ARCHIVUM MATHEMATICUM (BRNO)
Tomus 39 (2003), 27 – 36

SELF–CORRECTING ITERATIVE METHODS FOR COMPUTING {2}–INVERSES

PREDRAG S. STANIMIROVIĆ

Abstract. In this paper we construct a few iterative processes for computing {2}-inverses of a linear bounded operator. These algorithms are extensions of the corresponding algorithms introduced in [11] and a method from [8]. A few error estimates are derived.

1. Introduction and preliminaries

Let $\mathbb{C}^{m\times n}$ be the set of $m\times n$ complex matrices and $\mathbb{C}_r^{m\times n}$ denote the set of $m\times n$ complex matrices whose rank is r. By I we denote an appropriate identity matrix.

There are well–known properties of generalized inverses of A:

$$(1)\ AXA=A,\ \ (2)\ XAX=X,\ \ (3)\ (AX)^*=AX,\ \ (4)\ (XA)^*=XA.$$

For a subset $\mathcal{S}$ of the set $\{1,2,3,4\}$, the set of operators obeying the conditions contained in $\mathcal{S}$ is denoted by $A\{\mathcal{S}\}$. An operator in $A\{\mathcal{S}\}$ is called an $\mathcal{S}$-inverse of A and is denoted by $A^{(\mathcal{S})}$. In particular, for any A, the set $A\{1,2,3,4\}$ consists of a single element, the Moore-Penrose inverse of A, denoted by $A^\dagger$. Any element from the class $A\{1,2\}$ is also called the reflexive generalized inverse of A.

Let A be $m\times n$ matrix and $\operatorname{rank}(A)=r$. Then all $\{2\}$-inverses of A can be represented in the form [15]

$$A\{2\}=\{X: X=W_1(W_2AW_1)^{-1}W_2,\ W_1\in\mathbb{C}^{n\times s},\ W_2\in\mathbb{C}^{s\times m},\ s=1,\dots,r\}\,. \tag{1.1}$$

In the literature it is known a great number of iterative methods for computation of the usual matrix inverse and the Moore-Penrose inverse.

One class of these methods is based on the *hyper-power* iterative method. An application of the hyper-power method of the order 2 in usual matrix inversion

---

2000 *Mathematics Subject Classification*: 15A09, 65F20.
Key words and phrases: generalized inverses, Moore–Penrose inverse, error matrix.
Received December 13, 1999.

dates back to the paper of Schulz [12]. The hyper-power method of the order 2 coincides with the Newton iteration and it is investigated in [2–4] and [8]:

$$X_{k+1} = X_k(2I - AX_k) = (2I - X_kA)X_k\,. \tag{1.2}$$

Ben-Israel and Cohen shown that the iteration (1.2) converges to $A^\dagger$ provided that

$$X_0 = \alpha A^*, \tag{1.3}$$

where α is a positive and sufficiently small [2–4]. Altman devised the hyper-power method of an arbitrary order $q \geq 2$, for inverting a nonsingular bounded operator on a Banach space [1]. In [9] the convergence of this method is proved under a weaker condition and some better error estimates are derived. Zlobec [20] and Petryshyn [10] shown that the q-th order hyper-power iterative method, with $q \geq 2$, can be used in computation of the Moore-Penrose inverse of an arbitrary matrix or a bounded linear operator with closed range. The hyper-power iterative method of the order 2 is studied in [13] in view of the singular value decomposition of a matrix. In [6] it is introduced a modification of the hyper-power iterative method which can be used in computation of the matrix product $A^\dagger B$, where A and B are arbitrary complex matrices with equal number of rows.

Also, a few iterative methods for computing the Moore-Penrose inverse are introduced and investigated in [11] and [8].

If L is the limit matrix and X_k is the k-th approximation of L, defined by an algorithm, then the convergence properties of the algorithm can be studied using the *error matrix* $E_k = X_k - L$, [11]. The matrix formula representing E_{k+1} is a sum of possible *zero-order* term consisting of a matrix which does not depend upon E_k, one or more *first-order* matrix terms in which E_k or E_k^* appears only once, and *higher-order* terms in which E_k and E_k^* appears at least twice [11]. All suitable algorithms have a zero-order term identical to 0, so that the first-order terms determine the convergence properties of the algorithm. Also, the method is self-correcting if the first-order terms are equal to 0.

Known iterative algorithms for inverting of nonsingular matrices have the first-order term equal to zero. However, iterative algorithms for computing the Moore-Penrose inverse have nonzero first-order term [11].

The paper is organized as follows. In Section 2 we construct several iterative methods for computing $\{2\}$-inverses of a singular complex matrix, generalizing iterative methods from [11], [8], using principles from [14] and the full-rank representation of $\{2\}$-inverses from [15]. In Section 3 we investigate error estimates of these methods. In view of these estimates we present a comparative study of introduced methods. Three of these methods are self-correcting.

In certain cases we get iterative methods for computation of the Drazin inverse and the generalized inverse $A^{(2)}_{T,S}$.

2. Iterative methods

In the following lemma we introduce a modification of the hyper-power iterative method of the order $q \geq 2$ which can be used in the construction of $\{2\}$-inverses.

Lemma 2.1. *Let $A \in \mathbb{C}^{m\times n}$, $\operatorname{rank}(A) = r \geq 2$, $s \leq r$, and $W_1 \in \mathbb{C}^{n\times s}$, $W_2 \in \mathbb{C}^{s\times m}$ are two arbitrary matrices, such that the matrix W_2AW_1 is invertible. If $q \geq 2$ is an integer, then the following two iterative processes:*

$$Y_0 = Y_0' = \alpha(W_2AW_1)^*, \qquad 0 < \alpha \leq \frac{2}{\operatorname{tr}((W_2AW_1)^* W_2AW_1)},$$

$$\begin{aligned} T_k &= I - Y_k W_2 A W_1 , \\ Y_{k+1} &= (I + T_k + \cdots + T_k^{q-1}) Y_k , \\ X_{k+1} &= W_1 Y_{k+1} W_2 \quad k = 0, 1, \ldots \end{aligned}$$

$$\begin{aligned} T_k' &= I - W_2 A W_1 Y_k' , \\ Y_{k+1}' &= Y_k'(I + T_k' + \cdots + T_k'^{\,q-1}) , \\ X_{k+1}' &= W_1 Y_{k+1}' W_2 \qquad k = 0, 1, \ldots, \;\; s = 1, \ldots, r \end{aligned}$$

generate $\{2\}$-inverses of A.

Proof. Follows from $Y_k \to (W_2AW_1)^{-1}$ and the general representation (1.1) of $\{2\}$-inverses. □

Remark 2.1. In the case $s = r$ the result of Lemma 2.1 reduces to known result, stated for $\{1,2\}$-inverses in [14, Lemma 2.1].

In the following lemma we introduce a few iterative methods for computing $\{2\}$-inverses.

Lemma 2.2. *Let A be $m \times n$ matrix of rank $\operatorname{rank}(A) = r \geq 2$ and $W_1 \in \mathbb{C}^{n\times s}$, $W_2 \in \mathbb{C}^{s\times m}$ are two arbitrary matrices, such that $W = W_2AW_1$ is an $s \times s$ invertible matrix. where $1 \leq s \leq r$. If converges, any of the following iterative processes*

$$Y_{k+1} = Y_k(2I - WY_k) , \quad X_{k+1} = W_1 Y_{k+1} W_2 , \tag{2.1}$$

$$Y_{k+1} = Y_k W Y_k , \quad X_{k+1} = W_1 Y_{k+1} W_2 , \tag{2.2}$$

$$Y_{k+1} = Y_k W Y_k (2I - WY_k) , \quad X_{k+1} = W_1 Y_{k+1} W_2 , \tag{2.3}$$

$$Y_{k+1} = Y_k W Y_k (2I - WY_k W Y_k) , \quad X_{k+1} = W_1 Y_{k+1} W_2 , \tag{2.4}$$

$$Y_{k+1} = Y_k (WY_k)^* , \quad X_{k+1} = W_1 Y_{k+1} W_2 , \tag{2.5}$$

$$Y_{k+1} = (WY_k)^* Y_k , \quad X_{k+1} = W_1 Y_{k+1} W_2 , \tag{2.6}$$

$$Y_{k+1} = Y_k W Y_k (4I - 4WY_k + (WY_k)^2), \quad X_{k+1} = W_1 Y_{k+1} W_2 , \tag{2.7}$$

$$k = 0, 1, \ldots$$

generates $\{2\}$*-inverse of* A.

Proof. Equation (2.1) is a partial case $q = 2$ of Lemma 2.1. In the rest of the proof it is sufficient to prove $Y_k \to W^{-1} = W^{\dagger}$. Then the proof follows from the general representation (1.1) of $\{2\}$-inverses.

The first equation in (2.2) is an iterative method for approximation of the Penrose's equation $Y = YWY$. The first equation (2.3) is an application of the Algorithm ζ from [11] in computation of the inverse $W^{-1} = W^{\dagger}$. The first equation in (2.4) can be generated applying Algorithm θ from [11] in computation of the inverse W^{-1}. Similarly, (2.5) and (2.6) are derived as analogous adaptations of the algorithms γ and δ, respectively, established in [11]. Finally, algorithm (2.7) is derived from the following iterative process, investigated in [8]:

$$Y_{k+1/2} = (2I - Y_k A)Y_k, \qquad Y_{k+1} = Y_{k+1/2} A Y_{k+1/2}.$$

Indeed, in this case we get the following approximation Y_{k+1} of the matrix W^{-1}:

$$Y_{k+1} = (2I - Y_k W)Y_k W(2I - Y_k W)Y_k = Y_k W Y_k (4I - 4WY_k + (WY_k)^2). \qquad \square$$

Corollary 2.2. **(a)** *Let* A *be square matrix of the order* n, $l \geq \operatorname{ind}(A)$ *and* $A^l = P_{A^l} Q_{A^l}$ *be a full-rank factorization of* A^l. *Then in the case* $W_1 = P_{A^l}$, $W_2 = Q_{A^l}$, *each of the iterative processes* (2.1)–(2.7) *generates the Drazin inverse of* A.

(b) *Let* $A \in \mathbb{C}_r^{m\times n}$, *let* T *be a subspace of* $\mathbb{C}^n$ *of dimension* $s \leq r$ *and* S *be a subspace of* $\mathbb{C}^m$ *of dimension* $m - s$, *such that* $AT \oplus S = \mathbb{C}^m$. *In the case* $\mathcal{R}(W_1) = T$, $\mathcal{N}(W_2) = S$ *each of the iterative methods* (2.1)-(2.7) *generates the generalized inverse* $A^{(2)}_{T,S}$.

Proof. **(a)** After a comparison of the general representation (1.1) of $\{2\}$-inverses and the full-rank representation of the Drazin inverse from [15], we conclude that the Drazin inverse A^D is an element from $A\{2\}$ which satisfies $W_1 = P_{A^l}$, $W_2 = Q_{A^l}$.

(b) The condition $AT \oplus S = \mathbb{C}^m$ ensures the existence of the generalized inverse $A^{(2)}_{T,S}$, [5]. This part of the proof follows from the representation of the generalized inverse $A^{(2)}_{T,S}$ which is introduced in [17]. $\square$

Introduced methods are implemented by means of the following functions in the package MATHEMATICA.

```
Tr[g_List]:=
   Block[{s1=0,i,n1,m1},
     {n1,m1}=Dimensions[g];
     If[n1=!=m1, s1={},
        s1=Sum[g[[i,i]], {i,n}]
     ];
     s1
```

```
   ]

Hermit[a_]:=Conjugate[Transpose[a]]

norma[m_]:=
   Block[{k1,k2,i,j,u},
      {k1,k2}=Dimensions[m];
      u=Sqrt[Sum[m[[i,j]]^2,{i,k1},{j,k2}]];
      Return[N[u]]
   ]

L2i[a_List, w1_List, w2_List,i_, e_]:=
   Block[{y0,y1,n,w,alfa,eps=e,nor,k=0},
    w=w2.a.w1;
    {n,n}=Dimensions[w];
    If[(i==1) || (i==3) ,
          alfa=2/Tr[Hermit[w].w],
          alfa=1/Tr[Hermit[w].w]
    ];
    y0=alfa*Hermit[w];
    Switch[i,
     1,y1=N[y0.(2*IdentityMatrix[n]-w.y0)],
     2,y1=N[y0.w.y0],
     3,y1=N[y0.(2*IdentityMatrix[n]-w.y0)],
     4,y1=N[y0.w.y0.(2*IdentityMatrix[n]-w.y0.w.y0)],
     5,y1=N[y0.Hermit[w.y0]],
     6,y1=N[Hermit[w.y0].y0],
     7,y1=N[y0.w.y0.(4*IdentityMatrix[n]-4*w.y0+(w.y0)^2)]
    ];
    nor=norma[y1-y0];
    While[nor>=eps,
     y0=y1;  k++;
     Switch[i,
       1,y1=N[y0.(2*IdentityMatrix[n]-w.y0)],
       2,y1=N[y0.w.y0],
       3,y1=N[y0.(2*IdentityMatrix[n]-w.y0)],
       4,y1=N[y0.w.y0.(2*IdentityMatrix[n]-w.y0.w.y0)],
       5,y1=N[y0.Hermit[w.y0]],
       6,y1=N[Hermit[w.y0].y0],
       7,y1=N[y0.w.y0.(4*IdentityMatrix[n]-4*w.y0+(w.y0)^2)]
     ];
     nor=norma[y1-y0];
    ];
    Return[Simplify[w1.y1.w2]]
   ]
```

Example 2.1. Consider the matrix

$$a = \begin{bmatrix} -1 & 0 & 1 & 2 \\ -1 & 1 & 0 & -1 \\ 0 & -1 & 2 & 3 \\ 0 & 1 & -1 & -3 \\ 1 & -1 & 0 & 1 \\ 5 & 0 & -1 & -2 \end{bmatrix}$$

of rank 3. Let us choose the following matrices of rank 2:

$$w1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 1 & 0 \\ 4 & 2 \end{bmatrix}, \quad w2 = \begin{bmatrix} 3 & 1 & 3 & 1 & 2 & -1 \\ 0 & -1 & 0 & 0 & -2 & 1 \end{bmatrix}.$$

Various $\{2\}$-inverses of the matrix a are generated as follows:

```
x=L2i[a,w1,w2,1,0.00000000001]
{{0.214286,-0.314286,0.214286,0.0714286,-0.628571,0.314286},
{-0.107143,0.357143,-0.107143,-0.0357143,0.714286,-0.357143},
{0.107143,-0.157143,0.107143,0.0357143,-0.314286,0.157143},
{0.214286,0.0857143,0.214286,0.0714286,0.171429,-0.0857143}}
x=L2i[a,w1,w2,2,0.0001]
{{0.0862259,0.0318854,0.0862259,0.028742,0.0637708,-0.0318854},
{0.0213806,0.00790632,0.0213806,0.00712687,0.0158126,-0.00790632},
{0.043113,0.0159427,0.043113,0.014371,0.0318854,-0.0159427},
{0.215213,0.0795834,0.215213,0.0717377,0.159167,-0.0795834}}
x=L2i[a,w1,w2,3,0.00000000001]
{{0.214286,-0.314286,0.214286,0.0714286,-0.628571,0.314286},
{-0.107143,0.357143,-0.107143,-0.0357143,0.714286,-0.357143},
{0.107143,-0.157143,0.107143,0.0357143,-0.314286,0.157143},
{0.214286,0.0857143,0.214286,0.0714286,0.171429,-0.0857143}}
x=L2i[a,w1,w2,4,0.000000000000000001]
{{0.0866953,0.032059,0.0866953,0.0288984,0.064118,-0.032059},
{0.021497,0.00794936,0.021497,0.00716567,0.0158987,-0.00794936},
{0.0433477,0.0160295,0.0433477,0.0144492,0.032059,-0.0160295},
{0.216385,0.0800167,0.216385,0.0721282,0.160033,-0.0800167}}
x=L2i[a,w1,w2,5,0.0001]
{{0.0862259,0.0318854,0.0862259,0.028742,0.0637708,-0.0318854},
{0.0213806,0.00790632,0.0213806,0.00712687,0.0158126,-0.00790632},
{0.043113,0.0159427,0.043113,0.014371,0.0318854,-0.0159427},
{0.215213,0.0795834,0.215213,0.0717377,0.159167,-0.0795834}}
x=L2i[a,w1,w2,6,0.01]
{{0.0810205,0.0293177,0.0810205,0.0270068,0.0586355,-0.0293177},
{-0.00439517,-0.00158945,-0.00439517,-0.00146506,-0.0031789,0.00158945},
{0.0405102,0.0146589,0.0405102,0.0135034,0.0293177,-0.0146589},
{0.153251,0.0554566,0.153251,0.0510835,0.110913,-0.0554566}}
x=L2i[a,w1,w2,7,0.000000000000001]
{{0.0877326,0.0292442,0.0877326,0.0292442,0.0584884,-0.0292442},
{0.0217539,0.00725129,0.0217539,0.00725129,0.0145026,-0.00725129},
{0.0438663,0.0146221,0.0438663,0.0146221,0.0292442,-0.0146221},
{0.218973,0.072991,0.218973,0.072991,0.145982,-0.072991}}
```

3. Calculating errors of algorithms

All of the algorithms described in the previous section are of the general form $X_{k+1} = W_1 Y_{k+1} W_2$, where Y_k is the k-th approximation of the inverse matrix $W^{-1} = (W_2AW_1)^{-1}$. Since the iterative methods for computing the inverse $(W_2AW_1)^{-1}$ are self-correcting, it seems interesting to investigate self-correctness of the iterative methods (2.1)–(2.7).

Theorem 3.1. *Let $A \in \mathbb{C}^{m\times n}$ be of rank r and $s \le r$ be an arbitrary positive integer. Assume that $W_1 \in \mathbb{C}^{n\times s}$ and $W_2 \in \mathbb{C}^{s\times m}$ are chosen so that $W = W_2AW_1$ is an invertible matrix, and $Y_k = W^{-1} + E$ is the k-th estimate of an algorithm for computing W^{-1}. Consider the iterative methods (2.1)-(2.7) for generating the class $A\{2\}$. Denote by $error_0$, $error_1$ and $error_2$ the zero-order, first-order and the second-order terms, respectively, in the error estimates $X_{k+1} - A^{(2)}$. All these algorithms have zero-order terms equal to 0. Also, performances of these methods are represented in the following Table 1.*

Formula for X_{k+1}	$error_1$	$error_2$
$W_1Y_k(2I - WY_k)W_2$	*0*	$-W_1EWEW_2$
$W_1Y_kWY_kW_2$	$2W_1EW_2$	W_1EWEW_2
$W_1Y_kWY_k(2I - WY_k)W_2$	W_1EW_2	$-W_1EWEW_2$
$W_1Y_kWY_k(2I - WY_kWY_k)W_2$	*0*	$-4W_1EWEW_2$
$W_1Y_k(WY_k)^*W_2$	$W_1(W^{-1}E^*W^* + E)W_2$	$W_1EE^*W^*W_2$
$W_1(WY_k)^*Y_kW_2$	$W_1(W^*E^*W^{-1} + E)W_2$	$W_1W^*E^*EW_2$
$W_1Y_kWY_k(4I-4WY_k+(WY_k)^2)W_2$	*0*	$-2W_1EWEW_2$

Table 1.

Proof. For the iterative process (2.1) we can write

$$X_{k+1} = W_1Y_k(2I - WY_k)W_2.$$

Since $Y_k = W^{-1} + E$, we get

$$\begin{aligned} X_{k+1} &= W_1(W^{-1} + E)\left(2I - W(W^{-1} + E)\right)W_2 \\ &= A^{(2)} - W_1EWEW_2. \end{aligned}$$

This is a verification of the first row in the table.

The iterative process (2.2) satisfies

$$\begin{aligned} X_{k+1} &= W_1(W^{-1} + E)W(W^{-1} + E)W_2 \\ &= A^{(2)} + 2W_1EW_2 + W_1EWEW_2 \end{aligned}$$

which confirms the second row of the table.

For the iterative process (2.3) have

$$X_{k+1} = W_1Y_kWY_k(2I - WY_k)W_2,$$

$$\begin{aligned} X_{k+1} &= W_1(W^{-1}+E)W(W^{-1}+E)\left(2I - W(W^{-1}+E)\right)W_2 \\ &= A^{(2)} + W_1EW_2 - W_1EWEW_2 - W_1EWEWEW_2 \end{aligned}$$

which completes the third row of the table.

The fourth row of the table can be verified from the following:

$$\begin{aligned} X_{k+1} &= W_1(W^{-1}+E)W(W^{-1}+E)W_2\left(2I - W(W^{-1}+E)W(W^{-1}+E)\right) \\ &= A^{(2)} + W_1(-4EWE - 4EWEWE - 4EWEWEWE)W_2. \end{aligned}$$

The rest of the proof can be verified in a similar way. □

In the following table we present the number of basic matrix operations, required in each iteration of the introduced methods.

Formula for X_{k+1}	multiplications	other operations
$W_1Y_k(2I - WY_k)W_2$	6	2
$W_1Y_kWY_kW_2$	6	0
$W_1Y_kWY_k(2I - WY_k)W_2$	8	2
$W_1Y_kWY_k(2I - WY_kWY_k)W_2$	10	2
$W_1Y_k(WY_k)^*W_2$	6	1
$W_1(WY_k)^*Y_kW_2$	6	1
$W_1Y_kWY_k(4I - 4WY_k + (WY_k)^2)W_2$	10	3

Table 2.

4. Conclusion

We introduce a few iterative methods for computing $\{2\}$-inverses, which represent extensions of analogous algorithms from [11], [8] and [14]. Error estimates of these methods are investigated. In view of these results, we give a few concluding remarks.

Remark 4.1. It is known that an algorithm is *self-correcting* if it first-order error is 0 [13]. Consequently, between the iterative methods for computing $\{2\}$-inverses, the methods (2.1), (2.4) and (2.7) are self–correcting. Note that in [14] an algorithm similar to the algorithm (2.1) is established. This algorithm is applicable in construction of $\{1,2\}$-inverses, and it is also self-correcting. Between the self-correcting methods (2.1), (2.4) and (2.7), the method (2.1) has a smaller absolute value of the second-order error term. Also, (2.1) requires a smallest number of matrix multiplications per iteration with respect to (2.4) and (2.7). For this purpose, we recommend usage of the algorithm (2.1). Between the methods (2.2), (2.3), (2.5) and (2.6) which are not self-correcting, the algorithm (2.3) has minimal first-order term.

Remark 4.2. It is well-known that the hyper–power method for computing the inverse of an invertible operator is self–correcting, but it is not self–correcting for computing generalized inverses [18], [19]. There is the well-known Zielke's iterative refinement process which solves the self-correcting problem. Namely, the iterative refinement of the k-th order hyper-power method for computing the Moore–Penrose inverse of A has the following form [18], [19]:

$$\begin{aligned} \tilde{X}_k &= A^* X_k^* X_k X_k^* A^* \\ X_{k+1} &= \tilde{X}_k (2I - A\tilde{X}_k)\,. \end{aligned}$$

This modification is not necessary in each step.

Consequently, the following iterative refinements for the iterative processes (2.2), (2.3), (2.5) and (2.6) are available:

$$\tilde{Y}_k = W^* Y_k^* Y_k Y_k^* W^*$$

$$X_{k+1} = W_1 \tilde{Y}_k W \tilde{Y}_k W_2, \tag{2.2'}$$

$$X_{k+1} = W_1 \tilde{Y}_k W \tilde{Y}_k (2I - W\tilde{Y}_k) W_2, \tag{2.3'}$$

$$X_{k+1} = W_1 \tilde{Y}_k (W\tilde{Y}_k)^* W_2, \tag{2.5'}$$

$$X_{k+1} = W_1 (W\tilde{Y}_k)^* \tilde{Y}_k W_2. \tag{2.6'}$$

Remark 4.3. Tanabe in [16] generalized the iterative methods based on the hyper-power iterative method for computing reflexive g-inverses. Advantages of the method which is introduced in [14], analogous to the representation (2.1), with respect to Tanabe's method [16], are discussed in [14]. Over all of these advantages, the method (2.1) is more general, and can be used in the construction of {2}-inverses.

References

[1] Altman, M., *An optimum cubically convergent iterative method of inverting a linear bounded operator in Hilbert space*, Pacific J. Math. **10** (1960), 1107–113.

[2] Ben-Israel, A., *An iterative method for computing the generalized inverse of an arbitrary matrix*, Math. Comp. **19** (1965), 452–455.

[3] Ben-Israel, A., *A note on an iterative method for generalized inversion of matrices*, Math. Comp. **20** (1966), 439–440.

[4] Ben-Israel, A. and Cohen, D., *On iterative computation of generalized inverses and associated projectors*, SIAM J. Numer. Anal. **3** (1966), 410–419.

[5] Chen, Y., *Finite algorithms for* (2)*-generalized inverse* $A^{(2)}_{T,S}$, Linear and Multilinear Algebra **40** (1995), 61–68.

[6] Garnett, J., Ben-Israel, A. and Yau, S. S., *A hyperpower iterative method for computing matrix products involving the generalized inverse*, SIAM J. Numer. Anal. **8** (1971), 104–109.

[7] Herzberger, J., *Using error-bounds hyperpower methods to calculate inclusions for the inverse of a matrix*, BIT **30** (1990), 508–515.

[8] Pan, V. and Schreiber, R., *An improved Newton iteration for the generalized inverse of a matrix, with applications*, SIAM. J. Sci. Stat. Comput. **12** (1991), 1109–1130.

[9] Petryshyn, W. V., *On the inversion of matrices and linear operators*, Proc. Amer. Math. Soc. **16** (1965), 893–901.

[10] Petryshyn, W. V., *On generalized inverses and on the uniform convergence of* $(I - \beta K)^n$ *with application to iterative methods*, J. Math. Anal. Appl. **18** (1967), 417–439.

[11] Pierce, W. H., *A self-correcting matrix iteration for the Moore-Penrose inverse*, Linear Algebra Appl. **244** (1996), 357–363.

[12] Schulz, G., *Iterative Berechnung der reziproken Matrix*, Z. Angew. Math. Mech. **13** (1933), 57–59.

[13] Söderström, T. and Stewart, G. W., *On the numerical properties of an iterative method for computing the Moore-Penrose generalized inverse*, SIAM J. Numer. Anal. **11** (1974), 61–74.

[14] Stanimirović, P. S. and Djordjević, D. S., *Universal iterative methods for computing generalized inverses*, Acta Math. Hungar. **79(3)** (1998), 253–268.

[15] Stanimirović, P. S., *Block representation of* $\{2\}$, $\{1,2\}$ *inverses and the Drazin inverse*, Indian J. Pure Appl. Math. **29** (1998), 1159–1176.

[16] Tanabe, K., *Neumann-type expansion of reflexive generalized inverses of a matrix and the hyperpower iterative method*, Linear Algebra Appl. **10** (1975), 163–175.

[17] Wang, G., *The representations of the generalized inverses* $(A \otimes B)^{(1,2)}_{T,S}$ *and* $(A \otimes B)^{(2)}_{T,S}$ *and some applications*, J. Shanghai Univ. (Natural Sciences) **24** (1995), 1–6.

[18] Zielke, G., *Iterative refinement of generalized matrix inverses now practicable*, SIGNUM Newsletter **13.4** (1978), 9–10.

[19] Zielke, G., *A survey of generalized matrix inverses*, Computational Mathematics, Banach center Publications **13** (1984), 499–526.

[20] Zlobec, S., *On computing the generalized inverse of a linear operator*, Glasnik Matematički **2(22)** No 2 (1967), 265–271.

University of Niš
Faculty of Science, Department of Mathematics
Višegradska 33
18000 Niš, Yugoslavia
E-mail: pecko@pmf.pmf.ni.ac.yu