전력계통의 자코비안행렬 가우스소거의 병렬계산

서 의 석 , 오 태 규
한국전기연구소

Parallel Computation Algorithm of Gauss Elimination in Power system Analysis

Eui-Suk Suh , Tae-Kyoo Oh
Korea Electrotechnology Research Institute

ABSTRACT

This paper describes an parallell 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 using ND(nested dissection) ordering develops the parallelism in the Gauss elimination to have balance of computing load among processes and each processor uses the sequential computation method to preserve the sparsity of matrix.

1. 서론

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

전력계통에서 많이 취급하는 대수방정식에 있어 행렬 A는 매우 성김성이 강하고 대칭이며 정칙행렬이다. 대수방정식에 있어 성김성벡터 기법을 이용하여 행렬 A를 삼각분해하고 삼각분해된 행렬과 기지의 벡터를 가지고 순방향 역방향 대치에 의해 방정식의 해를 구하는 것이 일반적이다. 그렇지만 전력계통 문제에 있어 해를 구하는데는 많은 반복계산을 요하고 있고, 해석의 온라인화를 위해서는 보다 고속계산을 요한다.

성김성이 없고 비교적 작은 차원의 행렬에 있어서는 프로세서 어레이 구조의 병렬컴퓨터에 적합한 방식으로 병렬계산 소프트웨어가 개발되어 왔다. 그렇지만 전력계통에 있어 행렬은 규모가 크고 매우 성김성이 강하기 때문에 이런 방식을 사용할 경우 영요소의 계산을 포함하며, 프로세서간 데이타통신량이 많아 매우 비효율적이다.

본 연구에서는 각 프로세서 내에서는 시컨시얼 계산방식의 성김성기법을 유지하면서 프로세서간 데이타통신을 최소화하도록 가우스소거의 병렬성을 개발하여 병렬계산 하는 기법을 제시한다. 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.

2. 행렬 A의 가우스소거법을 이용한 삼각화

가우스소거는 선형대수방정식을 푸는데 잘알려진 방법이다. 열단위의 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 그리 문제가 발생하지 않는다.

다음과 같이 일부 열이 소거된 후의 행렬을 생각하자.

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

여기서 A(1)=A, 행렬의 공백부분은 소거된 요소이다. 이와 같이 나타낼 경우 삼각화과정의 k열 소거단계에서 행렬의 요소를 수식으로 나타내면 다음과 같이 된다.

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

여기서

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

식(2)에 의해 가우스소거를 완료하면 식 (3)과 같이 된다.

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

여기서 $U=A(n+1)$, $U_{kj} = A_{kj}(k)/A_{kk}(k) \equiv A_{kj}(k+1)$

따라서 행렬 A의 삼각분해는 식(3)에 의해 다음과 같이 된다.

$$A = LU \qquad (4)$$

여기서 $L = D_1 L_1{}^c D_2 L_2{}^c \cdots D_n L_n{}^c$, $L_{ik} = A_{ik}(k)$

이 삼각분해시 각요소들을 보면 식(2)에서

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

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

식 (6)에서 알수 있듯이 시컨시얼 알고리즘에서 k 열의 비

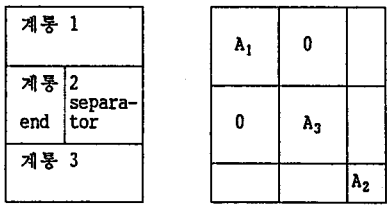
영요소를 소거할 때 그 이전 단계의 값이 작용한다.

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 같이 대각행렬의 형태가 되면 가능하다. 행렬의 구조는 모선의 오더링(ordering)에 따라 다르게 되며 본 논문에서는 수준구조(level strcture) 그래프와 분리자(separator)에 의해 모선을 구분하여 오더링하는 것에 의해 대각행렬의 구조가 되도록 하였다. 그리고 이 과정에 있어 보다 효과적으로 병렬성을 찾아내기 위해 경험적인 방법을 도입(end 모선사용)하였다.

3-1 .병렬성을 갖는 행렬구조

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



(a) 전력계통분리 (b) ND 방법에 의한 행렬의 구조

그림 1 전력계통 모선의 분류와 행렬의 구조

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(end 와 separator 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. 이를 위해 엔드모선(end buses: 맨마지막에 오더링 함)이라는 개념을 도입하였으며 이들 모선을 제거하면 계통은 보다 폭이 좁고 길게 된다. 이 엔드모선은 전력계통의 구조를 알고 있으므로 쉽게 선택할 수 있다.

계통에서 이 엔드모선을 제거한 후 주변모선을 시작모선으로 하여 전력계통의 수준구조(level structure)를 만들고 이 수준구조의 중간부분을 분리자(separator)로 선택한다. 이와 같이 하여 계통의 모선들을 분류하고 계통 1, 계통 3, 분리자 모선, 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 수준구조를 만든다. 만약 어떤 모선 v에 대해 수준구조가 r에 루트화된 수준구조보다 길으면 r은 v로 대치하여 step 3로 간다.

step 5 r을 준주변모선으로 한다.

2) 수준구조

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

$$Adj(L_i) \subseteq L_{i-1} \cup L_{i+1}, \quad 0<i<m$$
$$Adj(L_0) \subseteq L_1$$
$$Adj(L_m) \subseteq L_{m-1}$$

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. 그레프를 G=(V,E), 방문된 모선의 집합을 V_v라 하자.

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

step 2 만약 queue가 비어 있으면 stop. 달리는 queue의 rear에서 모선v를 제거하고, v에 근접한 방문하지 않은 모선의 집합S를 찾는다: $S = Adj(v) \cap (V-V_v)$

step 3 만약 S={}이면 step 2로 가고, 달리는 w∈S 인 각각의 w에 대해
(3a) queue의 front에 w를 더하라.
(3b) level(w)=level(v)+1
(3c) w를 방문된 것으로 나타내라: $V_v=V_v \cup \{v\}$

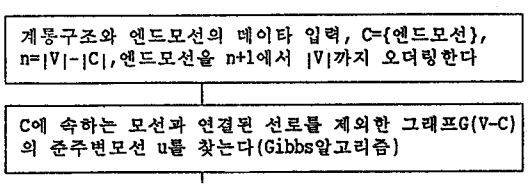
step 4 step 2로 가라.

3) 삼각화시 새로운 비영요소의 수를 줄이는 알고리즘

트리구조의 그레프를 갖는 행렬은 단조오더링될 때 가우스소거는 새로운 비영요소의 발생없이 실행할 수 있다. 행렬의 그래프가 트리가 아닌 경우는 일부 모선들을 하나로 묶어 트리구조를 갖게 할 수 있다. 이와같이 하여 오더링하면 새로운 비영요소의 발생을 최소로 할 수 있다.

3-3 모선의 오더링 알고리즘

행렬과 연관된 방향성이 없는 그래프가 주어지면 알고리즘은 준주변모선에 루트화된 수준구조를 만든다. 이때 최적분리자가 중간수준에서 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 하자.



G(V-C)의 u에 루트화된 수준구조 L_0, L_1,···,L_m을 만듬

계통을 분리하여 오더링 한다.i=0, k=0, n_1=0이라 하여
1) n/(p-k)의 반올림한 수를 n_p라 하고, 수준 L_i 부터 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 $R_k=\{L_1,\cdots,L_j-S_k\}$의 모선 수를 $n_2=|R_k|$라 하고, R_k에 속하는 모선을 삼각화시 비영요소가 최소가 되도록 n_1+1에서 n_1+n_2 까지 번호를 주어 오더링한 후 $n_1=n_1+n_2$로 한다.
4) 만약 k가 p-2이면
$R_{k+1}=\{L_{j+1},\cdots,L_m\}$, $n_2=|R_{k+1}|$로 하여 R_{k+1}을 같은 방법으로 n_1+1에서 n_1+n_2 까지 번호를 주어 오더링하고, $S_0,\cdots,S_{p-2}$의 순으로 분리자를 n_1+n_2+1에서 |V|-|C|까지 번호를 주어 오더링하고 마친다
만약 k<p-2이면
i=j+1, n=n-n_2-$|S_k|$, k=k+1로 하고 단계 1)으로 감

그림 2 모선의 오더링 알고리즘

4. 가우스소거의 병렬계산

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

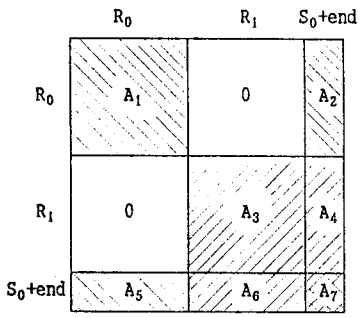


그림 3 제시한 오더링 방법에 의한 행렬구조

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

이 행렬의 구조를 살펴보면 가우스소거시 즉, 대각행렬 A_1의 하삼각 부분의 비영요소를 소거할 때 영향을 받는 부분은 A_1과 A_2 부분이며 A_5의 비영요소소거는 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

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

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

$$= \Delta A_{ij}' + \Delta A_{ij}''$$

여기서 Ω_1 = { A_1행렬에 속하는 모선 들 }
Ω_2 = { A_2행렬에 속하는 모선 들 }
Ω = { Ω_1,Ω_2 }

와 같이 분리되므로 두 부분의 영향을 각각 계산할 수 있다.

이들 계산과정을 도식적으로 나타내면 그림 4와 같다.

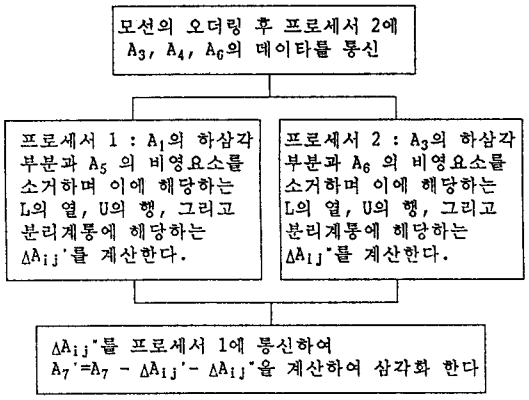


그림 4 병렬계산 알고리즘

5. 사례연구

본 논문에서 제시한 알고리즘을 IEEE 30모선 계통에 적용하였다. 계통도는 그림 5와 같으며 엔드모선을 모선 4와 10으로 선정하였다.

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

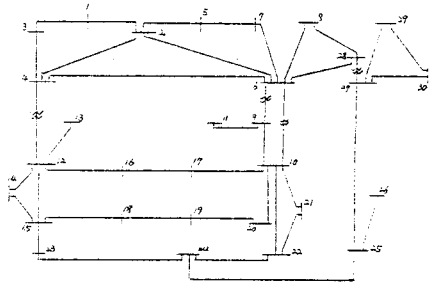


그림 5 IEEE 30 모선 계통도

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

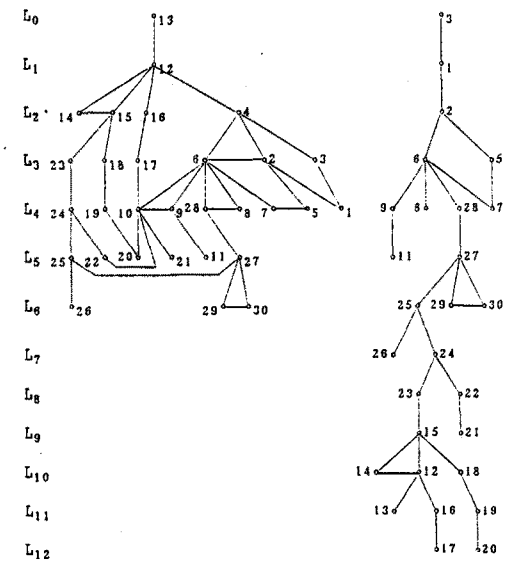
그림 6(a)의 경우 수준구조는 넓게 퍼지고 수준의 길이가 짧다. 이 경우 중간 수준을 L_4로 하여 계통분리를 하면 분리자 모선이 24, 19, 10, 9, 28로 5개가 되며, 프로세서 1에는 16개 모선이, 프로세서 2에는 9개의 모선이 할당되어 계통분리 모선이 많아지게 되고 프로세서간 계산부담이 불균형하게 된다. 따라서 가우스소거의 병렬계산은 비효과적일 수 있다.

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 수준구조 6(a)를 사용하여 삼각화과정시 새로운 비영요소의 수가 최소가 되도록 오더링했을 때 행렬의 구조를 나타내며, 그림 7(b), (c), (d)는 각각 프로세서 2, 3, 4개로 병렬계산하기 위해 그림 6(b)의 수준구조에 의해 계통분리를 하여 오더링했을 때의 행렬 구조를 나타낸다(o:원래의 비영요소, x:새로 발생한 비영요소).

이 행렬구조에 의해 소거해야할 비영요소의 수를 표 1에 나타내었다.

표 1을 보면 시컨시얼 계산의 경우 소거해야할 비영요소의 수는 77개이며 프로세서를 2, 3, 4개를 사용하여 병렬계산할 경우는 각각 38, 34, 45개로 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. 계통이 클수록 프로세서의 수가 많은 것이 더욱 효과적이고 계산시간을 더욱 줄일 수 있다.



(a) 앤드모선을 사용하지 않았을 때의 수준구조

(b) 4, 10 모선을 앤드모선으로 했을 때 수준구조

그림 6 IEEE 30 모선 계통의 수준구조

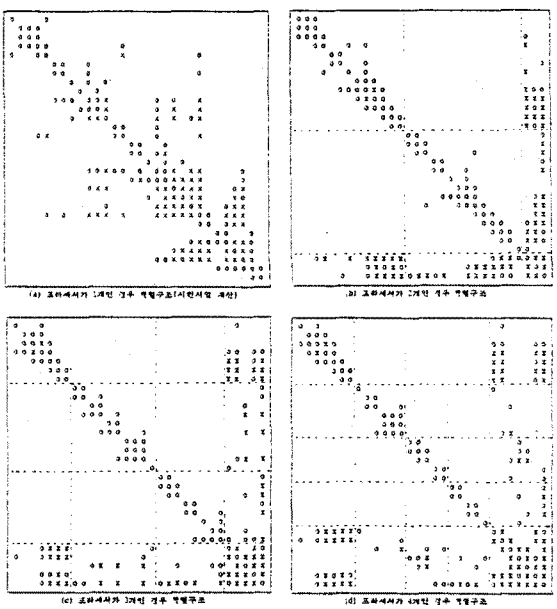


그림 7 프로세서가 1-4인 경우 오더링에 의한 행렬구조

표 1 모선 오더링과 소거할 비영요소 수(NZ)

계산방식	모선의 오더링(30모선)	NZ
시컨시얼	26,30,29,27,25,21,22,20,11,10,9,19,8,28 5,7,1,6,2,17,3,18,24,23,14,15,16,4,12,13	77
병렬 p1	29,30,27,28,11,9,8,7,6,5,2,1,3 (13)	(35
p2	20,19,18,17,16,13,14,12,15,23,24,22,21 ,26(14)	(34
sep+end	25 + 4,10(3)	(3) 38
병렬 p1	8,7,5,6,2,1,3(7)	(25
p2	21,22,23,24,26,25,29,30,27,11(10)	(20
p3	20,19,18,17,16,13,14,12(8)	(16
sep+end	9,28,15+4,10(5)	(9) 34
병렬 p1	8,7,6,5,2,1,3(7)	(29
p2	11,26,25,29,30,27(6)	(9)
p3	14,15,23,21,22(5)	(12
p4	17,16,20,19,13(5)	(9)
sep+end	9,28,24,12,18+4,10(6)	(16 45

6. 결론

본 논문에서는 가우스소거시 기존의 성김성기법을 유지하면서 병렬계산하는 알고리즘을 개발하였다.

본 논문에서 개발한 알고리즘은

1) 어레이 방식(full matrix사용)이 아닌 알고리즘 자체의 병렬성을 개발하여 멀티컴퓨터 개념의 병렬계산을 할 수 있도록 하였다.

2) 가우스소거의 병렬성을 개발하기 위해 행렬의 구조를 대각형으로 변경하는 ND(nested dissection)알고리즘의 적용에 있어 앤드모선(end bus)을 도입하여 폭이좁고 길다란 수준구조를 갖도록 하였다. 이것에 의해 프로세서간 계산부담을 비교적 평형이 되게 하였으며, 계통분리 모선 수를 적게할 수 있었다.

3) 제시한 알고리즘의 적용시 계통의 크기에 따라 적정한 프로세서의 수를 선정해야 함을 알 수 있었다.

실계통은 수백 모선 또는 천 모선 정도의 크기를 가지므로 대상계통에 따라 프로세서의 적정한 수를 검토사용하여 계산시간을 상당히 줄일 수 있을 것으로 기대된다.

참고문헌

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

[2] Danial J. Tylavsky, Fernando Alvarado, "Parallel Processing in Power Systems Computation", Transactions on Power Systems, Vol. 7, No. 2, May 1992.

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

[4] Yu and Wang, "A New Paralell LU Decomposition Method", IEEE Trans. on Power Systems, Feb. 1990.