AN IMPROVED BLOCK-PARALLEL NEWTON METHOD VIA EPSILON DECOMPOSITIONS FOR LOAD-FLOW CALCULATIONS

M. Amano, Member, IEEE
Hitachi Research Laboratory
Hitachi, Ltd.
Hitachi, 319-12 JAPAN

A. I. Zečević D. D. Šiljak, Fellow, IEEE
Department of Electrical Engineering
Santa Clara University
Santa Clara, CA 95053 USA

Abstract. The objective of this paper is to present a new method for parallel load-flow calculations based on an effective decomposition of the network. In the solution process we utilize the block-parallel Newton method developed in [1], which involves only diagonal blocks of the Jacobian. The underlying structure is obtained by applying the epsilon decomposition algorithm [2,3], which eliminates weak coupling elements from the matrix. We demonstrate that the iterative process can be significantly accelerated by making certain modifications in mismatch evaluation for buses connecting different blocks. Experiments on the hypercube confirm that the proposed method is indeed effective, particularly for problems where a good initial approximation is available (such as outage assessment).

Keywords: load-flow problem, parallel processing, block-iterative solutions, network decomposition.

1. INTRODUCTION

Parallel algorithms for solving systems of algebraic equations are generally classified as either *direct* or *iterative*. Direct methods are based on a variety of principles, such as the inverse matrix lemma [4], decomposition into the bordered block diagonal form [5] or elimination trees [6]. There are no convergence problems associated with these methods, but it may be difficult to achieve a high level of parallelism and maintain a low communication overhead. In contrast, iterative procedures such as the Jacobi and Gauss-Seidel methods [7] typically possess straightforward parallelism and require only minimal communications, but exhibit slow convergence. In such methods, the number of iterations usually increases with the number of processors used.

In this paper we will utilize the block-parallel Newton iterative method [1] to solve load-flow equations, and propose a new acceleration technique to overcome the problem of slow convergence. The parallelization is based on the *epsilon decomposition* algorithm [2, 3], which partitions large scale systems into weakly coupled subsystems. More specifically, given matrix A and a sufficiently small number $\varepsilon > 0$, the decomposition results in a permuted matrix:

$$\tilde{A} = A_D + \varepsilon A_N \tag{1.1}$$

where A_D is block diagonal and all elements of A_N have magnitude less than or equal to one.

In order to use epsilon decompositions in solving systems of nonlinear algebraic equations:

$$f(x) = 0 \tag{1.2}$$

it is necessary to consider the *simplified* Newton method [7]:

$$x^{(k+1)} = x^{(k)} - J^{-1}(x^{(0)})\, f(x^{(k)}) \tag{1.3}$$

in which the Jacobian is computed only at the initial value $x^{(0)}$. It has been shown in [1] that whenever $J(x^{(0)})$ has an epsilon decomposition:

$$J(x^{(0)}) = J_D + \varepsilon J_N \tag{1.4}$$

for a sufficiently small ε, the weak coupling term can be discarded; equation (1.3) then reduces to:

$$x_i^{(k+1)} = x_i^{(k)} - J_i^{-1} f_i(x^{(k)}) \quad (i = 1, 2, \ldots, n) \tag{1.5}$$

where J_i ($i = 1, 2, \ldots, n$) are the diagonal blocks of matrix J_D. This modified procedure is ideally suited for parallel computing, since each block can be factorized independently by a different processor and the only communication involves updating $f_i(x^{(k)})$ in each iteration. We should point out, however, that (1.5) also results in slower convergence, so an effective parallelization ultimately depends on our ability to secure faster convergence. Consequently, a key objective of the following sections will be to present an acceleration method that can achieve this in load-flow calculations.

A variety of methods, such as successive overrelaxation and extrapolation [8], are available for accelerating iterative procedures. However, none of these methods was able to sufficiently reduce the number of iterations in the block-parallel Newton process. This motivated the development of a new method, which accelerates the iterative process by modifying the mismatch evaluation of buses which connect

95 SM 598-3 PWRS A paper recommended and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the 1995 IEEE/PES Summer Meeting, July 23-27, 1995, Portland, OR. Manuscript submitted December 28, 1993; made available for printing June 20, 1995.

different blocks. Numerical experiments have shown that this technique can significantly reduce the number of iterations; in the following, iterative process (1.5) with such an acceleration will be referred to as the *improved* block-parallel Newton method. Computational results using the IEEE 118 bus system and an 888 bus network on an 8-node hypercube will be presented to illustrate that the proposed method is indeed effective, particularly when a good initial approximation is available (as in the case in outage assessment).

2. APPLICATION OF THE BLOCK-PARALLEL NEWTON METHOD TO THE LOAD FLOW PROBLEM

The load-flow problem is described by a system of nonlinear algebraic equations:

$$F(x; P, Q) = 0 \tag{2.1}$$

where x is a variable vector of bus voltages, and P and Q are input vectors of injected powers. For an n-bus system without the slack bus, (2.1) can be reformulated as:

$$F_i(x_1, x_2, ..., x_n; P_i, Q_i) = 0 \quad (i = 1, 2, ..., n) \tag{2.2}$$

where

$$F_i = [F_{P_i}, F_{Q_i}]^T, \quad x_i = [f_i, e_i]^T \tag{2.3}$$

and

$$F_{P_i} = P_i - \mathrm{Re}\left(E_i \sum_{k=1}^{n} Y_{ik}^* E_k^*\right) \tag{2.4}$$

$$F_{Q_i} = Q_i - \mathrm{Im}\left(E_i \sum_{k=1}^{n} Y_{ik}^* E_k^*\right) \tag{2.5}$$

In the above equations, $E_i = e_i + jf_i$ represents the complex node voltages, $S_i = P_i + jQ_i$ the injected power and $Y_{ik} = G_{ik} + jB_{ik}$ the admittance. For PV buses, where the voltage magnitude V_i is fixed, equation (2.5) is replaced by

$$F_{Q_i} = V_i^2 - (e_i^2 + f_i^2) \tag{2.6}$$

The standard Newton method

$$x^{(k+1)} = x^{(k)} - J^{-1}(x^{(k)})\, F(x^{(k)}) \tag{2.7}$$

is the most general method for solving such systems of nonlinear equations. Iterations (2.7) are typically updated using LU factorization and forward/backward substitution in conjunction with sparse matrix techniques [8]. In most applications, however, method (1.3) is known to be an adequate substitute for (2.7) [9]. Furthermore, as demonstrated earlier, when $J(x^{(0)})$ has an epsilon decomposition (1.3) can be additionally simplified, resulting in procedure (1.5) which is ideally suited for parallel processing.

Obtaining an epsilon decomposition for the Jacobian is a fundamental step in the parallel solution of (1.5). Choosing *flat start* values

$$e_i = 1, \quad f_i = 0 \quad (i = 1, 2, ..., n) \tag{2.8}$$

as the initial approximation, and assuming (for simplicity) that:

$$\sum_{k=1}^{n} G_{ik} = 0 \quad (i = 1, 2, ..., n) \tag{2.9}$$

$$\sum_{k=1}^{n} B_{ik} = 0 \quad (i = 1, 2, ..., n) \tag{2.10}$$

the Jacobian computed at $x^{(0)}$ becomes:

$$J(x^{(0)}) = \left[\begin{array}{cc:cc:c:cc} B_{11} & -G_{11} & B_{12} & -G_{12} & \cdots & B_{1n} & -G_{1n} \\ G_{11} & B_{11} & G_{12} & B_{12} & & G_{1n} & B_{1n} \\ \hdashline \vdots & & \vdots & & & \vdots & \\ \hdashline B_{n1} & -G_{n1} & B_{n2} & -G_{n2} & \cdots & B_{nn} & -G_{nn} \\ G_{n1} & B_{n1} & G_{n2} & B_{n2} & & G_{nn} & B_{nn} \end{array}\right] \tag{2.11}$$

For PV nodes, the 2×2 blocks are substituted by:

$$\begin{bmatrix} B_{ik} & -G_{ik} \\ 0 & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} B_{ii} & -G_{ii} \\ 0 & -2 \end{bmatrix} \tag{2.12}$$

Although $J(x^{(0)})$ is a $2n \times 2n$ matrix, for decomposition purposes it is convenient to treat each 2×2 block as a single element. In particular, since in most cases B_{ik} is substantially larger than G_{ik}, we will generally apply epsilon decomposition to matrix $B=[B_{ik}]$ and subsequently use the result to partition the original Jacobian; this reduces the problem dimension by a factor of 2.

As an illustration, we will consider the IEEE 14 bus system [10], with matrix B shown in Fig. 1. The epsilon decomposition algorithm [2, 3] first removes the weak coupling by eliminating the off-diagonal elements which have magnitude less than or equal to ε, then collects the buses in the same blocks which keep connections after the elimination. Selecting ε=4.1, we obtain four diagonal blocks shown in Fig. 1. It can be seen that all off-diagonal block elements have magnitude less than or equal to ε.

As for the selection of the parameter ε, there is a simple rule that a larger ε results in smaller blocks, and the desired value can be found by changing the parameter according to the magnitude of the off-diagonal elements. In the example, ε=4.1 corresponds to the off-diagonal element "-4.1". If more blocks are needed, the element "-4.8" can be found, and by setting ε=4.8 the first diagonal block is divided as shown by the dotted lines (note that choosing ε=4.4 does not change the decomposition). If the number of blocks is specified, the desired decomposition can be derived changing ε in such manner. A brief additional explanation of the algorithm using a simple example is presented in the Appendix.

For efficient parallel processing, it is important to balance the block sizes. Since the blocks obtained in Fig. 1 are unbalanced due the wide range of values of B_{ik}, we now propose to *scale* equations (2.4) and (2.5) by B_{ii}. Setting ε=0.48, the epsilon decomposition results in the matrix shown in Fig. 2. The diagonal blocks are obviously more balanced than those in Fig. 1.

Next we show its application to the IEEE 118 bus network. The result of the epsilon decompositions is

$B =$

30.3	-5.2	-4.8	-5.1	0	0	0	0	0	0	0	0	0
-5.2	34.9	0	-21.6	0	0	0	0	0	0	-4.0	0	0
-4.8	0	9.8	-5.1	0	0	0	0	0	0	0	0	0
-5.1	-21.6	-5.1	38.3	-4.8	0	-1.8	0	0	0	0	0	0
0	0	0	-4.8	19.6	-5.7	-9.1	0	0	0	0	0	0
0	0	0	0	-5.7	5.7	0	0	0	0	0	0	0
0	0	0	-1.8	-9.1	0	24.3	0	-10.4	-3.0	0	0	0
0	0	0	0	0	0	0	8.5	-4.4	0	-4.1	0	0
0	0	0	0	0	0	-10.4	-4.4	14.8	0	0	0	0
0	0	0	0	0	0	-3.0	0	0	5.3	0	-2.3	0
0	-4.0	0	0	0	0	0	-4.1	0	0	17.3	-6.1	-3.2
0	0	0	0	0	0	0	0	0	-2.3	-6.1	10.7	-2.3
0	0	0	0	0	0	0	0	0	0	-3.2	-2.3	-5.4

Fig. 1. Decomposition of 14-bus system (without scaling).

$\bar{B} =$

1	-0.17	-0.16	-0.17	0	0	0	0	0	0	0	0	0
-0.15	1	0	-0.62	0	0	0	0	0	0	-0.11	0	0
-0.49	0	1	-0.52	0	0	0	0	0	0	0	0	0
-0.13	-0.56	-0.13	1	-0.12	0	-0.05	0	0	0	0	0	0
0	0	0	-0.24	1	-0.29	-0.46	0	0	0	0	0	0
0	0	0	0	-1.00	1	0	0	0	0	0	0	0
0	0	0	-0.07	-0.37	0	1	0	-0.43	-0.12	0	0	0
0	0	0	0	0	0	0	1	-0.52	0	-0.48	0	0
0	0	0	0	0	0	-0.70	-0.30	1	0	0	0	0
0	0	0	0	0	0	-0.57	0	0	1	0	-0.43	0
0	-0.23	0	0	0	0	0	-0.24	0	0	1	-0.35	-0.18
0	0	0	0	0	0	0	0	0	-0.22	-0.57	1	-0.21
0	0	0	0	0	0	0	0	0	0	-0.59	-0.41	1

Fig. 2. Decomposition of 14-bus system (with scaling).

Table I. Decompositions and iterations.

Decompositions	Block size	Iterations
Epsilon	17 17 17 16 16 13 12 9	164
Heuristic	15 15 15 15 15 14 14 14	237

depicted in Fig. 3 with the shaded circles, in case ε=0.40 is chosen in the scaled matrix. Twenty-six blocks are obtained, but the sizes of the blocks are somewhat unbalanced in spite of the scaling. To obtain the balanced blocks, we now *reconnect* the blocks. The boundary lines in Fig. 3 show eight *balanced* blocks derived from the reconnection. It should be pointed out that the reconnection procedure is *not* difficult, since the decomposed blocks are already available and only the block sizes need to be considered. The reconnection process has not yet automated, but for on-line use, it will be necessary to develop an algorithm to select the ε and to reconnect the blocks according to the designated blocks number and block sizes.

The effectiveness of the decomposition procedure is demonstrated in Table I, in which we present a comparison with a heuristic partitioning that maps the 118 bus system onto 8 processors. The heuristic method is based on the geography of the network and considers only the balance of block sizes (not the coupling strength). We compared the number of iterations required by the block-parallel Newton process using a flat start and a mismatch of 0.001 p.u. as the termination criterion; expectedly, the epsilon decomposition required fewer iterations.

3. THE ACCELERATION METHOD

In order to obtain an efficient parallel implementation of the block-parallel Newton method, it is critical to reduce the number of iterations. Several standard acceleration methods such as successive overrelaxation and extrapolation did not show significant improvement in our experiments with

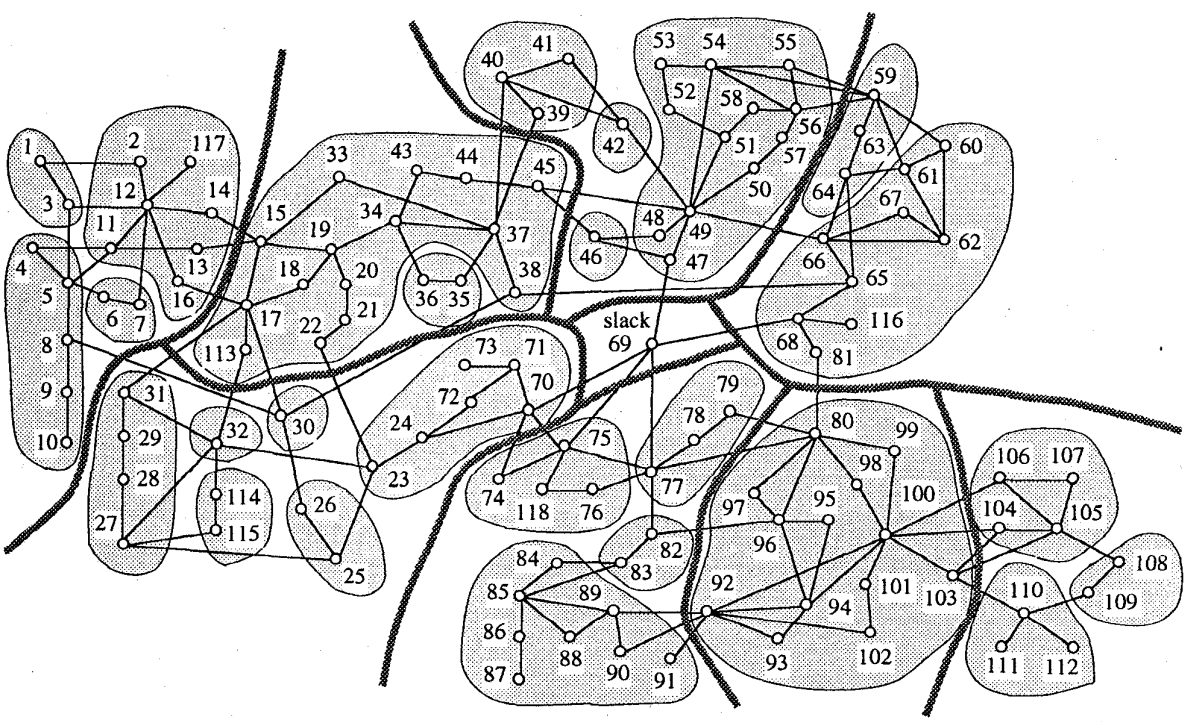

Fig. 3. Decomposition of 118-bus system.

iterations (1.5). We therefore propose a new acceleration method, based on certain modifications in mismatch evaluation.

We should first observe that equations (2.4) and (2.5) can be rewritten as:

$$F_{Pi} = P_i - \mathrm{Re}\left\{E_i Y_{ii}^* E_i^* - E_i \sum_{k=1}^{n} Y_{ik}^* (E_i - E_k)^*\right\} \quad (3.1)$$

$$F_{Qi} = Q_i - \mathrm{Im}\left\{E_i Y_{ii}^* E_i^* - E_i \sum_{k=1}^{n} Y_{ik}^* (E_i - E_k)^*\right\} \quad (3.2)$$

where the term $(E_i - E_k)$ corresponds to the voltage difference between the buses. Let us now assume that bus A and bus B belong to blocks A and B respectively, as illustrated in Fig. 4. We will also assume that the bus voltages are converging to their respective solutions E_A and E_B, as shown in Fig. 5. If solution E_B is already known, using the voltage difference $L_1 = (E_A^{(k)} - E_B)$ instead of $L_2 = (E_A^{(k)} - E_B^{(k)})$ is clearly the best way to evaluate mismatches. However, since E_B is *not* known at this point, we propose to use $L_3 = (E_A^{(k)} - \tilde{E}_B^{(k)})$ with $\tilde{E}_B^{(k)}$ given by

$$\tilde{E}_B^{(k)} = E_B^{(k)} + \alpha\,(E_A^{(k)} - E_A^{(k-1)}) \quad (0 < \alpha < 1) \quad (3.3)$$

As shown in Fig. 5, $\tilde{E}_B^{(k)}$ is an advancement of $E_B^{(k)}$ in the direction of $(E_A^{(k)} - E_A^{(k-1)})$. This is based on the supposition that $E_B^{(k)}$ was computed reflecting the previous value $E_A^{(k-1)}$, and with the update from $E_A^{(k-1)}$ to $E_A^{(k)}$, $E_B^{(k)}$ may also advance in the same direction. Numerical experiments show that the scheme (3.3) can improve the iterative process when a proper value of α is chosen. In the following section we will show how to select α and will demonstrate the effectiveness.

As for the assumption that the two bus voltages are converging to the similar directions as shown in Fig. 5, it should be pointed out that this is not always the case. When the voltages are converging to the opposite directions, the scheme (3.3) may deteriorate the convergency. However, we did not experience the deterioration in our experiments when a proper α is selected. Although further investigation is needed, we suppose the reason is that the approximate voltage difference is established in the first several iterations, and that the process does not usually work as the voltages

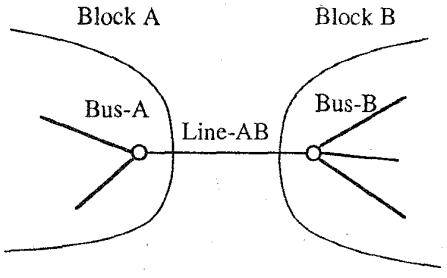

Fig. 4. Buses having a connection to another block.

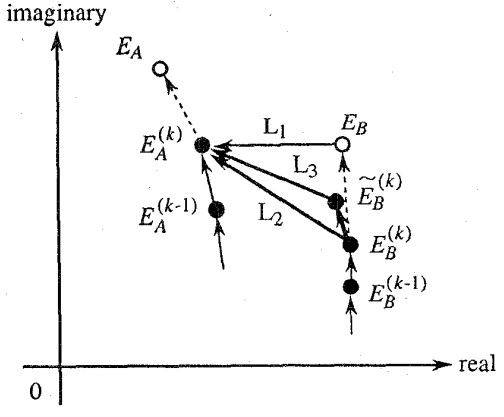

Fig. 5. Voltage difference between buses.

Table II. Number of iterations.

a. IEEE 118-bus, flat start

Number of processors	Without acceleration	With acceleration	
		Optimal α	Dynamic α
single	8	—	—
2	35	18 (α=0.4)	20
4	78	24 (α=0.6)	37
8	165	34 (α=0.7)	58

b. IEEE 118-bus, approximate start

Number of processors	Without acceleration	With acceleration	
		Optimal α	Dynamic α
single	2	—	—
2	10	5 (α=0.2)	6
4	20	5 (α=0.2)	6
8	38	7 (α=0.2)	8

c. 888-bus, flat start

Number of processors	Without acceleration	With acceleration	
		Optimal α	Dynamic α
single	12	—	—
2	199	59 (α=0.7)	95
4	666	96 (α=0.8)	92
8	NC	104 (α=0.8)	119

NC : not convergent after 2000 iterations.

d. 888-bus, approximate start

Number of processors	Without acceleration	With acceleration	
		Optimal α	Dynamic α
single	5	—	—
2	77	6 (α=0.2)	9
4	460	6 (α=0.2)	7
8	182	7 (α=0.2)	9

converge to the opposite directions with gradually increasing the voltage difference.

4. COMPUTATIONAL RESULTS

We now provide experimental results obtained for the IEEE 118 bus system and the 888 bus network. In Table II we present the number of iterations required by the block-parallel Newton process with a mismatch of 0.001 p.u. as the termination criterion. Two types of initial values were used - the standard *flat start* and the *approximate start*. The approximate start was obtained by adding random values ranging from -0.001 to 0.001 to the solutions. It was used to evaluate the cases where good initial approximations are provided. As seen from the table, the number of iterations is significantly reduced by the acceleration method when the optimal parameter α is selected. The optimal α was chosen by scanning from 0.1 to 0.9 in steps of 0.1. A smaller α is suitable for the approximate start, and a larger α is chosen for the cases where many iterations are required.

However, since it is impractical to select the optimal α at each load-flow computation, the following rule to decide α dynamically is introduced;

$$\alpha = \begin{cases} 0.02\,k + 0.3 & (1 \le k \le 25) \\ 0.8 & (k > 25) \end{cases} \qquad (k : \text{iteration step}) \qquad (4.1)$$

The process starts with a smaller α and increases α with the iteration steps. The parameters 0.02, 0.3, and 0.8 are derived empirically. The required iterations for the dynamic α are also shown in Table II. Although a few more iterations are required compared to those of the optimal α, substantial acceleration is achieved. The scheme (4.1) and the parameters were tested only for the two example systems, but we expect that the scheme can be applied to other networks because the same parameters are effective for the two different networks. Further investigation is necessary to confirm it.

Computation times using the Intel iPSC/860 8-node hypercube system are shown in Table III. The approximate start and acceleration with the dynamic α were used. The computation time includes ordering, LU factorization and block-Newton iterations (the time for data input and output is not included); the communication time is shown separately and is seen to be very modest. The speed-up was calculated with respect to computations using a *single* processor and the complete (unpartitioned) Jacobian; in all cases, a mismatch of 0.001 p.u. was used as the termination criterion. Table III shows that a substantial speed-up is obtained for both systems. In some cases, the speed-up actually exceeds the number of processors. It is because LU factorization of the decomposed Jacobian matrices needs less operations than that of the whole Jacobian, and if the convergence rate is very high, the total number of operations of the decomposed process (1.5) can be smaller than that of the process (1.3).

Table III. Computation time of approximate start cases.

a. IEEE 118-bus

Proc.	Iter.	Time I	Time II	Total	Speed-up
single	2	0.106	0	0.106	1
2	6	0.055	0.001	0.056	1.89
4	6	0.022	0.003	0.025	4.24
8	8	0.010	0.008	0.018	5.89

b. 888-bus

Proc.	Iter.	Time I	Time II	Total	Speed-up
single	5	0.873	0	0.873	1
2	9	0.291	0.006	0.297	2.94
4	7	0.146	0.001	0.147	5.94
8	9	0.077	0.007	0.084	10.39

Time I : Computation time (s),
Time II : Communication time (s).

The results in Table III indicate that the proposed method is extremely effective in cases when a good initial condition is available. It is therefore desirable to apply the method to problems such as outage assessment and voltage security assessment, where load-flow calculations are performed using good initial approximations. The results of outage assessment cases for the IEEE 118 bus system are shown in Table IV. Five cases were selected by randomly

Table IV. Computation time of outage assessment cases (IEEE 118-bus).

	Single		2		4		8	
	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)
Case 1	3	0.109	3	0.043	5	0.022	6	0.015
Case 2	3	0.112	5	0.053	12	0.036	19	0.036
Case 3	4	0.114	8	0.061	17	0.047	17	0.033
Case 4	4	0.115	11	0.071	21	0.056	31	0.056
Case 5	4	0.112	8	0.059	5	0.020	8	0.017
Total		0.562		0.287		0.181		0.157
Speed Up		1		1.96		3.10		3.58

choosing transmission lines and assuming line outages (removing the only line between the buses). The base case solutions were used as initial values and the parameter α was chosen according to the rule (4.1). The decompositions of the network were not changed for each contingency because we supposed that one line removal would not largely decrease the effectiveness of the base case decompositions. The Table shows that in Case 1 and 5 almost the same iterations are required as in the approximate start cases. The other cases require more iterations and the speed-up is smaller, but totally substantial speed-up is achieved.

We should note here that our load-flow calculation is based on (1.3) where the Jacobian is computed only once. It is pointed out that the method shows a slower convergence for heavy loaded systems. In such cases, the Jacobian should be computed at each iteration step (i.e. full Newton method), and the block-parallel process should be also modified to compute Jacobian more frequently. Further investigation is required to modify the iterative process and to verify the effectiveness of the proposed method in such heavy loaded cases.

We should also mention about the fast decoupled method where the voltage magnitude and the phase angle are updated in a decoupled manner [9]. The main difference is that the initial Jacobian matrix is replaced with the constant approximate matrices B' and B''. Therefore the decomposition must be made to the constant matrices instead of the Jacobian. Since the matrices B' and B'' have almost the same structure as the matrix B discussed in the previous section, we expect that the same decomposition procedure can be applied with a couple of modifications. We also suppose that the same acceleration method can be applied, although the expected speed-up may be a little smaller because the order of the Jacobian matrix is already reduced.

5. CONCLUSIONS

An improved block-parallel Newton method for load-flow calculations was presented. The number of iterations was significantly reduced by utilizing epsilon decompositions and the newly developed acceleration method. The method is suitable for on-line security assessment of outages and voltage stability, where many cases of load-flow with good initial approximations are performed. The proposed method improves the block-iterative process in general, and the same approach can be utilized for solving other large scale problems arising in networks, such as optimal power flow and transient stability analysis.

ACKNOWLEDGEMENTS

The research reported herein has been partially supported by the National Science Foundation under Grant ECS-9114872.

REFERENCES

[1] A. I. Zečević and D. D. Šiljak, "A block-parallel Newton method via overlapping epsilon decompositions," *SIAM Journal on Matrix Algebra and Applications*,1994.

[2] M. E. Sezer and D. D. Šiljak, "Nested epsilon decompositions and clustering of complex systems," *Automatica*, vol.22, pp.321-331, 1991.

[3] D. D. Šiljak, *Decentralized control of complex systems*, Cambridge, MA: Academic Press, 1991.

[4] R. Kasturi and M. Potti, "Piecewise Newton-Raphson load flow - an exact method using ordered elimination," *IEEE Trans. PAS*, vol.95, pp.1244-1253, 1976.

[5] A. Yokoyama *et al.*, "Automatic nonuniform decomposition of large-scale power systems for parallel processing," *Proc. of 9-th PSCC*, 1987.

[6] F. L. Alvarado, W. F. Tinney and M. K. Enns, "Sparsity in large-scale network computation," in *Control and Dynamic Systems*, vol.41, C.T.Leondes, Ed., San Diego CA: Academic, pp.207-272, 1991.

[7] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, New York: Academic Press, 1970.

[8] W. F. Tinney and C. E. Hart, "Power flow solution by Newton's method," *IEEE Trans.PAS*, vol.86, pp.1449-1456, 1967.

[9] B. Stott, "Review of load-flow calculation methods," *Proc. IEEE*, vol.62, pp.916-929, 1974.

[10] Y. Wallach, *Calculations and Programs for Power System Networks*, Englewood Cliffs, NJ: Prentice Hall, 1986.

BIOGRAPHIES

M. Amano (M'92) received his B.Eng. and M.Eng. degrees in electrical engineering from Kyoto University, Japan, in 1983 and 1985, respectively. He joined Hitachi Research Laboratory, Hitachi, Ltd. in 1985. In 1990 and 1991 he stayed in Santa Clara University as a visiting researcher. He is currently engaged in developing computer systems for power system analysis and control.

A. I. Zečević received the B.S. degree in electrical engineering from the University of Belgrade, Yugoslavia, in 1984 and his M.S. and Ph.D. degrees from Santa Clara University in 1990 and 1993, respectively.

He is currently teaching electronic circuits at Santa Clara University. His research interests include graph theoretic decomposition algorithms and parallel computation of large scale systems. He is particularly interested in applications to large electronic circuits, such as those arising in power systems and VLSI design.

D. D. Šiljak (F'81) received his Ph.D. degree in 1963 from the University of Belgrade, Belgrade, Yugoslavia.

Since 1964 he has been with Santa Clara University, Santa Clara, CA, where he is presently the B&M Swig Professor at the School of Engineering and teaches courses in system theory and applications. His research interest is in the theory of large-scale systems and its applications to problems in control engineering, power systems, economics, aerospace, and model ecosystems. He is the author of the monographs *Nonlinear Systems* (Wiley, 1969), *Large-Scale Dynamic Systems* (North-Holland, 1978), and *Decentralized Control of Complex Systems* (Academic Press, 1991).

Dr. Šiljak is an honorary member of the Serbian Academy of Sciences and Arts, Belgrade, Yugoslavia.

APPENDIX

The following is a brief explanation of the epsilon decomposition algorithm.

Fig.A.1 shows a simple example matrix and its network diagram. The first step of the algorithm is to eliminate the off-diagonal elements which have magnitude less than or equal to ε. The numbers with shade and the dotted arrows in Fig.A.2 indicate the eliminated elements when ε=0.4 is chosen.

The next step is to form blocks by combining the nodes which keep connections after the elimination. Necessary permutation of the corresponding rows and columns of the matrix is performed along with the clustering. As shown in Fig.A.2, two diagonal blocks are obtained by permuting node 2 and node 3.

The parameter ε can be selected according to the designated blocks number or block sizes regarding the magnitude of the off-diagonal elements. In the example, the non-zero off-diagonal elements are 0.2, 0.3, 0.4, 0.5, and 0.8. Selecting ε=0.2 or ε=0.3 does not provide a decomposition, and selecting ε=0.4 results in Fig.A.2. Choosing a larger value ε=0.5 provides more blocks as shown in Fig.A.3. If ε=0.8 is chosen, all nodes are decomposed into single element blocks.

More general and precise explanation of the algorithm including the overlapping decompositions or the nested decompositions is presented in [2] and [3].

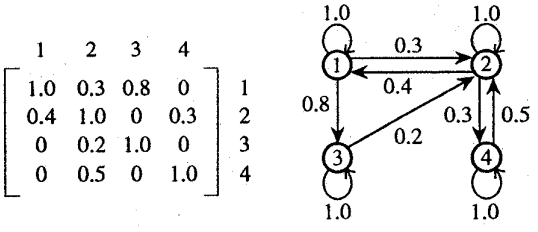

Fig. A.1. Example network.

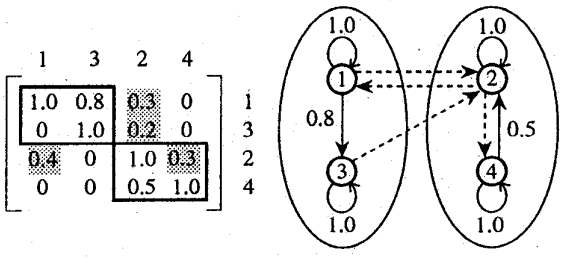

Fig. A.2. Decomposition with ε=0.4.

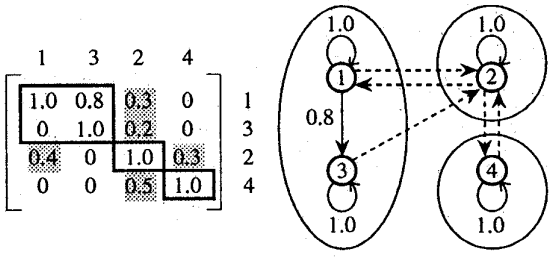

Fig. A.3. Decomposition with ε=0.5.

Discussion

F. L. Alvarado (The University of Wisconsin–Madison, ECE Department, 1415 Engineering Drive, Madison, WI 57706):

For some years we have been tantalizingly close to a new class of solvers to replace conventional ordered factored LU decomposition as the workhorse for power system engineering computations, but, in the opinion of this discusser, we are not there yet. This papers adds significant and interesting material to this debate. Here are some of the basic facts and issues, from the perspective of this reviewer:

- For direct methods based on LDU factorization, serial computation time grows slightly faster than linearly but less than quadratically for power system problems characterized by topology-symmetric matrices. This makes the direct method very hard to beat for most applications.
- When unsymmetric matrices are used (as is the case when exact Newton's method is used for formulations of the power flow problem that include unconventional controls, for example), the effectiveness of direct methods can be reduced sharply as a result of the need for off-diagonal pivoting.
- Recently, very effective direct method have been developed for unsymmetric direct matrices. Some of these remain to be tested in power system matrices of realistic size with a variety of realistic matrix entry values.
- Conventional implementations of direct methods are notoriously bad for paralell environments. Some row/column blocking techniques have been devised that improve things significantly.
- Blocking and partitioning can also improve the paralelism of direct methods. In a few cases, partitioning can actually improve the speed even in serial environments.
- There have been significant improvements (due to these and other authors) on methods for partitioning and blocking in recent years.
- Specialized ordering techniques enhance the ability to parallelize direct methods, at the expense of additional fill-in.
- Partitioned inverses (W-matrices) greatly enhance the ability to parallelize direct methods, at least for the repeat solution phase.
- Iterative linear solver methods are easy to implement and can be used to solve most power system problems.
- The convergence of iterative methods is dependent on the numerical values of the matrix entries.
- The number of iterations is a function of matrix condition and matrix dimension.
- For well-behaved (diagonal-dominant, symmetric, well conditioned) matrices, iterative methods work remarkably well, and are very easy to implement.
- Iterative methods are ideally suited for parallel environment or for automated vectorization and parallelization.
- For unsymmetric matrices, there are a number of good and effective iterative methods but their convergence properties are far more erratic.
- Preconditioners, which can be viewed as approximate direct solvers, can improve the performance of almost any iterative method.
- Approximate partitioned inverses are extremely effective as preconditioners for iterative methods.
- Explicit parallel programming has failed to gain widespread popularity due to a number of practical reasons. The future of parallel computation will arguably be with the use of parallizable algorithms rather than explict coding of parallel computations on specialized platforms.

We would like the authors reactions and views of these (admittedly biased) opinions and their views of the direction that parallel computation research should take in the next 5 to 10 years, both within the research community and within actual applications to real power systems. We also are interested in comments regarding how the industry might proceed with rather definitive testing and comparizons to establish the feasibility of all these new classes of methods (the author's own included) for truly practical systems and applications in practical portable environments.

Manuscript received August 21, 1995.

M. Amano, A. I. Zečević, D. D. Šiljak: The Authors would like to thank Professor Alvarado for his valuable comments, as well as for initiating a very interesting discussion on problems and future trends in parallel computation for power systems. Our response is given below.

1. We have long been of the opinion that blocking and partitioning are indeed the most promising ways to parallelize direct methods. Much of our recent research in this area has been devoted to parallel LU factorization based on balanced BBD decompositions [A1]. We found that the inherent parallelism of balanced BBD structures allows for an easy implementation of parallel factorization on various architectures, which is quite unlike some of the conventional implementations of direct methods.

In very sparse and moderately sized matrices (such as those arising in power systems) the balanced BBD approach proved to be most advantageous in cases where multiple factorizations need to be performed. The method was also effective for repeated forward/backward substitution, although the partitioned inverse technique seems hard to beat in that respect. We should also point out that the balanced BBD structure does *not* necessarily come at the expense of additional fill-in. Extensive experiments have shown that the amount of fill-in is, in fact, usually similar to that resulting from symmetric minimal degree ordering [A2].

As far as the LU factorization of nonsymmetric matrices is concerned, pivoting will clearly complicate the algorithm. However , we have found that if the balanced BBD decomposition is applied to an unsymmetric matrix, the pivoting can generally be confined to the diagonal blocks (and thus be performed in parallel). Our preliminary results in this area have been very encouraging, although further testing needs to be done before a definitive conclusion is reached.

2. We agree that iterative methods are ideally suited for parallel computation, and that convergence is the key issue in their effective implementation. Our experience with epsilon decomposition has shown that the availability of a good initial approximation is the most critical factor in reducing the number of iterations; although such an approximation typically exists in transient stability simulation and (to a lesser extent) in outage assessment, we recognize the need to reduce this sensitivity. Effective preconditioners such as approximate partitioned inverse are certainly a good way to resolve this problem. We have also considered the selective use of epsilon decomposition in conjunction with the balanced BBD decomposition; this form of incomplete factorization can significantly improve the convergence properties, while maintaining a high level of sparsity.

Regarding future trends in parallel computing, we agree with Professor Alvarado that inherently parallelizable algorithms will probably become prevalent. In our opinion, combinations of direct and iterative algorithms (e.g. balanced BBD and epsilon decompositions with acceleration) are also promising, particularly for very large systems. Considering practical use, the transient stability simulations as shown in [A3] will be the most possible applications. We belive that the improvement of the parallel algorithms together with the up-to-date processor technologies will enable a real-time or super real-time simulation of large power systems. It would be also important for the power industry to develop new supervisory controls using such extensively fast simulation techniques. As for definitive testing and comparisons of the parallel methods, the use of bench mark network models based on actual large-scale power systems will be indispensable.

[A1] A.I. Zečević and D.D. Šiljak, "Balanced decompositions for multilevel parallel processing," *IEEE Trans. on Circuits and Systems*, vol.41, pp.220-233, 1994.

[A2] A.I. Zečević and D.D. Šiljak, "A balanced decomposition algorithm for parallel solutions of very large sparse systems," *Proc. of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, San Francisco, CA, pp.436-441, February 1995.

[A3] H. Taoka, et al., "Real-time digital simulator for power system analysis on a hypercube computer," *IEEE Trans. on Power Systems*, vol.7, pp.1-7, 1992.

Manuscript received October 24, 1995.