Unified frameworks for high order Newton-Schulz and Richardson iterations: a computationally efficient toolkit for convergence rate

Citation for the original published paper (version of record):
Stotsky, A. (2019). Unified frameworks for high order Newton-Schulz and Richardson iterations: a computationally
efficient toolkit for convergence rate improvement. Journal of Applied Mathematics and Computing, 60(1-2): 605-623. http://dx.doi.org/10.1007/s12190-018-01229-8

Journal of Applied Mathematics and Computing (2019) 60:605–623
https://doi.org/10.1007/s12190-018-01229-8

BRIEF REPORTS

Unified frameworks for high order Newton-Schulz and Richardson iterations: a computationally efficient toolkit for convergence rate improvement

Alexander Stotsky[1]

Received: 2 October 2018 / Published online: 1 January 2019

Abstract
Convergence rate and robustness improvement together with reduction of computational complexity are required for solving the system of linear equations $A\theta_* = b$ in many applications such as system identification, signal and image processing, network analysis, machine learning and many others. Two unified frameworks (1) for convergence rate improvement of high order Newton-Schulz matrix inversion algorithms and (2) for combination of Richardson and iterative matrix inversion algorithms with improved convergence rate for estimation of θ_* are proposed. Recursive and computationally efficient version of new algorithms is developed for implementation on parallel computational units. In addition to unified description of the algorithms the frameworks include explicit transient models of estimation errors and convergence analysis. Simulation results confirm significant performance improvement of proposed algorithms in comparison with existing methods.

Keywords Richardson iteration · Neumann series · High order Newton-Schulz algorithm · Least squares estimation · Harmonic regressor · Strictly Diagonally Dominant Matrix · Symmetric positive definite matrix · Ill-conditioned matrix · Polynomial preconditioning · Matrix power series factorization · Computationally efficient matrix inversion algorithm · Simultaneous calculations

1 Introduction

Least squares method is widely used in system identification, signal processing, adaptive control and in many other areas [1]. For accurate solution many least squares problems can be associated with estimation of the parameter vector θ_*, which satisfies algebraic equation

✉ Alexander Stotsky
alexander.stotsky@telia.com

1 Chalmers Industriteknik, Chalmers Teknikpark, Sven Hultins gata 9, 412 58 Gothenburg, Sweden

$$A\theta_* = b \tag{1}$$

where b is the vector, and A is SPD (Symmetric and Positive Definite) matrix, which can often be presented in the form:

$$A = \sum_{i=1}^{w} \varphi_i \varphi_i^T \tag{2}$$

where φ_i is the regressor (for example harmonic regressor, [2–4]) and w is the window size. Small window size which is often used (a) for detection of fast transients in time series (time frequency) analysis, [5,6] (b) in regression analysis for small datasets, [1] and (c) in many other cases implies ill-conditioning, [7] and even rank deficiency in some cases of the matrix A.

Iterative methods for solving (1) are often preferable (especially for large-scale systems) due to simplicity, better accuracy and robustness, less processor time and memory space compared to direct methods, [8]. However, ill-conditioning of information matrix implies very slow convergence of the recursive procedures for matrix inversion, [9]. Moreover, limited computational capacity for on-board applications for example, results in significant loss of accuracy and error accumulation in finite-digit calculations, which in turn may result in numerical instability, [6].

Fast convergent, computationally efficient and robust (with respect to round-off, truncation and accumulation errors) matrix inversion algorithms are required in many time-critical applications. In particular, the problem of convergence rate and robustness improvement for widely used high order Newton-Schulz iterative procedure, [10,11] is especially relevant for ill-conditioned matrices, [12].

Convergence rate of Newton-Schulz algorithm can be improved by choosing higher order. However, this requires too many additional calculations and results in loss of accuracy due to error accumulation. A trade-off between convergence rate, computational complexity and robustness necessitates development of unified framework for convergence rate improvement of high order Newton-Schulz algorithm. This unified framework offers a number of solutions, where the trade-off between convergence rate and computational complexity can be easily quantified, see Sects. 3.2 and 4.

Moreover, accuracy and robustness associated with minimization of the residual error $A\theta_{k-1} - b$, where θ_k is the estimate of θ_* together with possibility of inclusion of matrix inversion algorithm with computational parallelization, [13] is motivation to look for combined solution, which involves Richardson iteration, [14] and improved Newton-Schulz algorithm, [15,16], see Sect. 2.

This paper is organized as follows. General framework for Richardson iteration, brief overview of the previous work and the problem statement for convergence rate improvement is presented in the next Sect. 2. Unified framework for matrix inversion algorithms (and their recursive computationally efficient parallel realization) and Richardson parameter estimation algorithms with improved convergence rate are presented Sects. 3, 4 and 5 respectively as main contributions of this paper. Simulation results, which confirm significant performance improvement of proposed algorithms in comparison with existing methods are presented in Sect. 6. Some concluding remarks and future directions are outlined in Sect. 7.

2 Previous work, unification and problem statement

2.1 Unified framework for performance improvement of Richardson iteration

For estimation of the parameter vector θ_* which satisfies (1) the following Richardson iteration is considered:

$$\theta_k = \theta_{k-1} - \underbrace{\left\{\sum_{d=0}^{q-1} F_k^d\right\}}_{\substack{\text{Neumann}\\ \text{Series}}} \underbrace{G_k}_{\substack{\text{Gain}\\ \text{Matrix}}} \underbrace{(A\theta_{k-1} - b)}_{\text{Residual Error}} \tag{3}$$

$$F_k = I - G_k A, \quad F_0 = I - G_0 A \tag{4}$$

$$\rho(F_0) = \rho(I - G_0 A) < 1 \tag{5}$$

$$\theta_0 = G_0 b \tag{6}$$

where θ_k is the estimate of θ_*, F_k is the error matrix, I is the identity matrix, ρ is the spectral radius which defines the convergence condition (5), $q = 1, 2, 3, \ldots$ is the order of Neumann series, G_k is a gain matrix, and G_0 is the preconditioner.

A general framework for modified Richardson iteration is associated with two multiplicative terms $\sum_{d=0}^{q-1} F_k^d$ and G_k introduced for convergence rate improvement. Indeed, original Richardson algorithm (3) has a simple form with $q = 1$ and $G_k = I$, Jacobi method is obtained with $q = 1$ and G_k, which is inverse diagonal of A (for Strictly Diagonally Dominant, SDD matrices), Gauss-Seidel method involves upper and lower triangular parts of A, [17], and finally Dubois–Greenbaum–Rodrigue method is associated with $G_k = G_0$ and $F_k = F_0$, where G_0 is the constant preconditioner, which satisfies (5), [15]. Notice that the convergence of the methods described above is slow, and the iteration can be even unstable in some cases, [17].

The convergence rate of the algorithm depends on the choice of the gain matrix G_k only (when decoupling Neumann series via the choice of $q = 1$) and can be improved via iterative preconditioning (dynamic preconditioning), [16]. The fastest convergence, where $\theta_k = \theta_*$ is achieved for optimal gain matrix $G_k = A^{-1}$. Therefore the gain matrix G_k can be associated with *the iterative procedure which converges as fast as possible to* A^{-1}, providing faster convergence compared to gradient methods, [18,19].

Neumann series which uses the most recent values of the gain matrix is applied for additional accuracy improvement of the estimate of A^{-1} and hence for convergence rate improvement, [16]. Notice that error accumulation problem (the loss of accuracy in finite-digit calculations) associated with Neumann series requires low orders $q = 1, 2$, [6] and therefore the potential of this term is limited. Therefore the design of fast convergent recursive matrix inversion algorithm G_k is considered as a main tool for convergence rate improvement of the Richardson iteration procedure in this paper.

2.2 Existing matrix inversion and parameter estimation algorithms

2.2.1 Splitting and preconditioning

Any positive definite and symmetric matrix A, whose inverse should be calculated can be split as follows (see for example [17] and references therein):

$$A = S - D \tag{7}$$

$$I - S^{-1}A = S^{-1}D \tag{8}$$

$$\rho(I - S^{-1}A) = \rho(S^{-1}D) < 1 \tag{9}$$

where the spectral radius $\rho(\cdot)$ defined in (9) is less than one for symmetric and positive definite matrices A and S (where S^{-1} is the preconditioner), provided that $2S - A$ is a positive definite matrix, [20,21].

For example, the matrix S can be chosen as a diagonal matrix, which contains the diagonal elements of SDD (Strictly Diagonally Dominant) and positive definite matrix A, [16]. For positive definite (not SDD) matrix A the simplest preconditioner can be chosen as $S^{-1} = I/\alpha$ with $\alpha = \|A\|_\infty/2 + \varepsilon$, where $\|\cdot\|_\infty$ is the maximum row sum matrix norm, and $\varepsilon > 0$ is a small positive number, [9].

Other types of preconditioning can be found in [17,22–25], see also references therein.

Notice that the spectral radius (9) is too close to one for ill-conditioned matrix A, which implies stability problem in the presence of error accumulation, [6]. Preconditioning can be modified by choosing easy invertible matrix S, where $\rho(S^{-1}D) > 1$ and applying stepwise calculations. Stepwise splitting method involves sequential application of matrix inversion algorithm with larger stability margins for robustness enhancement, [6]. In other words, the inversion procedure is divided into a number of steps to avoid instability and increasing computational time. The computational time of stepwise splitting method depends on the convergence rate of inversion algorithm, that necessitates the development of fast convergent inversion algorithms, which accelerate also Richardson iteration (3) for time-critical applications.

Existing matrix inversion and parameter estimation algorithms are described in the next Sect. 2.2.2.

2.2.2 High order Newton-Schulz matrix inversion algorithm

Inverse of the matrix A can be estimated by G_k with the following the most general and well known algorithm:

$$G_k = \left\{ \sum_{d=0}^{p-1} F_{k-1}^{d} \right\} G_{k-1} \tag{10}$$

where F_k is defined in (4) and $p = 2, 3, \ldots$ is the order of the algorithm, which has the following error model

$$F_k = (S^{-1}D)^{p^k} \tag{11}$$

and convergence condition (9) with the preconditioner $G_0 = S^{-1}$, $k = 1, 2, \ldots$.

Algorithm (10) (presented in different forms) is known as high order Newton-Schulz algorithm, [10,11,16] and hyperpower iterative matrix method, [26–29]. Notice that Newton-Schulz algorithm, $p = 2$, [30,31] is used more often due to significant error accumulation for higher orders, [10].

The vector of unknown parameters can be estimated via (10) as follows:

$$\theta_k = G_k b \tag{12}$$

$$\tilde{\theta}_k = (S^{-1}D)^{p^k - 1}\tilde{\theta}_0 \tag{13}$$

where θ_k is the estimate of θ_* defined in (12), and the model for the parameter mismatch $\tilde{\theta}_k = \theta_k - \theta_*$, where $\theta_0 = G_0 b$ is defined in (13).

Notice that the following error model for the parameter mismatch is valid for Richardson iteration (3) with (10) and $G_0 = S^{-1}$, [16]:

$$\tilde{\theta}_k = (S^{-1}D)^{\frac{q\,(p^{k+1} - p)}{(p-1)}}\tilde{\theta}_0 \tag{14}$$

Notice also that low orders, $p = 2, 3$ and $q = 1, 2$ of the algorithms are usually chosen for prevention error accumulation [6,10] .

Comparison of two error models (13) and (14) with $q = 1$ shows that Richardson iteration (3) with G_k defined in (10) converges faster than the algorithm (12). Simulation results show that the difference in convergence rates is more pronounced for low orders, $p = 2, 3$.

Notice that the initial mismatch $\tilde{\theta}_0$ can be essentially reduced in recursive least squares estimation for example, using information available in the previous steps, [1,9] and by polynomial preconditioning, [18], described in the next Sect. 2.2.3.

2.2.3 Computationally efficient matrix inversion algorithm

Matrix inversion algorithm of order $h = 1, 2, 3, \ldots$ which requires one matrix addition and one multiplication only in each step can be written as follows, [6]:

$$G_k = G_0 + (S^{-1}D)^{2h}G_{k-1} = \left\{\sum_{j=0}^{k}(S^{-1}D)^{2hj}\right\}G_0 \tag{15}$$

with the error model:

$$G_k - A^{-1} = (S^{-1}D)^{2hk}\{G_0 - A^{-1}\} \tag{16}$$

where the polynomial preconditioner

$$G_0 = \left\{ \sum_{j=0}^{h-1} (S^{-1}D)^{2j} \right\} \{(S^{-1}D) + I\} S^{-1}$$
$$= (I - (S^{-1}D)^{2h}) A^{-1} \tag{17}$$

and $(S^{-1}D)^{2h}$ are precalculated. Notice that the algorithm (15) is very computationally efficient since the complexity depends on the order h in precalculations only.

Notice also that similar algorithms based on the approximate inverse chains and fast matrix series expansion are described in [32] and [33] respectively.

Unfortunately, the algorithm (15) has significantly slower convergence compared to algorithm (10), [5]. However, error accumulation problem is more pronounced for algorithm (10), especially for higher orders, [6]. A unified framework for combined algorithms which takes the advantages of both approaches and includes high order Newton-Schulz algorithm and power series associated with (15) are designed in the next Sect. 3 for convergence rate improvement and reduction of error accumulation.

3 Matrix inversion algorithms with improved convergence rate

3.1 General power series expansion

The following relation holds for general power series in each step, $k = 0, 1, \ldots$:

$$\left\{ \sum_{j=0}^{2h\omega-1} (S^{-1}D)^j \right\} S^{-1} = (I - (S^{-1}D)^{2h\omega}) A^{-1} \tag{18}$$

where $\omega = \omega(k) = c_0 + c_1 k + c_2 k^2 + \ldots \geq 1$ is the polynomial, for example, and the coefficients $c_i \geq 0$, $i = 0, 1, 2, \ldots$ are integers.

The simplest variant of this factorization is presented in (17), [6], where $\omega = 1$ with $c_0 = 1$ and $c_i = 0$, $i = 1, 2, \ldots$. Notice that higher orders of the polynomial ω result in significant computational complexity of the algorithms based on the factorization (18). Therefore the first order polynomial $\omega = (k + 1)$, where $c_0 = c_1 = 1$ and $c_i = 0$, $i = 2, 3, \ldots$ is considered further in this paper.

Notice that (18) defines the model for power series expansion, which can easily be presented in the recursive form with minimal number of calculations in each step and computationally efficient low degree polynomial preconditioning (for example preconditioning (17)), see Sect. 4.

3.2 Unified matrix inversion algorithms

The following unified algorithm:

$$G_k = \underbrace{T}_{\substack{\text{Polynomial}\\ \text{Preconditioning}}} + \underbrace{\Gamma}_{\substack{\text{Convergence}\\ \text{Accelerator}}} \underbrace{\left\{\sum_{d=0}^{n-1} F_{k-1}^d\right\} G_{k-1}}_{\substack{\text{High Order Newton-Schulz}\\ \text{Iteration}}} \tag{19}$$

$$\Gamma = I - T\ A \tag{20}$$

$$F_k = I - G_k A, \quad \rho(F_0) = \rho(I - G_0 A) < 1 \tag{21}$$

has the error model:

$$F_k = \Gamma\ F_{k-1}^n \tag{22}$$

where Γ is the gain matrix, which accelerates the convergence ($\rho(\Gamma) < 1$ for convergence rate improvement) and the polynomial preconditioning T satisfies equation (20).

Error model (22) is proved by evaluation of the Neumann series $\sum_{d=0}^{n-1} F_{k-1}^d$, taking into account (20) together with multiplication of both sides of the Eq. (19) by A.

Different choices of the matrices Γ and T, which obey equation (20) result in the following algorithms:

1. High order Newton-Schulz iteration (10):

$$\Gamma = I,\ T = 0,\ n = 2, 3, 4, \ldots \tag{23}$$

which can also be presented in the following form, [16]:

$$\Gamma = I - G_{k-1}A,\ T = G_{k-1},\ n = 1, 2, 3, \ldots \tag{24}$$

2. High order algorithm with polynomial factorization (18) and $\omega = 1$:

$$\Gamma = (S^{-1}D)^{2h},\ T = \left\{\sum_{j=0}^{2h-1} (S^{-1}D)^j\right\} S^{-1}$$
$$\rho(S^{-1}D) < 1,\ n = 1, 2, 3, \ldots \tag{25}$$

Notice that algorithm (19), (25) with $n = 1$ becomes the algorithm (15).

3. High order algorithm with polynomial factorization (18) and $\omega = (k + 1)$:

$$\Gamma_k = (S^{-1}D)^{2h(k+1)},\ T_k = \left\{\sum_{j=0}^{2h(k+1)-1} (S^{-1}D)^j\right\} S^{-1}$$
$$\rho(S^{-1}D) < 1,\ n = 1, 2, 3, \ldots \tag{26}$$

4. High order algorithm with exponentiation:

$$\Gamma_k = (S^{-1}D)^{m^k}, \ T_k = \left\{ \sum_{j=0}^{m^k - 1} (S^{-1}D)^j \right\} S^{-1}$$
$$\rho(S^{-1}D) < 1, \ n, m = 1, 2, 3, \ldots \tag{27}$$

   Notice that algorithm (19) with exponentiation gets the form $G_k = S^{-1} + (S^{-1}D)G_{k-1}$, [5,19], where (20) becomes (8) with $n = m = 1$.
5. Modification of the high order algorithm with recursive fast calculation of the matrix series expansion proposed in [33]:

$$\Gamma_k = (S^{-1}D)^{h2^k}, \ T_k = \left\{ \sum_{j=0}^{h2^k-1} (S^{-1}D)^j \right\} S^{-1}$$
$$\rho(S^{-1}D) < 1, \ h = 1, 2, 3, \ldots, n = 1 \tag{28}$$

Notice that algorithm (19) covers both high order Newton-Schulz algorithm (10) and high order algorithm (15) and provides unified framework for new algorithms with improved performance. Known and new matrix inversion algorithms based on unified framework (19) with error models are summarized in Table 1.

Notice also that the relation (20) can be seen as extended error model associated with the error (21). The choice of Γ and T which satisfies the condition (20) and involves G_{k-1} should not contradict with the equations (19), (21). The framework (19) covers mostly the case where T is power series expansion associated with (18), which provides inaccurate estimate of A^{-1} and Γ quantifies this inaccuracy, accelerating the convergence. Therefore the convergence analysis [similar to the cases (23)–(28), see also Table 1] should be performed for each choice of Γ and T.

Comparison of the error models in Table 1 shows that the algorithm (19), (26) provides the best convergence rate with reasonable computational complexity compared to other choices. This algorithm is considered in the next Sect. 3.3.

3.3 Fast matrix inversion algorithm

The algorithm (19), (26) can be presented in the following form:

$$G_k = \left\{ \sum_{j=0}^{2h(k+1)-1} (S^{-1}D)^j \right\} S^{-1}$$
$$+ (S^{-1}D)^{2h(k+1)} \left\{ \sum_{d=0}^{n-1} F_{k-1}^d \right\} G_{k-1} \tag{29}$$

Table 1 A family of known and new matrix inversion algorithms

	Unified matrix inversion algorithms and error models
N	$G_k = T + \Gamma \left\{ \sum_{d=0}^{n-1} F_{k-1}^d \right\} G_{k-1}, \Gamma = I - TA, F_k = \Gamma F_{k-1}^n$
	$F_k = I - G_k A, A = S - D, \rho(I - S^{-1}A) = \rho(S^{-1}D) < 1, G_k \to A^{-1}$ as $k \to \infty$
1	$\Gamma = I, T = 0, G_0 = S^{-1}, F_k = F_{k-1}^n, F_k = (S^{-1}D)^{n^k}, n = 2, 3, 4, \ldots$
2	$\Gamma = (S^{-1}D)^{2h}, T = \left\{ \sum_{j=0}^{2h-1} (S^{-1}D)^j \right\} S^{-1}, G_0 = \left\{ \sum_{j=0}^{h-1} (S^{-1}D)^{2j} \right\} \{(S^{-1}D) + I\} S^{-1},$
	$F_k = (S^{-1}D)^{2h} F_{k-1}$
	$F_k = (S^{-1}D)^{2h(k+1)}, h = 1, 2, 3, \ldots, n = 1$
3	$\Gamma = (S^{-1}D)^{2h}, T = \left\{ \sum_{j=0}^{2h-1} (S^{-1}D)^j \right\} S^{-1}, G_0 = \left\{ \sum_{j=0}^{h-1} (S^{-1}D)^{2j} \right\} \{(S^{-1}D) + I\} S^{-1},$
	$F_k = (S^{-1}D)^{2h} F_{k-1}^n$
	$F_k = (S^{-1}D)^{2h \frac{(n^{k+1} - 1)}{(n-1)}}, h = 1, 2, 3, \ldots, n = 2, 3, 4, \ldots$
4	$\Gamma = (S^{-1}D)^{2h(k+1)}, T = \left\{ \sum_{j=0}^{2h(k+1)-1} (S^{-1}D)^j \right\} S^{-1},$
	$G_0 = \left\{ \sum_{j=0}^{h-1} (S^{-1}D)^{2j} \right\} \{(S^{-1}D) + I\} S^{-1}$
	$F_k = (S^{-1}D)^{2h(k+1)} F_{k-1}, F_k = (S^{-1}D)^{2h\{k \frac{(k+3)}{2} + 1\}}, h = 1, 2, 3, \ldots, n = 1$
5	$\Gamma = (S^{-1}D)^{2h(k+1)}, T = \left\{ \sum_{j=0}^{2h(k+1)-1} (S^{-1}D)^j \right\} S^{-1},$
	$G_0 = \left\{ \sum_{j=0}^{h-1} (S^{-1}D)^{2j} \right\} \{(S^{-1}D) + I\} S^{-1}$
	$F_k = (S^{-1}D)^{2h(k+1)} F_{k-1}^n, F_k = (S^{-1}D)^{2h \left\{ \frac{n^{k+2} - n^3 - (k-1)(n-1)}{(n-1)^2} + (n+2) \right\}}$
	$h = 1, 2, 3, \ldots, n = 2, 3, 4, \ldots$
6	$\Gamma = (S^{-1}D)^{m^k}, T = \left\{ \sum_{j=0}^{m^k - 1} (S^{-1}D)^j \right\} S^{-1}, G_0 = S^{-1}, F_k = (S^{-1}D)^{m^k} F_{k-1}^n$
	$F_k = (S^{-1}D)^{\frac{(m^{k+1} - n^{k+1})}{(m-n)}}$, if $m \neq n$, and $F_k = (S^{-1}D)^{(k+1)n^k}$, if $m = n$, where $n, m = 1, 2, 3, \ldots$
7	$\Gamma = (S^{-1}D)^{h2^k}, T = \left\{ \sum_{j=0}^{h2^k - 1} (S^{-1}D)^j \right\} S^{-1}, G_0 = \left\{ \sum_{j=0}^{h-1} (S^{-1}D)^j \right\} S^{-1}$
	$F_k = (S^{-1}D)^{h2^k} F_{k-1}, F_k = (S^{-1}D)^{h(2^{k+1} - 1)}, h = 1, 2, 3, \ldots, n = 1$

with the following error model (22):

$$F_k = (S^{-1}D)^{2h(k+1)} F_{k-1}^n \tag{30}$$

where the error matrix $F_k = I - G_k A$, $F_0 = (S^{-1}D)^{2h}$, $h = 1, 2\dots$ and $n = 1, 2, \dots$ are the orders, $k = 1, 2, \dots$. Polynomial preconditioner G_0 is defined in (17).

The error model (30) can also be presented in the following form:

$$F_k = (S^{-1}D)^{2h\left\{k\frac{(k+3)}{2}+1\right\}} \tag{31}$$

for $n = 1$, and in the following form:

$$F_k = (S^{-1}D)^{2h\left\{\frac{n^{k+2}-n^3-(k-1)(n-1)}{(n-1)^2}+(n+2)\right\}} \tag{32}$$

for $n > 1$. Comparison of the error models (11) and (32) quantifies convergence rate improvement associated with combined algorithm. The trade-off between convergence rate improvement and computational complexity necessitates development of recursive and computationally efficient version of the algorithm (29).

4 Recursive realization of the algorithm (29)

Algorithm (29) can be divided in two independent parts for simultaneous computations. The first part includes the following terms: $T_k = \left\{\sum_{j=0}^{2h(k+1)-1}(S^{-1}D)^j\right\} S^{-1}$ and $\Gamma_k = (S^{-1}D)^{2h(k+1)}$, and the second part is associated with Neumann series $\sum_{d=0}^{n-1} F_{k-1}^d$.

4.1 Recursive calculations for part I

Suppose that the following series $T_0 = G_0 = \left\{\sum_{j=0}^{h-1}(S^{-1}D)^{2j}\right\}\{(S^{-1}D)+I\}S^{-1}$ is precalculated recursively for $h > 1$ as follows:

```
$for\ i = 1 : (h-1)$
$U_i = (S^{-1}D)^2\, U_{i-1} + U_0$
$end$                                                    (33)
```

where $U_0 = \{(S^{-1}D)+I\}S^{-1}$ and $U_{h-1} = T_0$. The algorithm (33) requires h matrix additions and multiplications only.

Then the following recursive algorithm reduces computational complexity of the first part:

$$\Gamma_k = (S^{-1}D)^{2h}\ \Gamma_{k-1}, \quad \Gamma_0 = (S^{-1}D)^{2h} = I - T_0\ A \tag{34}$$

$$T_k = T_{k-1} + \Gamma_{k-1}\ T_0 = \left\{ \sum_{j=0}^{k} \Gamma_0^j \right\}\ T_0 \tag{35}$$

where Γ_k in (34) is recursive realization of $\Gamma_k = (S^{-1}D)^{2h(k+1)}$, and T_0 is calculated via (33). Equation (35) is the recursive realization of T_k and similar to (15) it requires one matrix addition and one multiplication only in each step, $k = 1, 2, 3, \ldots$.

Notice that T_k and Γ_k can also be calculated as follows:

$$T_k = T_0 + \Gamma_0\ T_{k-1} \tag{36}$$

$$\Gamma_k = I - T_k\ A \tag{37}$$

which is more robust in finite-digit calculations (in some cases) compared to (34), (35).

Notice also that the computational complexity of the recursive realization T_k does not depend on the order h, but requires order dependent precalculations, which are also used as initial condition G_0 in (17) and calculated only once.

Notice also that all the terms in Part I can be precalculated and memorized as a function of step number, if the number of steps is predetermined. The desired number of steps can be calculated using error models (31) and (32) with desired estimation accuracy. Moreover, for reduction of error accumulation in main calculations the precalculations can be performed with higher accuracy.

4.2 Recursive calculations for part II

Computational complexity of the second part can be reduced via recursive algorithm as follows:

$$\begin{aligned} R_0 &= C_1^{n-1}\ I + C_0^{n-1}\ (G_{k-1}A) \\ &\quad for\ i = 1 : (n-2) \\ R_i &= (G_{k-1}A)\ R_{i-1} + C_{i+1}^{n-1}\ I \\ &\quad end \end{aligned} \tag{38}$$

where $n > 2$ which corresponds to the following factorization of the Neumann series:

$$\begin{aligned} &\sum_{d=0}^{n-1} F_{k-1}^d = C_{n-1}^{n-1} + (G_{k-1}A)\ (C_{n-2}^{n-1}I + (G_{k-1}A) \\ &(C_{n-3}^{n-1}I + (G_{k-1}A)\ (C_{n-4}^{n-1}I\ \ + \ldots + \\ &(G_{k-1}A)\ \underbrace{(C_1^{n-1}I + C_0^{n-1}(G_{k-1}A)}_{= R_0}))))) \end{aligned} \tag{39}$$

Table 2 Neumann series factorization

Order n	Neumann series factorization $\sum_{d=0}^{n-1} F_{k-1}^{d},\ F_k = I - (G_{k-1}A)$
2	$2I - (G_{k-1}A)$
3	$3I + (G_{k-1}A)(-3I + (G_{k-1}A))$
4	$4I + (G_{k-1}A)(-6I + (G_{k-1}A)(4I - (G_{k-1}A))$
5	$5I + (G_{k-1}A)(-10I + (G_{k-1}A)(10I + (G_{k-1}A)(-5I + (G_{k-1}A))$
...	...
n	$C_{n-1}^{n-1} + (G_{k-1}A)\ (C_{n-2}^{n-1}I + (G_{k-1}A)\ (C_{n-3}^{n-1}I + (G_{k-1}A)\ (C_{n-4}^{n-1}I\ + ... +$ $(G_{k-1}A)\ (C_{1}^{n-1}I + C_{0}^{n-1}(G_{k-1}A)))))$ $C_d^{n-1} = \frac{n!}{d!\,(n-d)!}(-1)^{n-d-1},\ d = 0, \dots, (n-1)$

where $C_d^{n-1} = \dfrac{n!}{d!\,(n-d)!}(-1)^{n-d-1}$ are the coefficients, and $d = 0, \dots, (n-1)$, $n = 2, 3, 4, \dots$. Factorization (39) requires $(n-1)$ matrix multiplications and additions of precalculated diagonal matrices in each step. Notice that addition of the diagonal matrices is computationally simpler operation than addition of conventional matrices.

The factorization of Neumann series (39) is presented in Table 2 for different orders. Notice that the number of matrix multiplications can be reduced further for some orders, see for example [11,26,28,34] and references therein.

4.3 Recursive realization of the matrix inversion algorithm (29)

Computationally efficient version of the matrix inversion algorithm (29) can be presented for simultaneous computations as follows:

$$\Gamma_k = (S^{-1}D)^{2h}\ \Gamma_{k-1}, \quad \Gamma_0 = (S^{-1}D)^{2h} \tag{40}$$
$$T_k = T_{k-1} + \Gamma_{k-1}\ T_0 \tag{41}$$
$$for\ i = 1 : (n-2)$$
$$R_i = (G_{k-1}A)\ R_{i-1} + C_{i+1}^{n-1}\ I \tag{42}$$
$$end$$
$$G_k = \underbrace{T_k}_{\text{Power Series Expansion}} + \underbrace{\Gamma_k}_{\text{Gain Matrix}} \underbrace{R_i\ G_{k-1}}_{\text{Newton-Schulz Iteration}} \tag{43}$$

with corresponding initial conditions. Recursive algorithm consists of two parts, (40), (41) and (42) which can be calculated simultaneously, and the common part (43).

Notice that the factorization $T_k = \left\{\sum_{j=0}^{k} \Gamma_0^j\right\} T_0$ in (43) is rapidly expanding power series, and the gain matrix $\Gamma_k = (S^{-1}D)^{2h(k+1)}$ determines the position (the bias) of Newton-Schulz iteration R_i in the resulting power series.

The orders h and n should not be high to prevent error accumulation, and h should be chosen higher than (or equal to) n since the number of operations in (40), (41) does not depend on h and error accumulation problem is more pronounced for Neumann series [6,10].

Notice that computational complexity of recursive calculation of Neumann series $\sum_{d=0}^{n-1} F_{k-1}^d$ is relatively low, especially for low orders. This series can be calculated in different ways, depending on the order and the way realized in (42) requires $(n-1)$, $(n \geq 2)$ matrix multiplications and additions of the diagonal matrices in each step. Therefore the recursive realization of (10) requires one additional multiplication.

The computational complexity and time of the algorithm (10) can be taken as the reference for evaluation of the performance of the algorithm (40)–(43). Therefore the additional computational complexity of the algorithm (40)–(43) compared to the computational complexity of (10) is three matrix multiplications, two additions (independent of the order) in each step and precalculations. The computational time is almost the same for both algorithms provided that two parts mentioned above are calculated simultaneously. This increase in computational burden in the algorithm (40)–(43) is a reasonable computational price for significant convergence rate improvement.

Notice that the recursive realizations can also be unified within the framework (19)–(22). Unified recursive realizations of the algorithms 5–7 from the Table 1 are presented in Table 3.

5 Combined Richardson iteration and matrix inversion algorithms

Consider Richardson algorithm (3) with $q = 1$ for estimation of the parameter vector θ_*, which satisfies algebraic equation (1). Then the following error models are valid for the parameter mismatch $\tilde{\theta}_k = \theta_k - \theta_*$:

(1) for G_k which satisfies (15)

$$\tilde{\theta}_k = (S^{-1}D)^{2h(k+1)}\tilde{\theta}_{k-1} \tag{44}$$

$$\tilde{\theta}_k = (S^{-1}D)^{h\,k\,(k+3)}\tilde{\theta}_0 \tag{45}$$

(2) for G_k which satisfies (29) with $n = 1$

$$\tilde{\theta}_k = (S^{-1}D)^{2h\left\{k\,\frac{(k+3)}{2}+1\right\}}\tilde{\theta}_{k-1} \tag{46}$$

$$\tilde{\theta}_k = (S^{-1}D)^{h\left\{\frac{k(k+1)(2k+10)}{6}+2k\right\}}\tilde{\theta}_0 \tag{47}$$

(3) for G_k which satisfies (29) with $n > 1$

$$\tilde{\theta}_k = F_k\,\tilde{\theta}_{k-1} \tag{48}$$

Table 3 Unified recursive realizations of matrix inversion algorithms

Unified matrix inversion algorithms	Unified recursive realizations
$G_k = T_k + \Gamma_k \left\{\sum_{d=0}^{n-1} F_{k-1}^d\right\} G_{k-1},\ \Gamma_k = I - T_k\ A$	$G_k = T_k + \Gamma_k\ R_i\ G_{k-1}$
	R_i is calculated in (38)
$T_k = \left\{\sum_{j=0}^{2h(k+1)-1} (S^{-1}D)^j\right\} S^{-1}, \Gamma_k = (S^{-1}D)^{2h(k+1)}$	$T_0 = G_0 = \left\{\sum_{j=0}^{h-1} (S^{-1}D)^{2j}\right\} \{(S^{-1}D) + I\} S^{-1}$ is calculated in (33)
$h = 1, 2, 3, \ldots,\ n = 1, 2, 3, \ldots$	$\Gamma_0 = I - T_0\ A$
	$T_k = T_0 + \Gamma_0\ T_{k-1}$
	$\Gamma_k = I - T_k\ A$
$T_k = \left\{\sum_{j=0}^{m^k-1} (S^{-1}D)^j\right\} S^{-1},\ \ \Gamma_k = (S^{-1}D)^{m^k}$	$T_0 = G_0 = S^{-1},\ \Gamma_0 = I - T_0\ A$
$n = 1, 2, 3, \ldots,\ m = 1, 2, 3, \ldots$	$H_i = \Gamma_{k-1}(H_{i-1} + I)$, for $i = 1, \ldots (m-2),\ H_0 = \Gamma_{k-1},\ m > 2$
	$T_k = T_{k-1} + H_i\ T_{k-1}$
	$\Gamma_k = I - T_k\ A$
$T_k = \left\{\sum_{j=0}^{h2^k-1} (S^{-1}D)^j\right\} S^{-1},\ \ \Gamma_k = (S^{-1}D)^{h2^k}$	$T_0 = G_0 = \left\{\sum_{j=0}^{h-1} (S^{-1}D)^j\right\} S^{-1},\ \ \Gamma_0 = I - T_0\ A$
$h = 1, 2, 3, \ldots,\ n = 1$	$T_k = T_{k-1} + \Gamma_{k-1}\ T_{k-1}$
	$\Gamma_k = I - T_k\ A$

Table 4 A family of Richardson algorithms

Richardson iteration
$\theta_k = \theta_{k-1} - G_k\,(A\theta_{k-1} - b),\ \theta_0 = G_0\, b,\ A\,\theta_* = b,\quad \tilde{\theta}_k = \theta_k - \theta_*,\ A = S - D,\ \rho(S^{-1}D) < 1,$
$\theta_k \to \theta_*$ as $k \to \infty$

N	Gain update algorithm G_k and error model
1	$G_k = G_0 + (S^{-1}D)^{2h} G_{k-1}, G_0 = \left\{\sum_{j=0}^{h-1}(S^{-1}D)^{2j}\right\}\{(S^{-1}D) + I\}S^{-1},$ $\tilde{\theta}_k = (S^{-1}D)^{2h(k+1)}\tilde{\theta}_{k-1},\ \tilde{\theta}_k = (S^{-1}D)^{h\,k\,(k+3)}\tilde{\theta}_0,\ h = 1, 2, 3, \ldots$
2	$G_k = \left\{\sum_{j=0}^{2h(k+1)-1}(S^{-1}D)^j\right\} S^{-1} + (S^{-1}D)^{2h(k+1)} G_{k-1},$ $G_0 = \left\{\sum_{j=0}^{h-1}(S^{-1}D)^{2j}\right\}\{(S^{-1}D) + I\}S^{-1}$ $\tilde{\theta}_k = (S^{-1}D)^{2h\left\{k\frac{(k+3)}{2}+1\right\}}\tilde{\theta}_{k-1},$ $\tilde{\theta}_k = (S^{-1}D)^{h\left\{\frac{k(k+1)(2k+10)}{6}+2k\right\}}\tilde{\theta}_0,\ h = 1, 2, 3, \ldots$
3	$G_k = \left\{\sum_{j=0}^{2h(k+1)-1}(S^{-1}D)^j\right\} S^{-1} + (S^{-1}D)^{2h(k+1)}\left\{\sum_{d=0}^{n-1}F_{k-1}^d\right\} G_{k-1},$ $F_k = I - G_k A$ $G_0 = \left\{\sum_{j=0}^{h-1}(S^{-1}D)^{2j}\right\}\{(S^{-1}D) + I\}S^{-1}$ $\tilde{\theta}_k = (S^{-1}D)^{2h\left\{\frac{n^{k+2} - n^3 - (k-1)(n-1)}{(n-1)^2} + (n+2)\right\}}\tilde{\theta}_{k-1}$ $\tilde{\theta}_k = (S^{-1}D)^{2h\left\{\frac{n^{k+3}-n^4}{(n-1)^3} - (k-1)\left\{\frac{n^3}{(n-1)^2} + \frac{k}{2(n-1)}\right\} + k(n+2)\right\}}\tilde{\theta}_0$ $n > 1,\ h = 1, 2, 3, \ldots$

where F_k is given in (32) and

$$\tilde{\theta}_k = (S^{-1}D)^{2h\,\gamma}\,\tilde{\theta}_0 \tag{49}$$

where $\gamma = \dfrac{n^{k+3} - n^4}{(n-1)^3} - (k-1)\left\{\dfrac{n^3}{(n-1)^2} + \dfrac{k}{2(n-1)}\right\} + k(n+2)$ and $k = 1, 2, 3, \ldots$.

The family of Richardson algorithms are summarized in Table 4 for selected matrix inversion algorithms presented in Table 1.

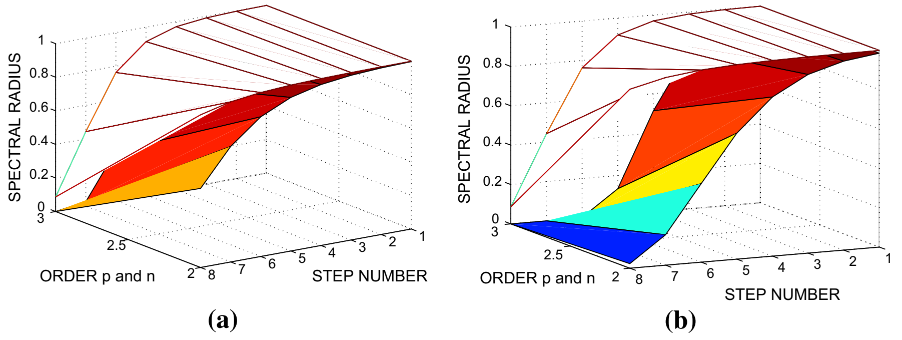


Fig. 1 Subplots show the spectral radius ρ^{p^k} for algorithm (10) (plotted as white surface) and $\rho^{2h\left\{\frac{n^{k+2}-n^3-(k-1)(n-1)}{(n-1)^2}+(n+2)\right\}}$ for algorithm (32) (plotted as colored surface) as a function of orders p and n (on the same axis) and step number k. The order $h = 1$ corresponds to Subplot **a**, and Subplot **b** shows the spectral radius for $h = 5$. (Color figure online)

6 Performance of the algorithms and comparisons

6.1 Comparison of matrix inversion algorithms (10) and (15)

The convergence rate of two matrix inversion algorithms (10) and (15) is compared via error models (11) and (16), where the former has the order p^k, $p = 2, 3 \ldots$ and the latter has the order $2hk$, $h = 1, 2, \ldots$ (where the order is associated with the most significant term in the power of the matrix $(S^{-1}D)$). Algorithm (10) converges much faster than algorithm (15). However, when these algorithms are applied in Richardsson iteration (3), which has the integration property the orders of convergence for parameter mismatch $p^{k+1}/(p-1)$ and hk^2 [see (14) and (45)] become comparable, but the method (10) is still faster especially for higher orders.

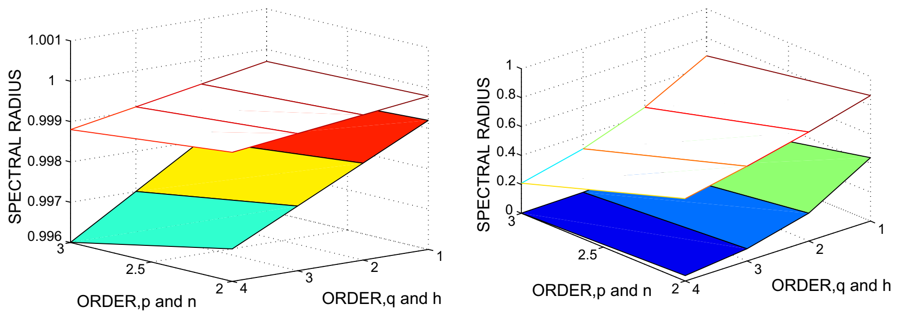
6.2 Comparison of matrix inversion algorithms (10) and (29)

The convergence rate of the algorithm (10) is compared to the convergence rate of the algorithm (29) via error models (11) and (32) for the same spectral radius ρ (which is close to one for ill-conditioned matrix A) as the function of step number k in Fig. 1a for $h = 1$ and in Fig. 1b for $h = 5$. The Fig. 1 shows that proposed matrix inversion algorithm (29) (plotted as colored surface) converges much faster than high order Newton-Schulz algorithm (10) (plotted as white surface), especially for higher order h, which essentially accelerates convergence.

Notice that the algorithm (29) has two orders n and h as parameters to be chosen, whereas algorithm (10) has only one order p as a parameter.

6.3 Comparison of Richardson algorithms (3) with G_k (10) and (29) as functions of four orders

Richardson iteration (3) with G_k defined in (10) has two orders $q = 1, 2, \ldots$ and $p = 2, 3, \ldots$ as parameters. The same iteration with $q = 1$ and G_k defined in (29) has



(a) (b)

Fig. 2 Subplots show the spectral radius $\rho^{\frac{q\,(p^{k+1}-p)}{(p-1)}}$ for Richardson iteration (3) with G_k defined in (10) (plotted as white surface) and the same iteration with $q = 1$ and G_k defined in (29) for $\rho^{2h\left\{\frac{n^{k+3}-n^4}{(n-1)^3}-(k-1)\left\{\frac{n^3}{(n-1)^2}+\frac{k}{2(n-1)}\right\}+k(n+2)\right\}}$ (plotted as colored surface) as a function of orders q and h and p and n. Subplot **a** corresponds to $k = 1$ and Subplot **b** shows the spectral radius for $k = 3$. (Color figure online)

also two orders $h = 1, 2, \ldots$, which is the counterpart of q and $n = 2, 3, \ldots$, which is counterpart of p as parameters. Therefore convergence of the algorithms can be compared via error models (14) and (49) as the function of orders for different steps. The Fig. 2 shows that Richardson iteration (3) with $q = 1$ and G_k defined in (29) (plotted as colored surface) converges much faster than algorithm (3) with G_k defined in (10) (plotted as white surface), especially for larger number of steps k.

6.4 Integration property of Richardson iteration and convergence rate improvement

In addition to robustness and accuracy improvement Richardson iteration (3) improves also the convergence rate compared to parameter estimation using recursive matrix inversion techniques only. Richardson iteration plays a role of integration loop for recursive matrix inversion algorithms and increases the order of convergence by one, which can be seen by comparison of the following pairs of the error models: (11), (14) and (16),(45) and (31), (47) and (32), (49), see also Table 4.

Notice that the same integration effect is present in the additional loop of modified high order Newton-Schulz algorithm proposed in [35]. Combination of high order Newton-Schulz and Richardson framework is preferable for more accurate and robust parameter calculation with improved convergence rate.

7 Conclusion

Increasing order of Newton-Schulz matrix inversion algorithm, which is a conventional way for convergence rate improvement for solving the system of linear equations results in error accumulation and the loss of accuracy in finite-digit calculations. Algorithm design for convergence rate improvement is considered in this paper as the trade-off achievement between convergence rate, computational complexity and robustness. The approach resulted in development of two unified frameworks for matrix inversion algorithms and for Richardson iteration, where the trade-off between convergence rate improvement and computational complexity is quantified using explicit transient error models, factorizations and recursive representations. The approach allows significant improvement of the convergence rate of estimated parameters with low orders, reduced error accumulation and efficient simultaneous computations.

References

1. Ljung, L.: System Identification: Theory for the User. Prentice-Hall, Upper Saddle River (1999)
2. Bayard, D.: A general theory of linear time-invariant adaptive feedforward systems with harmonic regressors. IEEE Trans. Autom. Control **45**(11), 1983–1996 (2000)
3. Stotsky, A.: Recursive trigonometric interpolation algorithms. J. Syst. Control Eng. **224**(1), 65–77 (2010)
4. Stotsky, A.: Harmonic regressor: robust solution to least-squares problem. Proc. IMechE Part I: J. Syt. Control Eng. **227**(8), 662–668 (2013)
5. Stotsky, A.: Towards accurate estimation of fast varying frequency in future electricity networks: the transition from model-free methods to model-based approach. J. Syst. Control Eng. **230**(10), 1164–1175 (2016)
6. Stotsky, A.: Grid frequency estimation using multiple model with harmonic regressor: robustness enhancement with stepwise splitting method. In: Conference Paper Archive, IFAC PapersOnLine 50-1, pp. 12817–12822, Elsevier (2017)
7. Chao, B. T, Li, H. L, Scott, E. J.: On the solution of ill-conditioned, simultaneous, linear, algebraic equations by machnine computation. In: Alan Kingery, R. (ed.) Engineering Experiment Station Bulletin No. 459, University of Illinois Bulletin (2007)
8. Brussino, G., Sonnad, V.: A comparison of direct and preconditioned iterative techniques for sparse, unsymmetric systems of linear equations. Numer. Methods Eng. **28**(4), 801–815 (1989)
9. Stotsky, A.: Accuracy improvement in least-squares estimation with harmonic regressor: new preconditioning and correction methods. In: 54-th CDC, Dec. 15–18, Osaka, Japan, pp. 4035–4040 (2015)
10. Isaacson, E., Keller, H.: Analysis of Numerical Methods. Wiley, New York (1966)
11. Stickel, E.: On a class of high order methods for inverting matrices. ZAMM Z. Angew. Math. Mech. **67**, 331–386 (1987)
12. Soleymani, F.: A new method for solving ill-conditioned linear systems. Opusc. Math. **33**(2), 337–344 (2013)
13. Missirlis, N.: A parallel iterative system solver. Linear Algebra Appl. **65**, 25–44 (1985)
14. Richardson, L.: The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a Masonry Dam. Philos. Trans. R. Soc. A **210**, 307–357 (1910)
15. Dubois, D., Greenbaum, A., Rodrigue, G.: Approximating the inverse of a matrix for use in iterative algorithms on vector processors. Computing **22**, 257–268 (1979)
16. Stotsky, A.: Combined high-order algorithms in robust least-squares estimation with harmonic regressor and Strictly Diagonally Dominant Information Matrix. Proc. IMechE Part I: J. Syst. Control Eng. **229**(2), 184–190 (2015)
17. Chen, K.: Matrix Preconditioning Techniques and Applications. Cambridge University Press, Cambridge (2005)
18. O'Leary, D.: Yet another polynomial preconditioner for the conjugate gradient algorithm. Linear Algebra Appl. **154–156**, 377–388 (1991)
19. Brezinski, C.: Variations on Richardson's method and acceleration. In: Numerical Analysis, A Numerical Analysis Conference in Honour of Jean Meinguet, Bull. Soc. Math. Belgium, pp. 33–44 (1996)
20. Horn, R., Johnson, C.: Matrix Analysis. Cambridge University Press, Cambridge (1985)
21. Hackbusch, W.: Iterative Solution of Large Sparse Systems of Equations. Springer Verlag, Heidelberg (1994)

22. Benzi, M.: Preconditioning techniques for large linear systems: a survey. J. Comput. Phys. **182**, 418–477 (2002)
23. O'Leary, D., White, R.: Multi-splittings of matrices and parallel solution of linear systems. SIAM J. Algebraic Discrete Methods **6**(4), 630–640 (1985)
24. Hadjidimos, A.A., Yeyios, A.: On multisplitting methods and m-step preconditioners for parallel and vector machines. In: Computer Science Technical Reports, Purdue University, Purdue e-Pubs, Report Number 92-060, Paper 981 (1992)
25. Ferronato, M.: Preconditioning for sparse linear systems at the dawn of the 21st century: history, current developments, and future perspectives. Int. Sch. Res. Netw. ISRN Appl. Math. **2012**, Article ID 127647, 49 pages. https://doi.org/10.5402/2012/127647
26. Pan, V., Soleymani, F., Zhao, L.: Highly efficient computation of generalized inverse of a matrix. arXiv:1604.07893v1 [math.RA] (2016)
27. Li, W., Li, Z.: A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix. Appl. Math. Comput. **215**(9), 3433–3442 (2010)
28. Chen, H., Wang, Y.: A family of higher-order convergent iterative methods for computing the Moore–Penrose inverse. Appl. Math. Comput. **218**, 4012–4016 (2011)
29. Climent, J., Thome, N., Wei, Y.: A geometrical approach on generalized inverses by Neumann-type series. Linear Algebra Appl. **332–334**, 533–540 (2001)
30. Schulz, G.: Iterative Berechnung Der Reziproken Matrix. Z. Angew. Math. Mech. **13**, 57–59 (1933)
31. Demidovich, B., Maron, I.: Basics of Numerical Mathematics. Fizmatgiz, Moscow (1963). (in Russian)
32. Peng, R., Spielman, D.: An efficient parallel solver for SDD linear systems, arXiv:1311.3286v1 [cs.NA], (2013)
33. Janiszowski, K.: Inversion of square matrices in processors with limited calculation abillities. Int. J. Appl. Math. Comput. Sci. AMSC **13**(2), 199–204 (2003)
34. Esmaeili, H., Erfanifar, R., Rashidi, M.: A fourth-order iterative inverse for computing the Moore–Penrose inverse. J. Hyperstructures **1**(6), 52–67 (2017)
35. Srivastava, S., Gupta, D.: A higher order iterative method for $A^{(2)}_{T,S}$. J. Appl. Math. Comput. **46**(1/2), 147–168 (2014)