A RANK-ORIENTED SETUP FOR THE COMPENSATION ALGORITHM

Robert A.M. van Amerongen

Delft University of Technology
Faculty of Electrical Engineering
Mekelweg 4, 2628 CD Delft, the Netherlands

Abstract - A procedure is given that solves modified network equations much more efficiently if the modification description suffers from rank deficiency. The key to the method lies in the application of a low-order product factorization to the modification matrix.
Consequently, the dimension of the reduced system will be determined by the rank of the modification and, thus, the computational process can be considerably speeded up.

Test results are given for networks up to 944 nodes. Two algorithms for the low-order product factorization are summarized in the appendix.

Keywords: network analysis, network modification, compensation algorithms.

1. INTRODUCTION

Many network computations in power system engineering require the solution of a large set of sparse linear or linearized equations all of which differ from a base case in respect of a small structural or parametrical change. Presuming that the coefficient matrix of the base case is held in its factorized form, and as the number of nodes that are related to this change usually is very small in comparison with the system size, a full refactorization of the modified coefficient matrix is wasteful and one has to choose from either compensation or partial matrix factor adaptation.

Basically, compensation [1] translates the network change into additional injections at those nodes that are involved with the modification. The current algorithms [2] take full advantage of the restricted size of the network modification by using sparse vector techniques [3]. With all these methods, the factorized coefficient matrix remains unchanged; the network change is taken into account before, during or after the substitutions depending on the specific type of compensation algorithm that is used. Applications of these techniques can be found in present-day procedures for fault analysis [4,5] and contingency screening [6].

On the other hand, partial matrix factor adaptation [7] directly affects the matrix factors by selectively refactorizing or updating only those rows and columns of the factors that undergo changes. Examples of applications can be found in algorithms for contingency screening [6] and for adjusted load flow solutions [8].

A modification setup is usually node oriented: a small matrix is made of which the dimension is equal to the number of nodes that are involved with the modification and which contains the description of this modification. With the aid of sparse matrices, consisting of unity vectors, the modification is connected to the full-sized matrix.

An alternative to this setup is the branch-oriented one: the modification is made of branch parameter changes and the connection matrices become branch-node incidence matrices [9]. Consequently, the dimension of the basic-modification matrix is now determined by the number of branches. This setup is preferred to the node-oriented one if the dimension of the basic modification is less than the number of nodes, because it will produce the fastest solution to the modified set of equations. This phenomenon always holds if the basic-modification matrix in a node-oriented setup is rank deficient.

There are, however, certain types of modifications that do suffer from rank deficiency but that are seemingly inconvertible to a form similar to the branch-oriented setup. The use of a node-oriented setup thus is inevitable and, as a consequence, prospective computational speed-up is relinquished. This situation can usually be found in structural network changes that are related to node creation: bus splits, breaker openings and so on.

This paper will fill this gap by using the concept of low-order product forms (l.o.p.f.). By l.o.p.f. we mean the decomposition of an m-dimensional basic-modification matrix of rank r into the product of two mxr matrices and an r-dimensional square (usually diagonal) matrix. This enables us to make a modification setup with the lowest possible dimension i.e. the rank r.

The idea of using l.o.p.f. is not a new one: presumably Alvarado was the first one who used it, and then in the context of a fault-analysis procedure [4]. In this paper we give a procedure that takes full advantage of its use. It is necessary also to give a different procedure to handle newly introduced nodes: current practice seems either to extend the reduced system in that new nodes can be handled explicitely or to reactivate dummy nodes in the full system; we give a procedure where the new nodes are eliminated from the description of the modification. This produces a modification with the smallest rank possible.

Although the technique is applicable both to partial matrix factor adaptation and to compensation, we restrict the treatment of this setup to the latter category, because our primary intention was to enhance current compensation algorithms. The question as to whether or not this new setup affects the discriminatory use of compensation and matrix factor adaptation (the former for simple and temporary modifications in a network solution that is solved only once, the latter for complicated or permanent changes or those that must be solved repeatedly) will remain unanswered.

The main part of this paper, therefore, concerns the introduction of the rank-oriented compensation algorithm (section 3) and its performance compared with the node-oriented one (section 4), that, itself, is summarized in section 2. Because rank deficiency often stems from structural changes related to node creation, section 5 uses the example of a bus split to show how to handle such cases and why the rank-oriented compensation algorithm is so beneficial in these cases. (The same procedure can be applied to fault-analysis procedures as well, but due to the complexity surrounding the use of different sequence networks and the occurrence of solid connections, it

89 SM 678-4 PWRS A paper recommended and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1989 Summer Meeting, Long Beach, California, July 9 - 14, 1989. Manuscript submitted January 30, 1989; made available for printing June 15, 1989.

will be the subject of another paper.) Finally, two algorithms for low-order product factorization are given in an appendix.

2. NODE-ORIENTED COMPENSATION ALGORITHM [2]

Let us assume that the following set of equations must be solved:

$$(A + M_1 \Delta M_2^T)\underline{x} = \underline{b} \tag{2.1}$$

where

- M_1, M_2 m-column connection matrices
- Δ a mxm basic-modification matrix
- $\underline{x}$, $\underline{b}$ vectors of unknowns and knowns respectively
- A a coefficient matrix that is held in its factorized form: A = LU

The modification concerns m nodes of the network. With a node-oriented setup of the modification, each connection matrix thus consists of m unity-vectors. The basic-modification matrix is rank deficient. The symbolical solution of (2.1) is given by the matrix-inversion lemma:

$$\underline{x} = (A^{-1} - A^{-1} M_1 \Delta S^{-1} M_2^T A^{-1})\underline{b} \tag{2.2}$$

where

$$S = (M_2^T A^{-1} M_1 \Delta + I) \tag{2.3}$$

- I a unity matrix

In actual computations the extreme sparsity of the connection matrices can most effectively be exploited by using the mid-compensation algorithm. The solution (2.2) must be rewritten into:

$$\underline{x} = U^{-1}(I - L^{-1} M_1 \Delta S^{-1} M_2^T U^{-1}) L^{-1} \underline{b} \tag{2.4}$$

where

- L, U the triangular factors of A: A = LU

In the computational procedure to obtain (2.4) we distinguish four steps that are relevant for the comparison with the new algorithm. These steps are:

I the preparatory step. This step mainly involves the calculation and the factorization of S (2.3) in the following way:

$$W_1 = L^{-1} M_1 \tag{2.5}$$

$$W_2^T = M_2^T U^{-1} \tag{2.6}$$

$$W_3 = W_2^T W_1 \tag{2.7}$$

$$S = (W_3 \Delta + I) \tag{2.8}$$

Both the W-matrices (2.5) and (2.6) are obtained by solving the related sets of linear equations by fast forward substitutions, followed by the calculation of their product (2.7). Finally, S is calculated and is factorized.

II the network forward solution. In this step an intermediate vector is calculated:

$$\underline{h} = L^{-1} \underline{b} \tag{2.9}$$

with a forward substitution.

III the compensation step. In this step the intermediate result (2.9) is modified to reflect the modification of the coefficient matrix:

$$\underline{h} := \underline{h} - W_1 \Delta S^{-1} W_2^T \underline{h} \tag{2.10}$$

by performing the multiplications and the low-dimensional substitutions (related to the factorized S) from the right to the left, and by using the results of step I.

IV the network back solution. This step completes the computation of (2.4) by performing the back substitution that yields:

$$\underline{x} = U^{-1} \underline{h} \tag{2.11}$$

Those steps that are relevant for a comparison with other modification setups are the steps I and III. Aside from the network size, the computational effort for the first step is roughly determined by the number of substitutions to obtain both (2.5) and (2.6) and by the number of multiplications to obtain (2.7). The first one depends linearly and the second one depends quadratically on the number of nodes m. On average, the effort increases progressively with m. The computational effort for the compensation step also depends progressively on m.

3. RANK-ORIENTED COMPENSATION ALGORITHM

With the node-oriented compensation setup as sketched in the previous section, the number of nodes that are involved in the modification (m) is the relevant factor that determines the computational effort. However, if the modification has a rank r and if r is less than m, then it is worthwhile remodelling the modification in that this rank becomes the determining factor. This conversion can be obtained by factorizing the basic-modification matrix according to:

$$\Delta = Q \tilde{\Delta} P^T \tag{3.1}$$

where

- Δ the basic-modification matrix of order m and of rank r
- P,Q matrices of order mxr
- $\tilde{\Delta}$ a diagonal matrix of order r

Such a factorization can be obtained with, for instance, a singular-value decomposition [10] or with a modified version of the usual LDU triangular factorization. In the appendix two algorithms of the latter type are given.

If we combine this factorization with the original problem (2.1) the new set of equations that is to be solved is given by:

$$(A + \tilde{M}_1 \tilde{\Delta} \tilde{M}_2^T)\underline{x} = \underline{b} \tag{3.2}$$

where

$$\tilde{M}_1 = M_1 Q \tag{3.3}$$

$$\tilde{M}_2^T = P^T M_2^T \tag{3.4}$$

From now on, we can globally use the same procedure as in the previous section. Thus, the symbolic solution is given by:

$$\underline{x} = (A^{-1} - A^{-1} \tilde{M}_1 \tilde{R}^{-1} \tilde{M}_2^T A^{-1})\underline{b} \tag{3.5}$$

where

$$\tilde{R} = (\tilde{M}_2^T A^{-1} \tilde{M}_1 + \Delta^{-1}) \quad (3.6)$$

With mid-compensation (3.5) becomes:

$$\underline{x} = U^{-1}(I - L^{-1}\tilde{M}_1 \tilde{R}^{-1} \tilde{M}_2^T U^{-1}) L^{-1} \underline{b} \quad (3.7)$$

Analogous to those used in the previous section, we give the four steps:

I the preparatory step. This one now includes and starts with the l.o.p.f. (3.1) and the setup of the connection matrices (3.3) and (3.4), and continues with the calculation of:

$$\tilde{W}_1 = L^{-1}\tilde{M}_1 \quad (3.8)$$

$$\tilde{W}_2^T = \tilde{M}_2^T U^{-1} \quad (3.9)$$

by solving the related set of equations with fast forward substitutions; their product:

$$\tilde{W}_3 = \tilde{W}_2^T \tilde{W}_1 \quad (3.10)$$

and, finally:

$$\tilde{R} = (\tilde{W}_3 + \Delta^{-1}) \quad (3.11)$$

This matrix is held in a factorized form.

II the network forward solution. This step is the same as the one in the previous section: a forward substitution yields an intermediate vector:

$$\underline{h} = L^{-1}\underline{b} \quad (3.12)$$

III the compensation step: the intermediate result is adapted according to:

$$\underline{h} := \underline{h} - \tilde{W}_1 \tilde{R}^{-1} \tilde{W}_2^T \underline{h} \quad (3.13)$$

IV the network back solution: this yields the final result:

$$\underline{x} = U^{-1}\underline{h} \quad (3.14)$$

This setup differs from the node-oriented one in respect to two points: first, the number of columns (the dimension of the reduced system) is now determined by the rank of the modification r instead of the number of nodes m; the lower the rank/node ratio the larger the computational advantage in prospect. Second, the basic modification has become of full-rank: additional (low-dimensional) matrix-matrix multiplication (see (2.8)) and matrix-vector multiplication (see (2.10)) are unnecessary. Note that the modification matrix Δ is in diagonal form; its inverse, therefore, is obtained at no cost.

4. THE PERFORMANCE OF THE NEW ALGORITHM

Many tests to establish the extent of the advantage of the rank-oriented setup over the node-oriented setup have been carried out. Each test for both setups involved the solution of a modified network problem where both the coefficient matrix and its modification were symmetric. To simulate rank deficiency, the modification matrix was artifically created for several values of the rank and of the size. In particular we were interested in the influence of these factors and of the network size on the CPU-time.

The networks that are used are synthetic ones made out of the 118-node test system. We experimented with a variety of the set of nodes that constitutes the area of the modification. Because each set was chosen so that the nodes are topologically related to each other, there was hardly any difference in the performances. (With randomly chosen nodes things are quite different.) This result is in accordance with the observations reported in [3].

Further, we performed experiments to find the fastest method to obtain the reduced system (2.7) in the node-oriented setup. There are two options: in the first, m separate sets of equations are solved in order to obtain the W-matrices, each with its own (conceptual) path, followed by sparse-vector inner-product calculations. In the second method we exploited the fact that the factorization paths of topologically connected nodes are strongly related in that they largely coincide. Therefore, the union of these paths is determined and this is used to perform all the fast forward substitutions "in parallel". Consequently, the calculation of (2.7) out of the W-matrices deteriorates into a simple matrix-matrix multiplication.

For all cases that were studied, the second method was faster than the first, in spite of the network size and in spite of the modification size. Therefore, we will take this second method as the preferential method for the node-oriented setup.

The rank setup automatically produces connection matrices where each column has an identical sparsity structure and, therefore, the use of the united path and "parallel" computations on all these columns (analogous to the second method above) speaks for itself. For the interpretetion of the results that are given below, it is important to note that both setups now are identical in respect of the calculation of the reduced systems (2.7) and (3.10) insofar as the precise computations and the sparsity pattern of the connection matrices are concerned, and that they differ only in the number of columns of the connection matrices. Therefore, we may expect that for a modification with a high rank/node ratio no significant profit can be gained, while modifications with extremely low rank/node ratios will largely gain.

For all the test cases we recorded the CPU-time for the four steps that are distinguished in the previous sections. A typical example is given in table 1. The network solution requires the same effort in each setup; differences in computation time are solely attributable to the preparatory step and to the compensation step. From this result, we can calculate two saving ratios:

- the saving as a percentage of the total CPU-time that is required for the solution with the node setup
- the saving as a percentage of the CPU-time that is required for the preparatory and the compensation steps in the node setup.

For the example of table 1 these percentages are 42% and 58% respectively.

Table 1. Elapsed time (ms) for the calculation of a modified network solution. Network size: 708 nodes; number of nodes involved in the modification: 15; rank of the modification: 7 (IBM 3083 mainframe; full optimizing VS-FORTRAN compiler).

	node-oriented CPU-time	%	rank-oriented CPU-time	%
I preparatory	16.40	61	6.44	24
II network forward	3.46	13	3.44	13
III compensation	3.23	12	1.81	7
IV network back	3.78	14	3.77	14
total	26.87	100	15.46	58

We calculated these saving percentages for many networks; the results are given in table 2.

Table 2. Savings (in percentages) for rank-oriented modified network solutions for various network sizes and various values for m (nodes involved) and for r (the rank). Each entry shows xx/yy where xx is the saving with respect to the total node-oriented CPU-time and yy is the saving with respect to the time needed for the steps I and III in the node-oriented setup.

	354-node network			472-node network			708-node network			944-node network		
r	m=15	m=10	m=5	m=15	m=10	m=5	m=15	m=10	m=5	m=15	m=10	m=5
14	10/12			9/12			9/12			8/11		
12	22/28			21/27			19/26			19/27		
9	38/48	8/12		37/48	6/10		34/47	9/15		32/45	7/12	
7	47/59	21/32		45/59	21/34		42/58	18/31		41/58	18/32	
4	57/73	37/56	7/16	55/73	37/59	4/ 9	54/73	34/57	7/17	52/74	32/56	5/14
2	65/82	47/71	19/43	62/82	44/71	19/44	59/81	42/71	17/43	58/82	40/71	17/44
1	68/86	50/77	24/53	64/84	48/76	23/53	63/86	46/78	22/55	61/86	45/79	22/57

They show a general trend that can be summarized as follows:

- for the saving expressed as a percentage of the total network solution CPU-time:
  - for each network size, the greatest savings are obtained for those cases with a large m and a small r; these saving percentages decrease as the network size grows
  - even modifications with a nearly full rank can gain from the rank setup
  - modifications that involve many nodes can more easily gain than modifications that involve only a small number of nodes
- for the saving expressed as a percentage of the preparatory step and the compensation step together:
  - again, the greatest saving occurs for the cases with a large m and a small r, the saving percentages stay mainly constant with growing network size
  - the saving as a function of the rank/node ratio is independent of the network size and increases with growing m.

These global results, stylized, are depicted in the figures 1a and 1b.

In these examples symmetry was assumed throughout; note that in the unsymmetric case the savings will be even greater.

5. AN EXAMPLE: THE SIMULATION OF A BUS SPLIT

In this section we will give an example of a typical power system computation that is perfectly suited to be solved with the rank-oriented compensation algorithm. It concerns the simulation of a bus split in a DC load flow context.

Assume a node, node 2 in the example depicted in figure 2, is split and one wants to know the voltage angles in this case. The split of this node creates a new node 1. If we use the subscripts e and n to refer to the new node and to the original network nodes respectively, the set of equations that are to be solved is given by:

$$\begin{bmatrix} \Delta_{ee} & \Delta_{en}M_2^T \\ M_1\Delta_{ne} & A + M_1\Delta_{nn}M_2^T \end{bmatrix} \begin{bmatrix} \underline{\theta}_e \\ \underline{\theta}_n \end{bmatrix} = \begin{bmatrix} \underline{P}_e \\ \underline{P}_n \end{bmatrix} \qquad (5.1)$$

where

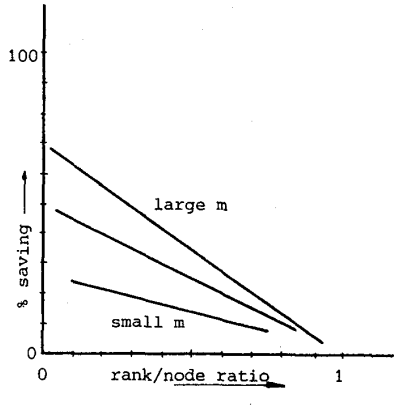


figure 1a. CPU-time saving of the rank-oriented setup as a percentage of the total CPU-time for the node-oriented setup.

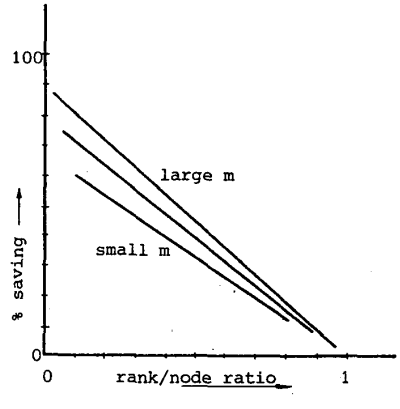


figure 1b. CPU-time saving of the rank-oriented setup as a percentage of the time for the preparatory step and the compensation step in the node-oriented setup.

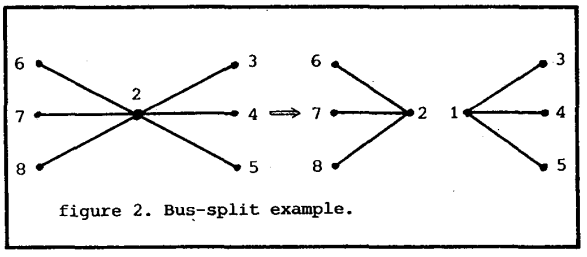


figure 2. Bus-split example.

$\underline{P}_n$ the injected powers at the nodes of the network

P_e the injected power at the new node

$\underline{\theta}_n$ the angles of the network nodes

θ_e the angle of the new node

Δ the 5x5 basic-modification matrix, partitioned into the 1x1 Δ_{ee}, the 1x4 Δ_{en}, the 4x1 Δ_{ne} and the 4x4 Δ_{nn} submatrices

M_1, M_2 the 4-column connection matrices

A the factorized DC load flow matrix

The solution is found by first eliminating the voltage angle of the new node 1 from this set. This causes the set (5.1) to be changed into:

$$(A + M_1 \Delta^{red} M_2^T)\underline{\theta}_n = \underline{P}_n + \underline{P}_e^{red} \qquad (5.2)$$

where the reduced basic-modification matrix is:

$$\Delta^{red} = \Delta_{nn} - \Delta_{ne}\Delta_{ee}^{-1}\Delta_{en} \qquad (5.3)$$

and the distributed power (among the adjacent nodes) is:

$$\underline{P}_e^{red} = -M_1 \Delta_{ne} \Delta_{ee}^{-1} P_e \qquad (5.4)$$

The elimination of the new node causes the problem to be brought into a standard modification form and it can be solved with the scheme as given in section 3. Finally, the angle of the eliminated node 1 can be found with a back substitution using the results of the elimination step.

The important point with regard to the core of this paper is, of course, the rank of the (reduced) basic-modification matrix (5.3). This rank is 1, regardless of the number of nodes involved. This can be seen as follows: the 5x5 (unreduced) modification matrix is:

$$\Delta = \begin{bmatrix} -\Sigma b & 0 & b_3 & b_4 & b_5 \\ 0 & \Sigma b & -b_3 & -b_4 & -b_5 \\ b_3 & -b_3 & 0 & 0 & 0 \\ b_4 & -b_4 & 0 & 0 & 0 \\ b_5 & -b_5 & 0 & 0 & 0 \end{bmatrix} \qquad (5.5)$$

where

b_3, b_4, b_5 the series susceptances of the lines connected to the nodes 3, 4 and 5 respectively (resistances ignored)

Σb the sum of these series susceptances.

After the reduction, the 4x4 matrix becomes:

$$\Delta^{red} = \begin{bmatrix} \Sigma b & -b_3 & -b_4 & -b_5 \\ -b_3 & \frac{b_3 b_3}{\Sigma b} & \frac{b_3 b_4}{\Sigma b} & \frac{b_3 b_5}{\Sigma b} \\ -b_4 & \frac{b_4 b_3}{\Sigma b} & \frac{b_4 b_4}{\Sigma b} & \frac{b_4 b_5}{\Sigma b} \\ -b_5 & \frac{b_5 b_3}{\Sigma b} & \frac{b_5 b_4}{\Sigma b} & \frac{b_5 b_5}{\Sigma b} \end{bmatrix} \qquad (5.6)$$

The low-order product factorization yields:

$$\Delta^{red} = \begin{bmatrix} 1 \\ \frac{-b_3}{\Sigma b} \\ \frac{-b_4}{\Sigma b} \\ \frac{-b_5}{\Sigma b} \end{bmatrix} \begin{bmatrix} \Sigma b \end{bmatrix} \begin{bmatrix} 1 & \frac{-b_3}{\Sigma b} & \frac{-b_4}{\Sigma b} & \frac{-b_5}{\Sigma b} \end{bmatrix} \qquad (5.7)$$

In this case it is even possible to obtain the desired decomposition from branch quantities directly instead of from numerical factorization.

It is interesting to note that the rank of (5.5) is 2: the reduction not only reduces the dimension from 5 to 4, it also reduces the rank from 2 to 1. This manner of handling new nodes, therefore, is the best one because it will facilitate a modification setup with the smallest rank.

We presented this example in a quite simple way, because we only wanted to show the relation between a node split and the very low rank of the modification that is caused by it. In an actual computation, however, the elimination (or the partial factorization followed by a partial forward substitution) and the back substitution must be applied to the 5x5 basic modification. Further, for this specific application it is worthwhile rearranging (5.1) in that the injected powers and the angles of the network nodes are not expressed in levels but in differences. This will permit the replacement of a full forward network substitution by a fast forward substitution.

6. CONCLUSIONS

In this paper a new variant of the compensation algorithm is given that is designed to profit from rank deficiency. The crux of the method is the factorization of a rank-deficient matrix into a low-order product form. This facilitates a modification setup with a reduced system of the smallest possible dimension.

Provided that the nodes that are involved in the modification are topologically related, this rank-oriented setup is nearly always much faster than the strict node-oriented one. Test results show that for network sizes ranging from 354 to 944 nodes, the computation time for a complete, modified, network solution reduces to a range from 20 to 35%. This reduction is largely due to a more efficient calculation of the reduced system.

Rank deficiency is usually caused by structural network modifications and, therefore, the most appealing application field can be found in procedures for fault analysis and contingency analysis.

REFERENCES

[1] Tinney, W.F.: "Compensation methods for Network Solutions by Optimally Ordered Triangular Factorization", IEEE Transactions on Power Apparatus and Systems, vol. PAS-91, 1972, pp. 123-127.

[2] Alsaç, O., B. Stott, W.F. Tinney: "Sparsity-oriented Compensation Methods for Modified Network Solutions", IEEE Transactions on Power Apparatus and Systems, vol. PAS-102, 1983, pp. 1050-1060.

[3] Tinney, W.F., V. Brandwajn, S.M. Chan: "Sparse Vector Methods", IEEE Transactions on Power Apparatus and Systems, vol. PAS-104, 1985, pp. 295-301.

[4] Alvarado, F.L., S.K. Mong, M.K. Enns: "A Fault Program with Macro's, Monitors and Direct Compensation in Mutual Groups", IEEE Transactions on Power Apparatus and Systems, vol. PAS-104, 1985, pp. 1109-1120.

[5] Brandwajn, V., W.F. Tinney: "Generalized Method of Fault Analysis", IEEE Transactions on Power Apparatus and Systems, vol. PAS-104, 1985, pp. 1301-1306.

[6] Brandwajn, V., M.G. Lauby: "Complete Bounding Method for AC Contingency Screening", Paper 88 SM 726-2, presented at the IEEE-PES 1988 Summer Meeting, Portland, Oregon, July 1988.

[7] Chan, S.M., V. Brandwajn: "Partial Matrix Refactorization", IEEE Transactions on Power Systems, vol. PWRS-1, 1986, pp. 193-200.

[8] Chang, S.-K., V. Brandwajn: "Adjusted Solutions in Fast Decoupled Load Flow", IEEE Transactions on Power Systems, vol. PWRS-3, 1988, pp. 726-733.

[9] Aitchison, P.W., J.L.R. Pereira, A. Brameller: "Extensions and Singularities in Compensated Network Solutions Applicable to Security Monitoring", Proc. IEE, vol. 134, prt. C, 1987, pp. 123-129.

[10] Wilkinson, J.H.: "Singular-value Decomposition-Basic Aspects", in: Jacobs, D.: Numerical Software-Needs and Availability, London, Academic Press, 1978.

APPENDIX

In this appendix two algorithms are presented for the determination of a low-order product form. Basically, they follow the well-known LDU factorization: at each step a new column of the lower triangular matrix and a new row of the upper triangular matrix is calculated. At the same time, the relevant right-lower part of the original matrix is modified accordingly (outer-product factorization). With the l.o.p.f. the procedure terminates if the remaining right-lower part of the matrix consists of zero elements (r<m) or if this matrix is empty (r=m). A permutation is carried out in order that the largest absolute diagonal element becomes the pivot element; this step could be restricted to the case where the current pivot element is too small.

The symmetric version (algorithm 2) differs from the unsymmetric one (algorithm 1) in that only the lower triangular part of the matrix needs to be set. After termination of the algorithm, r holds the rank of the matrix. The first r columns of Q (and P) and the first r diagonal elements of D contain the required decomposition.

Algorithm 1. Unsymmetric factorization $A=QDP^T$

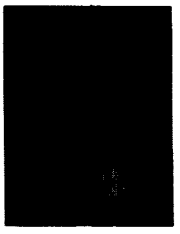
```
initialize:
iperm(i):= i        i=1,m
Q:= 0
P:= 0
D:= A
eps:= termination tolerance (f.i. 1.0d-05)

do for l=1,m
     search for the largest absolute diagonal entry
     d(i,i) for i=l,m; let d(k,k) be this entry;
     if (|d(k,k)| < eps) then
        r=l-1
        terminate
     end if
     if (k.ne.l) then
        interchange:
        iperm(k) <-> iperm(l)
        d(i,l)   <-> d(i,k)      i=l,m
        d(l,j)   <-> d(k,j)      j=l,m
     end if
     set columns l of Q and P:
     q(iperm(i),l):=d(i,l)/d(l,l)     i=l,m
     p(iperm(j),l):=d(l,j)/d(l,l)     j=l,m
     eliminate node l:
     d(i,j):=d(i,j)-d(i,l)*d(l,j)/d(l,l)  i=l+1,m;
                                          j=l+1,m
end do
r=m
terminate
```

Algorithm 2. Symmetric factorization $A=QDQ^T$

```
initialize:
iperm(i):= i        i=1,m
Q:= 0
D:= A (only lower triangular part need to be set)
eps:= termination tolerance (f.i. 1.0d-05)

do for l=1,m
     search for the largest absolute diagonal entry
     d(i,i) for i=l,m; let d(k,k) be this entry;
     if (|d(k,k)| < eps) then
        r=l-1
        terminate
     end if
     if (k.ne.l) then
        interchange:
        iperm(k) <-> iperm(l)
        d(i,l)   <-> d(k,i)      i=l+1,k-1
        d(l,l)   <-> d(k,k)
        d(i,l)   <-> d(i,k)      i=k+1,m
     end if
     set column l of Q:
     q(iperm(i),l):=d(i,l)/d(l,l)    i=l,m
     eliminate node l:
     d(i,j):=d(i,j)-d(i,l)*d(j,l)/d(l,l)   i=l+1,m;
                                           j=l+1,i
end do
r=m
terminate
```



Robert A.M. van Amerongen was born in Heemstede (the Netherlands) on May 3, 1950. He studied electrical engineering at the Delft University of Technology, and economics at the Erasmus University Rotterdam and the University of Amsterdam. He received his Msc. Electrical Engineering in 1978 and his Master's in Economics in 1987.

In 1978 he entered the power system laboratory of the TH Delft as a research assistent. Nowadays he is responsible for education and research. His main areas of interest are electric power-system analysis, including network calculation and its applications, optimization and estimation.