# 數理計劃 소프트웨어 *LinPro*의 설계 및 구현†

양광민*

# Design and Implementation of Mathematical Programming Software – *LinPro*†

Kwang Min Yang*

## ABSTRACT

This study addresses basic requirements for mathematical programming software, discusses considerations in designing these software, implementation issues facing in these types of applications development, and shows some examples of codes being developed in the course. This type of projects requires long and ever-changing evolutionary phases. The experience is therefore, valuable in suggesting some useful hints which may be salvaged for similar projects as well as providing reusable codes. In particular, scanning and parsing the free-format inputs, symbol table management, mixed-language programming, and data structures dealing with large sparse matrices are indispensable to many management science software development. Extensions to be made are also discussed.

## 1. 서 론

線型계획은 경영과학 기법 중 가장 널리 사용되는 대표적인 도구 중 하나다. 많이 이용되고 있는 이유가 여러가지가 있을 수 있겠으나 문제가 定型化될 수 있고, 해법이 반복적이어서 컴퓨터를 이용하는 것이 안성맞춤이므로 일찌기 이 문제를 다루는 소프트웨어가 컴퓨터의 등장과 함께 어느 것보다도 먼저 개발될 수 있었음에 기인한다.

선형계획 해법을 다루는 소프트웨어는 거의 컴퓨터가 商用化되는 시기부터 개발되기 시작하여 경영에 컴퓨터가 도입된 시기와 일치하는 역사를 갖고 있다. 이 해법에 관한 소프트웨어는 현재 서베이[18]된 것만도 크고 작은 것을 합쳐 수십개가 되나 최근의 노력[1, 2]을 제외하면 거의 대부분 상용의 외국 것으로서 기업, 연구소의 경우는

† 이 논문은 1994학년도 교내 일반교수연구비 지원에 의하여 연구되었음.
* 중앙대학교 경영학과

고액의 외화를 지불하고 구입하여 사용하고, 학교의 경우는 대체로 복제하여 사용하는 것이 우리의 현실이다. 본 연구는 이러한 소프트웨어를 자체 개발하기 위한 효과적인 방법으로 경영기법 소프트웨어에 공통적으로 사용될 수 있는 front-end processor로 쓸 수 있는 prototype에 해당하는 것을 개발하는 것에 초점을 맞추기로 한다. 대부분의 경영과학기법의 해법 자체는 이미 출판되어 public domain에 속하며, 효율적은 아니라도 많은 것이 이미 컴퓨터코드로서 공개되어 있는 실정임을 감안한다면 이러한 접근방법도 실효성이 있다고 보여진다.

외국의 소프트웨어가 결여한 한글메뉴의 사용과 사용자 설명서의 한글화로 이용자의 편의성을 높이고 결과물을 shareware로 공개하므로서 학생들의 교육과 연구에 부담없이 이용할 수 있도록 하고, 개발된 연구결과를 유사한 응용프로그램에도 移植하여 활용토록 하고자 함이 연구의 바람이다.

## 2. 소프트웨어 요구분석

### 2.1 수리계획 소프트웨어의 일반적 요구사항

Sharda의 탁상용 線型計劃 소프트웨어에 관한 매2년마다의 *MS/OR Today*의 서베이[18]에서 우리는 선형계획 소프트웨어의 수, 사용플랫폼, 대상문제크기, 입력양식, 가격 등 그 기능이 다양하게 변화되어 왔음을 알 수 있다.

위의 선형계획 소프트웨어의 서어베이에서는 다음과 같은 주요 속성을 따라 조사·정리됐다.

- 작동 플랫폼
- 문제 크기 (行수, 列수, 非零數)
- 가격
- 입력양식 (MPS, 계산표양식, 고유양식)
- 계산표 호환여부
- 모형화 언어와의 연계성
- 원시코드 제공여부
- 정수계획 포함여부
- 여타 제공기능

위의 속성 중 대상문제의 크기, 사용 플랫폼, 입력양식, 여타 프로그램과의 연계성 및 확장성이 수리계획 소프트웨어의 설계시 비중있게 고려되어야 할 일반적 사항이라고 생각되며, 이들 모두는 제안 소프트웨어 *LinPro*설계시에 고려되었다.

### 2.2 제안 소프트웨어 *LinPro* 요구사항

소프트웨어의 개발 및 구현이 점진적, 進化的 양상을 보인다는 것을 감안한다면 위의 수리계획 일반 요구사항에 한글 관련 요구를 추가한 것, 다른 파일을 가져오기, 여타 기존 소프트웨어와의 접속 등을 만족하는 것을 잠정적 요구로 정했다.

제안 소프트웨어의 구체적인 요구사항을 나열하면 다음과 같다.

- 기존 소프트웨어 도구사용의 극대화 (컴파일러기법 도입, 테스터, 인스톨러 등)
- 자유형 입력 방식
- 자료구조와 해법 알고리즘의 독립·조화
- 모델링언어의 수용
- 그래픽 출력
- User interface (friendliness, callability 등) 강조
- 타 응용프로그램과의 연계성

앞으로 확장을 예상하여 다음을 고려하기로 한다.

- generic programming (C++ STL) 고려
- 移植性 강화

# 3. 소프트웨어 설계

모든 소프트웨어의 개발이 그렇듯이 한번에 완벽한 것이 되리라는 기대는 희망사항일 뿐이다. 특히 인적, 시간적자원이 한정된 경우는 더욱 그러하다. 여기서도 시스템이 점진적으로 보완되어 진화하는 경로를 따르는 개방된 시스템으로 시작한다.

소프트웨어 설계와 관련한 기본 원칙으로 다음을 고려한다. 사용 언어는 C 또는 C++를 기본으로 하고 이미 만들어진 부분이나 효율성을 위해 피치 못한 부분은 assembly 또는 Fortran을 허용키로 한다. 이 경우 주의해야 할 점은 혼합언어프로그래밍이 허용되는 시스템을 선정하는 일이다. 표준언어, 표준라이브러리를 사용하는 것도 잊지 말아야 한다. 모든 코드는 재사용이 가능하도록 개발한다.

자료구조는 어떠한 해법에도 큰 무리없이, 아니면 간단한 조작의 추가로 지원되고, 작동될 수 있는 구조이어야 하며 이때 공간, 시간효율성이 간과되지 않도록 설계한다. 사용 수리계획알고리즘에 독립적으로 작동되도록 자료구조, 프로그램을 설계한다.

## 3.1 개관

여타 소프트웨어와 마찬가지로 수리계획소프트웨어도 입력된 문제를 푸는 주기능 이외에 다양한 서어비스 기능을 제공해야 한다. 이러한 여러 기능을 이용자에게 편리하게 제공하기 위해 소프트웨어는 대체로 커맨드방식(command-driven) 또는 메뉴방식(menu-driven)의 두가지 중 하나를 선택하여 설계된다. 두가지 방식이 각각 장단점이 있어 두가지의 장점을 택한 하이브리드형식인 메뉴방식에 핫키를 겸한 방식을 채택하기로 한다. 각 메뉴가 어떤 기능을 수행하는지를 알 수 있는 도움메뉴도 추가하여 따로 매뉴얼을 찾아보지 않아도 되도록 고안한다. 풀다운메뉴에 콘텍스트 센시티브(context-sensitive)형으로 설계하고 메뉴는 한글처리한다.

소프트웨어의 확장, 보수·유지를 위해 부품으로 모듈화하고 가능한한 generic 프로그래밍을 도입하여 再사용성을 제고한다. 기존의 루틴을 접착하기 쉽도록 혼합프로그래밍이 가능한 컴파일러들을 선정한다.

최적화해법을 효율적으로 수행함은 물론 실제적인 대규모문제를 간결하게 저장할 수 있고 여러가지 입력양식을 허용하는 자료구조를 가지게 설계한다. 자가제작 루틴의 사용보다 되도록 이미 성능이 입증된 소프트웨어공학적 도구 (商用의 YACC, installer, S/W 시험기, 에디터 등)를 가급적 많이 채용한다. 이를 위해서는 제안 소프트웨어가 유연한 인터페이스를 갖도록 설계해야 하는 어려움은 있으나, 이 방법은 성능이 향상된 부품의 교환을 용이하게 하여 추후의 업그레이드시 생산성을 높게하는 이점이 있다.

## 3.2 메뉴 구성

제안소프트웨어 ***LinPro***는 다음과 같은 계층적 메뉴구조를 갖도록 설계됐다.

◆ 도움말

도움말, 명령어, 버전, 소프트웨어 소개

◆ 입력

새 문제, 문제보기, 저장하기, 불러오기, ASCII로 저장하기

◆ 최적해

민감도분석없이, 민감도분석, 그래프해

◆ 편집

디렉토리에서 찾기, 파일이름으로 찾기

◆ 파일관리

ASCII, MPS형식, Tri-tuple형식, 읽어보기

◆ 인쇄

파일인쇄, 현WS, 프린터설정

◆ 전환

바꾸기, 원래대로, 훌터보기

◆ 도스명령어

디렉토리, 쉘

◆ 프로그램 끝

예, 아니오

필요한 경우 위의 부메뉴에 한계층 아래의 부메뉴가 포함된 경우도 있다. 도움말은 F1 key에 할당되어 있어 어느 때나 사용 가능하고 F1 사용 후 다시 아무 key나 누르면 원래 상태로 되돌아가도록 설계됐다.

풀다운 메뉴의 구현에 관한 것은 Dorfman[6]의 것을 참고로 하는 것도 효율적이다.

### 3.3 自由型 입력 방식

서론에서 지적했듯이 선형계획기법이 여타 경영계획기법과 비교해 실용성이 높은 기법으로 정착할 수 있었던 것은 모형화 가능성(定型性) 및 해법의 컴퓨터 적용 용이성에 기인한다고 할 수 있다.

컴퓨터 이용의 역사에서 보아 왔듯이, 초기의 컴퓨터는 그 성능의 제약에서 모든 처리가 사용자 입장에서 보다도 컴퓨터 기계 중심에서 처리되어 오다가 차츰 인간에게 편리하도록 바뀌어 왔다는 사실에 유념할 필요가 있다. 즉 인간과 컴퓨터와의 의사소통을 우리는 컴퓨터프로그래밍 언어란 매체를 통해 수행해 왔고 이 프로그래밍 언어는 아주 원시적인 기계어로 부터 출발해서 어셈블리어, 컴파일러어, 간이언어 등을 거치면서 가능한 한 우리의 사고가 투명하게 전달될 수 있는 자연어를 목표로 진화해 왔다. 한편 정보처리 과정의 설계에서 중요한 고려사항 중 하나는 잡음이 혼입되지 않도록 처리경로를 단축하는 것이다. 이들을 고려해 선형계획의 수식표현 그대로가 컴퓨터에서 (중간 변환과정 없이) 입력되도록 하는 형식이 自由型 입력(free-format input) 방식이다.

선형계획의 입력방식에는 여러가지가 있다: 즉, 固定列형식, IBM의 MPS형식[13], 계산표형식, 자유형(*Lindo*형)형식 등이다.

제안소프트웨어 *LinPro*는 위의 입력형식을 모두 갖추도록 할 것이나 본 절에서는 특히 자유형 입력에 관해 심도있게 서술하기로 한다. 자유형식을 사용하는 소프트웨어는 이미 전에도 있었으나 통상적으로 *Lindo*[16]형식이라고 부른다. Cunningham의 *Lingo*[5]에서는 *Lindo*형식과 거의 동일하나 보다 일관성있는 구문법을 갖는 자유형식을 택하고 있다.

수리계획문제의 자유형 입력방식의 구문을 다음과 같이 정하기로 한다.

〈문제〉

maximize 3x + 5.2y

subject to

x−3y≤5

−2x + y ≥ 2.1

x, y ≥ 0

〈유효한 입력의 예〉

maximize 3x+5.2y st x−3y 〈5 −2x+y〉= 2.1

end

또는,

max 3x + 5.2y

subj to

x−3y<5

−2x + y>2.1

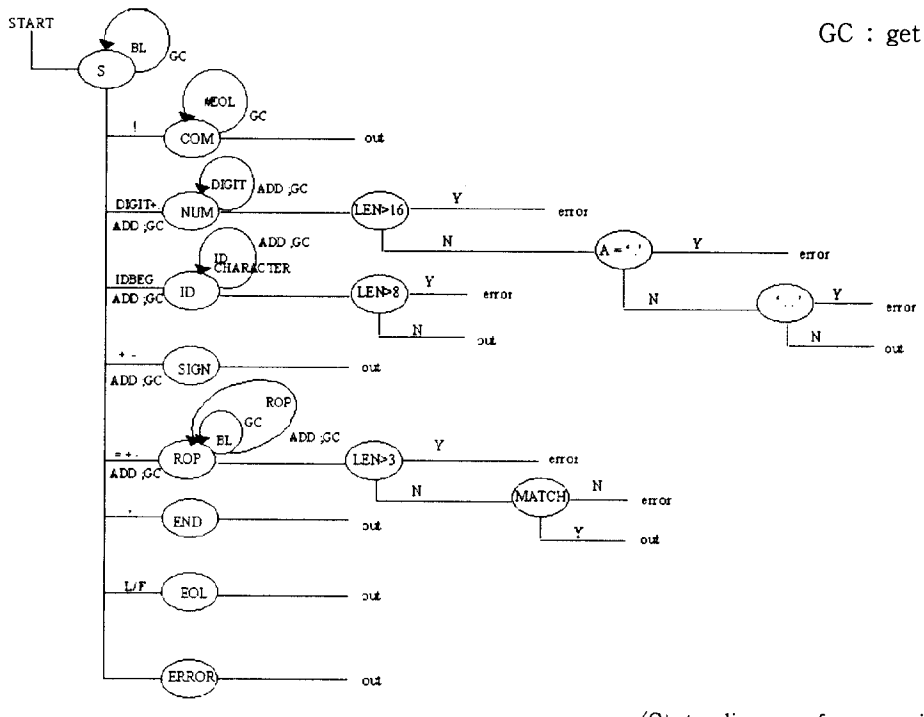
end
또는,
max
3x+5.2
y
st x−
3y<5 −2x + y>2.1 end

〈delimiters〉
목적함수를 표시하는 심볼 (max 또는 min)
제약식을 표시하는 심볼 (subject to, subj to, st, 또는 st)
문제가 끝났음을 표시하는 심볼 (end)

위의 구문을 갖는 문제(원시프로그램)를 최적화(optimizer)모듈의 입력(목표프로그램)으로 변환하는 과정은 통상적으로 컴파일러의 기본요소인 스캐너와 파서가 담당하는 부분이며 이의 구성과 설계를 설명하도록 한다.

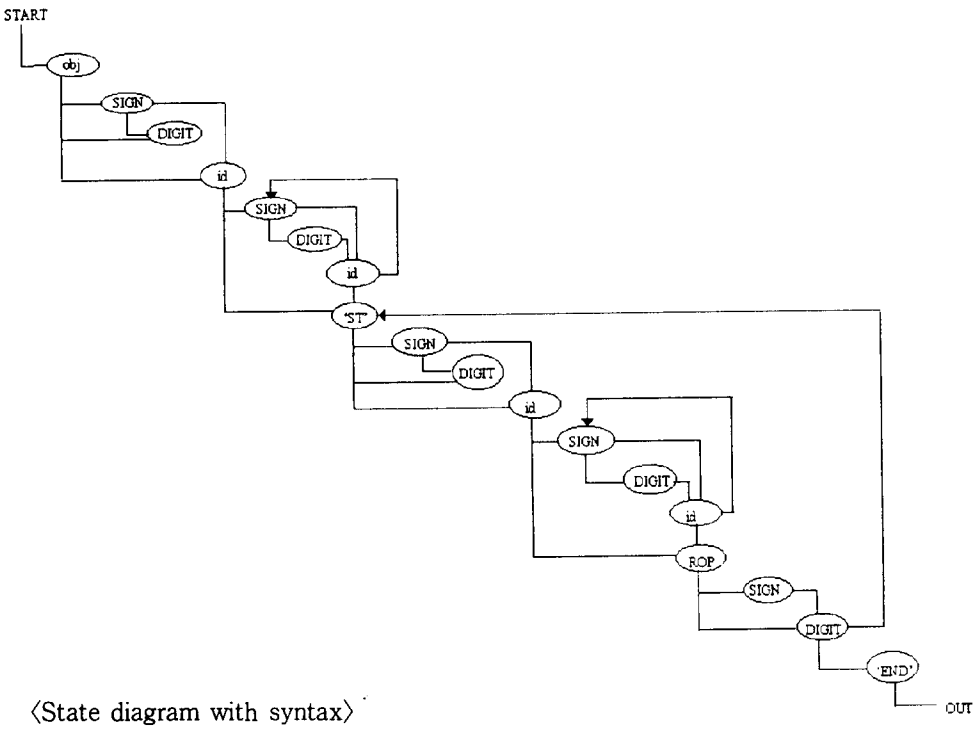
〈grammer〉
〈comment〉::=!|〈comment〉 letter
|〈comment〉 digit
| 〈comment〉 blank
〈identifier〉::= letter|〈identifier〉 letter
|〈identifier〉 digit
〈number〉 ::= digit|〈number〉 digit
〈sign〉 ::= + | −
〈relational operator〉 ::= = | 〈 | 〉
〈comment〉 ::= letter|〈comment〉 letter
|〈comment〉 digit
〈end of line〉 ::= L/F
〈end of problem〉 ::= END

기호들을 스캐닝하기 위한 상태도를 보이면 다음과 같다.




〈State diagram for scanning symbols〉

syntax를 포함한 상태도를 다음에 보인다.




〈State diagram with syntax〉

실제로 Microsoft C로서 구현한 스캐너와 파서를 〈부록-1〉에 예로서 보였다. 제안시스템에서는 실행효율성을 위해 수작업으로 코딩했으나 시스템들이 제공하는 YACC (예로서 PCYACC) 등을 이용하는 것이 생산성도 높고 보수·유지에도 능률적이다.

### 3.4 텍스트 편집기

컴퓨터에서 처리를 위해 자료 입력시 착오는 불가피하다. 어떤 경우는 시간이 경과하면서 자료의 유효성이 변화하여 자연적으로 수정이 요구된다. 어떤 형식의 입력방식이거나 입력된 것을 수정하는 일은 피할 수 없고 그러기 위해서는 편집기(text editor)가 필수적이다.

편집기는 대체로 行편집기 (line editor; EDLIN), 화면편집기 (screen editor; EDIT), 워드프로세서 등으로 구별할 수 있으나 화면편집기와 워드프로세서의 구별은 명확하지 않다고 보겠으며 통상적인 워드프로세서의 잡다한 기능은 수리계획 소프트웨어에는 필요하지 않다.

입력형식에 따라 필요한 기능이 다를 수 있으나 대체로 행편집기 또는 간단한 프로그래머용 화면편집기 정도면 충분하다. 자유형 입력양식을 채택하는 경우는 모든 기능을 갖춘 일반 화면편집기 수준의 편집기가 요구된다. 이미 만들어진 편집기를 필요시마다 불러서 사용하는 것은 별도로 편집기를 만들지 않아도 되는 장점은 있으나 메모리 사용량, 실행속도 면에서 불리하므로 편집기를 프로그램에 통합한 내장 프로그램으로 설계

함이 타당하다.

수리계획 소프트웨어에서 필요한 편집기 설계시에는 다음의 것들이 포함되면 적절하다.

- 문자 입력 및 삽입
- 문자 삭제
- 행 삭제
- 행 이동
- 상하 스크롤
- 페이지 상하 이동
- 파일 맨앞으로
- 파일 끝으로
- 스트링 찾기
- 찾기 및 바꾸기

제안 소프트웨어의 편집기를 살펴보아도 알 수 있듯이 편집기를 만드는 작업이 대단히 어려운 일이라고 말할 수는 없겠으나 그런대로 몇가지 신경을 써야 하는 부분이 있음은 사실이다. 그 하나는 편집코자하는 파일과 스크린을 同調하는 일이다. 즉 두개의 별개의 물리적파일을 다루므로 현재의 위치를 변화할 때 항상 同期化해야 하는 것을 잊지 말아야 한다. 또한 편집코자 하는 파일은 일차원적 파일인 반면 스크린 파일은 2차원이어 이 또한 주의를 요구하는 부분이다. 한행의 끝에는 C/R 그리고/또는 L/F와 같은 과외의 문자가 있음도 어렵게 하는 부분의 하나다. 편집되는 문자들은 메모리 버퍼에 있게 되는데 우리는 항상 이 버퍼의 앞뒤를 벗어나서 움직이지 않도록 하는 일을 점검해야한다. 즉 경계조건(boundary conditions)의 만족여부를 항상 주의해야 하는 일도 프로그램의 골치 아픈 부분이다.

여타 워드프로세서, 응용프로그램에서 작성된 파일을 가져오는 경우에도 여러가지 예기치 못한 일이 발생할 수 있다. 이는 특정 응용 프로그램마다 서로 다른 파일형식을 채택하므로 인해 생기는 문제이다. 예를 들자면 파일의 처음과 끝을 표시하는 방법이 특별하여 이를 인식 못할 경우 경계를 넘어버려 예상치 못한 결과를 가져 오는 경우이다.

이미 만들어진 공개된 또는 상용의 편집기를 수정·이용하는 방안도 노력을 절감할 수 있는 방편이므로 생각해 볼 수 있다. 예로서 Schildt [15]의 screen editor subsystem, Sub Systems의 TE Developer's Kit, Thomson-Davis의 tde Editor, Gary Osborn의 SE Editor (CUG #331) 등을 이용해 볼 수 있다.

### 3.5 문제 저장을 위한 자료구조

학생들이 수리계획모형을 학습하는데 필요한 정도의 소형문제를 저장하고 해를 구하는 소프트웨어는 문제저장을 위한 자료구조로 어떤 형식을 택하거나 별다른 문제를 제기하지 않으나, 현장에서 만나는 실제의 대규모문제를 풀 수 있는 소프트웨어는 문제행렬을 저장하고 다루는데 있어 시간효율적, 공간효율적인 자료구조를 갖추는 것이 필요불가결하다. 최근에는 몇십만 변수를 갖는 대규모문제도 개인용컴퓨터에서 처리할 수 있는 정도가 되었다[3].

자료구조를 고안함에 있어 고려해야 할 사항은 다음과 같다.

- 문제의 특성
- 문제의 저장 및 검색방향
- 연산순서
- 소요시간
- 可用메모리 용량

공학분야의 응용이 아닌 경영에서 다루는 문제는 그 규모가 매우 큰 반면 문제행렬이 성글다는 것을 그 특성으로 들 수 있다. 이는 유사한 구조의 것이 부서별 또는 분기별로 반복되는 블록형이라는 기업의 적용대상 문제의 특성에서 기인한

다. 문제행렬의 非零계수의 밀도가 5%미만인 것이 전형적이다. 이 특성은 해법소프트웨어를 설계하는데도 많은 것을 시사한다.

예로서 간단히 변수 1000, 제약식 1000인 문제를 생각해 볼 때, 문제행렬의 계수를 full matrix 형태에 real*8로 보관한다고 할 경우 적어도 문제표현에만 8MB의 메모리가 필요하다는 계산이 된다. 따라서 full matrix형태로는 실제적인 문제에 접근할 수 없다는 사실을 알 수 있고 좀 더 효율적인 방안이 요구됨을 실감하게 된다. 물론 2차 기억장치에 문제를 보관하는 방안을 생각할 수 있으나 이는 실행속도의 저하를 수반하므로 고려할 필요성이 없다.

상업용 수리계획 소프트웨어는 모두 문제행렬의 표현에서 非零계수만을 보관한다. 비영계수만을 저장하는 방식은 full matrix에는 필요치 않은 비영계수의 위치 즉 행 및 열의 값(整數)을 추가로 저장해야하는 문제가 있으나 이는 비영계수당 행 및 열에 각각 2 bytes씩 4 bytes면 해결되어 비영계수밀도가 낮은 경우에는 효율적이다. 무작위가 아니고 列別(column-wise) 또는 行別로 저장할 경우는 비영계수당 2 bytes로 줄일 수도 있음은 물론이다. 대부분의 상용 소프트웨어가 비영계수를 모두 저장하는 방식을 취함으로 필요기억공간은 비영계수의 수에 비례하게 된다. 그러나 우리가 취급해야 하는 실제 문제를 깊이 분석해 보면 저장공간을 더욱 줄일 수 있는 방안을 고안해 낼 수 있다.

실제에서 보는 현실 문제의 계수의 분포를 분석해 보면 전혀 중복되지 않는 서로 다른 계수의 수는 대규모 문제의 경우에도 1000개를 넘지 않는 것이 일반적이다. 즉 .0, 1.0, −1.0, 10.0, 2.0과 같은 계수의 빈도가 높다. 이 사실에서 비영계수를 모두 저장할 것이 아니라 문제를 입력하면서 계수를 점검하여 겹치지 않는 서로 다른 계수만을 저장하고 이미 입력된 숫자는 실수가 저장된 위치의 포인터만을 저장하는 방안을 생각할 수 있음을 안다. 이 방법은 문제입력시 二進探索이 요구되고 알고리즘 수행시 간접 어드레싱을 요해 실행시간의 추가가 없도록 코딩시 주의를 요해야 하는 점은 있으나, 實數풀에다 非零係數당 2 bytes씩만 할당하면 충분하여 통상적인 방법의 대략 1/4 정도의 메모리 공간만 요구되는 효율적 방법이다. 알고리즘의 구현 예를 〈부록-2〉에 보인다.

성긴 행렬을 표현하는 가장 편리한 방법은 아래의 예에서 보인 바와 같이 무순의 $(i, j, A_{ij})$ 방법이다. 즉, 행렬 $A_{5\times5}$를

$$A=\begin{bmatrix} 1. & 0 & 0 & 1. & 0 \\ 2. & 0 & -2. & 0 & 3. \\ 0 & -3. & 0 & 0 & 0 \\ 0 & 4. & 0 & -4. & 0 \\ 5. & 0 & 5. & 0 & 6. \end{bmatrix}$$

라고 할 때, 實數 풀(pool) rnp는

번지 1 2 3 4 5 6 7 8 9
rnp 1. 3. 2. 6. -3. -4. 5. -2. 4.

로 이루어 지고 非零 triples $(i, j, A_{ij})$는

번지 1 2 3 4 5 6 7 8 9 10 11
irn 1 2 2 1 5 3 4 5 2 4 5
jcn 4 5 1 1 5 2 4 3 3 2 1
ival 1 2 3 1 4 5 6 7 8 9 7

로 이루어 진다. 이를 성긴 列벡터(sparse row vectors)로 표현하면

번지 1 2 3 4 5 6 7 8 9 10 11
lenCOL 3 2 2 2 2
JCOLst 1 4 6 8 10
irn 2 1 5 3 4 5 2 1 4 2 5
ival 3 1 7 5 9 7 8 1 6 2 4

위의 표현에서 열은 1열 부터 순차로 저장된다.

물론 lenCOL()과 JCOLst()을 둘다 유지할 필요는 없다. 하나만 유지해도 나머지 하나는 다른 하나에서 계산이 되기 때문이다. 위의 표현방식은 前방향이든 後방향이든 순차적으로 접근하는 경우에는 공간효율적인 방식이나 삽입에는 적합치 못한 단점이 있다. 그러나 문제를 보관하는데는 삽입·삭제가 없고 2 bytes정수 배열로 충분하므로 효율적인 표현방식이다. 참고로 $(i, j, A_{ij})$ triples로 부터 위의 列벡터로 전환하는 코드를 〈부록-3〉에 보인다.

### 3.6 인터페이스

위에서 보았듯이 간단한 수리계획 소프트웨어라도 설계에 여러 국면이 관계한다. 이 중 어떤 것은 이미 만들어진 소프트웨어 루틴을 再사용해야 하는 경우이다. 만들어진 소프트웨어 루틴이 사용코자하는 프로그래밍언어와 같을 경우는 별 문제가 없겠으나 그렇지 않을 경우에는 異種언어간의 인터페이스를 사전에 점검해야 한다. 문자열을 조작하는 데 편리한 언어와 數配列을 조작하는데 효율적인 언어는 다를 수가 있고 이의 인터페이스 방안을 개발을 시작하기 전에 미리 확인하는 것이 현명하다.

개발에 시스템이 제공하는 특수한 도구(예로 소프트웨어 검사기, installer 등)를 사용하면 생산성을 높일 수 있다. 이 경우에도 사용시스템과의 호환성 등을 사전에 고려하여 도구를 선택하도록 해야 한다. 앞으로의 수정, 확장성 등을 예상하여 사용 소프트웨어, 하드웨어 플랫폼과의 인터페이스를 고려하여 설계에 임하여야 한다. 예로서 그래픽인터페이스, 운영시스템 등이 여기에 속한다.

## 4. 소프트웨어 구현

수리계획모형을 다루는 간단한 소프트웨어라고 하더라도 설계하고 개발·수정 및 여러가지 기능을 추가하고 버전을 관리하는 일은 사소한 작업이 아니다. 서로 다른 언어로 구현된 여러 모듈을 어려움없이 통합하고 유연하게 동작되게 하기 위해 가장 다양한 프로그래밍 언어와 도구를 제공하는 Microsoft환경을 채택했다. 제공언어의 종류의 다양성과 혼합프로그래밍의 용이성, 구득성이 주된 선정이유이다. Borland의 Turbo계열도 고려했으나 이미 구현된 Fortran 루틴과의 인터페이스가 여의치 못해 배제됐다.

### 4.1 슈퍼바이저 루틴

가용자원의 점검, 프로그램의 보호, 메뉴의 관리 및 여러 루틴을 접착하는 기능의 상위 프로그램으로 슈퍼바이저를 두었다.

할당가능 메모리용량, 비디오카드의 종류 등 하드웨어의 점검과 프로그램의 수정여부, 텍스트/그래픽모드의 전환, 메뉴 선택에 따른 관련 모듈과의 인터페이스 및 내비게이션을 관장하도록 C로 구현되었다.

### 4.2 편집기

편집기는 물론 어떤 프로그래밍 언어로도 작성이 가능하나 여타 다른 부문과의 호환성과 스트링조작의 효율성을 고려하여 제안 시스템에서는 대부분 C로 구현했으나 스크린 입출력부분의 효율성을 고려하여 어셈블리를 혼용했다. 이 편집기의 일부는 최적화모듈의 출력을 브라우징 하는데도 일부 공용된다. EOL 또는 EOF을 표현하는 방식은 파일에 따라 여러방식이 채용되고 있으나

여기서는 제한된 방식 몇가지만이 존재하는 것으로 상정하고 구현했다.

### 4.3 최적화모듈

최적화모듈의 바탕이 되는 문제저장을 위한 자료구조는 위 3장에서 설명한 바대로 성긴 열벡터 방식이 채용되었고 해법을 수행하는데 필요한 基底($B^{-1}$)는 전체행렬형 및 linked list방식을 모두 구현했으나 $B^{-1}$가 아닌 유효한 제약식만으로 구성되는 kernel만을 저장하는 전체행렬형이 효율적인 것으로 판단된다. 이 최적화모듈은 이미 만들어진 코드를 사용하기 위한 것과 실행효율성의 제고를 위해 C 및 Fortran이 혼합 사용된 유일한 부분이다. C언어는 원래 배열에 대한 고려가 없이 설계되었기 때문에 행렬의 조작이 대부분인 최적화모듈에서는 Fortran에 비해 C는 비효율적이다. Fortran은 副프로그램을 부를 때 크기조정가능 배열(adjustable dimensions)을 전하는 효율적인 방법이 구비되어 있어 범용 Fortran副프로그램이 가능하나, C함수는 그러하지 않다. 多次元배열에서 C는 마지막 디멘션의 크기를 常數로 표현하여야만 하도록 요구한다. C를 사용언어로 결정했다면 Ross[14]의 제안을 참고하여 우회하는 방안을 모색해 볼 수 있다.

수리계획 소프트웨어 실행시 소요시간의 대부분에 해당하는 행렬의 積을 계산할 때 multiplicand와 multiplier의 디멘션을 고려하여 內積이 좋을지 外積으로 계산하는 것이 효율적일지를 신중히 고려하여 전체적으로 시간효율적인 코드를 구현하는 것이 필요불가결하다.

### 4.4 그래픽출력

한글메뉴 및 2변수 문제의 그래프해를 위한 루틴에 텍스트모드가 아닌 그래픽모드가 사용된다. 출력모드의 교체를 매끄럽게 해야하는 일과 그래픽과 관련한 시스템루틴의 가용성이 사전에 검토되어야 한다. 入門者를 위한 선형문제(2변수)의 그래프解에서는 그래프를 그리는 방법론과 더불어 효율성도 문제이나 여러가지 병적인 경우가 발생할 수 있어 이러한 모든 경우에 대비한 알고리즘을 설계하고 테스트하는 것이 적어도 사소한 코딩 정도가 아님을 유념해야 한다.

### 4.5 혼합언어프로그래밍

제안시스템이 한가지언어로 구현되지 않고 이미 짜여진 프로그램이나 라이브러리를 사용해야 하는 경우에는 異種언어프로그래밍이 불가피하다. 따라서 사용언어 선택시 이종언어프로그래밍이 허용되는 컴파일러인가를 확인한 후 선택해야 하고 코딩시에 패스하는 매개변수의 데이터타입, 순서 및 수에 각별한 주의가 요청된다. C와 어셈블리의 혼합은 별다른 문제를 제기하지 않으나 실제로 Microsoft환경에서 Fortran을 C에서 부르는 것은 여러가지 문제, 특히 시스템수준에서 문제가 발생하기도 하여 시스템전문가 수준의 전문지식이 요구되는 부분이다. DOS환경에서는 메모리모델도 각별한 주의가 필요한 부분이다.

### 4.6 기타

입력양식 중 자유형과 MPS양식의 경우에는 변수를 hashing 또는 二進탐색하는 루틴과 더불어 입력을 스캐닝하고 구문을 분해하는 부분이 필요하여 효율적인 코드를 구현하기 위해서는 컴파일러를 작성하는 정도의 기술이 요구된다. 이부분은 자체적으로 만들 수도 있고 아니면 이러한 기능을 제공하는 도구 즉, PCYACC 등과 같은

상업용 도구를 사용하여 해결할 수도 있다.

프로그램이 완성되고 버그가 있는지를 테스트하기 위한 절차도 시간을 소모하는 부분이다. 자동적으로 내비게이트하며 오류를 logging하는 상업용 테스팅소프트웨어를 이용하면 이부분을 수작업에 비해 거의 완벽하고 용이하게 해낼 수 있다.

제안 *LinPro*를 임의로 수정하는 일이나 바이러스의 침입을 방지하기 위한 수단으로 프로그램을 DES로 encription하는 방안을 채택했다.

## 5. 결 론

제안시스템이 제공하는 여러 기능의 작동성, 유용성 및 기능의 추가, 최적화모듈의 robustness 등은 앞으로 사용하면서 점검되고 보완되어야 할 부분이다. 특히 수리계획과 관련하여 GAMS 또는 AMPL 등과 같은 모델링언어와의 연계 및 최적화모듈의 결과를 목적에 맞게 가공할 수 있는 RPG의 개발 및 연계를 추가 구현하여 실제문제에 이용할 수 있게 하는 것이 앞으로의 과제이다.

수리계획문제가 계산집중적인 응용임을 고려하여 32 bit 하드웨어, 운영시스템 및 프로그래밍언어가 PC에서도 널리 보급됨에 따라 32 비트 네이티브코드를 이용하여 소기의 목적을 이루는 시도가 이루어져야 한다. 여러 商用 수리계획 소프트웨어를 구해 사용자 편의성, 계산속도 등에서의 상호비교실험도 앞으로 이루어져야 할 과제이다.

이러한 소프트웨어 프로젝트를 설계하고 구현한 결과물은 이와 유사한 프로젝트 즉 여타 경영과학패키지, 통계패키지의 개발에도 그대로 적용되므로 서론에서 언급했던 기대효과와 더불어 의미있는 시도라 하겠다.

## 참 고 문 헌

[1] 김세헌, 김성륜, 장근녕, 여재현, "소규모 한글 선형계획법 소프트웨어 K-LP의 개발," 『경영과학』, 제11권, 제3호 (1994년 10월), pp. 27-34.

[2] 박순달, 김우제, 설동렬, "LIPED," 『경영과학』, 제11권, 제3호 (1994년 10월), pp. 47-54.

[3] Bixby, Robert E., et al., "Very Large-Scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods," *Operations Research*, Vol. 40, No. 5 (September-October 1992), pp. 885-897.

[4] Chang, Yih-Long and Robert S. Sullivan, ***Quant System (Version 2.0)***, Prentice Hall, 1991.

[5] Cunningham, Kevin and Linus Schrage, ***The LINGO Modeling Language***, LINDO Systems, 1988.

[6] Dorfman, Len, ***Optimizing Microsoft C Libraries***, McGraw-Hills, 1991.

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

[8] Gries, David, ***Compiler Construction for Digital Computers***, John Wiley & Sons, 1971.

[9] Hanson, R. J. and K. L. Hiebert, ***Sparse Linear Programming Subprogram, Sandia National Laboratory***, Report SAND 81-0297, 1981.

[10] Hirshfeld, David S., "Some Thoughts on Math Programming Practice in the '90s,"

*Interfaces*, Vol. 20, No. 4(July-August 1990), pp. 158-165.

[11] Marsten, Roy, "The Design of the XMP Linear Programming Library," *ACM Transactions on Mathematical Software*, Vol. 7, No. 4(December 1981), pp. 481-497.

[12] _____, *et al.*, "Interior Points Methods for Linear Programming: Just Call Newton, Lagrange, and Fiacco and McCormick!," *Interfaces*, Vol. 20, No. 4 (July-August 1990), pp. 105-116.

[13] Murtagh, Bruce A., *Advanced Linear Programming*, McGraw-Hill, 1981.

[14] Ross, John W., "Calling C Functions with Variably Dimensioned Arrays," *Dr. Dobb's Journal*, August 1993, pp. 52-56.

[15] Schildt, Herbert, *Born to Code in C*, McGraw-Hill, 1989.

[16] Schrage, Linus, *User's Manual for Linear, Integer, and Quadratic Programming with LINDO Release 5.0*, The Scientific Press, 1991.

[17] Shamir, Ron, "The Efficienty of the Simplex Method: A Survey," *Management Science*, Vol. 33, No. 3 (March 1987), pp. 301-334.

[18] Sharda, Ramesh, "Linear Programming Software for Personal Computers: 1992 Survey," *OR/MS Today*, June 1992, pp. 44-60.

[19] Stockman, Harlan W., "Optimizing Matrix Math on the Pentium," *Dr. Dobb's Journal*, May 1994, pp. 52-66.

[20] Yurkiewicz, J., "Educational Operational Research Software: A Review," *Interfaces*, Vol. 18, No. 4 (July-August 1988), pp. 59-71.

〈부록－1〉

```
//       scanner.c
#include "linpro.h"

typedef struct node {
  unsigned short int node_no;
  struct node *left, *right;
} node;

int scanner(int, ELEMENT *, char **, int *);
int edit_line(char *), get_line(char *);
int get_token(char *), look_up(char *);
int jstate(int state, int token_type);
int parser(int state, int *class, char **ptab);
void error_msg(int err);
int var_no(char *symbol, node **root, char **p_varname, int *nsymb);
void free_tree(node *n);
void view(ELEMENT *, char **, int *);
int setup(int *, char **, ELEMENT *, char **, int *);

extern char *varnames, *top_names;
extern FILE *lu6, *lu9;
/* This main is for testing purpose
  char *p_varname[MAX_VARS];
  int   out[4+MAX_CONSTR];
  ELEMENT *A;

void main(void)
{
   FILE *lu0, *lu2;
   int  nz;
   int  i;

   if((A=malloc(MAX_NZ*sizeof(ELEMENT)))==NULL) {
      puts("not enough heap storage - A!");
      exit(1);
   }

   if((varnames=malloc(9*MAX_VARS))==NULL) {
      puts("not enough heap storage - varnames!");
      exit(1);
   }
   top_names = varnames;

    nz = scanner(1,A,p_varname,out);
    printf("   nz= %d, nsymb=%d\n", nz,out[2]);
   for (i=0; i<4+out[1]; i++) printf("  %d ", out[i]);  printf("\n");
   for (i=0; i<nz; i++) printf("  %d %d %f", A[i].row,A[i].col,A[i].value);
   printf("\n   variable names:");
   for (i=0; i<=out[2]; i++)
     printf("\n i=%d  str=%s    loc=%d  str2=%c",
               i, p_varname[i],(int)(p_varname[i]-varnames)
                  ,varnames[(int)(p_varname[i]-varnames)]);
   printf("\n");
   for(i=0; i<(int)(top_names-varnames); i++)printf("%c",varnames[i]);
   printf("\n\n");
   view(A, p_varname, out);
}
*/
 unsigned int rhs, sign, constr, nsymb;
 double digit;
 int nz, row;
 char *prog;
 node *root;

int scanner(int keybd, ELEMENT *A, char **p_varname, int *out)
{
  char token[20], line[81], tab[160], *loc;
  int i, err, pos, state=0, err_count=0;
  int  class[80];
  char *ptab[80];
```

```
    nz = constr = rhs = sign = nsymb = 0;
    root=NULL;
    top_names=varnames;
    digit = 1.;
    row=1;
    if(keybd) clrscr();          //       _clearscreen(_GCLEARSCREEN);

    while(state!=18) {                        //  not 'END'
      if((i=keybd ? edit_line(line) : get_line(line))==0) {       // ESC pressed
//      if(!keybd) error_msg(2);          // msg hidden
        nz = 0;
        break;
      }
      prog=line;


    loc=tab;
    err=pos=0;
    while((i=get_token(token))!=EOL) {     //  EOL==COMMENT
       if(i>0 && err==0) {
          class[pos]=i;
          ptab[pos++]=loc;
          strcpy(loc,token);
          loc += strlen(token) + 1;
       }
       else if(i<0) {
          error_msg(err=-i);
          err_count++;
       }
    }
    class[pos]=0;
    if(!err) {
        if((i = parser(state,class,ptab)) >= 0) {
            state = i;
            nz=setup(class, ptab, A, p_varname, out);
            error_msg(0);                 // clear message area
            if(keybd) fputs(strcat(line,"\n"),lu9);
        }
        else {
            error_msg(3);
            err_count++;
        }
    }
    if(!keybd && err_count) {
        error_msg(2);
        nz = 0;
    }
  }
  free_tree(root);
//out[0]                          //  max/min designator
  out[1] = constr - 1;            //  # of constraints
  out[2] = nsymb;                 //  # of variables
  out[3] = 0;                     //  print option
  return nz;                      //  A[0:nz-1]   i.e. nz units
}

int get_token(char *token)
{
  int token_type;
  register char *temp;

  temp = token;

  while(isspace(*prog)) prog++;

  if(*prog=='\0')
    return(token_type=EOL);

  if(*prog=='!')
    return (token_type=COMMENT);

  if(strchr("+-",*prog)) {
    *temp++ = *prog++;
    *temp='\0';
    return (token_type=SIGN);
  }
```

```
if(strchr("<=>", *prog)) {
  *temp++ = *prog++;
  while(isspace(*prog)) prog++;
  if(*prog && strchr("<=>",*prog)) *temp++ = *prog++;
  *temp='\0';
  token_type = REL_OPERATOR;
  if(strlen(token)==2
      && strcmp(token,"<=") && strcmp(token,">=")
      && strcmp(token,"=<") && strcmp(token,"=>")) token_type=-9;
  return token_type;
}

if(isdigit(*prog) || *prog=='.') {
  while(isdigit(*prog) || *prog=='.') *temp++ = *prog++;
  *temp = '\0';
  token_type = NUMBER;
  if(strlen(token)>16)  token_type = -5;
  if(!strcmp(token,"."))  token_type = -6;
  if(strchr(token,'.')!=strrchr(token,'.'))  token_type = -7;
  return token_type;
}

if(isalpha(*prog)) {
  while(isalnum(*prog)) *temp++=*prog++;
  *temp = '\0';
  token_type = look_up(token) ? COMMAND : VARIABLE;

   if(strlen(token) > 8 ) token_type = -8;
   return token_type;
 }

 prog++;
 return (token_type=ILLEGAL);
}

int look_up(char *s)
{
 char *command[]= {
   "MAXIMIZE",
   "MINIMIZE",
   "ST",
   "SUBJECT",
   "TO",
   "END",
   ""
 };

 register char *p;
 register int i;

 for(p=s; *p; *p = toupper(*p), p++);
 for(i=0; *command[i]; i++)
   if(!strncmp(command[i], s,3)) return 1;
 return 0;
}

void error_msg(int err)
{
 char *e_msg[]={
   "                                                    ",
   " Error(01) - Use carriage return for overlength line!    ",
   " Error(02) - Error(s) detected in the file               ",
   " Error(03) - Syntactic error encountered                 ",
   " Error(04) - Expecting keyword 'SUBJECT TO'              ",
   " Error(05) - Numeric fields are too long                 ",
   " Error(06) - Decimal point must be with digit(s)         ",
   " Error(07) - More than one decimal point within a numeric",
   " Error(08) - Variable name is too long                   ",
   " Error(09) - Illegal set of relational operators         ",
   " Error(10) - Illegal character
 };
  short color;                long bcolor;
#define RED 4
```

```
    color=_gettextcolor();          bcolor=_getbkcolor();
    _settextcolor(RED);             _setbkcolor(0);
    _settextposition(25,1);
    _outtext(e_msg[err]);
    _settextcolor(color);           _setbkcolor(bcolor);
}

int parser(int state, int *class, char **ptab)
{
  register int p;
  for(p=0; state>=0 && class[p] && state<18; p++) {
    state = jstate(state,class[p]);
    if(state==1 && strncmp(ptab[p],"MAX",3) && strncmp(ptab[p],"MIN",3))
       state=-1;
    if(state==8) {
       class[p]=KEYWORD;
       if(class[p-1]!=KEYWORD) {
         if(strncmp(ptab[p],"ST",2) && strncmp(ptab[p],"SUBJ",3)) state=-1;
         if(!strncmp(ptab[p],"SUBJECT",3)) state=7;
       }
       else if(strncmp(ptab[p],"TO",2))  state=-1;
    }
  }
  return state;
}

int jstate(int state, int token_type)
{
  int ist[]={1,2,5,7,8,10,12,13,15,18,20,21,23,25,26,28,30,31,35},
      tok[]={0,4,5,1,2,1,2,2,5,4,1,2,2,5,4,
             5, 1, 2, 1, 2, 2, 5, 6, 1, 2, 2, 5, 6, 5, 1, 1,5, 1, 2, 4},
      jst[]={0,1,2,3,4,3,4,4,5,8,6,7,7,5,8,
             9,10,11,10,11,11,12,15,13,14,14,12,15,16,17,17,9,10,11,18};
  register int j;

  for(j=ist[state]; j<ist[state+1]; j++)
     if(token_type==tok[j]) return jst[j];
  return (-1);
}

void free_tree(node *n)

{
  if(n!=NULL) {
     free_tree(n->left);
     free_tree(n->right);
     free(n);
  }
}

int var_no(char *symbol, node **root, char **p_varname, int *nsymb)
{
#define MEMORY_ALLOC(left_right) { \
          if((n=malloc(sizeof(node)))==NULL) { \
             puts("not enough heap storage - node!"); \
             exit(1); \
          } \
          n->left = n->right = NULL; \
          left_right = n; \
          n->node_no = ++*nsymb; \
          if(*nsymb==MAX_VARS) {puts("exceeds MAX_VARS!"); exit(1);} \
          strcpy(top_names,symbol); \
          p_varname[n->node_no] = top_names; \
          top_names += strlen(symbol)+1; \
          return n->node_no; \
   }

   node *n, *p=*root;

      if (*root == NULL)
         MEMORY_ALLOC(*root);

   while(1) {
      register int compare;
      if((compare=strcmp(symbol,p_varname[p->node_no]))<0) {
        if (p->left != NULL)
           n = p->left;
```

```
        else
           MEMORY_ALLOC(p->left);
      }
      else if(compare>0) {
        if (p->right != NULL)
           p = p->right;
        else
           MEMORY_ALLOC(p->right);
      }
      else
           return p->node_no;
   }
}

int setup(int *class, char **ptab, ELEMENT *A, char **p_varname, int *out)
{
#define SIGNED(sign,digit) (sign) ? -(digit) : (digit)

  register int p;

  for(p=0; class[p]; p++) {                          // not eol
    switch(class[p]) {
      case NUMBER:                                   // 1:
        digit = atof(ptab[p]);
        if(rhs) {
           A[nz].row = constr++;
           A[nz].col = 0;
           A[nz++].value = SIGNED(sign,digit);
           sign = 0;
           digit = 1.;
           rhs = 0;
        }
        break;
      case VARIABLE:                                 // 2:
        A[nz].row = constr;
        A[nz].col = var_no(ptab[p],&root,p_varname,&nsymb);
        A[nz++].value = SIGNED(sign,digit);
        sign = 0;
        digit = 1.;
        break;
      case KEYWORD:                                  // 3: ST
        constr = 1;
        break;
      case COMMAND:                                  // 4: MAX/MIN/END
        if(constr) break;                            // exclude "END"
        out[0] = !strncmp(ptab[p],"MAX",3);
        break;
      case SIGN:                                     // 5:
        if(!strcmp(ptab[p],"-")) sign = 1;
        break;
      case REL_OPERATOR:                             // 6: <=
        if(strchr(ptab[p],'<'))      out[3+constr] = 1;
        else if(strchr(ptab[p],'>')) out[3+constr] = 2;
        else                         out[3+constr] = 0;
        rhs = 1;
        break;
    }
    if(nz==MAX_NZ) {puts("exceeds MAX_NZ!"); exit(1);}
    if(constr==MAX_CONSTR) {puts("exceeds MAX_CONSTR!"); exit(1);}
  }
  return nz;                        // A[0:nz-1]   i.e. nz units
}

int get_line(char *line)
{
  if(fgets(line,80,lu6)==NULL)  return 0;
  return 1;
}
```

〈부록 - 2〉

```
      function iptr(value,rnp,linkL,linkR,maxrn,nrn)
      implicit  real*8(a-h,o-z), integer*2(i-n)
      dimension  rnp(*),linkL(*),linkR(*)
      do while (nrn.eq.0)
         rnp(1) = 0.
         linkL(1) = 0
         linkR(1) = 0
         nrn = 1
      end do
      iptr = 1
    2 if (value .lt. rnp(iptr)) go to 3
      if (value .gt. rnp(iptr)) go to 4
      if (value .eq. rnp(iptr)) return
    3 if (linkL(iptr) .eq. 0) go to 5
         iptr = linkL(iptr)
         go to 2
    4 if (linkR(iptr) .eq. 0) go to 5
         iptr = linkR(iptr)
         go to 2
    5 nrn = nrn + 1
      if (nrn .gt. maxrn) stop ' Error-RNP Exceed'
      rnp(nrn) = value
      linkL(nrn) = 0
      linkR(nrn) = 0
      if (value .lt. rnp(iptr)) linkL(iptr) = nrn
      if (value .gt. rnp(iptr)) linkR(iptr) = nrn
      iptr = nrn
      return
      end
```

〈부록 - 3〉

```
      subroutine sparse(n,nz,iVAL,irn,jcn,jcolst,lencol)
      implicit  real*8(a-h,o-z), integer*2(i-n)
      dimension irn(*),jcn(*),ival(*),lenCOL(*),JCOLst(*)
c     convert an unordered set of triples(i,j,Aij) into sparse format
      do 10 ic=1,n
   10 LENCOL(ic) = 0
      iz = 0
      do while (iz.lt.nz)
          iz = iz + 1
          lenCOL(jcn(iz)) = lenCOL(jcn(iz)) + 1
      end do
c   Set JCOLst() to point to where the NEXT col would start
      JCOLst(1) = lenCOL(1) + 1
      do 20 ic=2,n
   20 JCOLst(ic) = JCOLst(ic-1) + lenCOL(ic)
      do 30 ik=1,nz
          do while (JCN(ik).gt.0)
            ip=JCOLst(jcn(ik)) - 1
            itemp = irn(ik)
            jtemp = jcn(ik)
            ktemp =ival(ik)
            irn(ik) = irn(ip)
            jcn(ik) = jcn(ip)
            ival(ik)=ival(ip)
            irn(ip) = itemp
            jcn(ip) =-jtemp
            ival(ip)= ktemp
            JCOLst(jtemp)=ip
          end do
   30 continue
      return
      end
```