Int. J. Appl. Math. Comput. Sci., 2003, Vol. 13, No. 2, 199–204

INVERSION OF SQUARE MATRICES IN PROCESSORS WITH LIMITED CALCULATION ABILLITIES

KRZYSZTOF B. JANISZOWSKI*

* Institute of Automatic Control and Robotics, Warsaw University of Technology
ul. Narbutta 87, 02–525 Warsaw, Poland
e-mail: kjanisz@mchtr.pw.edu.pl

An iterative inversion algorithm for a class of square matrices is derived and tested. The inverted matrix can be defined over both real and complex fields. This algorithm is based only on the operations of addition and multiplication. The numerics of the algorithm can cope with a short number representation and therefore can be very useful in the case of processors with limited possibilities, like different neuro-computers and accelerator cards. The quality of inversion can be traced and tested. The algorithm can be used in the case of singular matrices, and then it automatically produces a result that contains the inverse of this part of the processed matrix which can be inverted. An example of the inversion of a six-order square matrix is presented and discussed.

Keywords: exponential matrix series, matrix inversion, neuro-processors

1. Introduction

Hardware neuro-processor and accelerator cards have usually a limited scope of mathematical operations, e.g. the addition and multiplication of float numbers with limited precision. This limitation reduces a variety of optimisation algorithms which can be performed on these processors. Many numerical procedures and algorithms need an inversion of square, real (or complex) matrices, cf. all gradient optimisation algorithms. The high capability of multiplication in the parallel computing of neuro-processors (Golub *et al.*, 1992) is now challenged by a very high frequency of standard PC-processors. Consequently, the rarely used and produced neuro-cards are too expensive for applications in different problems. The presented algorithm can perhaps change this situation and enlarge the application area of neuro-computers.

The inversion of a square matrix is a very frequent operation in various algorithms, which always engages programmers' attention due to problems which can arise during the calculations, e.g. bad conditioning, stiffness or large dimensions. There are various refined algorithms, e.g. Singular Value Decomposition (SVD) (Bodewig, 1965; Collar and Simpson, 1987; Kiełbasiński and Szczepik, 1992), Birmann's (BF) or Cholesky's (CF) factorisations (Bodewig, 1965; Collar and Simpson, 1987; Higham, 1996; Kiełbasiński and Szczepik, 1992; William *et al.*, 1992) or the well-known Gauss reduction method (Bodewig, 1965; Kiełbasiński and Szczepik, 1992). All of these algorithms are complicated and need a definite precision of float number operations to guarantee acceptable results. An estimate of the inversion error is not calculated as a standard result. There are some tests used for the qualification of a rank defect in the procedure application (SVD, BF, CF), but there is no direct evaluation of the "quality of inversion", and it is hard to check the error in calculations, e.g. in the case of a limited number representation. The proposed approach is based on some form of iterative expansion and it is possible to define a measure of the error as the difference between the actual state of calculations and an "ideal" solution. From this point of view, the proposed approach can be interesting even in the case of ordinary PC algorithms, e.g. in the case of matrices of very high dimensions or the necessity of limiting the calculation effort.

2. Problem Statement and Its Solution

Consider a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, defined over the real number field (or the complex number field $\mathbb{C}$). In the sequel, we will refer to the real numbers as the field of matrices, but the algorithm is capable of inversing complex matrices as well. We will look for a matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ that will satisfy the equation

$$\boldsymbol{B}\,\boldsymbol{A} = \boldsymbol{A}\,\boldsymbol{B} = \boldsymbol{I}, \quad \boldsymbol{I} \in \mathbb{R}^{n\times n}. \tag{1}$$

Consider a square matrix $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ that has all the eigenvalues σ_i satisfying the condition

$$|\sigma_i| < 1, \quad i = 1, \ldots, n. \tag{2}$$

The above condition will assure the convergence

$$\|\boldsymbol{D}^k\| \to 0 \quad \text{as} \quad k \to \infty \tag{3}$$

for any measure $\|\cdot\|$ defined for matrix $\boldsymbol{D}$, e.g. the trace or the determinant. If the above condition is satisfied, we can introduce the following definition:

$$(\boldsymbol{I}-\boldsymbol{D})^{-1} \triangleq \boldsymbol{I}+\boldsymbol{D}+\boldsymbol{D}^2+\cdots+\boldsymbol{D}^q+\cdots \quad \text{as} \quad q \to \infty, \tag{4}$$

which can be easily verified by multiplying both the sides of (4) by $\boldsymbol{I}-\boldsymbol{D}$, where $\boldsymbol{I}$ is the identity matrix defined in (1). Now, introduce a scaling coefficient $\alpha > 0, \quad \alpha \in \mathbb{R}$, which is determined for the matrix $\boldsymbol{A}$ such that

$$\boldsymbol{D} = \boldsymbol{I} - \alpha\boldsymbol{A} \tag{5}$$

satisfies (2). This relation is equivalent to determining

$$\boldsymbol{A} = \alpha^{-1}(\boldsymbol{I} - \boldsymbol{D}). \tag{6}$$

The matrix $\boldsymbol{B}$, being the inverse of the matrix $\boldsymbol{A}$, can be calculated as

$$\begin{aligned}\boldsymbol{B} = \boldsymbol{A}^{-1} &= \left(\alpha^{-1}(\boldsymbol{I} - \boldsymbol{D})\right)^{-1} = \alpha(\boldsymbol{I} - \boldsymbol{D})^{-1} \\ &= \alpha(\boldsymbol{I} + \boldsymbol{D} + \boldsymbol{D}^2 + \cdots),\end{aligned} \tag{7}$$

and it is determined by the series of the right-hand side of (4). The series expansion in (7) has to be limited in a way. As for the truncation error of this series for the order p, we obtain

$$\begin{aligned}\Delta\boldsymbol{B} &= \alpha\boldsymbol{D}^p(\boldsymbol{I} + \boldsymbol{D} + \boldsymbol{D}^2 + \cdots + \boldsymbol{D}^q + \cdots) \\ &= \alpha\boldsymbol{D}^p(\boldsymbol{I} - \boldsymbol{D})^{-1} \cong \boldsymbol{D}^p\boldsymbol{A}^{-1}.\end{aligned} \tag{8}$$

The error estimate is then dependent on the first term $\boldsymbol{D}^p$ neglected in the series (7) and the inverse matrix $\boldsymbol{A}^{-1}$. As a measure of the convergence, we introduce

$$\begin{aligned}\|\Delta\boldsymbol{B}\| &= \sum_{i,j}|\alpha_{i,j}|\sum_{i,j}|\rho_{i,j}|, \\ \boldsymbol{A}^{-1} &= \{\alpha_{i,j}\}, \quad \boldsymbol{D}^p = \{\rho_{i,j}\}, \quad i,j = 1,\ldots,n.\end{aligned} \tag{9}$$

In iterative calculations, the first term of (9), i.e. $\sum_{i,j}|\alpha_{i,j}|$, is not known, but it can be approximated by the part of the series already included in expression (7). The second term $\sum_{i,j}|\rho_{i,j}|$ can be determined directly, as the matrix $\boldsymbol{D}^p$ is the last matrix included into the series expansion.

In the case of the properties and coefficients of the matrix $\boldsymbol{A}$ which are not known *a priori*, small values of the scaling factor α will be used for the preservation of the condition (2) and then the series in (7) or (8) can decrease very slowly. The efficiency of the algorithm will be limited. The way of a fast calculation of the series $\boldsymbol{I} + \boldsymbol{D} + \boldsymbol{D}^2 + \cdots + \boldsymbol{D}^q + \cdots$ will be investigated in the next section.

3. Algorithm of a Fast Calculation of the Matrix Series Expansion

This aspect of calculation will be investigated by determining the number of matrix operations, e.g. the products or additions of matrices. These operations are performed by neuro-processors (Beernaert and Roose, 1991; Synapse 3, 1997) very fast and in a comparable time.

The iterations of the algorithm will start with the determination of the following expression for $\boldsymbol{G}_0$:

$$\boldsymbol{G}_0 = \boldsymbol{I} + \boldsymbol{D} + \cdots + \boldsymbol{D}^{m-1}, \quad m > 2, \quad \boldsymbol{G}_0 \in \mathbb{R}^{n\times n} \tag{10}$$

and a multiplier $\boldsymbol{H}_{1,m} \in \mathbb{R}^{n\times n}$ equal to

$$\boldsymbol{H}_{1,m} = \boldsymbol{D}^m. \tag{11}$$

In the second step the following expressions are calculated:

$$\begin{aligned}\boldsymbol{G}_1 = \boldsymbol{G}_0 + \boldsymbol{G}_0\boldsymbol{H}_{1,m} &= \boldsymbol{I} + \boldsymbol{D} + \cdots + \boldsymbol{D}^{2m-1}, \\ \boldsymbol{H}_{2,m} &= \boldsymbol{H}_{1,m}\,\boldsymbol{H}_{1,m}.\end{aligned} \tag{12}$$

Each step corresponding to (12) introduces three matrix operations: one addition and two multiplications. All k future steps will be determined as in (12):

$$\begin{aligned}\boldsymbol{G}_k = \boldsymbol{G}_{k-1} + \boldsymbol{G}_{k-1}\boldsymbol{H}_{k,m} &= \boldsymbol{I} + \boldsymbol{D} + \cdots + \boldsymbol{D}^{2^k m-1}, \\ \boldsymbol{H}_{k+1,m} &= \boldsymbol{H}_{k,m}\,\boldsymbol{H}_{k,m},\end{aligned} \tag{13}$$

and the result of the calculations will contain $2^k m$ elements of the series (7). At each step of iterations, matrix $\boldsymbol{G}_k$ can be used for evaluating the first term in (9) and the matrix $\boldsymbol{H}_{k+1,m}$ will yield the evaluation of the second term in the error measure (9). The rate of convergence is very fast, e.g. in the case of $m = 4$ and $k = 8$ steps of the algorithm, the number of matrix operations is equal to 29 and the series (7) contains 1024 components.

The only problem is a proper choice of the scaling factor α that will preserve the condition (2). The evaluation of the scale of matrix elements is usually performed in the case of neuro-processors to achieve the best calculation accuracy; hence some hints regarding the proper choice are available (Synapse 3, 1997). The choice of a small value of α will perhaps demand more steps of iterations, but one additional step of the algorithm (13) introduces only 3 additional matrix operations.

There is a possibility of estimating a proper scaling factor α if the eigenvalues $\lambda_i, \quad i = 1,\ldots,n$ of the inverted matrix $\boldsymbol{A}$ are known. The eigenvalues σ_i of the matrix $\boldsymbol{D}$ (5) are

$$\sigma_i = 1 - \alpha\lambda_i, \quad i = 1,\ldots,n, \tag{14}$$

and the condition for the convergence of the series $\boldsymbol{I} + \boldsymbol{D} + \boldsymbol{D}^2 + \cdots + \boldsymbol{D}^q + \cdots$ amounts to the limitation of the absolute value of σ_i to the unit circle, in spite of the Mieses theorem (Collar and Simpson, 1987). The boundary value of the scaling factor is defined by

$$\alpha < \frac{2}{|\lambda_{\max}|}, \tag{15}$$

where $\lambda_{\max}$ is the maximal absolute eigenvalue of the matrix $\boldsymbol{A}$. This value can be estimated by the investigation of the convergence of the subsequent power expression of $\boldsymbol{A}$ (Collar and Simpson, 1987).

The condition (15) can be directly used in the case of real, symmetric matrices and, primarily the aim of the proposed algorithm was the inversion of the matrices that yield a Hessian matrix, in a gradient-like approach.

4. Results of Testing

The presented algorithm was used for the inversion of symmetric matrices. But it can also be used in the case of non-symmetric matrices, provided that the condition (2) is satisfied. Various aspects of the introduced scheme were investigated: the sensitivity of the algorithm to the choice of the scaling factor α, the convergence rate of the algorithm, the possibility of error estimation and the sensitivity to rounding in a number representation.

The derived error (9), due to the impact of the truncation of the iteration number (8), was estimated in the form

$$\|\Delta \boldsymbol{D}\| = \sum_{i,j} |\rho_{i,j}|, \quad \boldsymbol{H}_{k+1,m} = \{\rho_{i,j}\}, \tag{16}$$

where $i, j = 1, \ldots, n$, which can be easily determined after the k-th step of the iterations (12)–(13). As the reference, we used a measure of the difference between the identity matrix $\boldsymbol{I}$ and the product of the matrix $\boldsymbol{A}$ and the matrix $\boldsymbol{A}_{\text{inv}}(k)$ inverted by the algorithm, within k steps of iterations:

$$\begin{gathered} \|\Delta_{\text{inv}}\| = \sum_{i,j} |v_{i,j}|, \quad \boldsymbol{V} = \{v_{i,j}\}, \\ \boldsymbol{V} = \boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}_{\text{inv}}(k). \end{gathered} \tag{17}$$

As an example, consider a covariance matrix calculated in some estimation algorithm for 6 parameters. The matrix $\boldsymbol{A} \in \mathbb{R}^{6\times 6}$ contains values of correlations between different signals scaled to the magnitude 1 on the main diagonal. The components of the matrix $\boldsymbol{A}$ are listed below:

$$\boldsymbol{A} = \begin{bmatrix} 1.0000 & 0.9170 & 0.8384 & 0.6063 & 0.5388 & 0.4799 \\ 0.9170 & 1.0000 & 0.9173 & 0.6787 & 0.6060 & 0.5367 \\ 0.8384 & 0.9173 & 1.0000 & 0.7316 & 0.6784 & 0.6061 \\ 0.6063 & 0.6787 & 0.7316 & 1.0000 & 0.9572 & 0.8546 \\ 0.5388 & 0.6060 & 0.6784 & 0.9572 & 1.0000 & 0.9572 \\ 0.4799 & 0.5367 & 0.6061 & 0.8546 & 0.9572 & 1.0000 \end{bmatrix}. \tag{18}$$

The inversion algorithm was derived for $m = 4$ at the start of the iterations (12). Table 1 presents inversion errors for different scaling factors α determined for consecutive numbers of iterations. The calculations were performed using a PC with a double representation of real numbers.

An inspection of the above results yields some important conclusions. The choice of the scaling factor α exerts strong influence on the convergence rate. An optimal choice (in the sense of a minimal calculation time) $\alpha = 0.428$, presented as the first column in Table 1, introduced very fast convergence, but increasing α resulted in some non-stability of iterations, since the condition (2) was not satisfied. The safe choice $\alpha = 0.1$ deteriorated a little the convergence rate, but one additional step of iterations yielded the same accuracy.

Very conservative choices of $\alpha = 0.01$ or 0.001 resulted in an increasing number of necessary iterations with 2 to 4 additional steps. In the case of very small scaling values α, the first steps of iterations consisted in multiplication (12) of matrices that were very close to the identity matrix, and the rate of convergence was poor, which is represented by increasing errors (the last columns in Table 1). For $\alpha = 0.001$ a large number 4^4 of elements in the series (4) was necessary to begin the reduction of resolution errors. The dependence of the convergence rate on the scaling factor α constitutes a drawback of the algorithm, but a proper value of this factor can be estimated when scaling matrices during the calculation in neuro-processors (Golub *et al.*, 1992; Masters, 1993; Synapse 3, 1997). Another conclusion from the inspection of Table 1 concerns a good evaluation of the inversion error $\|\Delta_{\text{inv}}\|$ by the value of $\|\Delta \boldsymbol{D}\|$ defined in (16). In the last steps of the algorithm this correspondence was violated, as the inversion errors remained constant while the rounding errors $\|\Delta \boldsymbol{D}\|$ were permanently decreasing. This was a direct effect of a limited representation of double numbers in the computer memory. In the case of a float representation, the inversion errors $\|\Delta_{\text{inv}}\|$ were kept constant at a level approximately equal to 10^{-6}.

The choice of the scaling factor α can be performed by a evaluation of (15). The maximal absolute eigenvalue

Table 1. Effect of different choices of the scaling factor α.

Scale factor	Iteration k									
α/errors	1	2	3	4	5	6	7	8	9	10
$\alpha = 0.428$										
$\|\Delta_{\text{inv}}\|$	7.758	5.815	2.624	6.5×10^{-1}	6.8×10^{-3}	8.1×10^{-11}	2.0×10^{-13}	2.0×10^{-13}		
$\|\Delta \boldsymbol{D}\|$	7.758	5.815	2.624	6.5×10^{-1}	6.8×10^{-3}	8.0×10^{-11}	1.6×10^{-42}	2×10^{-169}		
$\alpha = 0.1$										
$\|\Delta_{\text{inv}}\|$	8.294	6.636	4.112	2.243	7.2×10^{-1}	1.0×10^{-2}	2.0×10^{-3}	4.1×10^{-10}	3.1×10^{-13}	
$\|\Delta \boldsymbol{D}\|$	8.294	6.636	4.112	2.243	7.2×10^{-1}	1.0×10^{-2}	2.0×10^{-3}	4.0×10^{-47}	1×10^{-129}	
$\alpha = 0.01$										
$\|\Delta_{\text{inv}}\|$	7.051	8.347	7.725	5.807	3.439	1.719	3.1×10^{-1}	3.4×10^{-4}	1.9×10^{-12}	1.9×10^{-12}
$\|\Delta \boldsymbol{D}\|$	7.051	8.347	7.725	5.807	3.439	1.719	3.1×10^{-1}	3.4×10^{-4}	5.0×10^{-16}	3×10^{-122}
$\alpha = 0.001$										
$\|\Delta_{\text{inv}}\|$	6.124	6.466	7.460	8.474	7.204	4.797	2.795	1.206	7.9×10^{-2}	1.5×10^{-6}
$\|\Delta \boldsymbol{D}\|$	6.124	6.466	7.460	8.474	7.204	4.797	2.795	1.206	7.9×10^{-2}	1.5×10^{-6}

Table 2. Effect of different number representations.

Mantissa representation	double	2^{22}	2^{16}	2^{14}	2^{12}	2^{10}
$\|\Delta_{\text{inv}}\|$	4.1×10^{-10}	9.8×10^{-4}	1.0×10^{-1}	3.5×10^{-1}	9.9×10^{-1}	4.37

determined for the matrix $\boldsymbol{A}$ is $\lambda_{\max} = 4.651$. Due to (15) the power series of $\boldsymbol{D}$ converges if $\alpha < 2/4.651 = 0.430$, and for the value of $\alpha = 0.428$ the convergence rate of the algorithm was tested, cf. Table 1. If the evaluation of a proper value of the scaling factor α, based on (15), can be performed before the procedure for matrix inversion, then the algorithm (12)–(13) will be very fast, consist of the smallest possible number of steps and therefore preserve the smallest error in calculations, cf. Table 1.

Another test was performed for the evaluation of inversion errors at different precisions of float operations. In various neuro-processors, the precision of the number representation is remarkably worse than in the standard PC-processors, and the proposed algorithm may produce some unexpected effects. The scaling factor α was equal to 0.1. The inversion errors were determined at the step $k = 8$, when iterations yielded constant inverted matrix coefficients, cf. Table 2.

The above results seem to be very poor, but calculations were stable and produced reasonable values, e.g. the product of $\boldsymbol{A}$ and the calculated inversion $\boldsymbol{A}_{\text{inv}}$ for the 14-digit mantissa was equal to

$$\boldsymbol{A}\,\boldsymbol{A}_{\text{inv}} = \begin{bmatrix} 0.9978 & 0.0039 & 0.0015 & 0.0004 & -0.0017 & 0.0007 \\ 0.0019 & 0.9934 & 0.0033 & 0.0033 & -0.0010 & 0.0027 \\ 0.0008 & 0.0032 & 0.9957 & -0.0014 & 0.0021 & -0.0015 \\ -0.0038 & -0.0040 & -0.0134 & 0.9825 & 0.0217 & -0.0213 \\ 0.0074 & 0.0096 & 0.0258 & 0.0292 & 0.9595 & 0.0426 \\ -0.0038 & -0.0052 & -0.0126 & -0.0137 & 0.0211 & 0.9763 \end{bmatrix}. \tag{19}$$

In the case of a limited number representation, a proper choice of the scaling factor α seems to be important. The scaling factor equal to an optimal value of $\alpha = 0.428$ results in the reduction of $\|\Delta_{\text{inv}}\|$ to the value of 0.12 for the 14-digit representation. Another aspect of the limited number representation can be observed, since the end of the calculations was achieved in five steps of the iteration (13). It was not necessary to continue iterations when the number representation introduced definite errors.

The operation of matrix inversion is very sensitive to various numerical defects, e.g. the singularity of the processed matrix. This defect will lead to many problems in the case of the standard algorithms (Bodewig, 1965; Golub *et al.*, 1992; Higham, 1996), and will generally induce high magnitudes of the elements of the resulting matrix. In the presented example, in the matrix $\boldsymbol{A}$ (18), the last column replaced the fifth one. The inversion of such a matrix (for the float precision and $\alpha = 0.1$) yielded the matrix $\boldsymbol{A}_{\mathrm{inv}}$, which, multiplied by $\boldsymbol{A}$, produced

$$\boldsymbol{A}\,\boldsymbol{A}_{\mathrm{inv}} = \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ -0.0159 & -0.02943 & 0.02275 & 0.35954 & 0.3370 & 0.3370 \\ 0.0159 & 0.02943 & -0.02275 & -0.3595 & 0.6629 & 0.6629 \end{bmatrix}. \tag{20}$$

This result is remarkable, i.e. the upper part of the matrix $\boldsymbol{A}$ was inverted and the last two rows are linearly dependent. This form is useful and does not introduce non-stability problems at subsequent calculations, e.g. in the case of determining a direction for an optimisation algorithm based on the inverse of the Hessian.

The proposed algorithm was primarily invented for the inversion of symmetric matrices by neuro-processors but it can be used for the inversion of any matrix that satisfies (2), e.g. for a skew matrix

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} 1.0000 & 0.9170 & 0.8384 & 0.6063 & 0.5388 & 0.4799 \\ -0.9170 & 1.0000 & 0.9173 & 0.6787 & 0.6060 & 0.5367 \\ -0.8384 & -0.9173 & 1.0000 & 0.7316 & 0.6784 & 0.6061 \\ -0.6063 & -0.6787 & -0.7316 & 1.0000 & 0.9572 & 0.8546 \\ -0.5388 & -0.6060 & -0.6784 & -0.9572 & 1.0000 & 0.9572 \\ -0.4799 & -0.5367 & -0.6061 & -0.8546 & -0.9572 & 1.0000 \end{bmatrix} \tag{21}$$

the inverse was determined after $n = 4$ steps of the iteration (12), for the scaling factor $\alpha = 0.1$ and the double representation, with the inversion error of $\|\Delta_{\mathrm{inv}}\| = 7.36 \times 10^{-14}$. The convergence rate was remarkably better than in the case of the original matrix $\boldsymbol{A}$ (18).

Finally, another possibility of the proposed algorithm was checked, namely the inversion of a complex matrix. The only limitation in the convergence of the algorithm is in fact the condition (2), which can be satisfied by a complex matrix. As an example, the 6×6 complex matrix

$$\boldsymbol{A} = \begin{bmatrix} 1.0 + i0.24 & 0.9170 + i0.22 & 0.8384 + i0.201 \\ 0.9170 + i0.22 & 1.0 + i0.24 & 0.9173 + i0.22 \\ 0.8384 + i0.201 & 0.9173 + i0.22 & 1.0 + i0.24 \\ 0.6063 + i0.145 & 0.6787 + i0.163 & 0.7316 + i0.176 \\ 0.5388 + i0.129 & 0.6060 + i0.145 & 0.6784 + i0.163 \\ 0.4799 + i0.115 & 0.5367 + i0.128 & 0.6061 + i0.145 \end{bmatrix}$$

$$\begin{bmatrix} 0.6063 + i0.145 & 0.5388 + i0.129 & 0.4799 + i0.115 \\ 0.6787 + i0.163 & 0.6060 + i0.145 & 0.5367 + i0.128 \\ 0.7316 + i0.176 & 0.6784 + i0.163 & 0.6061 + i0.145 \\ 1.0 + i0.24 & 0.9572 + i0.230 & 0.8546 + i0.205 \\ 0.9572 + i0.230 & 1.0 + i0.24 & 0.9572 + i0.230 \\ 0.8546 + i0.205 & 0.9572 + i0.230 & 1.0 + i0.24 \end{bmatrix} \tag{22}$$

was used (imaginary parts of numbers were set to 24% of real parts in (18)). The algorithm for the inversion outlined in (12) and (13) was performed exactly in the same way. The only difference was the representation of the matrix $\boldsymbol{A}$. The elements were complex numbers and the corresponding procedures of multiplication and addition were adapted to complex numbers. After eight steps of the algorithm with the scaling factor $\alpha = 0.1$, the inverse was determined with the error (17) equal to 3.07×10^{-13}. Investigations of other aspects regarding complex number calculations do not introduce substantially different results in comparison with the effects observed for real number calculations.

5. Conclusions

The proposed algorithm was primarily invented for chips applied in neuro-processors or net-accelerators used in calculation for the optimisation of artificial neural nets. Its application can increase the convergence rate of various algorithms used for the optimisation of net weights and based on the application of gradient approaches or back-propagation methods. The iterative way of the inverse calculation induces an increase in the numerical effort in the case of common PC applications. However, it can be an alternative in the case of large matrices, where the numerical effort increases faster than the square of a matrix dimension. A remarkable advantage of this procedure is its stable behaviour in the case of singular matrices in which the exact place and dimension of the rank defect are not known. Another interesting feature is the calculation error, which is produced by rounding the number representation. The magnitude of this error seems to be

independent of the numerical conditioning of the inverted matrix. The value of this error can be traced at the k-th step of the algorithm and, after the level of the number representation has been reached, the procedure can be terminated.

The most visible drawback of this approach is the scaling (5)–(6) of the inverted matrix. It can be performed simultaneously with preparing the matrix for computations in the case of neuro-processors or in the starting phase of calculation like in the presented example. In the case of the possible knowledge of the eigenvalues of the inverted matrix the relation (15) can be used for the determination of a proper value of α. If the scaling factor is not predefined, after scaling the components the value of the rounding error (16) can be controlled for the detection of rapid increments in the case of non-stable behaviour in the scheme (12) and (13).

References

Beernaert L. and Roose D. (1991): *Parallel Gaussian elimination, iPSC/2 hypercube versus a transputer network*, In: Numerical Linear Algebra (G. Golub, Ed.). — NATO ASI Series, Vol. 70, Berlin: Springer.

Bodewig E. (1965): *Matrix Calculus*. — Amsterdam: North-Holland.

Bjorck A. (1991): *Error analysis of least squares algorithms*, In: Numerical Linear Algebra (G. Golub, Ed.). — NATO ASI Series, Vol. 70, Berlin: Springer.

Collar A.R. and Simpson A. (1987): *Matrices and Engineering Dynamics*. — New York: Wiley.

Golub G., Greenbaum A. and Luskin M. (1992): *Recent Advances in Iterative Methods*. — New York: Springer.

Higham J.H. (1996): *Accuracy and Stability of Numerical Algorithms*. — Philadelphia: SIAM.

Kiełbasiński A. and Szczepik H. (1992): *Numerical Algebra*. — Warsaw: WNT, (in Polish).

Masters T. (1993): *Practical Neural Network Recipes in C++*. — London: Academic Press.

Synapse 3 (1977): *PC Siemens Card- Technical Documentation*. — Dresden: Siemens.

William H., Flannery B., Teukolsky S. and Vetterling W. (1992): *Numerical Recipes in C*. — New York: Cambrige University Press.

Received: 23 September 2002
Revised: 31 March 2003