrest-Tomlin 방법의 경우, 하나의 소거대상행에 관련된 선회연산이 연속적으로 일어난다. 두가지 방법간의 이러한 차이는 L^{-1} eta 의 보관방식의 차이를 발생시킨다. 즉, Bartels-Golub 방법의 경우에는 선회연산의 정보를 일련으로 나열하는 식으로 보관하여야 하는 반면, Forrest-Tomlin 방법이나 수정 Forrest-Tomlin 방법의 경우 선회행과 승수를 일련으로 저장하고 소거대상행에 대해서는 같은 소거대상행이 연속되는 길이와 위치를 따로 보관하면 되므로 자료 보관의 효율성이 더 크다고 할 수 있다. Reid 방법이나 Saunders 방법의 경우는 Bartels-Golub 방법과 선회연산의 형태를 미리 알 수 없으므로 선회연산의 정보를 그냥 일련으로 보관하여야 한다. 따라서, 하삼각행렬의 저장형태로 볼 때는 Forrest-Tomlin 방법이나 수정 Forrest-Tomlin 방법이 가장 우수하다고 할 수 있다.

상하분해 시에 얻어지는 초기 하삼각행렬은 Saunders 방법의 경우에는 L_0 형태가 되고 다른 방법의 경우에는 L_0^{-1} 형태가 된다. Saunders 방법에서는 L_0 의 특수한 형태 때문에 그 저장을 효율화 할 수 있다. 즉, L_0 는 도입열이 발생하지 않는 부분에서 기저행렬과 동일하므로 그 열지수만 보관하는 것으로 충분하다. Saunders 방법의 이외의 방법에서는 가우스 소거법을 이용하여 상하분해를 수행할 때 초기 L_0^{-1} 이 얻어지는데 이에 대해 갱신과정의 L^{-1} 와는 달리 행·열위주로 동시에 보관할 수 있다. 따라서, 입력 벡터의 희소도를 충분히 활용할 수 있다. 즉, 입력 벡터의 영에 관계되는 선회연산을 모두 생략할 수 있는 것이다.

6. 계산 속도

상하분해 요소의 갱신연산은 단체법에서 매회 일어나므로 그 계산 속도도 중요하다고 할 수 있다. 각 갱신방법에 따라 그 계산 복잡도는 많이 다르다. Bartels-Golub 방법이나 Forrest-Tomlin 방법, 수정 Forrest-Tomlin 방법은 그 계산이 아주 간단하다. 즉, 몇번의 간단한 치환연산만으로 끝난다. 그러나 그 이후 발생하는 가우스 소거연산의 양은 미리 예측할 수 없다. 반면, Reid 방법의 경우 범프의 크기를 줄이는 순서화 방법의 과정은 상당히 복잡하다고 할 수 있다. 최악의 경우 네 번의 검색이 필요하게 된다. 이러한 큰 복잡도는 효율적인 구현에 저해요인이 되기도 한다. 마지막으로 Saunders 방법의 경우, 초기 상하분해시에 P3, P4 순서화를 먼저 수행해야 하므로 계산 부하가 크다. 또한, 범프를 소거하기 전 Reid 방법을 적용하는 데에도 많은 계산량이 필요하므로 가장 계산 복잡도가 큰 방

법이라고 할 수 있다. 그러나, Reid 방법이나 Saunders 방법의 경우 가우스 소거연산을 수행하여야할 범프의 크기를 줄이는 과정을 거치게 됨에 따라 일반적으로 타 방법에 비해 가우스 소거연산이 적게 일어난다고 예측할 수 있다.

7. 전체 성능의 비교

궁극적으로 단체법의 수행속도를 높이고자 함이 우리의 목적이므로 각 수정방법의 특성들이 어떤 식으로 단체법의 수행속도에 영향을 미치는지 알아보아야 할 것이다. 다음 [표 4]은 각 수정방법에 따른 단체법의 수행속도를 서로 비교한 것이다. 실험은 이전 절에서의 실험 환경과 동일하며, 상하분해 요소 수정방법 이외의 단체법의 모든 구성 요소는 똑같은 상황에서 실험이 수행되었다. 즉, 평가방법으로는 dynamic steepest-edge 방법이 사용되었고, 자료구조로는 Gustavson 구조가 사용되었다. 또한 문제를 풀기 전에 사전처리 및 규모화가 수행되었다.

이 실험 결과를 바탕으로, 각 방법 간의 단체법 수행 속도가 서로 차이 나는지 알아보기 위해 t-검정법(쌍체비교)을 실시하였다. 유의수준 5%로 검정을 실시한 결과, 수정 Forrest-Tomlin 방법을 사용하는 경우에 가장 빠른 속도를 나타내었고, Bartels-Golub 방법을 사용하는 경우에 가장 느린 속도를 나타냈었다. 한편, Forrest-Tomlin 방법과 Reid 방법은 수행 속도면에서 서로 유의한 차이를 보이지 못했다. 이에 대한 보다 자세한 검정 결과는 부록에 수록하였다.

[표 4] 각 수정방법에 따른 단체법 수행속도 비교

문제	Bartels-Golub	Forrest-Tomlin	수정 Forrest-Tomlin	Reid
25fv47	17.93	7.18	6.74	9.18
bnl2	21.65	7.07	7.10	8.64
80bau3b	113.84	48.16	44.21	58.41
cycle	9.76	3.59	2.70	3.48
degen3	48.33	19.48	18.12	23.18
maros	10.89	3.85	2.51	4.12
nesm	18.95	10.74	9.64	9.91
piloja	21.13	10.38	8.65	10.44
woodlp	30.17	23.82	18.49	22.14
평균	32.52	14.92	13.13	16.61

(단위 : 초)

8. 결론

본 연구에서는, 기저행렬을 상하분해하여 유지하는 상하분해 단체법에서 상하분해 요소를 매회 갱신하여 유지하는 방법들에 대한 비교·분석을 수행하였다.

각 방법들을 비교하기 위해서 희소도 유지, 수치적 안정성, 자료구조와 관련 연산, 계산 속도 그리고 단체법의 전체 수행 성능 등의 기준을 통해 평가하였고, 주로 실험적 결과를 통해 분석하였다. 그 결과, 희소도 유지 측면에서는 Reid 방법 및 수정 Forrest-Tomlin 방법이 가장 우수하였고, 수치적 안정성 측면에서는 Reid 방법이 가장 우수한 성능을 보였다. 반면, 단체법의 전체적인 수행 성능 측면에서는 수정 Forrest-Tomlin 방법이 가장 우수하였다. Reid 방법의 경우, 수정 Forrest-Tomlin 방법에 비해 희소도 측면에서 유사한 특성을 보였

고 수치적 안정성 측면에서는 우수하였지만, 그 계산 복잡도로 인해 수정연산의 시간 소요가 비교적 커서, 단체법의 수행 속도면에서 수정 Forrest-Tomlin 방법에 비해 성능이 열등한 것으로 분석된다.

참고문헌

[1] 박순달, 「선형계획법(3정판)」, 민영사, 1992

[2] 박순달, 「LPAKO ver 4.8 사용자 매뉴얼」. 2000

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

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

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

[6] Bixby, R.E., "Progress in Linear Programming", *ORSA Journal on Computing*, 6, pp. 15-22, 1994

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

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

[9] Gay, D.M., "Electronic mail distribution of linear programming test problems", *Mathematical Programming Society Committee on Algorithms Newsletter*, 13, 1985

[10] Karmarkar, N. K., "A New Polynomial-Time Algorithm for Linear Programming", *Combinatorica*, 4, 4, pp. 373-395, 1984

[11] Lustig, I. J., R. E. Marsten and D. F. Shanno, "Interior Point Methods for Linear Programming: Computational State of the Art", *ORSA Journal on Computing*, 6, 1, pp. 1-14, 1994

[12] Megiddo, N., "On finding primal- and dual-optimal bases", *ORSA Journal on Computing*, 3, 1, 1991.

[13] 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

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

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

[16] Suhl U. H., L. M. Suhl, "Computing Sparse LU Factorizations for Large-Scale Linear Programming Bases", *ORSA Journal on Computing*, 2, 4, 1990

[17] Terlaky T., S. Zhang, "Pivot rules for linear programming : A survey on recent theoretical developments", *Annals of Operations Research* 46, pp. 203-233, 1993

<부록> T-검정법(쌍체비교) 결과

(※ 프로그램 : Microsoft Excel, 유의수준 : 5%)

• 각 수정방법 간의 평균 비영요소 증가율 비교

구분	Bartels-Golub	Forrest-Tomlin
평균	75.80333333	41.38444444
분산	508.84755	243.9870278
관측수	9	9
피어슨 상관 계수	0.677229871	
가설 평균차	0	
자유도	8	
t 통계량	6.219971205	
P(T<=t) 단측 검정	0.000126925	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.00025385	
t 기각치 양측 검정	2.306005626	

구분	Forrest-Tomlin	수정 Forrest-Tomlin
평균	41.38444444	40.11333333
분산	243.9870278	244.0966
관측수	9	9
피어슨 상관 계수	0.996803726	
가설 평균차	0	
자유도	8	
t 통계량	3.053050333	
P(T<=t) 단측 검정	0.007873952	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.015747904	
t 기각치 양측 검정	2.306005626	

구분	Bartels-Golub	수정 Forrest-Tomlin
평균	75.80333333	40.11333333
분산	508.84755	244.0966
관측수	9	9
피어슨 상관 계수	0.655374888	
가설 평균차	0	
자유도	8	
t 통계량	6.276612255	
P(T<=t) 단측 검정	0.000119368	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.000238736	
t 기각치 양측 검정	2.306005626	

구분	Forrest-Tomlin	Reid
평균	41.38444444	36.20333333
분산	243.9870278	313.067625
관측수	9	9
피어슨 상관 계수	0.904391302	
가설 평균차	0	
자유도	8	
t 통계량	2.056095206	
P(T<=t) 단측 검정	0.036903135	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.073806269	
t 기각치 양측 검정	2.306005626	

구분	Bartels-Golub	Reid
평균	75.80333333	36.20333333
분산	508.84755	313.067625
관측수	9	9
피어슨 상관 계수	0.663142574	
가설 평균차	0	
자유도	8	
t 통계량	6.945625761	
P(T<=t) 단측 검정	5.94697E-05	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.000118939	
t 기각치 양측 검정	2.306005626	

구분	수정 Forrest-Tomlin	Reid
평균	40.11333333	36.20333333
분산	244.0966	313.067625
관측수	9	9
피어슨 상관 계수	0.872005351	
가설 평균차	0	
자유도	8	
t 통계량	1.3540039	
P(T<=t) 단측 검정	0.10637003	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.212740059	
t 기각치 양측 검정	2.306005626	

• 각 수정방법 간의 평균 growth factor 비교

구분	Bartels-Golub	Forrest-Tomlin
평균	152.3333333	253.0111111
분산	48665.015	99907.35861
관측수	9	9
피어슨 상관 계수	0.985458185	
가설 평균차	0	
자유도	8	
t 통계량	-2.861066186	
P(T<=t) 단측 검정	0.010558391	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.021116781	
t 기각치 양측 검정	2.306005626	

구분	Forrest-Tomlin	수정 Forrest-Tomlin
평균	253.0111111	190.9333333
분산	99907.35861	56934.335
관측수	9	9
피어슨 상관 계수	0.985162614	
가설 평균차	0	
자유도	8	
t 통계량	2.051606296	
P(T<=t) 단측 검정	0.037161284	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.074322567	
t 기각치 양측 검정	2.306005626	

구분	Bartels-Golub	수정 Forrest-Tomlin
평균	152.3333333	190.9333333
분산	48665.015	56934.335
관측수	9	9
피어슨 상관 계수	0.996640579	
가설 평균차	0	
자유도	8	
t 통계량	-4.447469003	
P(T<=t) 단측 검정	0.001073256	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.002146511	
t 기각치 양측 검정	2.306005626	

구분	Forrest-Tomlin	Reid
평균	253.0111111	105.6333333
분산	99907.35861	37369.66
관측수	9	9
피어슨 상관 계수	0.98222318	
가설 평균차	0	
자유도	8	
t 통계량	3.3668708	
P(T<=t) 단측 검정	0.004915543	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.009831087	
t 기각치 양측 검정	2.306005626	

구분	Bartels-Golub	Reid
평균	152.3333333	105.6333333
분산	48665.015	37369.66
관측수	9	9
피어슨 상관 계수	0.987967901	
가설 평균차	0	
자유도	8	
t 통계량	3.329195332	
P(T<=t) 단측 검정	0.00519842	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.01039684	
t 기각치 양측 검정	2.306005626	

구분	수정 Forrest-Tomlin	Reid
평균	190.9333333	105.6333333
분산	56934.335	37369.66
관측수	9	9
피어슨 상관 계수	0.97546224	
가설 평균차	0	
자유도	8	
t 통계량	3.895439705	
P(T<=t) 단측 검정	0.00228699	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.00457398	
t 기각치 양측 검정	2.306005626	

• 각 수정방법에 따른 단체법 수행속도 비교

구분	Bartels-Golub	Forrest-Tomlin
평균	32.51666667	14.91888889
분산	1062.399675	201.9286361
관측수	9	9
피어슨 상관 계수	0.960295112	
가설 평균차	0	
자유도	8	
t 통계량	2.727096384	
P(T<=t) 단측 검정	0.012981066	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.025962132	
t 기각치 양측 검정	2.306005626	

구분	Forrest-Tomlin	수정 Forrest-Tomlin
평균	14.91888889	13.12888889
분산	201.9286361	168.8261111
관측수	9	9
피어슨 상관 계수	0.995882658	
가설 평균차	0	
자유도	8	
t 통계량	3.099776879	
P(T<=t) 단측 검정	0.00733536	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.014670719	
t 기각치 양측 검정	2.306005626	

구분	Bartels-Golub	수정 Forrest-Tomlin
평균	32.51666667	13.12888889
분산	1062.399675	168.8261111
관측수	9	9
피어슨 상관 계수	0.979310045	
가설 평균차	0	
자유도	8	
t 통계량	2.901896658	
P(T<=t) 단측 검정	0.00991697	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.01983394	
t 기각치 양측 검정	2.306005626	

구분	Forrest-Tomlin	Reid
평균	14.91888889	16.61111111
분산	201.9286361	293.8834361
관측수	9	9
피어슨 상관 계수	0.991217607	
가설 평균차	0	
자유도	8	
t 통계량	-1.414525221	
P(T<=t) 단측 검정	0.09746362	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.19492724	
t 기각치 양측 검정	2.306005626	

구분	Bartels-Golub	Reid
평균	32.51666667	16.61111111
분산	1062.399675	293.8834361
관측수	9	9
피어슨 상관 계수	0.987727176	
가설 평균차	0	
자유도	8	
t 통계량	3.003113173	
P(T<=t) 단측 검정	0.00849544	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.01699088	
t 기각치 양측 검정	2.306005626	

구분	수정 Forrest-Tomlin	Reid
평균	13.12888889	16.61111111
분산	168.8261111	293.8834361
관측수	9	9
피어슨 상관 계수	0.997622097	
가설 평균차	0	
자유도	8	
t 통계량	-2.443406063	
P(T<=t) 단측 검정	0.020174786	
t 기각치 단측 검정	1.85954832	
P(T<=t) 양측 검정	0.040349572	
t 기각치 양측 검정	2.306005626	