Algorithms **2015**, *8*, 982-998; doi:10.3390/a8040982

OPEN ACCESS

algorithms

ISSN 1999-4893

www.mdpi.com/journal/algorithms

Article

Some Matrix Iterations for Computing Generalized Inverses and Balancing Chemical Equations

Farahnaz Soleimani [1,*], Predrag S. Stanimirović [2] and Fazlollah Soleymani [3]

[1] Department of Chemistry, Roudehen Branch, Islamic Azad University, 39731 Roudehen, Iran

[2] Faculty of Sciences and Mathematics, University of Niš, Višegradska 33, 18000 Niš, Serbia; E-Mail: pecko@pmf.ni.ac.rs

[3] Department of Applied Mathematics, Ferdowsi University of Mashhad, 91779 Mashhad, Iran; E-Mail: fazlollah.soleymani@gmail.com

* Author to whom correspondence should be addressed; E-Mail: fz_soleimani@yahoo.com; Tel.: +98-912-348-2263.

Academic Editors: Alicia Cordero, Juan R. Torregrosa and Francisco I. Chicharro

Received: 25 June 2015 / Accepted: 26 October 2015 / Published: 3 November 2015

Abstract: An application of iterative methods for computing the Moore–Penrose inverse in balancing chemical equations is considered. With the aim to illustrate proposed algorithms, an improved high order hyper-power matrix iterative method for computing generalized inverses is introduced and applied. The improvements of the hyper-power iterative scheme are based on its proper factorization, as well as on the possibility to accelerate the iterations in the initial phase of the convergence. Although the effectiveness of our approach is confirmed on the basis of the theoretical point of view, some numerical comparisons in balancing chemical equations, as well as on randomly-generated matrices are furnished.

Keywords: generalized inverses; balancing chemical equations; hyper-power method; order of convergence; matrix inverse

---

1. Introduction

A chemical equation is only a symbolic representation of a chemical reaction and represents an expression of atoms, elements, compounds or ions. Such expressions are generated based on balancing through reactant or product coefficients, as well as through reactant or product molar masses [1].

In fact, equilibrating the equations that represent the stoichiometry of a reacting system is a matter of mathematics, since it can be reduced to the problem of solving homogeneous linear systems.

Balancing chemical equations is an important application of generalized inverses. To discuss further, the reflexive g-inverse of a matrix has been successfully used in solving a general problem of balancing chemical equations (see [2,3]). Continuing in the same direction, Krishnamurthy in [4] gave a mathematical method for balancing chemical equations founded by virtue of a generalized matrix inverse. The method used in [4] is based on the exact computation of reflexive generalized inverses by means of elementary matrix transformations and the finite-field residue arithmetic, as it was described in [5].

It is well known that the symbolic data processing, including both rational arithmetic and multiple modulus residue arithmetic, are time consuming, both for the implementation and for execution. On the other hand, the finite-field exact arithmetic is inapplicable to chemical reactions, which include atoms with fractional and/or integer oxidation numbers. This hard class of chemical reactions is investigated in [6].

Additionally, the balancing chemical equations problem can be readily resolved by computer algebra software, as was discussed in [7]. The approach used in [7] is based on the usage of the Gaussian elimination (also known as the Gauss–Jordan algorithm), while the approach that is exploited in [2] is based on singular value decomposition (SVD). On the other hand, it is widely known that Gaussian elimination, as well as SVD require a large amount of numerical operations [8]. Furthermore, small pivots that could appear in Gaussian elimination can lead to large multipliers [9], which can sometimes lead to the divergence of numerical algorithms. Two methods of balancing chemical equations, introduced in [10], are based on integer linear programming and integer nonlinear programming models, respectively. Notice that the linear Diophantine matrix method was proposed in [11]. The method is applicable in cases when the reaction matrices lead to infinite stoichiometrically-independent solutions.

In the present paper, we consider the possibility of applying a new higher order iterative method for computing the Moore–Penrose inverse in the problem of balancing chemical equations. In general, our current research represents the first attempt to apply iterative methods in balancing chemical reactions.

A rapid numerical algorithm for computing matrix-generalized inverses with a prescribed range and null space is developed in order to implement this global idea. The method is based on an appropriate modification of the hyper-power iterative method. Furthermore, some techniques for the acceleration of the method in the initial phase of its convergence is discussed. We also try to show the applicability of proposed iterative schemes in balancing chemical equations. Before a more detailed discussion, we briefly review some of the important backgrounds incorporated in our work.

The outer inverse with prescribed range T and null space S of a matrix $A \in \mathbb{C}_r^{m\times n}$, denoted by $A^{(2)}_{T,S}$, satisfies the second Penrose matrix equation $XAX = X$ and two additional properties: $\mathcal{R}(X) = T$ and $\mathcal{N}(X) = S$. The significance of these inverses is reflected primarily in the fact that the most important generalized inverses are particular cases of outer inverses with a prescribed range and null space. For example, the Moore–Penrose inverse $A^\dagger$, the weighted Moore–Penrose inverse $A^\dagger_{M,N}$, the

Drazin inverse A^D and the group inverse $A^{\#}$ can be derived by means of the appropriate choices of subspaces T and S in what follows:

$$\begin{array}{llll} A^{\dagger} = A^{(2)}_{\mathcal{R}(A^*),\mathcal{N}(A^*)}, & A^{\dagger}_{M,N} = A^{(2)}_{\mathcal{R}(A^{\sharp}),\mathcal{N}(A^{\sharp})} & & \\ A^{\mathrm{D}} = A^{(2)}_{\mathcal{R}(A^k),\mathcal{N}(A^k)}, & k \geq \mathrm{ind}(A), & A^{\#} = A^{(2)}_{\mathcal{R}(A),\mathcal{N}(A)}, & \mathrm{ind}(A) = 1 \end{array} \tag{1}$$

wherein $A^{\sharp} = MAN^{-1}$ and $\mathrm{ind}(A)$ denotes the index of a square matrix A (see, e.g., [12]).

Although there are many approaches to calculate these inverses by means of direct methods, an alternative and very important approach is to use iterative methods. Among many such matrix iterative methods, the hyper-power iterative family has been introduced and investigated (see, for example, [13–15]). The hyper-power iteration of the order p is defined by the following scheme (see, for example, [13]):

$$X_{k+1} = X_k(I + R_k + \cdots + R_k^{p-1}) = X_k \sum_{i=0}^{p-1} R_k^i, \quad R_k = I - AX_k \tag{2}$$

The iteration Equation (2) requires p matrix-matrix multiplications (from now on denoted by mmm) to achieve the p-th order of convergence. The adoption $p = 2$ yields to the Schulz matrix iteration, originated in [16],

$$X_{k+1} = X_k(2I - AX_k) \tag{3}$$

with the second rate of convergence. Further, choice $p = 3$ gives the cubically-convergent method of Chebyshev [17], defined as follows:

$$X_{k+1} = X_k(3I - AX_k(3I - AX_k)) \tag{4}$$

For more details about the background of iterative methods for computing generalized inverses, please refer to [18].

The main motivation of the paper [19] was the observation that the inverse of the reaction matrix cannot always be obtained. For this purpose, the author used an approach based on row-reduced echelon forms of both the reaction matrix and its transpose. Since the Moore–Penrose inverse always exists, replacement of the ordinary inverse by the corresponding pseudoinverse resolves the drawback that the inverse of the reaction matrix does not always exist. Furthermore, two successive transformations into the corresponding row-reduced echelon forms are time-consuming and badly-conditioned numerical processes, again based on Gaussian elimination. Our intention is to avoid the above-mentioned drawbacks that appear in previously-used approaches.

Here, we decide to develop an application of the Schulz-type methods. The motivation is based on the following advantages arising from the use of these methods. Firstly, they are totally applicable on sparse matrices possessing sparse inverses. Secondly, the Schulz-type methods are useful for providing approximate inverse preconditions. Thirdly, such schemes are parallelizable, while Gaussian elimination with partial pivoting is not suitable for the parallelism.

It is worth mentioning that an application of iterative methods in finding exact solutions that involve integers or rational entries requires an additional transformation of the solution and utilization of tools for symbolic data processing. To this end, we used the programming package Mathematica.

The rest of this paper is organized as follows. A new formulation of a very high order method is presented in Section 2. The method is fast and economical at the same time, which is confirmed by the fact that it attains a very high rate of convergence by using a relatively small number of mmm. Acceleration of the convergence via scaling the initial iterates is discussed, and some novel approaches in this trend are given in the same section. An application of iterative methods in balancing chemical equations is considered in Section 3. A comparison of numerical results obtained by applying the introduced method is shown against the results defined by using several similar methods. Some numerical experiments concerning the application of the new iterations in balancing chemical equations are presented in Section 4. Finally, some concluding remarks are drawn in the last section.

2. An Efficient Method and Its Acceleration

A Schulz-type method of high order $p = 31$ with two improvements is derived and chosen as one of the options for balancing chemical equations. The first improvement is based on a proper factorization, which reduces the number of mmm required in each cycle. The second straightening is based on a proper accelerating of initial iterations.

Toward this goal, we consider Equation (2) in the case $p = 31$ as follows:

$$X_{k+1} = X_k \sum_{i=0}^{30} R_k^i, \quad R_k = I - AX_k \tag{5}$$

In its original form, the hyper-power iteration Equation (5) is of the order 31 and requires 31 mmm. It is necessary to remark that a kind of effectiveness of a computational iterative (fixed point-type) method can be estimated by the real number (called the computational efficiency index) $EI = p^{\frac{1}{\theta}}$, wherein θ and p stand for the whole computational cost and the rate of convergence per cycle, respectively. Here, the most important burden and cost per cycle is the number of matrix-matrix products.

Clearly, in Equation (5), this proportion between the order of convergence and the needed number of mmm is not suitable, since its efficiency index:

$$EI = 31^{\frac{1}{31}} \approx 1.1171 \tag{6}$$

is relatively small. This shows that Equation (5) is not a useful iterative method. To improve the applicability of Equation (5) and, so, to derive a fast matrix iteration with a reduced number of mmm, *i.e.*, to obtain an efficient method, we keep going as in the following subsection.

2.1. An Efficient Method

We rewrite Equation (5) as:

$$\begin{aligned} X_{k+1} = X_k(I + R_k(I + R_k)(I - R_k + R_k^2)(I + R_k + R_k^2)(I - R_k + R_k^2 - R_k^3 + R_k^4) \\ \times (I + R_k + R_k^2 + R_k^3 + R_k^4)(I - R_k + R_k^3 - R_k^4 + R_k^5 - R_k^7 + R_k^8) \\ \times (I + R_k - R_k^3 - R_k^4 - R_k^5 + R_k^7 + R_k^8)) \end{aligned} \tag{7}$$

Subsequently, Equation (7) results in the following formulation of Equation (5):

$$X_{k+1} = X_k \left(I + (R_k + R_k^2 + R_k^3 + R_k^4 + R_k^5 + R_k^6)\left(I + R_k^6 + R_k^{12} + R_k^{18} + R_k^{24}\right)\right) \tag{8}$$

Now, we could deduce our final fast matrix iteration by simplifying Equation (8) more:

$$X_{k+1} = X_k \left(I + (R_k + R_k^2)(I + R_k^2 + R_k^4)\left(I + (R_k^2 + R_k^8)(R_k^4 + R_k^{16})\right)\right) \tag{9}$$

where only nine mmm are required. Therefore, the efficiency index of the proposed fast iterative method becomes:

$$EI = 31^{\frac{1}{9}} \approx 1.4645 \tag{10}$$

The efficiency index 1.4645 of Equation (9) is higher than the efficiency index 1.4142 of Equation (3), higher than the efficiency index 1.4422 of Equation (4) and finally higher than the efficiency index 1.4592 of the 30th order method proposed recently by Sharifi *et al.* [20].

At this point, it would be useful to provide the following theorem regarding the convergence behavior of Equation (9).

Theorem 1. *Let $A \in \mathbb{C}_r^{m\times n}$ be a given matrix of rank r and $G \in \mathbb{C}_s^{n\times m}$ be a given matrix of rank $0 < s \leq r$, which satisfy $\mathrm{rank}(GA) = \mathrm{rank}(G)$. Then, the sequence $\{X_k\}_{k=0}^{k=\infty}$ generated by the iterative method Equation (9) converges to $A^{(2)}_{R(G),N(G)}$ with 31st-order if the initial approximation $X_0 = \alpha G$ satisfies:*

$$\|F_0\| = \|AA^{(2)}_{T,S} - AX_0\| < 1 \tag{11}$$

Proof. The proof of this theorem would be similar to the ones in [21]. Hence, we skip it over and just include the following error bound:

$$\|E_{k+1}\| \leq \|A^{(2)}_{R(G),N(G)}\| \, \|A\|^{31} \, \|E_k\|^{31} \tag{12}$$

wherein $E_k = A^{(2)}_{R(G),N(G)} - X_k$. □

The derived iterative method is very fast and effective, in contrast to the existing iterative Schulz-type methods of the same type. However, as was pointed out by Soleimani *et al.* [22], such iterative methods are slow at the beginning of the iterative process, and the real convergence rate cannot be observed. An idea to remedy this disadvantage is to apply a multiple root-finding algorithm on the matrix equation $F(X) = X^{-1} - A = 0$ and to try to accelerate the hyper-power method in its initial iterations. Such a discussion about a scaled version of the hyper-power method is the main aim of the next subsection.

2.2. Accelerating the Initial Phase

Another useful motivation of the present paper is processed here. The iterative scheme:

$$X_{k+1} = X_k \left((\beta + 1)I - \beta AX_k\right), \; 1 \leq \beta \leq 2 \tag{13}$$

was applied in [22] to achieve the convergence phase in the main (modified Householder) method much more rapidly and to accelerate the beginning of the process. In the second iteration phase, it is sufficient

to apply the introduced fast and efficient modified Householder method, which then reaches its own full speed of convergence [22].

In the same vein, the iterative expression Equation (13) can be rewritten in the following equivalent, but more practical form:

$$X_{k+1} = X_k \left((1-\beta)I + \beta \sum_{i=0}^{1} R_k^i \right), \quad R_k = I - AX_k, \ 1 \le \beta \le 2 \tag{14}$$

One can now observe that Equation (14) is the particular case ($p = 2$) of the following new scheme:

$$\begin{aligned} X_{k+1} &= X_k \left(I + \beta \left((p-1)I + \sum_{i=1}^{p-1} (-1)^i \binom{p}{i+1} (AX_k)^i \right) \right) \\ &= X_k \left((1-\beta)I + \beta \left(\sum_{i=0}^{p-1} (-1)^i \binom{p}{i+1} (AX_k)^i \right) \right) \\ &= X_k \left((1-\beta)I + \beta \sum_{i=0}^{p-1} R_k^i \right), \quad R_k = I - AX_k, \ 1 \le \beta \le 2 \end{aligned} \tag{15}$$

Therefore, Equation (15) can be considered as an important acceleration of the Schulz-type method Equation (2) in the initial phase, before the convergence rate is practically achievable. We note that such accelerations are useful for large matrices, whereas the iterative methods require too much iterations to converge. Particularly, by following Equation (15), one can immediately notice that the accelerating in the initial phase of iteration Equation (9) is of the form:

$$X_{k+1} = X_k \left((1-\beta)I + \beta \left(I + (R_k + R_k^2)(I + R_k^2 + R_k^4) \left(I + (R_k^2 + R_k^8)(R_k^4 + R_k^{16}) \right) \right) \right) \tag{16}$$

Remark 1. *The particular choice $\beta = 1$ in Equation (15) reduces these iterations to the usual hyper-power family of the iterative methods possessing the order $p \ge 2$:*

$$\begin{aligned} X_{k+1} &= X_k \left(pI - \binom{p}{2} AX_k + \cdots + (-1)^{p-1} \binom{p}{p} (AX_k)^{p-1} \right) \\ &= X_k \sum_{i=0}^{p-1} R_k^i, \quad R_k = I - AX_k \end{aligned} \tag{17}$$

The choice $p = 2$ in Equation (15) leads to the scaled Schulz matrix iteration considered recently in [23], and the choice $p = 2$, $\beta = 1$ produces the original Schulz matrix iteration, originated in [12].

Finally, a hybrid algorithm may be written now by incorporating Equations (9) and (16) as follows.

Algorithm 1 The new hybrid method for computing generalized inverses.

1: The input is given $X_0 \in \mathbb{C}^{n \times m}$.
2: Use Equation (16) until $\|X_{l+1} - X_l\| < \delta$ (for an inner loop counter l or $\epsilon < \delta$).
3: set $X_0 = X_l$
4: for $k = 0, 1, \ldots$ until convergence ($\|X_{k+1} - X_k\| < \epsilon$), use Equation (9) to converge with high order.
5: end for

Instead of the hybrid Algorithm 1, based on the usage of Equation (16) in the initial phase and Equation (9) in the final stage, our third result here is to define a unique iterative method, which can be derived by applying variable acceleration parameter $\beta = 1 + \beta_k$, $0 \leq \beta_k \leq 1$. This approach yields scaled hyper-power iterations of the general form:

$$\begin{aligned} X_0 &= \alpha G \\ X_{k+1} &= X_k \left(-\beta_k I + (1+\beta_k) \sum_{i=0}^{p-1} R_k^i \right), \ 0 \leq \beta_k \leq 1, \ k \geq 0 \\ &= X_k \left(I + (1+\beta_k) \sum_{i=1}^{p-1} R_k^i \right) \end{aligned} \tag{18}$$

where the initial approximation $X_0 = \alpha G$ is chosen according to Equation (11).

Furthermore, it is possible to propose various modifications of β_k in a manner that guarantees $1 + \beta_k \to 1$. For example:

$$\beta_0 = 1, \quad \beta_{k+1} = \frac{\beta_k}{2}, \ k \geq 0 \tag{19}$$

3. Balancing Chemical Equations Using Iterations

In accordance with the intention that was motivated in the first section, in this section, we investigate the applicability of some iterations from the hyper-power family in balancing chemical equations. It is shown that the iterative methods can be applied successfully without any limitations.

It is assumed that a chemical system is modeled by a single reaction of the general form (see, for example, [2]):

$$\sum_{j=1}^{r} x_j \prod_{i=1}^{m} \Psi_{a_{ij}}^i \to \sum_{j=r+1}^{r+s} x_j \prod_{i=1}^{m} \Omega_{b_{ij}}^i \tag{20}$$

In Equation (20), $x_j, j = 1, \ldots, r$ (resp. $x_j, j = r+1, \ldots, r+s$) are unknown rational coefficients of the reactants (resp. the products), Ψ^i, Ω^i, $i = 1, \ldots, m$ are chemical elements in reactants and products, respectively, and $a_{ij}, i = 1, \ldots, m, j = 1, \ldots, r$ and $b_{ij}, i = 1, \ldots, m, j = r+1, \ldots, r+s$ are the numbers of atoms Ψ^i and Ω^i, respectively, in the j-th molecule.

3.1. Balancing Chemical Equations Using Iterative Methods

The coefficients x_i are integers, rational or real numbers, which should be determined on the basis of three basic principles: (1) the law of conservation of mass; (2) the law of conservation of atoms; (3) the time-independence of Equation (20), an assumption usually valid for stable/non-sensitive reactions. Let there be m distinct atoms involved in the chemical reaction Equation (20) and $n = r + s$ distinct reactants and products. It is necessary to form an $m \times n$ matrix A, called the reaction matrix, whose columns represent the reactants and products and the rows represent the distinct atoms in the chemical reaction. More precisely, the (i, j)-th element of A, denoted by $a_{i,j}$, represents the number of atoms of type i in each compound/element (reactant or product). An arbitrary element $a_{i,j}$ is positive or

negative according to whether it corresponds to a reactant or a product. Hence, the balancing chemical equation problem can be formulated as the homogeneous matrix equation:

$$Ax = 0 \tag{21}$$

with respect to the unknown vector $x \in \mathbb{R}^n$, where $A \in \mathbb{R}^{m \times n}$ denotes the reaction matrix and 0 denotes the null column vector of the order m. In this way, an arbitrary chemical reaction can be formulated as a matrix equation.

We would like to use the symbolic and numerical possibilities of the *Mathematica* computer algebra system in conjunction with the above-defined iterative method(s) for computing generalized inverses to automatize the chemical reactions balancing process.

The general solution of the balancing problem in the matrix form Equation (21) is given by:

$$s = \left(I - A^{\dagger}A\right) c \tag{22}$$

where c is the arbitrarily-selected n-dimensional vector. Let us assume that the approximation of $A^{\dagger}$ generated by an arbitrary iterative method is given by $X := X_{k+1}$.

If the iterative method for computing X is performed in the floating point arithmetic, it is necessary to perform a transition from the solution whose coordinates are real numbers into an exact (integer and/or rational) solution. Thus, the iterative approach in balancing chemical equations assumes three general algorithmic steps, as is described in Algorithm 2.

Algorithm 2 General algorithm for balancing chemical equations by an iterative solver.

1: Apply (for example) Algorithm 1 and compute the approximation $X := X_{k+1}$ of $A^{\dagger}$.
2: Compute the vector s using Equation (22).
3: Transform real numbers included in s into an exact solution.

A clear observation about Algorithms 2 is the following:

- Steps 1 and 2 require usage of real arithmetic (with very high precision);
- Step 3 requires usage of symbolic processing and exact arithmetic capabilities to deal with rational numbers.

As a result, in order to apply iterative methods to the problem of balancing chemical equations, it is necessary to use a software that meets two diametrically-opposite criteria: the ability to carry out numerical calculations (with very high precision) and the ability to apply the exact arithmetic and symbolic calculations. The programming language *Mathematica* possesses both of these properties. More details about this programming language can be found in [24].

The following (sample) *Mathematica* code can be used to determine the exact solution using real values contained in the vector s (defined in Equation (22)).

```
Id = IdentityMatrix[n]; s = (Id - X.A).ConstantArray[1, n];
s = Rationalize[s, 10^(-300)]; c = s*LCM @@ Denominator /@ s;
(* Multiply s by the Least Common Multiple of denominators in s *)
s = c/Min @@ Numerator /@ c
(* Divide c by the Minimum of numerators in c *)
```

The standard *Mathematica* function `Rationalize[x,dx]` yields the rational number with the smallest denominator within a given tolerance dx of x. Sometimes, to avoid the influence of round-off errors and possible errors caused by the usage of the function `Rationalize`, it is necessary to perform iterative steps with a very high precision.

An improvement of the vector s can be attained as follows. It is possible to propose an amplification of the vector $s = \left(I - A^{\dagger}A\right)c$, where c is an n-dimensional column vector. The improvement can be obtained using $s = \left(I - A^{\dagger}A\right)\left(\left(I - A^{\dagger}A\right)c\right)$. In the practical implementation, it is necessary to replace the expression `s = (Id - X.A).ConstantArray[1, n]` by `s = (Id - X.A).((Id - X.A).ConstantArray[1, n]).`

This replacement can be explained by the fact that $A(I - A^{\dagger}A)s$ is closer to the zero vector zero than As.

3.2. Balancing Chemical Equations in Symbolic Form

As was explained in [6], balancing ℵ chemical reactions that possess atoms with fractional oxidation numbers and non-unique coefficients is an extremely hard problem in chemistry. The case when the system Equation (21) is not uniquely determined can be resolved using the *Mathematica* function `Reduce`. If a ℵ chemical reaction includes n reaction molecules and m reaction elements, then the reaction matrix A is of the order $m \times n$. In the case $n > m$, the reaction has $\binom{n}{m}$ general solutions. All of them can be found applying the following expression:

```
Reduce[A.{{x1}, {x2}, ..., {xn}} == {0, 0,...,0},{xk1, xk2,...,xkm}]
```

where the zero vector in the right-hand side is of the length m and $\{xk1, xk2, ..., xkm\}$ is the list of dependent variables.

4. Experimental Results

Let us denote the iterations Equation (3) by NM (Newton's Method), the iterations Equation (4) by CM (Chebyshev's Method) and Equation (9) by PM (Proposed Method). Here, we apply different methods in the Mathematica 10 environment to compute some generalized inverses and to show the superiority of our scheme(s). We also denote the hybrid algorithm given in [22] by HAL (Householder Algorithm) and our Algorithm 1 is denoted by APM (Accelerated Proposed Method). Throughout the paper the computer characteristics are Microsoft Windows XP Intel(R), Pentium(R) 4 CPU, 3.20 GHz with 4 GB of RAM, unless stated otherwise (as in the end of Example 1).

4.1. Numerical Experiments on Randomly-Generated Matrices

Example 1. *[22] In this numerical experiment, we compute the Moore–Penrose inverse of a dense, randomly-generated* $m \times n = 800 \times 810$ *matrix, which is defined as follows:*

```
m = 800; n = 810; SeedRandom[12345]; A = RandomReal[{-10, 10}, {m, n}];
```

The numerical results corresponding to the number of iterations and the CPU time are illustrated in Table 1, wherein IT denotes the number of iterative steps. It shows that APM with $m = 2$ and five inner loops, while $p = 2$, is superior to the other existing methods. We employed HAL with $m = 2$ and eight inner loops. Note that the initial matrix is chosen as $X_0 = \frac{2}{\|A\|_F^2} A^*$, while the stopping criterion is defined by $\frac{||X_{k+1}-X_k||_\infty}{||X_{k+1}||_\infty} < 10^{-300}$. Here, $\|\cdot\|_F$ stands for the Frobenius norm (Hilbert–Schmidt norm), which is for an $m \times n$ matrix A defined as:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\text{trace}(A^*A)} = \sqrt{\sum_{i=1}^{\min\{m,n\}}\sigma_i^2} \tag{23}$$

where A^* denotes the conjugate transpose of A, σ_i are the singular values of A and the trace function is used.

Table 1. The results corresponding to the Moore–Penrose inverse of a randomly-generated matrix. IT, number of iterative steps.

Methods	NM	CM	HAL	APM
IT	28	18	14	9
Time	15.7656250	15.0000000	14.0156250	13.4687500

It is important to emphasize that the computational time is directly initiated by computer and software specifications. To clarify this, we execute our implemented algorithms/methods from Example 1 on a more recently-featured computer, whose characteristics are Windows 7 Ultimate, Intel(R) Core(TM) i5-4400 CPU 3.10 GHz with 8 GB of RAM and 64 Operating System. The results corresponding to this hardware/software configuration are given in Table 2 in terms of the elapsed CPU time. Furthermore, we re-ran Example 1 for an $m \times n = 1010 \times 1000$ matrix, which was randomly generated by the code:

```
m = 1010; n = 1000; SeedRandom[12345]; A = RandomReal[{-10, 10}, {m, n}];
```

to show that our schemes can also simply be applied on matrices satisfying $m \geq n$. The results generated by these values are arranged in Table 3, where $m = 3$ inner loops are considered for APM.

Table 2. The results corresponding to the Moore–Penrose inverse of randomly-generated matrices with a better equipped computer.

Methods	NM	CM	HAL	APM
Time	1.620093	1.570090	1.363078	1.279073

Table 3. The results corresponding to the Moore–Penrose inverse of the randomly-generated matrix $A_{1010\times1000}$.

Methods	NM	CM	HAL	APM
IT	28	18	14	8
Time	2.714405	2.605205	2.371204	2.293204

The numerical example illustrates the theoretical results presented in Section 2. It can be observed from the results included in Tables 1–3 that, firstly, like the existing methods, the presented method shows a stable behavior along with a fast convergence. Additionally, according to results contained in Tables 1–3, it is clear that the number of iterations required in the APM method during numerical approximations of the Moore–Penrose inverse is smaller than the number of approximations generated by the classical methods. This observation is in accordance with the fact that the efficiency index is clearly the largest in the case of the APM method. In general, APM is superior among all of the existing famous hyper-power iterative schemes. This superiority is in accordance with the theory of efficiency analysis discussed before.

In fact, it can be observed that increasing the efficiency index by a proper factorization of the hyper-power method is a kind of nice strategy that gives promising results in terms of both the number of iterations and the computational time on different computers.

Here, it is also worth noting that Schulz-type solvers are the best choice for sparse matrices possessing sparse inverses. Since, in such cases, the usual SVD technique in the software, such as Mathematica or MATLAB, ruins the sparsity pattern and requires much more time, hence such iterative methods and the SVD-type (direct) schemes are both competitive, but have their own fields of applications.

4.2. Numerical Experiments in Balancing Chemical Equations

In this subsection, we present some clear examples indicating the applicability of our approach in the balancing chemical equations. We also apply the following initial matrix $X_0 = \frac{1}{\sigma_1^2}A^*$.

Example 2. *Consider a specific skeletal chemical equation from [10]:*

$$x_1\mathrm{KNO_3} + x_2\mathrm{C} \rightarrow x_3\mathrm{K_2CO_3} + x_4\mathrm{CO} + x_5\mathrm{N_2} \tag{24}$$

where the left-hand side of the arrow consists of compounds/elements called reactants, while the right-hand side comprises compounds/elements called the products. Hence, Equation (24) is formulated as the homogeneous equation $Ax = 0$, wherein 0 denotes the null column vector and:

$$A = \begin{pmatrix} 1 & 0 & -2 & 0 & 0 \\ 1 & 0 & 0 & 0 & -2 \\ 3 & 0 & -3 & -1 & 0 \\ 0 & 1 & -1 & -1 & 0 \end{pmatrix} \tag{25}$$

The results generated after the comparison of numerical results derived in Example 2 are given in Table 4, using 300 precision digits, being large enough to minimize round-off errors, as well as to clearly observe the computed asymptotic error constants in the convergence phase. Although in all practical problems, the machine precision (double precision) is enough (just like Example (1)), here, our focus is to find very accurate coefficients for the chemical equation, since a very accurate tolerance, such as $\|X_k - A^\dagger\|_\infty \leq 10^{-150}$, must be incorporated.

The final exact coefficients are defined as $(x_1, x_2, x_3, x_4, x_5)^{\mathrm{T}} = (2, 4, 1, 3, 1)^{\mathrm{T}}$. Thus,

$$2\mathrm{KNO_3} + 4\mathrm{C} \rightarrow \mathrm{K_2CO_3} + 3\mathrm{CO} + \mathrm{N_2} \tag{26}$$

Experimental results clearly show that PM is the most efficient method for this purpose. In addition, we remark that since we use iterative methods in floating point arithmetic to obtain the coefficient, we must use the command `Round[]` in the last lines of our written *Mathematica* code, so as to attain the coefficients in exact arithmetic.

Table 4. The results corresponding to balancing Equation (24).

Methods	NM	CM	PM
IT	14	9	3
$\|X_{k+1} - A^{\dagger}\|$	2.59961×10^{-161}	1.1697×10^{-193}	9.09312×10^{-293}

In order to support the improvement described in Section 3, it is worth mentioning that (using Mathematica notations):

$$\texttt{A.(Id} - \texttt{X.A).ConstantArray[1,n]} = \{-3.9 * 10^{-293}, -5.0 * 10^{-294}, 2.0 * 10^{-293}, -0.0 * 10^{-295}\}$$

and $\texttt{A.(Id} - \texttt{X.A)((Id} - \texttt{X.A).ConstantArray[1,n])} = \{0. * 10^{-295}, 0. * 10^{-295}, 0. * 10^{-295}, 0. * 10^{-295}\}$.

Example 3. *Now, we solve the following skeletal chemical equation from [10]:*

$$\begin{aligned} x_1\mathrm{H_3PO_4} + x_2\mathrm{MgSiO_3} + x_3\mathrm{CF_2Cl_2} + x_4\mathrm{NaAlF_4} + x_5\mathrm{KI} + x_6\mathrm{PbCrO_4} + x_7\mathrm{FeSO_4} \\ + x_8\mathrm{BrCl} + x_9\mathrm{Ca(CN)_2} + x_{10}\mathrm{SO_2} + x_{11}\mathrm{H_2}, \\ \rightarrow x_{12}\mathrm{PI_3} + x_{13}\mathrm{MgCO_3} + x_{14}\mathrm{Na_2SiO_3} + x_{15}\mathrm{PbBr_2} \\ + x_{16}\mathrm{CrCl_3} + x_{17}\mathrm{KAl(OH)_4} + x_{18}\mathrm{Fe(SCN)_3} + x_{19}\mathrm{CaF_2} + x_{20}\mathrm{H_2O} \end{aligned} \tag{27}$$

Equation (27) is formulated as a homogeneous system of linear equations with the following coefficient matrix:

$$A_{19\times 20} = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & -2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 & 0 & 4 & 4 & 0 & 0 & 2 & 0 & 0 & -3 & -3 & 0 & 0 & -4 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -3 & 0 & 0 \\ 0 & 0 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 \end{pmatrix}$$

The results of this experiment generated by using the ordinary double precision arithmetic and the stopping termination $\|X_k - A^{\dagger}\|_{\infty} \leq 10^{-10}$ *are illustrated in Figure 1. Note that the final coefficients obtained in exact arithmetic are equal to* $(x_1, \ldots, x_{20})^{\mathrm{T}} =$

$(2,3,3,6,6,6,10,12,15,20,88,2,3,3,6,6,6,10,15,79)^{\mathrm{T}}$. *The results once again show that PM is the best iterative process.*

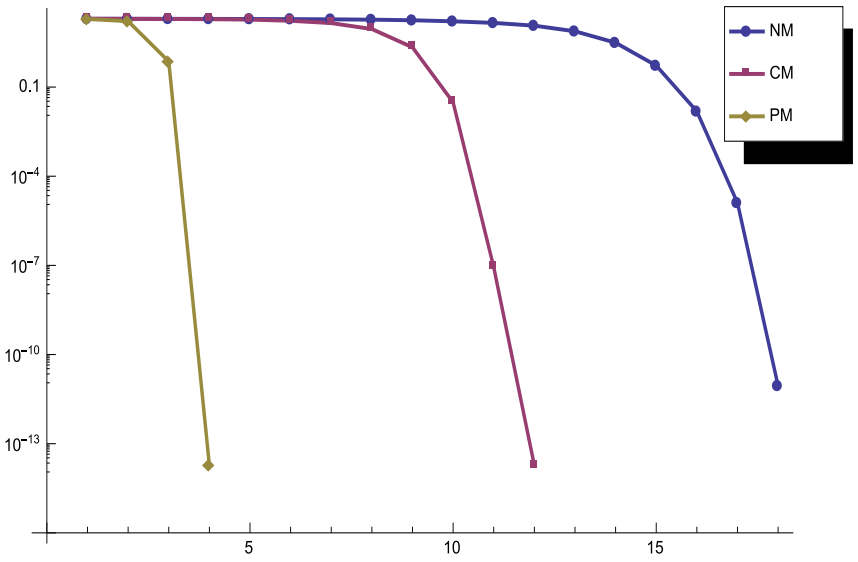

Figure 1. Convergence history for different methods used in Example 3.

Example 4. *Consider the following example from [19]:*

$$x_1\mathrm{MnO}_4^- + x_2\mathrm{H}^+ + x_3\mathrm{Fe}^{2+} \rightarrow x_4\mathrm{Mn}^{2+} + x_5\mathrm{H}_2\mathrm{O} + x_6\mathrm{Fe}^{3+}$$

The reaction matrix A can be derived by taking into account both the law of conservation of atoms and the law of electrical neutrality, and it is equal to (see [19]*):*

$$A = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 4 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -2 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ -1 & 1 & 2 & -2 & 0 & -3 \end{pmatrix}$$

As in the previous examples, let us denote by X the result derived by the iterative method Equation (9)*. The rational approximation of* $s = \texttt{A.(Id - XA)((Id - X.A).ConstantArray[1,n])}$ *is equal to* $\{\frac{2}{11}, \frac{16}{11}, \frac{10}{11}, \frac{2}{11}, \frac{8}{11}, \frac{10}{11}\}$*, and the exact solution coincides with the result given in [19]:* $\{1,8,5,1,4,5\}$.

Example 5. *In this example, it is shown that our iterative method is capable of producing the solution in the case when the real coefficients are used and the reaction is not unique within relative proportions. Let us consider Reaction 1 with one arbitrary element from [6]:*

$$x_1\mathrm{X} + x_2\mathrm{O}_2 \rightarrow x_3\mathrm{X}_{0.987}\mathrm{O} + x_4\mathrm{X}_2\mathrm{O}_3 + x_5\mathrm{X}_3\mathrm{O}_4 \tag{28}$$

The reaction matrix of the homogeneous system Equation (21) *is given by* $A = \{\{1,0,-0.987,-2,-3\},\{0,2,-1,-3,-4\}\}$. *The iterative method Equation* (9) *converges quickly, since the list of consecutive errors* $\|X_k - A^{\dagger}\|_{\infty}$ *is given by*

$\{0.163381, 2.220446049250313 \times 10^{-16},\ 2.7755575615628914 \times 10^{-16}\}$. *The approximate solution* $s = \mathtt{A}.(\mathtt{Id} - \mathtt{X.A})((\mathtt{Id} - \mathtt{X.A}).\mathtt{ConstantArray}[1, \mathtt{n}])$ *is equal to:*

$$s = \{1.40225926604, 0.890820221049, 0.657559993896, 0.35925113635, 0.0115817597876\}$$

and its fractional approximation is:

$$c = \left\{\frac{44517366795613798}{367684821979411}, \frac{103259812167103006}{1342504707428259}, \frac{67799434016501962}{1194167476431641}, \frac{56436772606792756}{1819443496152855}, 1\right\}.$$

Example 6. *All possible solutions of the problem considered in Example 5 with respect to* $x1$ *and* $x2$ *can be generated using the Mathematica function* `Reduce`*:*

```
Reduce[A.{{x1}, {x2}, {x3}, {x4}, {x5}} == {0, 0}, {x1, x2}]
```

All solutions in the symbolic form are given as follows (using Mathematica notations):

$$\mathrm{x1} = 0.987\mathrm{x3} + 2.0\mathrm{x4} + 3.0\mathrm{x5} + 0.0 \wedge \mathrm{x2} = 0.5\mathrm{x3} + 1.5\mathrm{x4} + 2.0\mathrm{x5} + 0$$

wherein $x3, x4, x5$ *are arbitrary real quantities. All possible solutions with respect to* $x1$ *and* $x3$ *can be generated using:*

```
Reduce[A.{{x1}, {x2}, {x3}, {x4}, {x5}} == {0, 0}, {x1, x3}]
```

All possible $\binom{5}{2} = 10$ *cases can be solved in the same way.*

Example 7. *As the last experiment and to show that the proposed iteration could preserve the sparsity pattern of the inverses if the inverses are sparse in nature, the following* 4000×4000 *matrix* $\mathtt{A} = \mathtt{ExampleData}[\text{“}\mathtt{Matrix}\text{”}, \text{“}\mathtt{Bai/tols4000}\text{”}]$ *has been taken from Matrix Market database with the stopping termination* $\frac{\|X_{k+1} - X_k\|_\infty}{\|X_{k+1}\|_\infty} \leq 10^{-10}$. *The new scheme Equation (9) converges in twelve iterations. The matrix plots of the approximate inverse for this case are brought forward in Figure 2.*

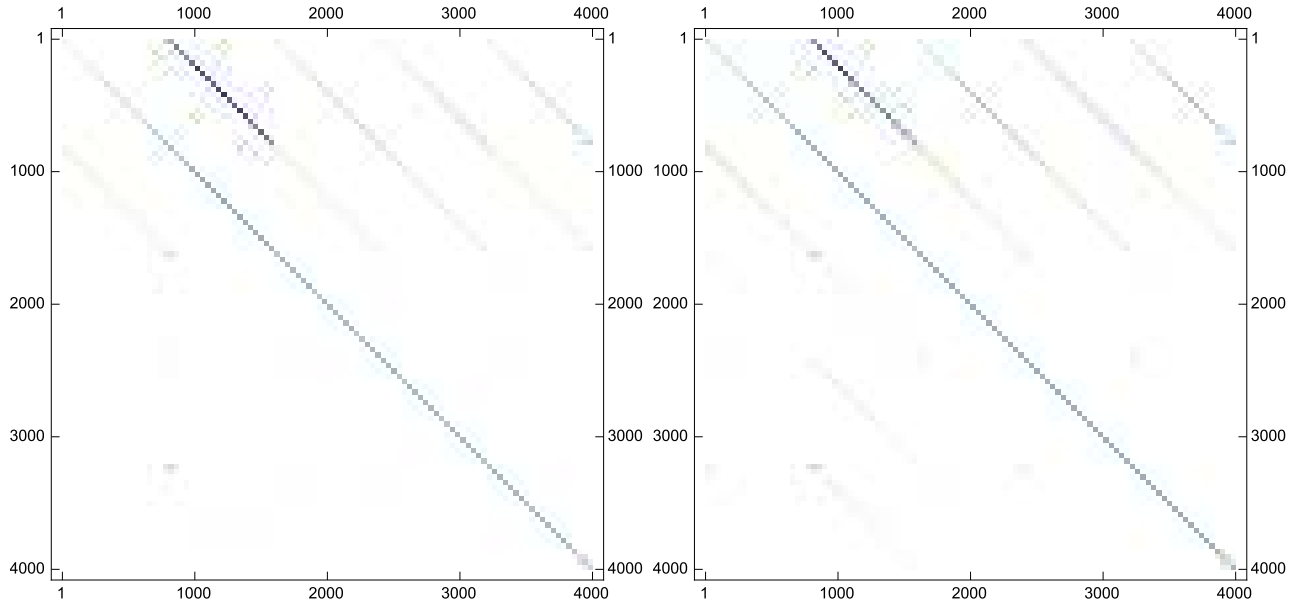

Figure 2. *Cont.*

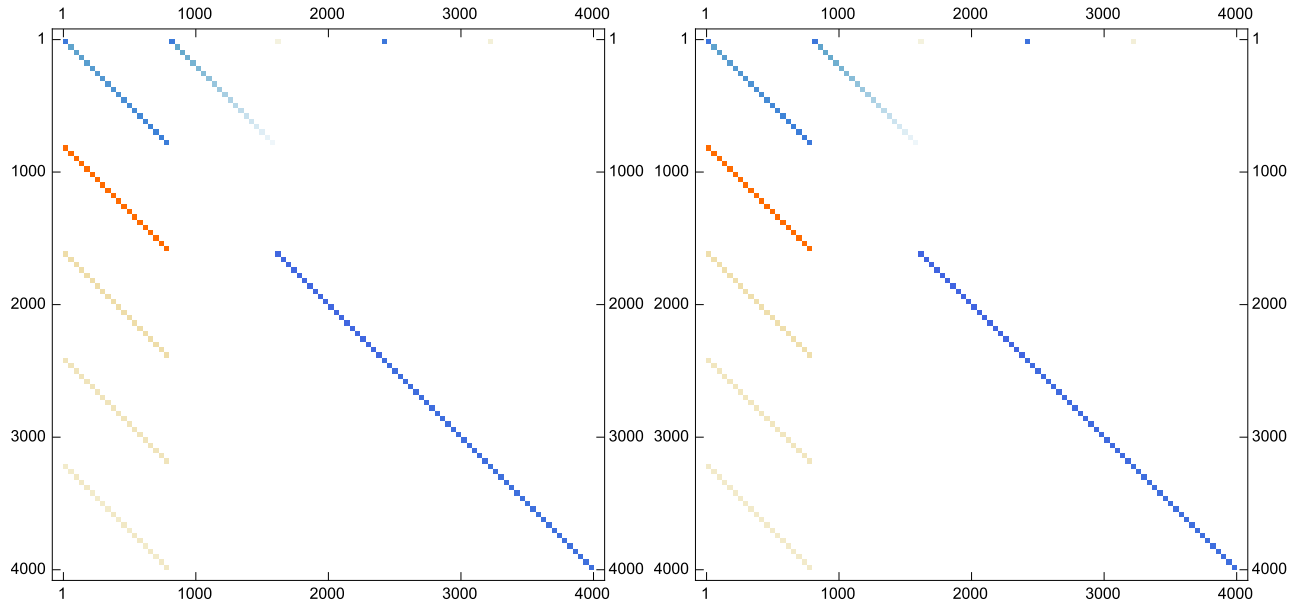

Figure 2. The sparsity pattern for the approximate inverses: X_1 (**top left**); X_2 (**top right**); X_{11} (**bottom left**); and X_{12} (**bottom right**).

5. Conclusions

In this paper, we have developed a matrix iterative method for computing generalized inverses. The derived scheme has been constructed based on the hyper-power iteration. We have shown that this scheme achieves the order of convergence equal to 31 by using only nine mmm, which hits a very high computational efficiency index.

We also provided further schemes by extending some of the known results so as to accelerate the initial phase of convergence. Furthermore, we applied our iterative schemes to balancing chemical equations as an important application-oriented area. The derived numerical results clearly upheld our theoretical findings to a great extent.

Further discussions and generalizations can be considered for future works to provide much more robust, reliable and fast hybrid algorithms for computing generalized inverses with potential applications, for example as in [25].

Acknowledgments

The research of the first author (Farahnaz Soleimani) is financially supported by Roudehen Branch, Islamic Azad University, Roudehen, Iran. Furthermore, the second author (Predrag S. Stanimirović) gratefully acknowledges support from the Research Project 174013 of the Serbian Ministry of Science. Interested readers may contact the corresponding author to obtain the Mathematica programs used in the paper.

Author Contributions

The contributions of all of the authors have been similar. All of them have worked together to develop the present manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Phillips, J.C. Algebraic constructs for the graphical and computational solution to balancing chemical equations. *Comput. Chem.* **1998**, *22*, 295–308.
2. Risteski, I.B. A new generalized matrix inverse method for balancing chemical equations and their stability. *Bol. Soc. Qum. Mex.* **2008**, *2*, 104–115.
3. Risteski, I.B. A new pseudoinverse matrix method for balancing chemical equations and their stability. *J. Korean Chem. Soc.* **2008**, *52*, 223–238.
4. Krishnamurthy, E.V. Generalized matrix inverse approach for automatic balancing of chemical equations. *Int. J. Math. Educ. Sci. Technol.* **1978**, *9*, 323–328.
5. Mahadeva, R.T.; Subramanian, K.; Krishnamurthy, E.V. Residue arithmetic algorithms for exact computation of g-inverses of matrices. *SIAM J. Numer. Anal.* **1976**, *13*, 155–171.
6. Risteski, I.B. A new generalized algebra for the balancing of ℵ chemical reactions. *Mater. Technol.* **2014**, *48*, 215–219.
7. Smith, W.R.; Missen, R.W. Using Mathematica and Maple to obtain chemical equations. *J. Chem. Educ.* **1997**, *74*, 1369–1371.
8. Xia, Y. A novel iterative method for computing generalized inverse. *Neural Comput.* **2014**, *26*, 449–465.
9. Higham, N.J. Gaussian elimination. *WIREs Comp. Stat.* **2011**, *332–334*, 230–238.
10. Sen, S.K.; Agarwal, H.; Sen, S. Chemical equation balancing: An integer programming approach. *Math. Comput. Model.* **2006**, *44*, 678–691.
11. Balasubramanian, K. Linear variational Diophantine techniques in mass balance of chemical reactions. *J. Math. Chem.* **2001**, *30*, 219–225.
12. Ben-Israel, A. An iterative method for computing the generalized inverse of an arbitrary matrix. *Math. Comput.* **1965**, *19*, 452–455.
13. Climent, J.-J.; Thome, N.; Wei, Y. A geometrical approach on generalized inverses by Neumann-type series. *Linear Algebra Appl.* **2001**, *332–334*, 533–540.
14. Liu, X.; Jin, H.; Yu, Y. Higher-order convergent iterative method for computing the generalized inverse and its application to Toeplitz matrices. *Linear Algebra Appl.* **2013**, *439*, 1635–1650.
15. Soleymani, F.; Stanimirovic, P.S.; Haghani, F.K. On hyper-power family of iterations for computing outer inverses possessing high efficiencies. *Linear Algebra Appl.* **2015**, *484*, 477–495.
16. Schulz, G. Iterative Berechnung der Reziproken matrix. *Z. Angew. Math. Mech.* **1933**, *13*, 57–59.
17. Soleymani, F.; Salmani, H.; Rasouli, M. Finding the Moore–Penrose inverse by a new matrix iteration. *J. Appl. Math. Comput.* **2015**, *47*, 33–48.

18. Soleymani, F. An efficient and stable Newton-type iterative method for computing generalized inverse $A^{(2)}_{T,S}$. *Numer. Algorithms* **2015**, *69*, 569–578.
19. Ramasami, P. A concise description of an old problem: Application of matrices to obtain the balancing coefficients of chemical equations. *J. Math. Chem.* **2003**, *34*, 123–129.
20. Sharifi, M.; Arab, M.; Khaksar Haghani, F. Finding generalized inverses by a fast and efficient numerical method. *J. Comput. Appl. Math.* **2015**, *279*, 187–191.
21. Stanimirović, P.S.; Soleymani, F. A class of numerical algorithms for computing outer inverses. *J. Comput. Appl. Math.* **2014**, *263*, 236–245.
22. Soleimani, F.; Soleymani, F.; Cordero, A.; Torregrosa, J.R. On the extension of Householder's method for weighted Moore–Penrose inverse. *Appl. Math. Comput.* **2014**, *231*, 407–413.
23. Petković, M.D.; Stanimirović, P.S. Two improvements of the iterative method for computing Moore–Penrose inverse based on Penrose equations. *J. Comput. Appl. Math.* **2014**, *267*, 61–71.
24. Wolfram, S. *The Mathematica Book*, 5th ed.; Wolfram Media: Champaign, IL, USA, 2003.
25. Soleymani, F.; Sharifi, M.; Karimi Vanani, S.; Khaksar Haghani, F. An inversion-free method for finding positive definite solution of a rational matrix equation. *Sci. World J.* **2014**, *2014*, 560931.