전력계통해석을 위한 자코비안행렬 가우스소거의 병렬계산 알고리즘

論文
43~2~2

Parallel Computation Algorithm of Gauss Elimination in Power system Analysis

徐 義 錫* · 吳 泰 圭**
(Eui-Suk Suh · Tae-Kyoo Oh)

Abstract – This paper describes a parallel computing algorithm in Gauss elimination of Jacobian matrix to large-scale power system. The structure of Jacobian matrix becomes different according to ordering method of buses. In sequential computation buses are ordered to minimize the number of fill-in in the triangulation of the Jacobian matrix. The proposed method develops the parallelism in the Gauss elimination by using ND(nested dissection) ordering. In this procedure the level structure of the power system network is transformed to be long and narrow by using end buses which results in balance of computing load among processes and maximization of parallel computation. Each processor uses the sequential computation method to preserve the sparsity of matrix.

Key words : Parallel Computation. (병렬계산), Causs Elimination (가우스 소거)

1. 서 론

최근 시컨시얼 컴퓨터의 계산속도의 한계를 극복하기 위한 방법으로 병렬컴퓨터가 개발되고 있고 많은 응용분야에서 응용소프트웨어의 개발 연구가 진행되고 있다.

전력계통에서 많이 취급하는 선형방정식에 있어 행렬 A는 매우 성김성이 강하고 대칭이며 정칙행렬이다. 선형방정식 해법에 있어 성김성벡터 기법을 이용하여 행렬 A를 삼각분해하고 삼각분해된 행렬과 기지의 벡터를 가지고 순방향 역방향 대치에 의해 방정식의 해를 구하는 것이 일반적이다.[1, 2]. 그러나 전력계통 문제에 있어 해를 구하는데는 많은 반복계산을 필요로 하고, 해석의 온라인화를 위해서는 보다 고속계산을 요한다.

*正 會 員:韓國電氣研究所 研究員
**正 會 員:韓國電氣研究所 電力系統研究部長
接受日字:1993年 8月 16日
1次修正:1993年 11月 19日

이 전력계통 해석의 고속화에 필요한 병렬알고리즘의 개발을 두가지 형태로 나누어 살펴 볼 수 있다. 하나는 주로 과도안정도 해석에 응용되고 있는 블럭행렬을 이용한 방법으로 대상이 되는 행렬을 여러 블럭으로 나누는 것이다.[3-5]. 블럭의 수는 프로세서의 수와 관계되며 이들 블럭을 나누는 방법은 계통내에 연계가 약한 부분을 경계로 클러스터를 구성하는 것이다. 다른 방법은 시컨시얼 방법으로 해를 구하는 과정에서 병렬성을 찾는 방법이다. 이것은 각 요소계산간 병렬성을 개척하는 것으로 많은 프로세서간 데이타통신을 요한다[6-8].

본 연구에서는 전자의 방법으로 블럭화에 의한 병렬알고리즘을 개발한다. 클러스터에 의한 병렬화는 계통의 연결상태에 의해 블럭이 정해지기 때문에 프로세서간 계산부담의 평형을 이루기가 어렵게 된다. 여기에서는 행렬을 블럭화하는 방법으로 ND(nested dissection) 오더링 방법을 적용한다. 이 방법은 계통을 나타내는 수준구조(level structure)를 사용하여 계통을 분리하는 것이다. 보통 루프로 된 계통은 수준구조가

짧고 폭이 넓게 되므로 계통분리자가 커지게 되어 순차적 계산이 증가하게 되며 프로세서간 계산부담이 불평형될 수 있어 보통의 효율적인 시컨시얼 계산에 비해 계산시간을 크게 향상하기가 어렵다. 따라서 수준구조를 길고 폭이 좁게 만들 필요가 있다. 이를 위해 엔드모선(end bus)이라는 개념을 정의하고 이를 사용하여 수준구조를 길고 폭이 좁게 만들어 ND오더링에 의해 블럭화하는 방법을 제시하고자 한다. 그리고 각 프로세서 내에서는 시컨시얼 계산방식의 성김성 기법을 사용하도록 한다.

우선 시컨시얼 계산과정을 살펴보고 가우스소거시 병렬성을 갖는 행렬의 구조를 만들기 위해 모선을 오더링(ordering)하는 방법과 병렬계산을 하는 과정을 설명한다. 그리고 프로세서의 수에 따른 병렬계산의 효과와 제시한 병렬계산의 효용성을 검토한다. (4)

2. 행렬 A의 가우스소거법을 이용한 삼각화

가우스소거는 선형방정식을 푸는데 잘 알려진 방법이다. 열(column) 단위의 소거가 이 알고리즘의 가장 일반적인 방법이며 열방향의 형태로 기억된다. 전력계통 해석시 만들어지는 행렬 A는 정칙이며 대각요소가 지배적(diagonal dominant)이고 구조적인 대칭(incidence symmetric)이다. 따라서 대각요소를 피버트로 사용하여 소거해도 해의 안정성에는 거의 문제가 발생하지 않는다.[1]

다음과 같이 일부 열이 소거된 후의 행렬을 생각하자.

$$A^{(3)}=\begin{array}{c|cccccc|} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 1 & x & x & x & x & x \\ 2 & & 1 & x & x & x & x \\ 3 & & & x & x & x & x \\ 4 & & & x & x & x & x \\ 5 & & & x & x & x & x \\ 6 & & & x & x & x & x \end{array} \quad (1)$$

여기서 $A^{(1)}=A$, 행렬의 공백부분은 소거된 영요소를 나타낸다. 이와같이 나타낼 경우 삼각화 과정의 k열 소거단계에서 행렬의 요소를 수식으로 나타내면 다음과 같이 된다.

$$A^{(k+1)}=(L_k^c)^{-1}D_k^{-1}A^{(k)} \quad (2)$$

여기서

$$D_k=\begin{vmatrix} 1 & & & & & \\ & 1 & & & & \\ & & A_{kk}^{(k)} & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{vmatrix} \qquad (L_k^c)^{-1}=\begin{vmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & \cdot & 1 & & \\ & & \cdot & & 1 & \\ & & & & -A_{nk}^{(k)} & 1 \end{vmatrix}$$

식 (2)에 의해 가우스소거 과정을 정리하면 식(3)과 같이 된다.

$$(L_n^c)^{-1}D_n^{-1}\cdots(L_2^c)^{-1}D_2^{-1}(L_1^c)^{-1}D_1^{-1}A=U \quad (3)$$

여기서 $U=A^{(n+1)}$, $U_{ki}=A_{ki}^{(k)}/A_{kk}^{(k)}$

$\equiv A_{ki}^{(k+1)}$

따라서 행렬 A의 삼각분해는 식 (3)에 의해 다음과 같이 된다.

$A=LU$

여기서 $L=D_1L_1^cD_2L_2^c\cdots D_nL_n^c$, $L_{ik}=A_{jk}^{(k)}$

이 삼각분해시 계산을 요하는 $k+1$번째 이상의 행과 열의 각요소들은 식 (2)와 (3)에서 다음과 같이 나타내진다.

$$A_{ij}^{(k+1)}=A_{ij}^{(k)}-L_{ik}U_{kj},\ i,\ j>k \quad (5)$$

$$A_{ij}^{(k)}=A_{ij}-\sum_{m=1}^{k-1}L_{im}U_{mj},\ i,\ j>k \quad (6)$$

식 (6)에서 알수 있듯이 시컨시얼 알고리즘에서 k열의 비영요소를 소거할 때 $k+1$번째 열 이상의 요소들은 그 이전 단계의 값이 작용한다.

3. 병렬계산을 위한 모선의 오더링

앞절에서 설명한 열의 소거과정을 병렬컴퓨터의 각 프로세서에 할당하여 병렬계산을 하기 위해서는 각 프로세서의 계산과정이 다른 프로세서의 소거과정과 서로 독립되도록 해야한다. 이는 삼각화하려는 행렬 A의 구조가 그림 1(b)와 같이 대각 행렬의 형태가 되면 가능하다. 행렬의 구조는 모선의 오더링(ordering)에 따라 다르게 되며 본 논문에서는 수준구조(level structure) 그래프와 분리자(separator)에 의해 모선을 분류하여 오더링하는 것에 의해 대각행렬의 구조가 되도록 하였다. 그리고 이 과정에 있어 수준구조를 길고 폭이 좁게 만들기 위해 엔드모선을 사용한 경험적인 방법을 도입하였다.

3.1. 병렬성을 갖는 행렬구조

그림 1(a)와 같은 전력계통을 두개의 프로세서에 의해 병렬계산한다고 하자. 이때 부분계통 1, 3 그리고 2순으로 모선을 오더링(ND 방법)하는 것에 의해 두개의 독립적인 계산부분을 갖는 행렬의 구조, 그림 1(b)와 같이 만들 수 있다.

그림 1(b)에서 행렬 A_1과 A_3에 대해 가우스소거를 할 때 서로간에 영향이 없다. 따라서 각 부분을 다른 프로세서에 할당하여 독립적으로 병렬계산할 수 있으며, 계통 2(end와 seperator 모선으로 구성)에 속하는 모선의 수를 최소화하고 계통 1과 3에 속하는 모선수가 평형을 이루도록 하는 것이 독립적인 계산부분을 극대화할 수 있으며 프로세서간 계산부담을 줄일 수 있다.

전력계통의 폭이 좁고 길다란 것이 폭이 넓은 계통보다 계통분리를 하는 계통2의 모선수가 작아지게 되며 계통 1과 3의 모선수가 평형을 이루게 할 수 있다. 일반적으로 운전시 계통의 연결상태를 보면 원거리 전력송전을 위한 기간계통은 루프로 연결하여 운전하며, 지역급전을 위한 지역계통은 레디얼로 운전된다. 따라서 루프에 속하는 몇개의 모선을 제거하면 계통의 연결상태는 레디얼로 운전된다. 따라서 루프에 속하는 몇개의 모선을 제거하면 계통의 연결상태는 레디얼 계통으로 만들수 있으며, 이것의 수준구조는 폭이 좁고 길게 된다. 본 연구에서 이들 제거모선을 엔드모선(end bus : 맨 마지막에 오더링함)이라 정의하고 병렬성 개발을 위한 모선의 오더링 과정에 도입하였다. 이 엔드모선은 일반적으로 전력계통의 구성을 알고 있으므로 쉽게 선택할 수 있다.

계통에서 이 엔드모선을 제거한 후 주변모선을 시작모선으로 전력계통의 수준구조(level structure)를 만들고 이 수준구조의 중간부분을

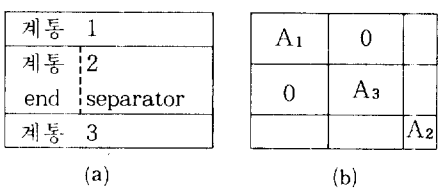


그림 1 전력계통 모선의 분류와 행렬의 구조
(a) 전력계통 분리
(b) ND 방법에 의한 행렬의 구조

Fig. 1 Bus classification and matrix structure
(a) Power system dissection
(b) matrix structure by ND ordering method

분리자(separator)로 선택한다. 이와같이 하여 계통의 모선들을 분류하고 계통 1, 계통 3, 분리자모선, 그리고 엔드모선 순서로 오더링하면 병렬계산에 매우 효과적인 행렬의 구조를 얻을 수 있다. 이들 오더링방법을 설명하기 전에 적용된 그래프이론을 요약 설명한다.

3.2 관련 그래프 이론[1]

1) 준주변모선 탐색

이 알고리즘은 Gibbs에 의해 제안된 것으로 큰 이심을 갖는 모선, 예를들어 길다란 계통의 경우 맨끝에 있는 모선을 찾는데 많이 사용되고 있다.

step 1 최소 디그리(degree)의 모선 r을 찾는다.

step 2 r에 루트화된 수준구조를 만든다.

step 3 현 수준구조의 마지막 수준에서 부분그래프를 찾는다.

step 4 각 부분그래프에서 최소 디그리의 모선을 찾고 이것에 루트화된 수준 구조를 만든다. 만약 어떤 모선 v에 대해 수준구조가 r에 루트화된 수준구조 보다 길으면 r은 v로 대치하여 step 3로 간다.

step 5 r을 준주변모선으로 한다.

2) 수준구조

가우스소거시 계산의 병렬성을 의해 전력계통의 모선들을 수준구조(level structure)를 사용하여 여러 부분집합으로 분해한다. $m+1$의 수준을 갖는 수준구조 $L_0, L_1 \cdots L_m$의 부분집합은 다음과 같이 정의된다.

$$Adj(L_i) \subseteq L_{i-1} \cup L_{i+1},\ 0<i<m$$

$$Adj(L_0) \subseteq L_1$$

$$Adj(L_m) \subseteq L_{m-1}$$

m은 수준구조의 길이(length)이며, 각 수준에 속하는 모선의 수 중 가장 큰 것을 폭(width)이라 한다. $L_0=\{u\}$ 이면 수준구조는 모선 u에서 루트화 되었다고 한다.

루트화된 수준구조를 만들어 내기위한 알고리즘은 다음과 같다. 그래프를 G=(V, E), 방문된 모선의 집합을 V_v라 하자.

step 1 queue가 시작모선 s를 포함하도록 초기화하고, level(s)=0, $V_v=\{s\}$ (여기서 queue는 연속적으로 기억하는 장소로 front라 하는 한끝에서 첨가되며, rear라 하는 다른 한끝에서 제거된다.)

step 2 만약 queue가 비어 있으면 stop. 그렇지 아니하면 queue의 rear에서 모선 v를 제거하고, v에 근접한 방문하지 않은 모선의 집합S를 찾는다.

계통구조와 엔드모선의 데이타 입력, C={엔드모선}, $n=|V|-|C|$, 엔드모선을 $n+1$에서 $|V|$까지 오더링한다.

C에 속하는 모선과 연결된 선로를 제외한 그래프 G(V−C)의 준주변모선 u를 찾는다(Gibbs알고리즘)

G(V−C)의 u에 루트화된 수준구조 $L_0, L_1, \cdots L_m$을 만듬

모선을 분류하여 오더링 한다. $i=0$, $k=0$, $n_1=0$이라 하여

1) $n/(p-k)$의 반올림한 수를 n_p라 하고, 수준 L_i부터 시작하여 n_p 번째 모선이 속하는 수준 j를 찾는다.
2) 최적이 되는 분리자$S_k \subseteq L_j$ 를 찾는다 (최적분리자는 L_j 모선 중 L_{j-1}과 L_{j+1}의 양측 모선에 근접한 모선만을 선택하면 된다.)
3) 분리된 부분그래프 $R_k=\{L_0,\cdots,L_1-S_k\}$의 모선 수를 $n_2=|R_k|$라 하고 R_k에 속하는 모선을 삼각화시 비영요소가 최소가 되도록 n_1+1에서 n_1+n_2까지 번호를 주어 오더링(MD 알고리즘)한 후 $n_1=n_1+n_2$로 한다.
4) 만약 k가 $p-2$이면 $R_{k+1}=\{L_{j+1},\cdots,L_m\}$, $n_2=|R_{k+1}|$로 하여 R_{k+1}을 같은 방법으로 n_1+1에서 n_1+n_2까지 번호를 주어 오더링하고, 분리자 $S_0, \cdots, S_{p-2}$에 속하는 모선들을 같은 방법으로 n_1+n_2+1에서 $|V|-|C|$까지 번호를 주어 오더링하고 마친다.
   만약 $k<p-2$이면 $i=j+1$, $n=n-n_2-|S_k|$, $k=k+1$로 하고 단계 1)으로 감

그림 2 모선의 오더링 알고리즘
Fig. 2 Bus ordering algorithm

$S=Adj(v)\cap(V-V_v)$

step 3 만약 S={ }이면 step 2로 가고, 그렇지 아니하면 w∈S인 각각의 w에 대해
(3a) queue의 front에 w을 더한다.
(3b) level(w)=level(V)+1
(3c) w을 방문된 것으로 나타낸다 : $V_v=V_v\cup\{w\}$

step 4 step 2로 돌아간다.

3) 삼각화시 새로운 비영요소의 수를 줄이는 알고리즘(MD 알고리즘)

대칭이고 정칙이며 성김성 행렬에 대한 가우스소거에 있어 이 MD(minimum degree) 알고리즘이 가장 일반적으로 사용되는 오더링 방법이다. 중심이 되는 개념은 소거과정에서 최소 디그리를 갖는 행 또는 열을 선택하여 새로운 비영요소의 생성을 줄이는 알고리즘이다. 이는 최소의 디그리를 갖는 모선을 우선적으로 오더링하고 이 모선이 소거될 때 새로운 비영요소의 발생에 따라 영향을 받는 모선의 디그리를 수정한다. 그리고 다시 오더링하지 않은 모선 중 최소의 디그리를 갖는 모선을 선택하여 오더링하는 과정을 모든 모선이 오더링될 때 까지 반복하는 것이다.

3.3 모선의 오더링 알고리즘

행렬과 연관된 방향성이 없는 그래프(실제에는 계통의 연결상태 데이타가 주어짐)가 주어지면 알고리즘은 준주변모선에 루트화된 수준구조를 만든다. 이때 최적분리자는 프로세서의 계산부담이 평형이 되도록 선택된다. 분리된 그래프의 모선집합을 간략히 R, 분리자에 해당하는 모선집합을 S, 각 단계에서 이전 및 현재 분리계통에 포함되는 모선의 집합을 C, 전체 계통 모선집합을 V, 모선수를 |V|, 프로세서의 수를 P라 할 때 모선의 오더링은 다음과 같이 한다.

4. 가우스소거의 병렬계산

$p=2$인 경우의 예를들어 제시한 방법에 의해 오더링하면 행렬의 구조는 그림 3와 같이 된다.

그림에서 비영요소는 사선으로 그려진 부분에 제한된다. 따라서 가우수소거시 첨가되는 비영요소도 이 영역에 한정된다.

이 행렬의 구조를 살펴보면 가우스소거시 즉, 대각행력 A_1의 하삼각 부분의 비영 요소를 소거할

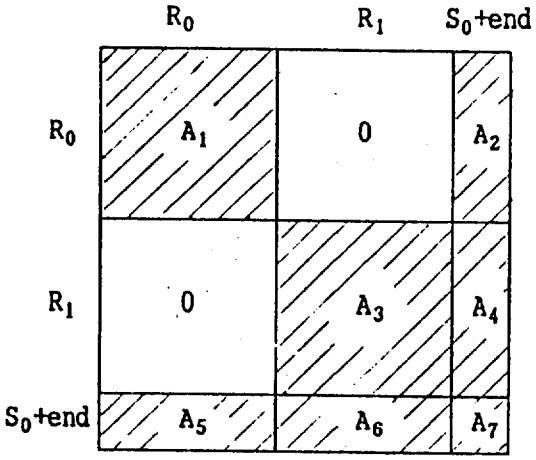


그림 3 제시한 오더링 방법에 의한 행렬구조
Fig. 3 Matrix structure by proposed ordering method

때 영향을 받는 부분은 A_1, A_2부분이며 A_5의 비영요소거는 A_5와 A_7에 영향을 미친다. 그리고 대각행렬 A_3의 하삼각 부분의 비영요소를 소거할 때는 A_3와 A_4 부분에, A_6의 비영요소소거는 A_6와 A_7에 영향을 미친다.

여기서 A_1, A_2, A_5와 A_3, A_4, A_6는 서로 독립적으로 계산되나 A_7은 각부분에 영향을 받는다. 그렇지만 위 두 독립된 부분의 모든 비영요소의 소거후 A_7의 변화는 식 (6)에 의해

$$\triangle A_{ij} = -\sum_{m\in\Omega} L_{im}U_{mj},\ i, j\text{는 분리계통에 속하는 모선}$$

$$= -\sum_{m\in\Omega_1} L_{im}U_{mj} - \sum_{m\in\Omega_2} L_{im}U_{mj} \tag{7}$$

$$\triangle A_{ij}' + \triangle A_{ij}''$$

여기서 $\Omega_1 = \{A_1$행렬에 속하는 모선들$\}$
$\Omega_2 = \{A_2$행렬에 속하는 모선들$\}$
$\Omega = \{\Omega_1, \Omega_2\}$

와 같이 분리되므로 두 부분의 영향을 각각 계산할 수 있다. 이들 계산과정을 도식적으로 나타내면 그림 4와 같다. 그림 4에서 알 수 있듯이 각 프로세서에 데이터 배정 후 가우스소거 과정에서는 분리계통 모선에 해당하는 행렬요소의 변화분에 대해서만 데이터통신이 필요하게 된다.

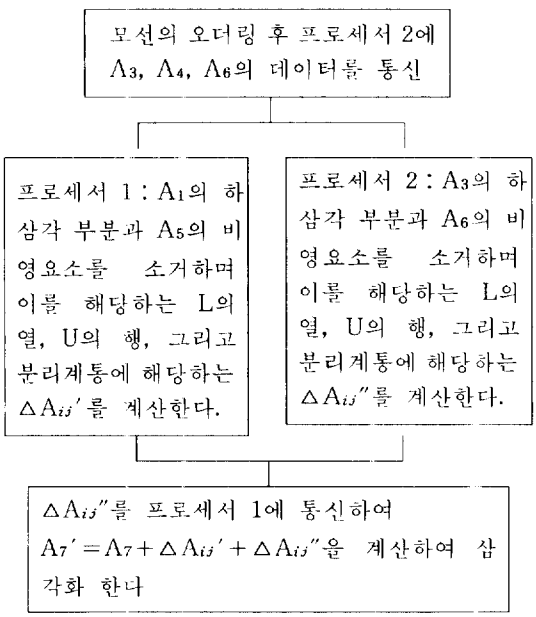


그림 4 병렬계산 알고리즘
Fig. 4 Parallel computing algorithm

5. 사례연구

본 논문에서 제시한 알고리즘을 IEEE 30모선 계통에 적용하였다. 계통도는 그림 5와 같으며 엔드모선을 모선 4와 10으로 선정하였다(이 두 모선을 제거할 경우 계통은 길게된다).

이 계통에 대해 시컨시얼 계산의 경우 및 엔드모선을 사용하여 프로세서를 2, 3, 4개를 사용할 경우에 대해 제시한 알고리즘의 효용성을 검토하였다.

이 계통의 준주변모선은 13, 26 등으로 여기서는 모선 13을 시작모선으로 하여 수준구조를 구하면 그림 6와 같이 된다.

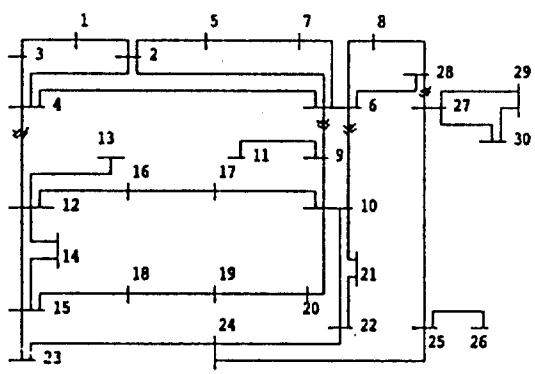


그림 5 IEEE 30모선 계통도
Fig. 5 IEEE 30 bus system

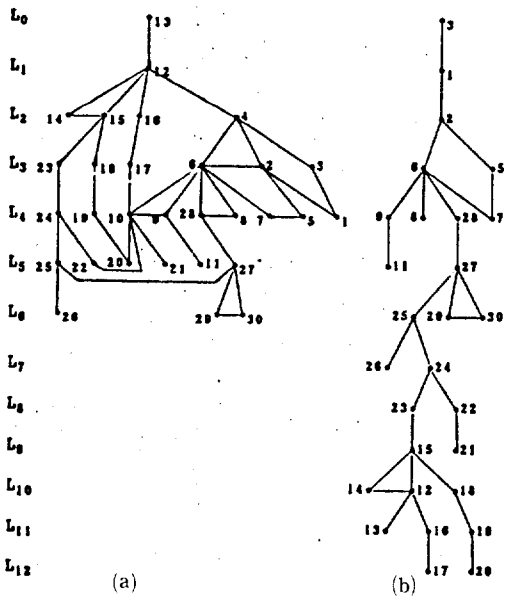
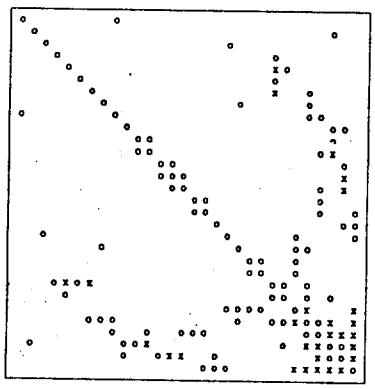
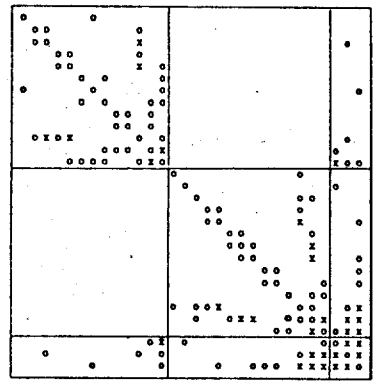


그림 6 IEER 30 모선계통의 수준구조
(a) 엔드모선을 사용하지 않았을 때의 수준구조
(b) 4, 10모선을 엔드모선으로 했을 때 수준구조

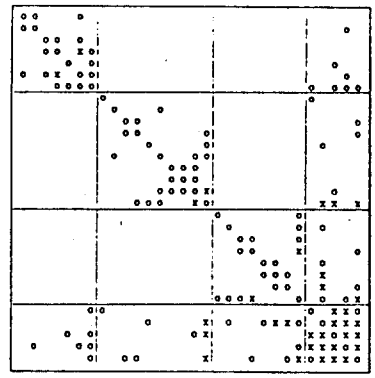
Fig. 6 Level structure of IEEE 30 bus test system
(a) Level structure of test system
(b) Level structure of end buses eliminated test systems (a) (b)



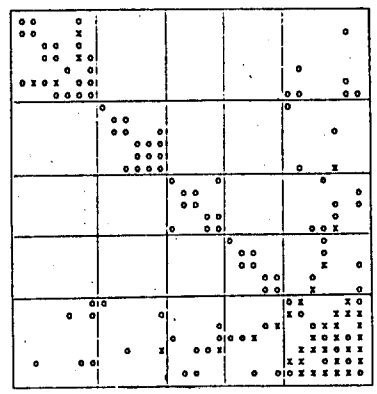
(a) 프로세서가 1개인 경우 행렬구조(시컨시얼 계산)
(a) Matrix structure in case of one procassor



(b) 프로세서가 2개인 경우 행렬구조
(b) Matrix structure in case of two procassors



(c) 프로세서가 3개인 경우 행렬구조
(c) Matrix structure in case of three processors



(d) 프로세서가 4개인 경우 행렬구조
(d) Matrix structure in case of four procssors

그림 7 프로세서가 1-4인 경우 오더링에 의한 행렬구조

Fig. 7 Matrix structures according to the number of processors

그림 6 (a)의 경우 수준구조는 넓게 퍼지고 수준의 길이가 짧다. 이 경우 중간 수준을 L_4로 하여 계통분리를 하면 분리자 모선이 24, 19, 10, 9, 28로 5개가 되며, 프로세서 1에는 16개 모선이, 프로세서 2에는 9개의 모선이 할당되여 계통분리 모선이 많아지게 되고, 프로세서간 계산부담이 불균형하게 된다. 따라서 가우스소거의 병렬계산은 비효과적일 수 있다.

엔드모선으로 4, 10 모선을 사용한 그림 6(b)의 경우 수준구조는 폭이 좁고 길어지게 되어 계통분리시 계통분리 모선의 수가 작아지게 되며, 프로세서간 계산부담이 비교적 평형이 된다.

그림 7(a)는 전계통의 모선을 MD 알고리즘을 사용하여 오더링했을 때 행렬의 구조를 나타내며, 그림 (b), (c), (d)는 각각 프로세서 2, 3, 4개로 병렬계산하기 위해 그림 6(b)의 수준구조에 의해 분리자를 선택하여 계통을 분리하고 각 분리된 계통에 속하는 모선, 분리자에 속하는 모선 그리고 엔드모선 순으로 오더링했을 때의 행렬 구조를 나타낸다(o : 원래의 비영요소, x : 새로 발생한 비영요소). 각 분류의 모선들은 시컨시얼 계산하여 일반적으로 사용하는 MD 알고리즘에 의해 오더링한다.

표 1 모선 오더링과 소거할 비영요소 수(NZ)
Table. 1 Results of bus ordering and No. of fill-in

계산 방식		모선의 오더링(30모선)	NZ
시컨시얼		11, 13, 26, 1, 3, 5, 7, 8, 9 14, 16, 17, 18, 19, 20, 21, 22, 23, 25, 28, 29, 30, 2, 4, 27, 6, 10, 12, 15, 24	53
병렬	p1	11, 1, 3, 5, 7, 8, 9, 28, 29, 30, 2, 27, 6(13)	(25)
	p2	13, 26, 14, 16, 17, 18, 19, 20, 21, 22, 23, 12, 15, 24(14)	(29)
sep+end		25+4, 10(3)	(3)
			32
병렬	p1	1, 3, 5, 7, 8, 2, 6(7)	(15)
	p2	11, 26, 21, 22, 23, 25, 29, 30, 27, 24(10)	(18)
			(16)
	p3	13, 14, 16, 17, 18, 19, 20, 12(8)	(9)
sep+end		9, 15, 28+4, 10(5)	27
병렬	p1	1, 3, 5, 7, 8, 2, 6(7)	(16)
	p2	11, 26, 25, 29, 30, 27(6)	(9)
	p3	14, 21, 22, 23, 15(5)	(11)
	p4	13, 16, 17, 19, 20(5)	(9)
sep+end		9, 28, 18, 12, 24+4, 10(6)	(15)
			31

이 행렬구조에 의해 소거해야 할 비영요소의 수를 표 1에 나타내었다.

표 1를 보면 시컨시얼 계산의 경우 소거해야 할 비영요소의 수는 53개이며 프로세서를 2, 3, 4개를 사용하여 병렬계산할 경우는 각각 32, 27, 31개로 계산시간을 약 반으로 줄일 수 있다(프로세서간 통신시간은 고려하지 않았음). 이계통의 경우는 프로세서의 수가 3개 정도가 적당하며 그 이상이면 계통분리시 분리계통에 속하는 모선수가 증가되어 소거할 비영요소의 수가 많아지게 된다. 이와같이 계통의 크기와 구성에 따라 프로세서의 수를 적절히 사용하는 것이 효과적이다.

6. 결 론

본 논문에서는 가우스소거시 기존의 성김성기법을 유지하면서 병렬계산하는 알고리즘을 개발하였다. 병렬알고리즘 개발에 중요한 점은 프로세서간 계산부담이 평형이 되고 통신량이 적으면서 병렬 계산부분을 극대화하는 것이다. 제시한 알고리즘은

1) 병렬성 개발을 위한 행렬의 블럭화 과정에 ND오더링 알고리즘을 적용하였다. 이 과정에서 엔드모선이라는 개념을 정의하고 이를 사용하여 해석하려는 계통의 수준구조를 폭이좁고 길다랗게 만드는 방법을 제시하여 분리자 모선수를 적게 함으로서 병렬 계산부분을 많게하고, 프로세서간 계산부담이 평형되도록 수준구조를 나눌 수 있게하였다.

2) 각 프로세서에 데이터 배정후 가우스소거 계산과정에 있어 분리계통 모선에 해당하는 행렬요소의 변화분 데이터만이 프로세서간 통신을 요하기 때문에 통신량에 의한 부담이 적다.

3) 사례연구를 통하여 알고리즘의 적용시 계통의 크기에 따라 적정한 수의 프로세서를 사용해야 최적으로 계산시간을 단축할 수 있음을 알 수 있었고, IEEE30 모선 계통의 경우 데이터통신시간을 포함하지 않을 때 계산시간은 약 반 정도로 단축할 수 있음을 해석적으로 알 수 있었다.

실계통은 수백 또는 천 모선 정도의 크기를 가지므로 대상계통에 따라 프로세서의 적정한

수를 검토 사용하여야 하며 계통의 폭이 좁고 길게 되도록 엔드모선을 선정하면 계산시간을 상당히 줄일 수 있을 것으로 기대된다.

참 고 문 헌

[1] Sergio Pissanetzky, "Sparse Matrix Technology," Academic Press, Inc., 1984.

[2] W.F. Tinney, V Brandwajn, "Sparse Vector Methods," IEEE Trans. on Power Apparatus and Systems, February,1985.

[3] Y. Wallach, "On block-parallel methods for solving linear equations," IEEE Trans. Comput., Vol. C-29, pp.354-359, May 1980.

[4] J. Fong, "Parallel processing of power system analysis problems via simple parallel microcomputer structures," IEEE Trans. Power App. Syst., Vol. PAS-97, pp. 1834-1841, Sept. 1978.

[5] W.L. Hatcher, "A feasibility study for the solution of transient stability problems by multiprocessor structures," IEEE Trans. Power App. Syst., Vol. PAS-96, pp. 1789-1796, Nov. 1977.

[6] John W. Huang, "Optimal Parallel Triangulation of a Sparse Matrix," IEEE Trans. CAS, Vol CAS-26, No. 9, Sept. 1979.

[7] A. Padilha, "A W-matrix Methodology for Solving Sparse Network Ewuations on Multiprocessor Computer," 91 SM 482-0 PWRS.

[8] Christopher P. Arnold, "An Efficient Parallel Algorithm for the Solution of Large Sparse Linear Matrix Equation," IEEE Trans. on Computers, Vol. C-32, No.3, March 1983.

저 자 소 개

서의석(徐義錫)

1959년 11월 4일생 1981년 한양대학교 공과대학 전기공학과 졸업 1983년 서울대학교 대학원 졸업(공학석사) 1992년 한양대학교 대학원 졸업(공학박사) 1983~1994. 2 한국전기연구소 근무

오태규(吳泰圭)

1951년 4월 30일 생 1978년 서울대학교 공과대학 공업교육과 졸업 1984년 Iowa 주립대 대학원 졸업(공학석사) 1986년 동 대학원 졸업(공학박사) 1978~1982년 현대건설 근무 1991~1992년 Pennsylvania 주립대 Visiting Scholar 현재 한국전기연구소 전력계통연구부장