PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 41, NO. 1, PP. 41-52 (1997)

IMPLEMENTATION OF A FAST MATRIX INVERSION METHOD IN THE ELECTRODYNAMIC SIMULATION PROGRAM

László BÜRGER

Department of Electric Power Systems
Technical University of Budapest
H-1521 Budapest, Hungary
Phone: + 36 1 463 2763
E-mail: burger@vmt.bme.hu

Received: January 10, 1996

Abstract

This paper describes the implementation of a fast matrix inversion algorithm that can be used efficiently in the electric network analysis.

The ElectroDynamic Simulation (EDS) is an IBM PC based computer program for studying short and middle term dynamics of the electric power system. The electric network is modelled by quasi-stationary representation. EDS has an interactive menu system. The user can change the operating conditions – like sending telecommands from a control center – or initiate simulation of disturbances. Simulation of some events requires changing the network matrices. The admittance matrix $\mathbf{Y}$ can be modified directly while the impedance matrix $\mathbf{Z}$ is updated by inverting the new $\mathbf{Y}$. The conventional inversion is very time consuming and freezes the simulation for unacceptably long intervals.

A new, faster method has been developed which takes into account that the change in the matrices due to the events is small and well-defined. In these cases the complete inversion is not needed, the impedance matrix can be updated directly, by the aid of the Sherman–Morrison theorem. The new method has been implemented in recent EDS versions and performs significantly faster than the conventional inversion.

Keywords: electrodynamic simulation, quasi-stationary modelling, matrix inversion.

1. Introduction

The ElectroDynamic Simulation program has been developed at the Department of Electric Power Systems. It is suitable for studying the steady state and dynamic processes of the interconnected electric power system. It was written in Turbo Pascal and runs on IBM PC/AT compatible computers.

EDS consists of two main modules. The first one is for the input, modification and documentation of model data, calculation and evaluation of the steady state: voltage conditions and power flows. In the second module, the dynamic simulation, transient processes can be superimposed over a previously calculated steady state.

In the dynamic simulation the electromechanical transients and regulation processes – local as well as system–wide ones – are represented with their differential equations. In details, these are swings of turbine–generator units, operation of turbine regulators, operation of automatic voltage regulator (AVR) of generators and transients inside high voltage direct current (HVDC) converters. The fourth–order Runge–Kutta method is used to solve these equations. The rest of the model, the electric network is represented by complex phasors of voltages and currents describing a quasi-stationary state in each time step (HORVÁTH and SZAKÁCS, 1991).

Time functions and trajectories of the requested quantities can be displayed graphically or in tables.

EDS is continuously used in the education at the Department of Electric Power System as well as in the power system analysis at the National Power Line Company and the National Dispatching Center.

2. Simulation of Events

In the simulation session transient processes start due to operator actions or events as load changes, system disturbances. All types of events can be initiated by interactive way via a menu system that is accessible all along the session. Load changes and disturbances can be scheduled as well. In these cases EDS is similar to a training simulator working with pre-built scenarios.

EDS calculates the node voltages and the currents in each time step according to the

$$\mathbf{u} = \mathbf{Z}\mathbf{i}, \tag{1}$$

$$I_{b,ij} = Y_{ij}(U_i - U_j) \tag{2}$$

equations. In *Eq.* (1) $\mathbf{u}$ is the complex vector of node voltages, $\mathbf{Z}$ is the nodal impedance matrix and $\mathbf{i}$ is the complex vector of the currents injected at the nodes. In *Eq.* (2) $I_{b,ij}$ stands for the current of the branch connecting the i-th node with the j-th one, Y_{ij} is the complex admittance of the branch, U_i and U_j are the phasors of the terminal voltages. Injected currents come mainly from generation and consumption, however, some events are also represented with injected currents, e.g. stepping of transformer tap changers. $\mathbf{Z}$ is the inverse of $\mathbf{Y}$ that contains series and shunt admittances of lines and transformers as well as transient reactance of generators, admittance-type loads and control devices at nodes.

Many of the possible events are represented with changes in $\mathbf{Z}$. These are the busbar short circuits, line short circuits and cut-offs, switching shunt devices on or off and automatic operation of static var compensators (SVC).

Since $\mathbf{Y}$ contains immediate topology information of the network it can be updated directly according to the type and the parameters of the event. $\mathbf{Z}$ is obtained by calculating the inverse of $\mathbf{Y}$. Most common EDS models of the Hungarian high voltage grid have about 70 nodes. Conventional complete inversion of a complex matrix of such a high order takes tens of seconds even on the modern 486-based PC's. A new, more efficient method has been implemented to speed up the simulation of the events.

3. Matrix Representation

The nodal admittance matrix $\mathbf{Y}$ is derived from the branch admittance matrix and the incidence matrix of the network. The change in $\mathbf{Y}$ caused by the change of the network state affects only a small and well-defined part of the matrix. For simplicity the following description will be restricted to symmetric three-phase conditions, although there are versions of EDS that are capable of handling unsymmetrical conditions. The description of symmetric three-phase conditions requires the positive sequence model circuit only.

Events related to only one node need the simplest modification to $\mathbf{Y}$. These are busbar three-phase short circuit and switching shunt devices installed at a node. These events only require modification of the diagonal element belonging to the given node. Series and shunt faults of lines need four elements of $\mathbf{Y}$ to be modified: diagonal elements of the terminal nodes and the elements representing the series connection.

The above mentioned changes can be expressed in dyadic form therefore $\mathbf{Z}$ can be updated in a straightforward way.

4. Method of Dyadic Corrections

The dyad is a special kind of matrix. It is the dyadic product of its two generating vectors:

$$\mathbf{u} * \mathbf{v}^T = \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{bmatrix} * [v_1\, v_2\, ..v_n] = \begin{bmatrix} u_1v_1 & u_1v_2 & \dots & u_1v_n \\ u_2v_1 & u_2v_2 & \dots & u_2v_n \\ \dots & \dots & \dots & \dots \\ u_nv_1 & u_nv_2 & \dots & u_nv_n \end{bmatrix}. \quad (3)$$

The rank of a dyad is always one unless the dyad is identically zero.

4.1 Theoretical Background

Linear algebra states that if a non-singular matrix is changed with a single dyad the inverse of the changed matrix will differ from that of the original one with a single dyad as well (Sherman–Morrison theorem). In formula:

$$\{\mathbf{A} + \mathbf{u}\mathbf{v}^T\}^{-1} = \mathbf{A}^{-1} - \{\mathbf{A}^{-1}(\mathbf{u}\mathbf{v}^T)\mathbf{A}^{-1}\}/(1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}). \qquad (4)$$

The statement can easily be proven by calculating the product of the changed matrix and its inverse as formulated above (RÓZSA, 1991). The theorem is well-known and widely applied in electric power system analysis (SZENDY, 1967), (GESZTI, BENKÓ and REGULY, 1974).

4.2 Estimation of the Expectable Speed Enhancement

The practical application of *Eq.* (4) (a single dyadic correction) requires significantly shorter CPU time than the conventional complete inversion. The speed of the method can be estimated simply using the following reasonable approximations:

- The average execution time of the floating point multiplication by hardware is close to that of the floating point division so they can be supposed to be equal. Further on these instructions will be referred to as FPMD (floating point multiplication or division).
- The average execution time of the floating point addition, subtraction and load-store operations by hardware can be ignored in comparison with either the multiplication or the division.

A complex multiplication requires 4 real multiplications (4 FPMDs) while a complex division requires 6 real multiplications and 2 real divisions (8 FPMDs). If N is the order of the matrices the calculation demand of a single dyadic correction consists of 1 dyadic multiplication of the generating vectors (N^2 complex multiplications), 3 matrix-with-vector multiplications ($3N^2$ complex multiplications), 1 scalar multiplication of the generating vectors (N complex multiplications) and 1 matrix-with-scalar division (N^2 complex divisions). The total is $24N^2 + 4N$ FPMDs.

The original inversion procedure of EDS takes about $6N^3 + 10N^2$ FPMDs. It relies on the sparsity of the admittance matrix thus its speed depends on the actual content of the matrix to be inverted. In order to prove the general usefulness of the new method a more common inversion procedure, the Gauss–Jordan elimination (GESZTI, 1986) will be used in comparisons. It takes about $8N^3 + 8N^2$ FPMDs.

These estimations show that expectable speed advantage of the new method is very considerable at N values about 70.

5. Implementation in the Dynamic Simulation

Modifications of $\mathbf{Y}$ due to the events have to be converted to dyadic form, that is, the generating vectors have to be specified for each case.

5.1 Events Associated with a Single Node

The modification matrix $\Delta\mathbf{Y}$ has only one non-zero element: the diagonal element belonging to the concerned node. Thus $\Delta\ \mathbf{Y}$ is very easy to produce in dyadic form, if k is the index of the node the generating vectors will have only zero elements except for the k-th position. The non-zero u_k and v_k elements can be chosen freely but their product must result Δy_{kk}. The trivial choice is

$$u_k = \Delta y_{kk},\ v_k = 1. \tag{5a-b}$$

In this case further CPU time can be saved taking into account the sparsity of the generating vectors. The result of the $\mathbf{A}^{-1}(\mathbf{u} * \mathbf{v}^T)\mathbf{A}^{-1}$ and $\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}$ multiplications can easily be foreseen and there is no need in fact to carry out the matrix–matrix or matrix–vector multiplications. E.g. using the trivial selection for the generating vectors $\mathbf{u}$ is Δy_{kk} times the k-th unity column vector and $\mathbf{v}$ is the k-th unity row vector. Multiplying a matrix with the k-th unity column or row vector gives the k-th column or row of the matrix, respectively. Thus

$$\mathbf{Y}^{-1}(\mathbf{u} * \mathbf{v}^T)\mathbf{Y}^{-1} = (\mathbf{Y}^{-1}\mathbf{u}) * (\mathbf{v}^T\mathbf{Y}^{-1}) = \Delta y_{kk}\mathbf{z}_k * \mathbf{z}_k^T \tag{6}$$

and

$$\mathbf{v}^T\mathbf{Y}^{-1}\mathbf{u} = \mathbf{v}^T(\mathbf{Y}^{-1}\mathbf{u}) = \mathbf{v}^T\Delta y_{kk}\mathbf{z}_k = \Delta y_{kk}\mathbf{z}_{kk}, \tag{7}$$

where $\mathbf{z}_k$ and $\mathbf{z}_k^T$ are the k-th column and row of $\mathbf{Y}^{-1}$, respectively, ($\mathbf{Y}^{-1}$ means $\mathbf{Z}$ here) and z_{kk} is the k-th diagonal element of $\mathbf{Y}^{-1}$. Finally

$$\begin{aligned}\{\mathbf{Y} + \Delta\mathbf{Y}\}^{-1} &= \mathbf{Y}^{-1} - \{\mathbf{Y}^{-1}(\mathbf{u} * \mathbf{v}^T)\mathbf{Y}^{-1}\}/(1 + \mathbf{v}^T\mathbf{Y}^{-1}\mathbf{u}) = \\ &= \mathbf{Y}^{-1} - (\Delta y_{kk}\mathbf{z}_k * \mathbf{z}_k^T)/(1 + \Delta y_{kk}z_{kk}).\end{aligned} \tag{8}$$

Obviously, the generating vectors are not needed at all. The algorithm calculates the new $\mathbf{Y}^{-1}$ when modifying y_{kk} with Δy_{kk} in the following way:

- the k-th row (or column) of $\mathbf{Y}^{-1}$ is saved in a $\mathbf{t}$ temporary vector,
- the $c = \Delta y_{kk}/(1 + \Delta y_{kk}z_{kk})$ constant factor is calculated,
- from each element of $\mathbf{Y}^{-1}$ (in the i-th row and the j-th column generally) $t_i * t_j * c$ is subtracted.

This special procedure is even faster than the normal dyadic correction. The number of FPMDs required is only $8\mathrm{N}^2 + 8$ (for $2\mathrm{N}^2$ complex multiplications and one complex division). Additional temporary storage is needed only for the $\mathbf{t}$ vector.

5.2 Events Associated with Two Nodes

This case covers faults, one-end or complete disconnection and re-connection of lines and transformers. The modification matrix $\Delta\mathbf{Y}$ has four non-zero elements: the diagonal elements Δy_{ii} and Δy_{jj} belonging to the concerned nodes, and off-diagonal elements representing the series connection between the nodes, Δy_{ij} and Δy_{ji}. (Since there are no phase shifters in the Hungarian network the matrices are symmetrical, $\Delta y_{ij} = \Delta y_{ji}$ in all cases. However, the method is also adaptable for unsymmetrical matrices.)

In this case $\Delta\mathbf{Y}$ cannot be produced as one dyad. Non-zero rows and columns of $\Delta\mathbf{Y}$ are linear independent in most cases while those of a dyad are definitely not. That is why $\Delta\mathbf{Y}$ must be provided in two steps.

5.2.1 The First Step

In the first step a dyad is presented which makes the proper modifications to the off-diagonal elements and one of the diagonal elements of $\mathbf{Y}$, Δy_{ii} for instance. Non-zero elements u_i, u_j, v_i and v_j of the generating vectors $\mathbf{u}$ and $\mathbf{v}$ must be chosen so that

$$u_i * v_i = \Delta y_{ii}, \tag{9a}$$

$$u_i * v_j = u_j * v_i = \Delta y_{ij} = \Delta y_{ji}. \tag{9b}$$

The trivial choice is

$$u_i = \Delta y_{ii}, \quad u_j = \Delta y_{ij}, \tag{10a-b}$$

$$v_i = 1, \quad v_j = \Delta y_{ij}/\Delta y_{ii}. \tag{11a-b}$$

The single dyadic correction with these generating vectors will obviously make an unintended $\Delta y_{ij}^2/\Delta y_{ii}$ modification to y_{jj} which must be corrected in the second step.

Similarly to the method in subsection 5.1 the resultant change of $\mathbf{Y}^{-1}$ can be calculated directly since $\mathbf{u}$ and $\mathbf{v}$ are very sparse. They have non-zero elements only at their i-th and j-th position so they are linear combinations of the i-th and the j-th unity vectors. If $\mathbf{e}_i$ stands for the i-th and $\mathbf{e}_j$ for the j-th unity column vector

$$\mathbf{u} = u_i * \mathbf{e}_i + u_j * \mathbf{e}_j, \tag{12a}$$

$$\mathbf{v} = v_i * \mathbf{e}_i + v_j * \mathbf{e}_j. \tag{12b}$$

If $\mathbf{z}_k$ stands for the k-th column of $\mathbf{Y}^{-1}$ as previously,

$$\begin{aligned} &\mathbf{Y}^{-1}(\mathbf{u} * \mathbf{v}^T)\mathbf{Y}^{-1} = (\mathbf{Y}^{-1}\mathbf{u}) * (\mathbf{v}^T\mathbf{Y}^{-1}) = \\ &= \mathbf{Y}^{-1}(u_i * \mathbf{e}_i + u_j * \mathbf{e}_j) * (v_i * \mathbf{e}_i + v_j * \mathbf{e}_j)\mathbf{Y}^{-1} = \\ &= (u_i * \mathbf{z}_i + u_j * \mathbf{z}_j) * (v_i * \mathbf{z}_i^T + v_j * \mathbf{z}_j^T). \end{aligned} \tag{13}$$

The $1 + \mathbf{v}^T\mathbf{Y}^{-1}\mathbf{u}$ constant denominator can also be calculated directly:

$$1 + \mathbf{v}^T\mathbf{Y}^{-1}\mathbf{u} = 1 + (v_i * \mathbf{z}_i^T + v_j * \mathbf{z}_j^T) * (u_i * \mathbf{e}_i + u_j * \mathbf{e}_j) =$$
$$= 1 + v_i\mathbf{z}_i^T * u_i\mathbf{e}_i + v_j\mathbf{z}_j^T * u_i\mathbf{e}_i + v_i\mathbf{z}_i^T * u_j\mathbf{e}_j + v_j\mathbf{z}_j^T * u_j\mathbf{e}_j. \quad (14)$$

After substituting for u_i, u_j, v_i and v_j from *Eqs.* (10) and (11):

$$1 + \mathbf{v}^T\mathbf{Y}^{-1}\mathbf{u} = 1 + \Delta y_{ii}z_{ii} + 2 * \Delta y_{ij}z_{ij} + (\Delta y_{ij}^2/\Delta y_{ii})z_{jj}. \quad (15)$$

The calculation of the $\mathbf{Y}^{-1}$ after the first step goes the following way:

- the $\mathbf{t}_1 = (u_i * \mathbf{z}_i + u_j * \mathbf{z}_j) = (\Delta y_{ii} * \mathbf{z}_i + \Delta y_{ij} * \mathbf{z}_j)$ and
- the $\mathbf{t}_2 = (v_i * \mathbf{z}_i + v_j * \mathbf{z}_j) = [\mathbf{z}_i + (\Delta y_{ij}/\Delta y_{ii}) * \mathbf{z}_j)$ temporary vectors are generated,
- the $c = 1/[1 + \Delta y_{ii}z_{ii} + 2 * \Delta y_{ij}z_{ij} + (\Delta y_{ij}^2/\Delta y_{ii})z_{jj}]$ constant factor is calculated,
- from each element of $\mathbf{Y}^{-1}$ (in the i-th row and the j-th column generally) $t_{1i} * t_{2j} * c$ is subtracted.

The temporary storage demand can be reduced choosing the generating vectors to be identical ($\mathbf{u} = \mathbf{v}$). In this case only one vector must be stored and the non-zero elements of the generating vector are .

$$u_i = \sqrt{\Delta y_{ii}}, \qquad u_j = \Delta y_{ij}/\sqrt{\Delta y_{ii}}. \quad (16a\text{-}b)$$

The difference in the procedure is that

- only the $\mathbf{v} = (u_i * \mathbf{z}_i + u_j * \mathbf{z}_j) = (\sqrt{\Delta y_{ii}} * \mathbf{z}_i + \Delta y_{ij}/\sqrt{\Delta y_{ii}} * \mathbf{z}_j)$ temporary vector must be generated instead of $\mathbf{t}_1$ and $\mathbf{t}_2$,
- from each element of $\mathbf{Y}^{-1}v_i * v_j * c$ must be subtracted instead of $t_{1i} * t_{2j} * c$ (where i is the row and j is the column index).

5.2.2 The Second Step

The second part of the procedure must undo the unintended $\Delta y_{ij}^2/\Delta y_{ii}$ modification made to y_{jj} in the previous step and make the intended Δy_{jj} modification at the same time. This means changing a single diagonal element of $\mathbf{Y}$ (and update $\mathbf{Y}^{-1}$ accordingly) which has already been described in subsection 5.1. The method introduced there must be followed.

5.2.3 The Calculation Demand

Regardless of the actual dyadic representation in the first step the calculation demand of the method consists of

- 2N complex multiplications to fill the vector $\mathbf{v}$ at the beginning of the first step,
- $2N^2$ complex multiplications to modify the elements of $\mathbf{Y}^{-1}$ in the first step,
- $2N^2$ complex multiplications in the second step

what give a total of $16N^2 + 8N+8$ FPMDs.

5.3 Changing the Number of Nodes

The number of nodes in the model network may change for several reasons. E.g.: load point is lost due to a fault and the related protective operation, new load point is connected, line fault simulation requires a new virtual node, previously independent nodes become equipotential due to switchgear operation and vice versa.

5.3.1 Decreasing the Number of Nodes

Decreasing the number of nodes is not a problem. The node whose last feeder is disconnected by making the proper modifications to the network matrices gets isolated automatically. If the node has no generation its voltage will become identically zero and the node can be excluded from the further simulation. The exclusion can be physical or logical.

Physical exclusion means deletion of the row and the column of the node and compaction of the matrices if necessary. If the node to be excluded was properly isolated previously – using the method described in subsection 5.2 – the rows and columns to be deleted will contain only zeros with the exception of the diagonal elements which means that the matrices will remain inverses of each other even after the deletion.

(This applies only with the assumption that the node has non-zero shunt admittance.)

The exclusion is logical if the row and the column of the node are actually not deleted. Logical exclusion requires an additional descriptor table to keep track of the state of each node. If the related flag in the table shows that the node is excluded no calculations should be done with the data of the node. However, the data still exist, the row and column belonging to the concerned node are not deleted from the matrices. This way the node can easily be included later again. In spite of the additional administration, if the number of nodes changes often during a simulation session the method may be even faster than the physical exclusion because the compaction is spared.

5.3.2 Increasing the Number of Nodes

Increasing the number of nodes can easily be done taking the steps of the physical exclusion in reverse order. First – if the node did not exist previously or the program uses physical deletion – a new row and a column must be joined to each matrix to increase their order. The rows and columns must be all zeros except the diagonal elements which must be the shunt admittance and its reciprocal, the shunt impedance of the new node, respectively. (If the node has no shunt device at all an arbitrary non-zero value can be used so that matrices are non-singular but it must be properly removed later when all the series connections have been included.) After this the matrices are still the inverses of each other. At this point the lines or transformers can be connected to the node using the method described in subsection 5.2. If c stands for the number of series connections of the node the successive application of the procedure method is faster than the inversion with Gauss–Jordan elimination if

$$8N^3 + 8N^2 > (16N^2 + 8N + 8) * c. \tag{17}$$

At $N = 70$ the formula gives $c < 35$. Since no node in the high voltage national grid models has as many as 35 connections (typical range is $2 \leq c \leq 6$) the method of successive dyadic corrections is much faster than the full inversion for network models of the usual size. See *Table 1* in Section 7 for benchmark results.

5.3.3 Switchgear Operation

Forthcoming versions of EDS are intended to support substation models. Switching devices are not to be modelled as parts of the electric network because they have very small – practically zero – impedance. Including them into the electric network model would make the matrices very ill-conditioned and might cause the calculation to become unstable. Switches will constitute a separate logical model for each modelled substation, their actual status will simply describe the potential conditions of nodes in the electric network model. When previously independent nodes are shorted by impedanceless switchgear they become equipotential and one of them must be excluded from the calculation. Its series connections should not only be removed but passed to the remaining node. Nodes with generation require additional administration so that injected current is also redirected to the remaining node.

6. Other Application Possibility

The method can be used efficiently in any other applications that similarly require the calculation of the inverse of a slightly modified matrix. Typical example is network reduction which is important means of speeding up all types of calculations related to network matrices. The reduced network not having the nodes that are unimportant from the point of view of the current study must produce the same impedance conditions for the remaining nodes. This is achieved conventionally in the following way:

- the $\mathbf{Y}$ admittance matrix of the complete network is created,
- the $\mathbf{Z}$ impedance matrix is calculated by inverting $\mathbf{Y}$,
- $\mathbf{Z}$ is truncated so that it does not contain rows and columns for unimportant nodes,
- after compactization the lower order $\mathbf{Z}'$ is inverted back into $\mathbf{Y}'$ that is to be decomposed to network components with new, equivalent parameters.

If relatively few nodes are to be eliminated at the same time the calculation of the inverse of the truncated impedance matrix can be speeded up because a single order reduction can be carried out doing two dyadic corrections. The impedance matrix is absolutely dense thus simplified methods described in Section 5 are useless, full dyadic corrections are needed.

When eliminating a single node the row and the column of the node must be filled with zeros with the exception of the diagonal element. If the index of the given node is k, correction with the dyad generated by the

$$\mathbf{u}_1 = [-1 \quad -1 \quad \ldots \quad -1]^T, \tag{18a}$$

$$\mathbf{v}_1^T = [z_{k1} \quad z_{k2} \quad \ldots \quad z_{k,k-1} \quad .. \quad z_{kN}] \tag{18b}$$

vectors will make the proper modifications to the k-th row and column (setting the diagonal element to unity) but will make an unintended modification to all other elements of $\mathbf{Z}$, too. The latter can be cancelled with a second dyadic correction by the

$$\mathbf{u}_2 = [1 \quad 1 \quad \ldots \quad 1]^T, \tag{19a}$$

$$\mathbf{v}_2^T = [z_{k1} \quad z_{k2} \quad \ldots \quad z_{k,k-1} \quad 0 \quad z_{k,k+1} \quad z_{kN}] \tag{19b}$$

vectors. Since the k-th element of both $\mathbf{v}$ and $\mathbf{u}$ is zero this step does not affect the k-th row and column of $\mathbf{Z}$. Similarly to that described in

subsection 5.2.1 the temporary storage demand can be reduced choosing the generating vectors of both dyads to be equal:

$$\mathbf{u}_1 = \mathbf{v}_1 = \left[j\frac{z_{k1}}{\sqrt{z_{kk}}}\, j\frac{z_{k2}}{\sqrt{z_{kk}}} \ldots j\frac{z_{k,k-1}}{\sqrt{z_{kk}}}\, j\sqrt{z_{kk}}\, j\frac{z_{k,k+1}}{\sqrt{z_{kk}}} \ldots j\frac{z_{kN}}{\sqrt{z_{kk}}} \right]^T,$$

$$\mathbf{u}_2 = \mathbf{v}_2 = \left[\frac{z_{k1}}{\sqrt{z_{kk}}}\, \frac{z_{k2}}{\sqrt{z_{kk}}} \ldots \frac{z_{k,k-1}}{\sqrt{z_{kk}}}\, 0,\, \frac{z_{k,k+1}}{\sqrt{z_{kk}}} \ldots \frac{z_{kN}}{\sqrt{z_{kk}}} \right]^T. \qquad \text{(20a-b)}$$

The calculation demand of two successive dyadic corrections is $48N^2+8N$ FPMDs. If the matrix is properly reordered so that the concerned node is first or the last one ($k = 1$ or $k = N$) the second dyadic correction can be restricted to the remaining matrix of $(N-1)$-th order thus reducing the calculation demand to $48N^2-40N+20$.

7. Benchmarks

The testing of the newly implemented methods in the EDS program has proven their accuracy and flexibility. The speed enhancement is very considerable but the pure execution time cannot be measured because EDS does screen refresh inside the calculation cycles. A separate benchmark program was written in order to compare the speed of the methods with that of the conventional inversion. Tests were run on a 40 MHz a80386DX / i80387DX PC, in DOS protected mode. *Table 1* shows the elapsed time in seconds. (The number of FPMDs can be found in brackets.)

Table 1
Benchmark results

Matrix order N	Gauss–Jordan elimination ($8N^3 3+8N^2$)	Full dyadic correction ($24N^2+4N$)	Diagonal element correction ($8N^2+8$)	π-model correction ($16N^2+8N+8$)
25	3.57	0.33	0.11	0.22
50	28.39	1.27	0.44	0.88
2 75	95.90	2.85	0.99	2.09
100	227.06	5.00	1.71	3.63
125	443.36	7.91	2.75	5.71
150	773.79	11.37	3.96	8.23
175	1216.66	15.44	5.38	11.15
200	1817.32	19.88	7.04	14.55

The results show that the preliminary estimation about the expectable speed enhancement is acceptable. The execution time for one FPMD shows

only very small differences among the newly developed methods but the difference is about 13.5% between the conventional method and any of the new methods. This may occur for the following reasons:

- The approximation refers to average execution times. The actual execution time largely depends on the way of addressing the operand which is compiler-specific and thus cannot be taken into account.
- The conventional inversion procedure contains a lot more additional instructions (load-store, addition, subtraction) relatively than the dyadic correction methods.
- Architectural features that began to appear with the 32-bit CPU's – like pipelining and caching – make the actual instruction execution time more occasional.

8. Summary

This paper described the theoretical background and the practical implementation of the method of dyadic corrections. The method facilitates the fast updating of the inverse of a slightly modified matrix. Problem specific versions optimized for use in the Electrodynamic Simulation program were also presented. A simple approximation was introduced to estimate the achievable speed enhancement.

The newly developed method was found to perform significantly faster than the conventional inversion when tested within the target application, EDS. Results of a separate benchmark program were presented to document the pure execution times for the different methods.

References

HORVÁTH, I. – SZAKÁCS, T. (1991): Interaktív elektrodinamikus szimuláció, Használati leírás. (Interactive Electrodynamic Simulation, User Manual), BME Villamosművek Tanszék, (in Hungarian).

GESZTI, P. O. (1986): Villamosenergia-rendszerek (Electric Power Systems). Budapest, Tankönyvkiadó (in Hungarian).

GESZTI, P. O. – BENKÓ, I. – REGULY, Z. (1974): Villamos hálózatok I. (Electric Power Networks, Vol. I). Budapest, Tankönyvkiadó, (in Hungarian).

RÓZSA, P. (1991): Lineáris algebra és alkalmazásai (Linear Algebra and Its Applications), Budapest, Tankönyvkiadó, (in Hungarian).

SZENDY, K. (1967): Korszerű hálózatszámítási módszerek (Modern Methods for Network Analysis), Budapest, Akadémiai Kiadó, (in Hungarian).