電算 副프로그램, SMS, 開發에 關한 硏究

趙 慶 鎬

A Study on the Development of the Subroutine Sparse Matrix Solver

Cho Kyung-ho

Summary

A useful subroutine named Sparse Matrix Solver (SMS) has been developed to treat a large size of sparse matrix.

In case that the coefficient matrix of the linear simultaneous equations is very large, there are usually two major difficulties, that is, memory-storage-overflow and a very long process-time.

In this paper, the algorithm of SMS is introduced, and through a sample problem it has been shown that it is possible to overcome these difficulties using SMS.

序 論

공학적으로 자주 접하게 되는 대형 연립방정식의 전산처리 과정에서 종종 야기되는 두가지 주요 난점은 첫째, 계수행렬이 너무 커서 기억용량을 초과하는 문제와 둘째, 전산처리 시간이 너무 오래 걸리는 문제 등을 들 수 있다.

이중에서도 요즈음 급격히 보급되고 있는 P/C 사용의 경우 기억용량 초과는 문제해결에 있어 거의 절대적인 장애요소가 된다. 반면에 중형 이상의 전산기종을 이용하는 경우는 기억용량 보다는 전산 사용료와 관련하여 전산처리 시간에 더 관심을 갖게되고, 경우에 따라선 이의 단축을 위한 번외의 노력이 문제 해결의 주요 쟁점이 되기도 한다.

대형 연립방정식의 주요 특성의 하나는 그 계수행렬의 많은 요소들이 零의 값을 갖는(zero-elements) sparse matrix라는 사실이다. 본고에서는 이러한 sparse matrix의 전산처리 과정중 위에 언급된 난점들을 효과적으로 극복하여, 필요 기억용량도 크게 절감시키면서 종래의 Gauss Elimination법이나 그 변형인 Cholesky의 LU-decomposition법(James, et al, 1985) 보다도 전산처

理工大學 專任講師

리 시간이 대폭 단축된 전산 부프로그램, SMS를 개발하여, 실제 계산 예를 통해 그 유용성을 검토하였다.

1. Sparse Matrix Solver(SMS)의 概要

선형 연립방정식

$$[A]\{x\} = \{b\} \quad \cdots\cdots(1)$$

에 대한 수치해법은 계수행렬 [A]의 특성에 따라 여러가지가 개발-보편화 되었다. 그 중에서도 [A]가 sparse matrix인 경우 간접법의 Gauss-Jordan 및 Gauss-Seidel Iterative Method 등이 효과적인 방법이 될 수 있으나, 이들은 해가 수렴하기 위해 [A]가 diagonally dominant 해야 하는 제한조건을 수반한다(Gerald, et al, 1984). 이러한 이유로 Gauss-Elimination과 같은 직접소거법이 (1)의 보편적인 수치해법으로 이용된다.

식 (1)의 직접소거 결과를

$$[UA]\{x\} = \{b'\} \quad \cdots\cdots(2)$$

where, [UA] : Upper Triangular Matrix of [A] after column elimination

$\{b'\}$: colunm elimination에 따라 $\{b\}$가 변한 것.

와 같이 표시하면, 해 $\{x\}$는 (2)에서 back-substitution으로 구해진다. 이때 크기 n×n인 계수행렬 [A]를 위한 기억요소 수는 n^2개로 대형 연립방정식에서 기억용량 초과라는 난점은 이에 기인함을 알 수 있다.

본 SMS에서는 [A]의 요소중에서 非零要素(non-zero elements)들 만을 사용하여 소거 과정이 수행될 수 있도록 다음과 같은 일차원 vector들을 설정-이용한다(Stoner, 1972).

[U] : [A]의 non-zero element만으로 구성된 compact vector

[JU] : [U]의 요소와 [A]의 요소 사이의 column 정보로 구성된 vector

[IU] : [U], [JU]의 요소와 [A]요소 사이의 row 정보로 구성된 vector

[IHOW] : [A]의 Gauss-Elimination 과정에서 나타난 pivot element의 위치, [U], [JU], [IU]의 대응 및 변화 과정 등의 모든 정보로 구성된 vector

즉, 종래의 Gauss-Elimination은 n×n의 2차원 배열 [A] 속에서 이루어졌으나, 본 SMS에서는 1차원 배열 [U], [IU], [JU] 및 [IHOW]의 상호 정보를 이용하여 elimination을 수행한다.

이로서, 종래의 零要素에 불필요하게 배당되었던 기억요소를 절약하고, 소거작업시 불필요하게 수반되었던 수많은 零要素에 대한 처리과정을 원천적으로 배제시킬 수 있어 전체적인 전산처리 시간의 단축효과를 기대할 수 있다.

2. [IHOW]의 生成 및 [U], [JU], [IU]의 變化

위에서 언급된 [U], [JU], [IU] 및 [IHOW]의 의미를 보다 쉽게 이해하기 위해 Figure 1-(a)와 같은 sparse matrix [A]에 대한 직접소거법을 생각해 보자. Figure 1-(b)는 소거가 시작되기 전의 [A]에 대응되는 1차원 vector들로 [U]는 [A]의 非零要素만으로 구성된 compact vector이고, [JU]는 [U]의 대응요소에 대한 column 정보를 수록하고 있다. [IU]는 각 row의 첫째 非零要素가 [U]에 저장된 위치를 나타내는 것으로, 예를 들면, IU(3)=5는 3번 row의 非零要素는 U(5)부터 저장되기 시작했음을 뜻한다.

Figure 2는 첫번째 column에 대한 소거작업이 완결됐을 때에 대한 [IHOW], [U], [JU] 및 [IU]를 보여준다. [IHOW]의 일반적인 구조는 IHOW(1)~IHOW(n)까지는 pivot row 및 row interchange에 대한 정보를 저장하고, IHOW(n+1)~IHOW(2n)까지는 각 column 소거시 발생되는 모든 정보, 즉, pivot요소의 위치, 소거대상이 되는 row수, 소거대상 요소의 위치 등에 관한 정보가 [IHOW]의 몇번째 요소부터 저장되기 시작하는 가를 가르킨다. 따라서, Figure 2의 IHOW(6)=11은 1번 column 소거에 대한 정보가 IHOW(11)부터 다음과 같이 기록되었음을 뜻한다.

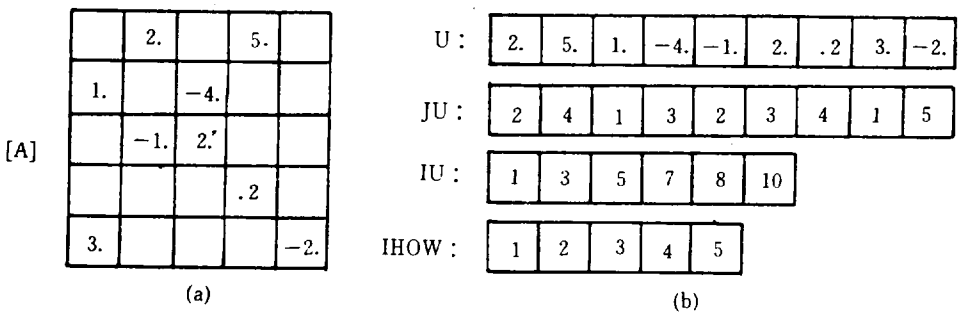


Fig. 1. Sample sparse matrix [A] and its correspondings [U], [JU], [IU] and [IHOW] before elimination

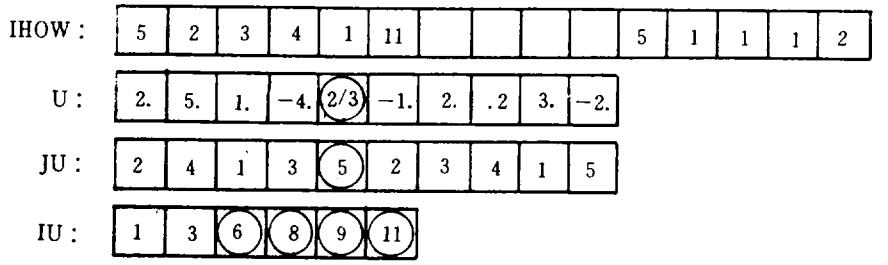


Fig. 2. [IHOW], [U], [JU] & [IU] after column 1 elimination.

- IHOW(11)=5, IHOW(12)=1 } pivot요소의 위치가 5번째 row의 첫번째 非零要素이다.
- IHOW(13)=1 소거대상 row가 한개뿐이다.
- IHOW(14)=1, IHOW(15)=2 } 소거대상 요소가 2번째 row의 첫번째 非零要素이다.

전산 프로그램 상에서는 이런 [IHOW]의 정보는 실질적으로 [U], [JU] 및 [IU]들 만의 검색을 통하여 얻어진다.

[IHOW]의 정보가 결정되면 그에 따라 소거작업이 수행되고 이는 [U], [JU] 및 [IU]의 값들을 변화시킨다. Figure 2의 ○로 표시된 [U], [JU] 및 [U]의 값들은 소거작업중 Figure 1의 A(5,5) =−2.에 의해 A(2,5)=A(2,5)−A(2,1)*A(5,5) /A(5,1)=0−1.×(−2.)/3.=2/3라는 非零要素가 발생되어 그에 따른 [U], [JU] 및 [IU] 값들의 변화를 의미한다.

이와같은 방법으로 마지막 5번째 column에 대한 소거작업을 완료했을 때의 [IHOW], [U], [JU] 및 [IU]들은 Figure 3와 같다.

Figure 4는 참고로 Figure 3의 [IHOW], [U], [JU] 및 [IU]의 최종 정보로 2차원 array를 재구성해 본 것으로, ○로 표시된 요소들을 zero로 대체하면 바로 (2)식의 upper triangular matrix [UA]와 같음을 알 수 있다. 결국 해 |x|는 여기서 back-substitution으로 구해지는 것이기 때문에, Figure 3의 [IHOW], [U], [JU] 및 [IU]로부터 해 |x|를 구하기에 앞서 |b| vector가 column 소거과정중 변화된 |b'|를 구하여야 한다. |b'|는 [IHOW]의 최종정보를 이용하여 |b|로부터 계산된다. |b'|를 구하는 다른 방법으로 |b|를 [A]에 포함시킨 augmented matrix [A']를 처음부터 사용해도 된다. 이상에서 소개된 SMS의 algorithm을 개략적으로 도식화하면 Figure 5와 같다.

IHOW :

5	1	2	3	4	11	16	21	26	31	5	1	1	1	2
1	1	1	1	3	2	2	1	2	3	3	3	1	1	4
4	2	0												

U :

2.	5.	1.	−4.	2/3	−1.	2.	5/2	1/3	.2	2/75	3.	−2.

JU :

2	4	1	3	5	2	3	4	5	4	5	1	5

IU :

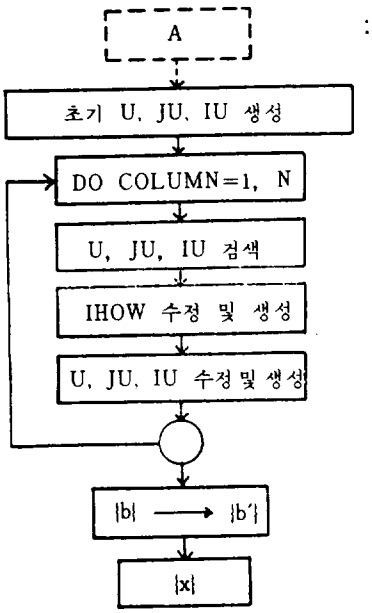
1	3	6	10	12	14

Fig. 3. Final [IHOW], [U], [JU] & [IU] after column 5 elimination

[UA] :

3.				−2.
	2.		5.	
1.		−4.		2/3
	−1.	2.	5/2	1/3
			.2	2/75

Fig. 4. A matrix regenerated from final [IHOW], [U], [JU] and [IU]



: search for pivot element and elements to be eliminated

: record all informations about elimination

: perform the elimination according to the records on IHOW, and update U, JU, & IU

Fig. 5. Schematic diagram of SMS algorithm

3. 非線型 聯立方程式과 SMS의 導入

n개의 비선형 연립방정식을

$$\begin{aligned} f_1(x_1,x_2,\cdots\cdots x_n)&=0\\ f_2(x_1,x_2,\cdots\cdots x_n)&=0\\ &\vdots\\ f_n(x_1,x_2,\cdots\cdots x_n)&=0 \end{aligned} \qquad \cdots\cdots\cdots\cdots\cdots\cdots (3)$$

라 하자. 식 (3)의 수치해를 구하기 위해 수렴속도가 빠르고 보편적으로 많이 이용되고 있는 Newton-Raphson's Iterative Method에 따르면, 다음과 같은 $|x|$에 대한 선형 연립방정식이 얻어진다.

$$f_1+\frac{\partial f_1}{\partial x_1}\Delta x_1+\frac{\partial f_1}{\partial x_2}\Delta x_2+\cdots\cdots\cdots+\frac{\partial f_1}{\partial x_n}\Delta x_n=0$$

$$f_2+\frac{\partial f_2}{\partial x_1}\Delta x_1+\frac{\partial f_2}{\partial x_2}\Delta x_2+\cdots\cdots\cdots+\frac{\partial f_2}{\partial x_n}\Delta x_n=0$$

$$f_n+\frac{\partial f_n}{\partial x_1}\Delta x_1+\frac{\partial f_n}{\partial x_2}\Delta x_2+\cdots\cdots\cdots+\frac{\partial f_n}{\partial x_n}\Delta x_n=0$$

이를 행렬식으로 다시 표현하면,

$$[A]\{\Delta x\}=\{b\} \qquad \cdots\cdots\cdots\cdots\cdots\cdots (4)$$

where, $A_{ij}=\dfrac{\partial f_i}{\partial x_j}$

$$\Delta x_j = x_j^{new}-x_j^{old}$$

$$b_i=-f_i$$

식 (4)의 계수행렬 [A]는 system size(n)가 증가함에 따라 zero값을 많이 갖는 sparse matrix 성향을 갖고 있으므로, 식 (4)의 수치해법으로 본고의 SMS를 이용하고자 한다.

Figure 6은 비선형 연립방정식의 수치해법을 개략적으로 도식화 한 것으로 Figure 6-(a)의 점선 부분은 기존의 직접소거법에 의한 식 (4)의 풀이과정을 뜻한다.

SMS를 이용하기 위해선 Figure 6-(a)의 점선 부분을 Figure 6-(b)로 대체하면 된다. Figure 6-(b)에서 보는 바와 같이 첫번째 iteration에서 얻어진 [IHOW]의 elimination 정보를 이용하면, 둘째 iteration부터는 pivot요소에 대한 검색, row interchange 등의 반복 작업을 생략할 수 있어, 대형 system의 경우 전체적인 전산처리시간을 크게 단축시킬 수 있을 것이다.

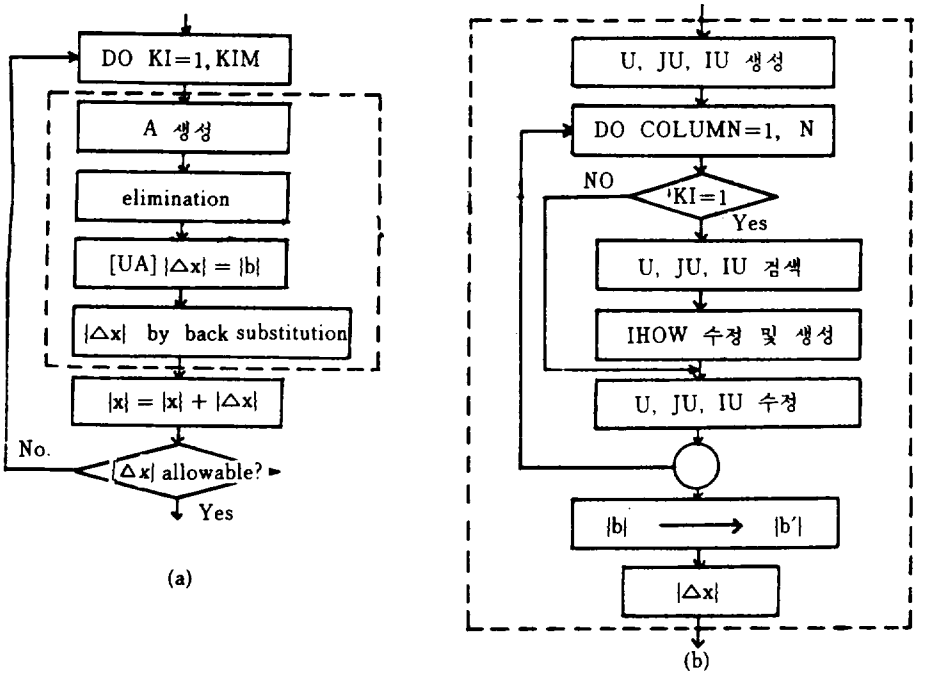


Fig. 6. Iteration for non-linear simultaneous eqs. and application of SMS

4. 計算 例 및 考察

비선형 연립방정식의 풀이에 SMS를 도입하여 그 유용성을 검토하기 위해 Figure 7과 같은 천연가스 배관망의 초기 정상상태에서의 유량 및 압력 분포를 구하는 문제를 선택했다.

본 예제는 Yow(1979)의 pipe 7개 및 junction 5개로 구성된 천연가스 배관망에 junction 7~junction 19를 더 설정하여 system size(n)을 38×38로 증가시킨 것으로 전체 pipe의 길이, pipe size, 마찰계수 및 초기 경계조건 등은 Yow이 모델과 동일하다.

pipe內의 천연가스 유동에 관한 지배방정식들은 다음과 같다(Wylie, 1978).

운동방정식 :

$$P_x+\frac{K\alpha^2}{A}M_t+\frac{Pg}{B^2}\sin\theta+\frac{K^2fB^2M^2}{2DA^2P}=0$$

연속방정식:

$$B^2/A\cdot M_x+P_t=0$$

여기서, P는 압력, M는 mass flow rate, K는 단위변환을 위한 상수, α는 inertial multiplier, A는 pipe 내부 단면적, g는 중력가속도, B는 wave speed, θ는 pipe line의 기울기, f는 마찰계수, D는 직경등을 뜻한다. 첨자 x 및 t는 각각 pipe 길이 방향 및 시간에 대한 변화율을 뜻한다.

정상상태에선 $M_t=0$ 및 $P_t=0$임을 고려하여 위의 지배방정식들을 적분하여 각각의 pipe 및 junction에 적용시킬 수 있는 꼴로 정리할 수 있다. 예를 들어, pipe 1에 대한 운동방정식을 f_1이라하면,

$$f_1(M_1,P_1,P_6)=P_1^2-e^{S1}P_6^2-\frac{K^2f_1B^2}{D_1A_1^2}L_1\cdot\frac{e^{S1}-1}{S_1}\cdot M_1^2=0$$

여기서, $S_1=(2g\cdot L_1\cdot\sin\theta_1)/B^2$, L_1=pipe 1의 길이, $\tan\theta_1$=pipe1의 기울기 등이다.

Junction ①에 대한 연속방정식을 g_1이라 하면,

$$g_1(M_1,M_2,M_3,MJ_1)=-M_1-M_2-M_3+MJ_1=0$$

여기서, M_i=pipe i의 mass flow rate, MJ_1=외부로부터 junction ①을 통해 유입되는 mass flow rate이다. 따라서 20개의 pipe에 대해 20개의 운동방정식과 18개의 junction에 대해 18개의 연속방정식이 존재하여, 총 38개의 미지수(M_1, M_2,…M_{20}, MJ_1, P_2, P_3,…P_{18})들은 이들 비선형 연립방정식의 해이다.

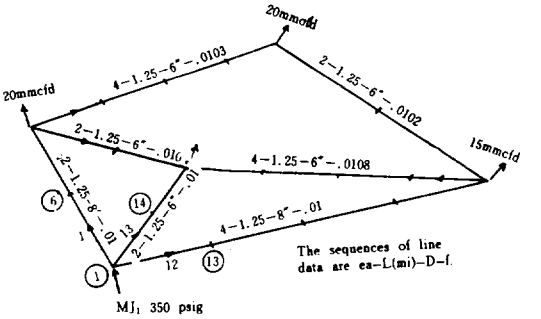


Fig. 7. Sample problem of natural gas pipeline networks

본 예제에 대한 식 (3)의 해는 적절한 초기 가정치 $|x_0|$에 대해 보통 5번 이내의 iteration으로 얻어지나, 본고에서는 해의 수렴 여하에는 관계없이 최대 iteration 수를 변화시키면서, Figure 6-(a), (b)의 두 방법에 대한 연산시간을 8bit P/C를 통해 비교하여 Figure 8과 같은 결과를 얻었다. 즉 두 방법 모두 연산시간은 iteration 횟수에 비례하여 증가하나, 본 예제에선 SMS가 기존의 직접소거법에 비해 5배 정도 연산시간이 빠름을 알 수 있다.

한편, 2차원 array A(38,38)은 1444개의 기억요소를 필요로 하는데 비해, SMS에선 IHOW(344), U(192), JU(192) 및 IU(39)로 총 767개의 기억요소로 작업 가능했기에, 본 예제에선 약 47% 정도의 기억요소 절감 효과가 있었음을 확인했다. 이러한 효과(연산시간 단축 및 기억요소 절감)는 연립방정식의 크기와 sparsity 정도에 따라 더 더욱 커지는 성향이 있어, 대형 system일수록 SMS

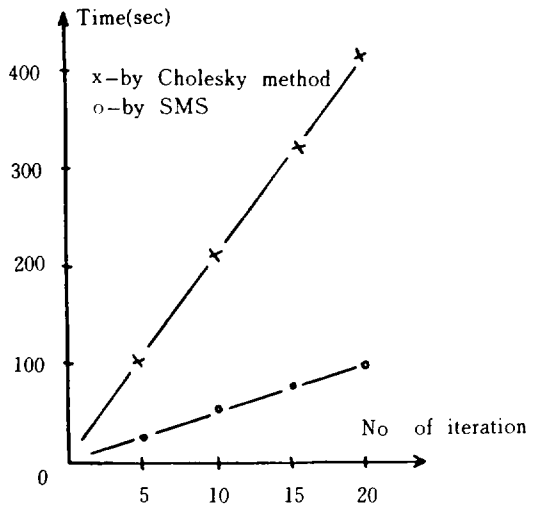


Fig. 8. Comparison between Choleskey Method and SMS

의 유용성이 증대된다고 할 수 있겠다.

摘 要

대용량의 비선형 연립방정식을 Newton-Raphson의 반복법으로 풀 때 나타나는 sparse matrix를 효과적으로 처리하기 위한 Sparse Matrix Solver(SMS)를 부프로그램 단위로 개발하였다.

또한, SMS를 이용함으로써 다음과 같은 난점의 극복 가능성을 실제 계산 예를 통하여 확인하였다.

-기억용량 초과문제
-반복연산시간의 장기화문제

SMS와 관련하여 앞으로 좀 더 연구되어야 할 과제로는 system size 및 sparsity와 SMS도입 사이의 경제적 분기점에 대한 연구와, 기존의 직접소거법과 비교한 SMS에서의 round-off error 누적에 대한 연구가 남아 있다고 사료된다.

參 考 文 獻

Gerald, C. F. and P. O. Wheatley, 1984. Applied Numerical Analysis, 3rd ed. p. 91-142. Addison-Wesley, Massachusetts.

James, M. L., G. M. Smith and J. C. Wolford. 1985. Applied Numerical Methods for Digital Computation, 3rd ed. p. 178~185. Harper & Row Pub. New York.

Stoner, M. A., 1972. Sensitivity Analysis Applied to a Steady-State Model of Natural Gas Transportation Sys. *Soc. of Pet. Eng. J.* pp. 115~125.

Wylie, E. B. and V. L. Streeter, 1978. Fluid Transient, p. 271~285. McGraw-Hill.

Yow, W., 1972. Numerical Error on Natural Gas Transient Calculations. *Tran. of ASME.* pp. 422~428.