Convergence Acceleration of Iterative Methods for Inverting Real Matrices Using Frobenius Norm Minimization

Ajinkya Borle
CSEE Department
University of Maryland Baltimore County
Baltimore, Maryland 21250
Email: aborle1@umbc.edu

Samuel J. Lomonaco
CSEE Department
University of Maryland Baltimore County
Baltimore, Maryland 21250
Email: lomonaco@umbc.edu

***Abstract*—The Schulz-type methods for computing generalized matrix inverses are a family of iterative methods that are popular for their high order of convergence (≥ 2). We propose two new scaled acceleration techniques for such type of iterative methods for real matrices (based on Frobenius norm minimization) and lay out efficient algorithms to implement these techniques. Test results show one of our techniques to be most effective for dense matrices but also works for sparse cases as well.**

***Index Terms*—Acceleration, Iterative Methods, Scaling, Generalized Inverses, Frobenius norm Minimization, Schulz Iteration, Approximate Inverses, Hyperpower Sequence**

I. Introduction

Generalized matrix inverses (which include inverses like the regular inverse, the Moore-Penrose inverse, the Drazin inverse and its varients, etc) have their applications in a lot of varied domains: such as wireless communication, 3d graphics, solving differential equations and many more [1]–[3]. Recent work done to compute these inverses has focused on the Schulz-type iterative methods that are attractive to mathematicians for their high order of convergence [4]–[6]. Essentially it is :

$$X_{i+1} = X_i + X_i(I - AX_i) + \cdots + X_i(I - AX_i)^{\rho-1} \quad (1)$$

Where ρ is the order of convergence that is desired (≥ 2). X_i is the current approximation of the inverse and X_{i+1} is the next one (output of the i$^{\text{th}}$ iteration). The I here represents the Identity matrix.

Upon putting $\rho = 2$ in the above equation, we get what is known as the Schulz Equation [7], [8] which has a quadratic convergence rate

$$X_{i+1} = X_i(2I - AX_i) \quad (2)$$

The problem with this family of iterative methods is that if someone were to implement them 'as is' for higher orders, It would be a very inefficient method because of the number of matrix multiplications involved in each step will be ρ. The current work in this area focuses on finding efficient ways to compute each iteration of these methods [5], [9]–[12] (primarily by finding ways to reduce the number of matrix multiplication calculations). These shall be discussed more in the next section and later.

When it comes to techniques that help to accelerate the convergence process, they exist mostly for the low-order methods [11], [13] . In 2015, Stanimirovic et al. provided multiple heuristics for the acceleration of the higher-order methods [11].

What we propose is an adaptive scaled acceleration technique that aims to work on any of these methods, no matter the convergence rate and independent of the starting point (i.e the initial approximation of the matrix inverse X_0). This is done iteratively with the goal of reducing the frobenius norm of the error per iteration.

We then outline algorithms that can be used to implement this technique in a time efficient manner. Finally, we test our technique against another acceleration technique for higher order methods [11].

II. Background and Related Work

The following are the terms that we'll use in the rest of the paper. We recommend the book 'Generalized inverses : theory and applications' [14] for an in-depth look at the topics covered here.

1) Input Matrix: We shall take $\mathbb{R}^{m\times n}$ to be the set of all $m \times n$ real rectangular matrices. Then $A \in \mathbb{R}^{m\times n}$ is the matrix that we want to find the generalized inverse for. I.e. it is the inverse to be used as the input

2) Generalized inverse: A generalized inverse of matrix A is a matrix $G \in \mathbb{R}^{n\times m}$ such that it has the following property.

$$AGA = A \quad (3)$$

Any matrix G that fulfills Eq. (3) would be termed as a generalized inverse. Thus, it becomes a class consisting various other inverses such as the Moore-Penrose inverse, the Drazin inverse, the Regular Inverse etc.

3) Approximate inverse: We have a matrix $X \in \mathbb{R}^{n\times m}$ that is a generalized inverse with some error. Thus, from (3) we can say that if

$$\|A - AGA\| = 0 \quad (4)$$

Then, we can say

$$\|A - AXA\| = \epsilon \tag{5}$$
$$\|I - AX\| = \epsilon_{right} \tag{6}$$
$$\|I - XA\| = \epsilon_{left} \tag{7}$$

Where ϵ,ϵ_{left} and ϵ_{right} are the error tolerance and Eq. (6) and (7) are for right and left approximate inverses respectively.

The Schulz-type methods use a starting approximate inverse X_0 in order to guarantee convergence governed by

$$X_0 = \mu A^T \tag{8}$$

Where A^T is the transpose of the input matrix and μ is an appropriate scalar that is sufficiently small [15], [16]. For example, we calculate μ for our numerical experiments by $\mu = \frac{1}{\|A\|_1 \|A\|_\infty}$ [13]. It should be noted that there are other ways to calculate μ [15], [16]

A. Related Work

A quadratically convergent method to compute approximate matrix inverses was first laid out by Günther Schulz in 1933 [7] and Harold Hotelling [8] in 1943. Ben-Israel and Cohen in 1966 [16] specified the conditions to formulate the initial approximate matrix X_0 .

In 1991, Pan and Schreiber published their seminal paper [13] on this method with multiple contributions that include : the most optimal initial approximate matrix X_0, a scaled acceleration technique that was proved to be optimal (but only applicable to when $\rho = 2$), etc.

Around this time, mathematicians also discovered the larger family of methods that calculate generalized matrix inverses with a high order of convergence ($\geq$ 2) [4]–[6]. This has lead to people in recent years creating more efficient ways to compute such methods. In 2013 Gonzales et al. wrote a paper [17] in which they reduced the error in a real approximate inverse using frobenius norm minimization. A part of their work bears some resemblance to how we find the scaling parameter for our first technique (although derived differently). However, their work is in pure theoretical domain and didn't consider it from an acceleration technique point of view.

In 2015, Stanimirovic et al. [11] proposed various acceleration heuristics that could work with the Schulz-type methods of various orders of convergence. Our work has the same motivation, but we have a different approach and a focus on real matrices in particular.

III. The acceleration techniques

Our objective here, broadly speaking, is to try to minimize the error in the approximate inverse that is produced iteratively. The way we do this to have a scalar value be multiplied to either the end result of the i$^{\text{th}}$ iteration's approximate inverse X_{i+1} , or to the difference between X_{i+1} and X_i. Thus, we can scale on the iterate X_{i+1} with a scalar ψ_{i+1}, such that,

$$X'_{i+1} = \psi_{i+1} X_i \tag{9}$$
$$\|A - AX'_{i+1}A\|_F \leq \|A - AX_{i+1}A\|_F \tag{10}$$

Where F denotes the Frobenius norm. Or we can scale on the difference between two successive iterates Δ_{i+1} with scalar ω_{i+1}, such that

$$\Delta_{i+1} = X_{i+1} - X_i \tag{11}$$
$$X^*_{i+1} = X_i + \omega_{i+1}\Delta_{i+1} \tag{12}$$
$$\|A - AX^*_{i+1}A\|_F \leq \|A - AX_{i+1}A\|_F \tag{13}$$

We can also apply (9) and (12) for other more specific types of generalized inverses. Eg : For regular/right inverses :

$$\|I - AX'_{i+1}\|_F \leq \|I - AX_{i+1}\|_F \tag{14}$$
$$\text{and } \|I - AX^*_{i+1}\|_F \leq \|I - AX_{i+1}\|_F \tag{15}$$

For left inverses

$$\|I - X'_{i+1}A\|_F \leq \|I - X_{i+1}A\|_F \tag{16}$$
$$\text{and } \|I - X^*_{i+1}A\|_F \leq \|I - X_{i+1}A\|_F \tag{17}$$

In the next subsections, we'll see how to calculate the values of ψ_{i+1} and ω_{i+1}.

A. Technique 1 : Scaling on the iterate (Scale on X)

We shall first focus on computing ψ_{i+1} for the case of right/regular inverses and then generalize it for others. Here A is a $m \times n$ Matrix of real values and X is a $n \times m$ matrix. If $m = n$ and A is non-singular, then we arrive at the special case of the right inverse being a regular inverse as well. We substitute Eq. (9) in (14) and square it to get , we get

$$\|I - \psi_{i+1}AX_{i+1}\|_F^2 \leq \|I - AX_{i+1}\|_F^2 \tag{18}$$

This shall help us simplify some calculations in the future. Now let us define Y_{i+1}, an $m \times m$ matrix, as

$$Y_{i+1} = AX_{i+1} \tag{19}$$
$$\text{such that, } \|I - \psi_{i+1}Y_{i+1}\|_F^2 \leq \|I - AX_{i+1}\|_F^2 \tag{20}$$

First, we can see that $I - \psi_{i+1}Y$ yields us something like the below

$$I - \psi_{i+1}Y_{i+1} = \begin{pmatrix} 1 - \psi_{i+1}y_{11}^{(i+1)} & \cdots & -\psi_{i+1}y_{1m}^{(i+1)} \\ -\psi_{i+1}y_{21}^{(i+1)} & \cdots & -\psi_{i+1}y_{2m}^{(i+1)} \\ \vdots & \cdots & \vdots \\ -\psi_{i+1}y_{m1}^{(i+1)} & \cdots & 1 - \psi_{i+1}y_{mm}^{(i+1)} \end{pmatrix}$$

Denoting the matrices in their elemental forms, we get

$$Y_{i+1} = (y_{pq}^{(i+1)}) \tag{21}$$
$$I = (\delta_{pq}) \tag{22}$$

Where p and q both range from 1 to m (no. of max rows) and δ_{pq} refers to the Kronecker delta here.

Using notations from Eq. (21) and (22), we calculate the Frobenius norm square

$$\|I - \psi_{i+1}Y_{i+1}\|_F^2 = \sum_{p=1}^{m}\sum_{q=1}^{m}(\delta_{pq} - \psi_{i+1}y_{pq}^{(i+1)})^2 \tag{23}$$

It is crucial to point out that because we are dealing with real numbers, the absolute squares (calculated by the Frobenius Norm) equals to just computing the squares (i.e. you needn't take the absolute value).
Simplifying the Kronecker deltas, we get

$$\|I - \psi_{i+1}Y_{i+1}\|_F^2 = \sum_{p=1}^{m}(1 + \psi_{i+1}^2 {y_{pp}^{(i+1)}}^2 - 2\psi_{i+1}y_{pp}^{(i+1)}) + \sum_{p=1}^{m}\sum_{q=1}^{m}{}_{p\neq q}(\psi_{i+1}^2 {y_{pq}^{(i+1)}}^2) \tag{24}$$

We can rearrange this equation in the form of a quadratic function with ψ_{i+1} as the unknown

$$\|I - \psi_{i+1}Y_{i+1}\|_F^2 = \psi_{i+1}^2 \sum_{p=1}^{m}\sum_{q=1}^{m} {y_{pq}^{(i+1)}}^2 - 2\psi_{i+1}\sum_{i=1}^{m} y_{pp}^{(i+1)} + \sum_{p=1}^{m} 1 \tag{25}$$

Thus if we consider

$$\|I - \psi_{i+1}Y_{i+1}\|_F^2 = f(\psi_{i+1}) = \alpha_{i+1}\psi_{i+1}^2 + \beta_{i+1}\psi_{i+1} + \gamma_{i+1} \tag{26}$$

$$\alpha_{i+1} = \sum_{p=1}^{m}\sum_{q=1}^{m} {y_{pq}^{(i+1)}}^2 \tag{27}$$

$$\beta_{i+1} = -2\sum_{p=1}^{m} y_{pp}^{(i+1)} = -2Tr(Y_{i+1}) \tag{28}$$

$$\gamma_{i+1} = \sum_{p=1}^{m} 1 = m \tag{29}$$

Then the values of $f(\psi_{i+1})$ and ψ_{i+1} (for which $f(\psi_{i+1})$ is the minimum) are given by the first derivative of the quadratic function. Alternatively the quadratic function can be thought of as a vertical concave-upwards parabola for which we find the coordinates of the vertex

$$f(\psi_{i+1}) = \gamma_{i+1} - \frac{\beta_{i+1}^2}{4\alpha_{i+1}} \tag{30}$$

$$\psi_{i+1} = -\frac{\beta_{i+1}}{2\alpha_{i+1}} \tag{31}$$

1) Left inverse special case: As for finding ψ_{i+1} for the left inverse, the main approach remains the same except for a few things. We redefine Eq. (19) and (20) as

$$Y_{i+1} = X_{i+1}A \tag{32}$$

$$\text{such that, } \|I - \psi_{i+1}Y_{i+1}\|_F^2 \leq \|I - X_{i+1}A\|_F^2 \tag{33}$$

Where Y_{i+1} is a $n \times n$ matrix. From this point on, the derivation for the values of ψ_{i+1} and the reduced Frobenius norm square is the same as for the right inverse case and we would use Eq. (30) and (31) here as well.

2) General inverse case: Combining Eq. (9) and (10) and squaring it, we get

$$\|A - \psi_{i+1}AX_{i+1}A\|_F^2 \leq \|A - AX_{i+1}A\|_F^2 \tag{34}$$

We define Y_{i+1} as a $m \times n$ matrix (same dimensions as A)

$$Y_{i+1} = AX_{i+1}A \tag{35}$$

$$\text{such that, } \|A - \psi_{i+1}Y_{i+1}\|_F \leq \|A - AX_{i+1}A\|_F \tag{36}$$

Expanding $A - \psi_{i+1}Y$, we get

$$A - \psi_{i+1}Y_{i+1} = \begin{pmatrix} a_{11} - \psi_{i+1}y_{11}^{(i+1)} & \cdots & a_{1n} - \psi_{i+1}y_{1n}^{(i+1)} \\ a_{21} - \psi_{i+1}y_{21}^{(i+1)} & \cdots & a_{2n} - \psi_{i+1}y_{2n}^{(i+1)} \\ \vdots & \cdots & \vdots \\ -a_{m1} - \psi_{i+1}y_{m1}^{(i+1)} & \cdots & a_{mn} - \psi_{i+1}y_{mn}^{(i+1)} \end{pmatrix}$$

Denoting A in its elemental form,

$$A = (a_{pq}) \tag{37}$$

Where p ranges from 1 to m and q ranges from 1 to n. Just like with Eq. (23), we can write out the Frobenius norm square of $A - \psi_{i+1}Y_{i+1}$ as

$$\|A - \psi_{i+1}Y_{i+1}\|_F^2 = \sum_{p=1}^{m}\sum_{q=1}^{n}(a_{pq}^2 + \psi_{i+1}^2 y_{pq}^{(i+1)2} - 2a_{pq}\psi_{i+1}y_{pq}^{(i+1)}) \tag{38}$$

Rewriting Eq. (26) for the general case, we get

$$f(\psi_{i+1}) = \alpha_{i+1}\psi_{i+1}^2 + \beta_{i+1}\psi_{i+1} + \gamma_{i+1} \tag{39}$$

$$\alpha_{i+1} = \sum_{p=1}^{m}\sum_{q=1}^{n} {y_{pq}^{(i+1)}}^2 \tag{40}$$

$$\beta_{i+1} = -2\sum_{p=1}^{m}\sum_{q=1}^{n} y_{pq}^{(i+1)}a_{pq} \tag{41}$$

$$\gamma_{i+1} = \sum_{p=1}^{m}\sum_{q=1}^{n} a_{pq}^2 \tag{42}$$

And the values for $f(\psi_{i+1})$ and ψ_{i+1} can be calculated by putting the above parameters in Eq. (30) and (31)

B. Technique 2 : Scaling on the difference between iterates (Scale on Δ)

Here, we do not scale on the iterate itself, but the difference between two successive iterates, as given by Eq. (11) and (12). As with the previous section, we shall start with finding ω_{i+1} for right/regular inverses

Substituting Eq. (12) in Eq. (15) and squaring it

$$\|I - A(X_i + \omega_{i+1}\Delta_{i+1})\|_F^2 \leq \|I - AX_{i+1}\|_F^2 \tag{43}$$

Now, Let us define Z_{i+1} as a $m \times m$ matrix

$$Z_{i+1} = A\Delta_{i+1} \tag{44}$$

$$Z_{i+1} = (z_{pq}^{(i+1)}) \tag{45}$$

Such that by combining Eq. (44),(19) and (12), we get

$$Y^*_{i+1} = Y_i + \omega_{i+1} Z_{i+1} \tag{46}$$

$$\|I - (Y_i + \omega_{i+1} Z_{i+1})\|_F^2 \leq \|I - AX_{i+1}\|_F^2 \tag{47}$$

We now calculate Eq. (43) as

$$\|I - (Y_i + \omega_{i+1} Z_{i+1})\|_F^2 = \sum_{p=1}^{m}\sum_{q=1}^{m}(\delta_{pq} - y_{pq}^{(i)} - \omega_{i+1} z_{pq})^2 \tag{48}$$

Expanding it and using the algebraic identity for $(a+b+c)^2$, we can get the Frobenius Norm square in the form of a quadratic function with ω_{i+1} as the unknown

$$\begin{aligned}\|I - (Y_i + \omega_{i+1} Z_{i+1})\|_F^2 &= f(\omega_{i+1}) \\ &= \alpha_{i+1}\omega_{i+1}^2 + \beta_{i+1}\omega_{i+1} + \gamma_{i+1}\end{aligned} \tag{49}$$

$$\alpha_{i+1} = \sum_{p=1}^{m}\sum_{q=1}^{m} {z_{pq}}^{(i+1)^2} \tag{50}$$

$$\beta_{i+1} = 2(\sum_{p=1}^{m}\sum_{q=1}^{m} y_{pq}^{(i)} z_{pq}^{(i+1)} - Tr(Z)) \tag{51}$$

$$\gamma_{i+1} = \sum_{p=1}^{m}\sum_{q=1}^{m} y_{pq}^{2^{(i)}} + m - 2Tr(Y) \tag{52}$$

Modifying Eq. (30) and (31) for the current case, we can get the reduced frobenius norm square and the acceleration scalar by $f(\omega_{i+1}) = \gamma_{i+1} - \frac{\beta_{i+1}^2}{4\alpha_{i+1}}$ and $\omega_{i+1} = -\frac{\beta_{i+1}}{2\alpha_{i+1}}$ respectively.

1) Left inverse special case: As with Scaling on the iterate scenario. Here we use $Y_i = X_i A$ and define Z as

$$Z = \Delta_{i+1} A \tag{53}$$

instead of $A\Delta_{i+1}$. The rest of the formulation would remain the same and Eq. (50) (51) and (52) would yield us the necessary values to find $f(\omega_{i+1})$ and ω_{i+1}

2) General inverse case: As with the General inverse case for the previous technique, we need to define Z_{i+1} in a manner compliant with Eq. (13)

$$Z_{i+1} = A\Delta_{i+1} A \tag{54}$$

Modifying Eq. (48) for the general case

$$\|A - (Y_i + \omega_{i+1} Z_{i+1})\|_F^2 = \sum_{p=1}^{m}\sum_{q=1}^{n}(a_{pq} - y_{pq}^{(i)} - \omega_{i+1} z_{pq})^2 \tag{55}$$

Expanding Eq. (55) for finding α_{i+1}, β_{i+1} and γ_{i+1}

$$\alpha_{i+1} = \sum_{p=1}^{m}\sum_{q=1}^{n} {z_{pq}}^{(i+1)^2} \tag{56}$$

$$\beta_{i+1} = 2(\sum_{p=1}^{m}\sum_{q=1}^{n} y_{pq}^{(i)} z_{pq}^{(i+1)} - \sum_{p=1}^{m}\sum_{q=1}^{n} a_{pq} z_{pq}^{(i+1)}) \tag{57}$$

$$\gamma_{i+1} = \sum_{p=1}^{m}\sum_{q=1}^{n}(a_{pq} - y_{pq}^{(i)})^2 \tag{58}$$

IV. Efficient computation of the techniques

A. Methods used

Before we begin with the algorithms that compute the acceleration techniques. We shall first lay out some of the Schulz-type methods that they can be used with. In the next section, we would be conducting some numerical experiments with these methods.

H3, a third order convergent method [10], is given by

$$X_{i+1} = X_i(3I - AX_i(3I - AX_i)) \tag{59}$$

PM 9 (P9 for short), a ninth order convergent method was laid out by Stanimirovic et al. [11]

$$\begin{aligned}T_1^{(i)} &= AX_i \\ T_2^{(i)} &= 3I + T_1^{(i)}(-3I + T_1^{(i)}) \\ T_3^{(i)} &= T_1^{(i)} T_2^{(i)} \\ X_{i+1} &= X_i T_2^{(i)}(-3I + T_3^{(i)}(-3I + T_3^{(i)}))\end{aligned} \tag{60}$$

And finally PM 11(P11 for short), which is a eleventh order convergent method was also outlined in the same paper as

$$\begin{aligned}R_i &= I - AX_i \\ X_{i+1} &= X_i[I + (R_i + R_i^2)(I + c * R_i^2 + R_i^4) \\ &\quad (I + d * R_i^2 + R_i^4)]\end{aligned} \tag{61}$$

where $c = 0.5(1 - \sqrt{5})$ and $d = 0.5(1 - \sqrt{5})$.

The way these methods are crafted makes it easier for us to find the α, β and γ for the right inverse. We shall re-interpret the above methods such that it is easier to find these acceleration parameters for the left-inverse without any additional matrix inverse.

Thus H3 becomes :

$$X_{i+1} = (3I - (3I - X_i A)X_i A)X_i \tag{62}$$

P9 becomes

$$\begin{aligned}T_1^{(i)} &= X_i A \\ T_2^{(i)} &= 3I + (-3I + T_1^{(i)})T_1^{(i)} \\ T_3^{(i)} &= T_2^{(i)} T_1^{(i)} \\ X_{i+1} &= (-3I + (-3I + T_3^{(i)})T_3^{(i)})T_2^{(i)} X_i\end{aligned} \tag{63}$$

As for P11, the only change is that $R_i = I - X_i A$, the second part of the method remains the same.

When it comes to finding the acceleration parameters in the case of the general inverse, we'll need an additional multiplication per iteration to get $Y_{i+1} = AX_{i+1}A$

B. Stopping condition(s)

Here we shall discuss our choice of the stopping condition for the Schulz-type method along with our acceleration technique. In 2015, Stanimirovic et al. gave the following stable stopping condition

$$\frac{\|X_{i+1} - X_i\|}{\rho^i \mu} < \epsilon_1 \tag{64}$$

Where ϵ_1 is a chosen tolerance of the difference between two successive iterates. Our acceleration techniques are focused on reducing the error in the approx. inverse X such that X is ever more closer to the ideal inverse. Thus the stopping condition that we choose is

$$\text{left inverse, } \|I - X_{i+1}A\| < \epsilon_2 \tag{65}$$

$$\text{right inverse, } \|I - AX_{i+1}\| < \epsilon_2 \tag{66}$$

$$\text{general inverse, } \|A - AX_{i+1}A\| < \epsilon_2 \tag{67}$$

Where ϵ_2 is the error tolerance in the approximate inverse. The idea behind these stopping conditions is that it guarantees us an approximate inverse that is below a set error threshold. The condition given in Eq. (64) is more optimized for giving a stopping point when the error difference between two iterates goes below ϵ_1. It however, does not focus on demanding a particular error tolerance in the approximate inverse X.

C. Algorithms

Here, we shall suggest algorithms that can calculate the acceleration parameters in an efficient manner. For the sake of brevity, we shall consider the right inverse case and mention in the discussion about how to modify it for the left inverse and the general case.

procedure SCALE ON X ACC.(A)
  Compute the scalar μ as governed by [13], [15], [16]
  Compute the initial approximation $X = \mu * A^T$
  loop
    $Y = AX$
    if not first iteration **then**
      if $\|I - Y\| < \epsilon_2$ **then**
        Break
      end if
      Calculate α and β by using Eq. (27) and (28)
      Calculate $\psi = -\frac{\beta}{2\alpha}$
      Compute $X = \psi X$ and $Y = \psi Y$
    end if
    Compute the next X by Eq. (1) or one of it's specific methods such as Eq. (59), (60) or (61).
  end loop
  return X ▷ Final approximate inverse
end procedure

We see here that the iterate X_{i+1} calculated as a result of the i$^{\text{th}}$ iteration has its acceleration parameters calculated and applied in the first part of the (i+1)$^{\text{th}}$ iteration. The algorithm also computes $Y = AX$ before the stopping condition and would be used in the calculation of the next X. Thus the algorithm in its entire runtime does only one extra matrix multiplication.

If we were to compute the left inverse, we'd put Eq. (65) for the stopping condition, define $Y = XA$ and use either (62), (63) or (61) for the iteration. The general inverse case would just change the stopping condition to (67) and the calculation of Y would be $Y_{temp} = AX$, $Y = Y_{temp}A$. The methods to be used for the actual iteration are same as in the right inverse case.

The cost of calculating the acceleration parameters for the right/left/regular inverse case is a total of $2n^2 + n$. Whereas for the general case it is $4n^2$. Now, we present an algorithm for the second acceleration technique,

procedure SCALE ON Δ ACC.(A)
  Compute the scalar μ as governed by [13], [15], [16]
  Compute the initial approximation $X = \mu * A^T$
  loop

    if first iteration **then**
      $Y = AX$
    else
      $Y = Y + \omega Z$
      if $\|I - Y\| < \epsilon_2$ **then**
        Break
      end if
    end if
    Compute X_{temp} by Eq. (1) or one of it's specific methods such as Eq. (59), (60) or (61).
    $\Delta = X_{temp} - X$
    $Z = A\Delta$
    Calculate α and β by using Eq. (50) and (51)
    Calculate $\omega = -\frac{\beta}{2\alpha}$
    Compute the new approx. inverse $X = X + \omega\Delta$
  end loop
  return X ▷ Final approximate inverse
end procedure

The challenge with the second acceleration technique is the need to calculate the Matrix Z for finding the acceleration parameters. This could mean doing an additional matrix multiplication per iteration. However, this is alleviated by then computing $Y = Y + \omega Z$ instead of $Y = AX$ for all the iterations but the first. Hence, this means instead of doing an operation in the scale of $n^{2.3729}$ (via coppersmith-winograd [18], other matrix multiplication algorithms are even more expensive) we are doing an additional $4n^2$ per iteration for the improved X and Y ($n^{2.3729} > 4n^2$ for $n \geq 43$).

The left inverse case would require us to start the first iteration with $Y = XA$ and (65) as the stopping condion. Z would need to be calculated as $Z = \Delta A$ per iteration. The actual iterative methods would be calculated by (62), (63) or Eq. (61).

For the general inverse case, the first iteration will require Y to be calculated by $Y_{temp} = AX$, $Y = Y_{temp}A$, while the rest of the iterations it would be $Y_{temp} = Y_{temp} + \omega Z_{temp}$ and $Y = Y + \omega Z$ (Z_{temp} is an intermediate matrix). Each iteration would require the matrix Z to be calculated by $Z_{temp} = A\Delta$, $Z = Z_{temp}A$. Finally, the stopping condition (67) is to be used here.

In the case of this particular acceleration technique, the cost of calculating the acceleration parameters is higher than the last one. It is about $4n^2 + n$ for the right/left/regular inverse case and $6n^2$ for the general case. However, the next section will show us that the benefits outweigh the costs for the most part.

V. Numerical experiments

Here we shall display the results of three experiments we did for calculating the Moore-Penrose inverse in order to see the performance of one of our acceleration technique (Scale on Δ) as compared to the baseline (i.e. no acceleration at all) and the second heuristic mentioned and tested in [11]. It is important to note that Examples 1 and 2 have been calculated in MATLAB™ and deal with dense matrices. Example 3 uses MATHEMATICA™ because in our experience, it handled sparse matrices better for our requirements. Another reason to use MATHEMATICA™ for Example 3 is because we are extending one of the experiments laid out in [11] and they used MATHEMATICA™ for it as well (example 2 in [11]) with its δ parameter set to 0.1.

We chose $\mu = \frac{1}{\|A\|_1 \|A\|_\infty}$ as the starting scalar and $\epsilon_2 = 10^{-10}$ as the error tolerance for all the experiments. We show the results in terms of Number of Iterations required to converge to ϵ_2 and the runtime (avg. of 3 runs) to do so. We use 'b/l' to denote using the Schulz-type method without any acceleration and 'Δ' to denote our 'Scale on Δ' technique. All the stopping conditions that we have used are computed using the Frobenius norm.

The computer that we observed these results was a laptop with Windows™ 8.1 (64 bit) OS, 8 GB of RAM and a 3rd gen Intel i7 Processor.

1) Example 1: Here we take 5 randomly generated uniform distribution matrices and 5 normal distribution matrices (also randomly generated) of varying dimensions. The main constraint here is that $m < n$.

TABLE I
Iterations to converge for Ex. 1 : B/L v. Ours(Δ)

Mat.	H2(b/l)	H2(Δ)	H3(b/l)	H3(Δ)	P9(b/l)	P9(Δ)	P11(b/l)	P11(Δ)
U1	23	12	15	9	8	6	7	5
U2	24	12	15	9	8	6	7	5
U3	25	13	16	9	8	6	7	6
U4	25	13	16	9	8	6	7	6
U5	25	13	16	9	8	6	8	6
N1	23	12	15	9	8	6	7	5
N2	23	17	15	10	8	6	7	5
N3	24	12	15	9	8	6	7	5
N4	24	13	15	9	8	6	7	6
N5	25	13	16	9	8	6	7	6

TABLE II
Iterations to Converge for Ex. 1 by [11] Heuristic 2

Mat.	H2	H3	P9	P11
U1	20	13	7	6
U2	20	13	7	6
U3	21	14	7	6
U4	21	14	7	6
U5	21	14	7	7
N1	20	13	7	6
N2	20	13	7	6
N3	20	13	7	6
N4	20	13	7	6
N5	21	14	7	6

TABLE III
Runtime (seconds) taken for Ex. 1: B/L v. Ours(Δ)

Mat.	H2(b/l)	H2(Δ)	H3(b/l)	H3(Δ)	P9(b/l)	P9(Δ)	P11(b/l)	P11(Δ)
U1	2.3307	1.4960	2.1615	1.5747	2.1183	1.8113	1.4920	1.2907
U2	3.0037	1.9250	2.7169	1.8836	2.8350	2.2365	2.2138	1.6201
U3	3.6846	2.4024	3.6449	2.3328	3.3832	2.7295	2.7436	2.4150
U4	4.5419	2.9836	4.3671	2.8470	4.0648	3.1712	3.0872	2.5299
U5	5.7587	3.6123	5.3756	3.4263	4.7504	3.9572	4.7112	3.2193
N1	2.3622	1.5687	2.2206	1.5151	2.2470	1.8700	1.9259	1.5011
N2	3.0596	2.5753	2.8609	2.0443	2.7304	2.3058	2.4881	1.9142
N3	3.7756	2.1573	3.5621	2.2960	3.4267	2.7409	2.9623	2.2497
N4	4.5266	2.8257	3.9542	2.7534	4.2393	3.2946	3.4243	3.0658
N5	5.8408	3.5446	5.3896	3.5549	5.2444	4.1281	4.2435	3.9802

TABLE IV
Runtime (seconds) taken for Ex. 1 by [11], Heuristic 2

Mat.	H2	H3	P9	P11
U1	2.3763	2.126	2.0187	1.5504
U2	2.7905	2.526	2.3613	2.0642
U3	3.5829	3.285	3.0928	2.7263
U4	4.2266	4.332	4.1068	3.4059
U5	5.3595	4.877	4.7842	4.4300
N1	2.2854	2.098	2.1243	1.8413
N2	2.6597	2.624	2.5679	2.1480
N3	3.3143	3.235	3.0780	2.7779
N4	4.1404	3.778	3.8093	3.1261
N5	5.1205	4.974	4.6028	3.9628

All of the input matrices are dense with full rank. I.e. the Moore-Penrose Inverse equals to Right inverse. We use (66) as the stopping condition. The following commands were used for the uniform distribution matrix generation :

```
rng(12345);
Uniform_Mat=20000*rand(rowsize,colsize)
-10000;
```

And for the normal distribution matrix generation :

```
rng(12345);
Normal_Mat=10000*randn(rowsize,colsize);
```

The dimensions of the Matrices used are as follows : U1 & N1 = (1000,1100), U2 & N2 = (1100,1200), U3 & N3 = (1200,1300), U4 & N4 = (1300,1400) and U5 & N5 = (1400,1500).

For the most part, our acceleration technique (denoted by Δ in the table) outperforms baseline case and the acceleration heuristic given by [11] both in terms of iterations required and the runtime of the method.

2) Example 2: In this example, we also randomly generate uniform and normal distribution based dense matrices, However here we have $m > n$, as our generation constraint.

Being full rank but having more columns than rows, the Moore-Penrose Inverse becomes equal to the Left Inverse. We use (65) for the stopping condition. We used the same instructions for the input matrix generation A but the only difference is the seed that we use : rng(54321).

The dimensions of the Matrices used are as follows : U'1 & N'1 = (1100,1000), U'2 & N'2 = (1200,1100), U'3 & N'3

TABLE V
ITERATIONS TO CONVERGE FOR EX. 2 : B/L V. OURS(Δ)

Mat.	H2(b/l)	H2(Δ)	H3(b/l)	H3(Δ)	P9(b/l)	P9(Δ)	P11(b/l)	P11(Δ)
U'1	23	12	15	9	8	6	7	5
U'2	24	15	15	10	8	7	7	6
U'3	24	13	15	9	8	6	7	6
U'4	25	13	16	9	8	6	7	6
U'5	25	13	16	9	8	6	8	6
N'1	23	12	15	9	8	6	7	5
N'2	24	12	15	9	8	6	7	5
N'3	24	13	15	9	8	6	7	6
N'4	24	13	15	9	8	6	7	6
N'5	25	13	16	9	8	6	8	6

TABLE VI
ITERATIONS TO CONVERGE FOR EX. 2 BY [11] HEURISTIC 2

Mat.	H2	H3	P9	P11
U'1	19	13	7	6
U'2	20	13	7	6
U'3	20	13	7	6
U'4	21	14	7	6
U'5	21	14	7	7
N'1	19	13	7	6
N'2	20	13	7	6
N'3	20	13	7	6
N'4	20	13	7	6
N'5	21	14	7	7

= (1300,1200), U'4 & N'4 = (1400,1300) and U'5 & N'5 = (1500,1400).

TABLE VII
RUNTIME (SECONDS) TAKEN FOR EXAMPLE 2: B/L V. OURS(Δ)

Mat.	H2(b/l)	H2(Δ)	H3(b/l)	H3(Δ)	P9(b/l)	P9(Δ)	P11(b/l)	P11(Δ)
U'1	1.8893	1.0896	1.7295	1.0963	2.3075	1.8593	2.0000	1.5448
U'2	2.5705	1.6373	2.2499	1.6241	2.8979	2.6826	2.5338	2.26359
U'3	2.9834	1.7546	2.8420	1.7143	3.7101	2.8650	3.0393	2.7486
U'4	3.8972	2.2174	3.6081	2.0223	4.3327	3.4639	3.8288	3.2045
U'5	4.8264	2.5054	4.4665	2.4030	5.4248	4.0336	5.0953	3.9852
N'1	1.9412	1.1022	1.7496	1.1360	1.5680	1.2494	2.0845	1.4648
N'2	2.6222	1.3007	2.3358	1.3991	1.9667	1.5076	2.4826	1.9638
N'3	3.2528	1.8165	2.8593	1.6920	2.7501	2.1897	3.0628	2.8310
N'4	3.6637	2.0070	3.3516	1.9999	3.4127	2.5295	3.5749	3.3964
N'5	4.7498	2.6287	4.4893	2.5971	4.4924	2.9150	4.9497	4.0340

TABLE VIII
RUNTIME (SECONDS) TAKEN FOR EXAMPLE 2 BY [11], HEURISTIC 2

Mat.	H2	H3	P9	P11
U'1	1.8106	1.8000	2.1192	1.8259
U'2	2.4346	2.1485	2.6234	2.2532
U'3	2.8245	2.6787	3.1062	2.5362
U'4	3.6417	3.3251	3.8772	3.2653
U'5	4.0719	3.8660	4.6057	4.6172
N'1	1.9517	1.6689	1.7232	1.7295
N'2	2.4150	2.0002	1.9790	2.1409
N'3	2.9650	2.6535	2.6257	2.7052
N'4	3.4541	3.0373	3.1733	3.2339
N'5	4.1158	3.9206	3.7189	4.5722

As with the previous results, we see again that our acceleration technique is very competitive with the baseline method as well as heuristic 2 of [11].

3) Example 3: Here we shall be testing with five sparse matrices of size $m \times n = 10000 \times 11000$ that have the same format as that of [11]'s Example 1. It is important to note that our matrices only contain real numbers, but the bands in which they are present are same as in their paper. In other words, these are not the same matrices as in [11] but are inspired from it.

```
m = 10000; n = 11000; number = 5;
SeedRandom[12345];
B = Table[
 SparseArray[{Band[{2000, -4000},{m, n}]
 -> Random[], Band[{10, 6000}, {m, n}]->
 {1.1, -Random[]}, Band[{-200, 9000}]->
 -0.02, Band[{3500, -7000}] -> 0.1},
 {m, n}, 0.], {1, 5}];
```

None of these five matrices are fully ranked so we go with (67) as the stopping condition.

For the case of Sparse Matrices we can see that our technique does not always provide an accelerated convergence for all of the higher order methods in all of the cases. Nonetheless, it does converge to an approximate matrix below the error tolerance level. In the future, we'll look to analyze the particular kinds of sparse matrices our acceleration technique is effective for.

TABLE IX
ITERATIONS TO CONVERGE FOR EX. 3 : B/L V. OURS(Δ)

Mat.	H2(b/l)	H2(Δ)	H3(b/l)	H3(Δ)	P9(b/l)	P9(Δ)	P11(b/l)	P11(Δ)
M1	20	14	13	10	7	6	6	6
M2	31	19	20	13	10	7	9	7
M3	22	15	14	11	7	7	7	6
M4	20	14	13	10	7	6	6	6
M5	21	15	14	11	7	7	6	6

TABLE X
ITERATIONS TO CONVERGE FOR EX. 3 BY [11] HEURISTIC 2

Mat.	H2	H3	P9	P11
M1	17	-	-	-
M2	28	-	10	-
M3	21	-	7	7
M4	-	13	7	6
M5	20	-	7	6

TABLE XI
RUNTIME (SECONDS) TAKEN FOR EXAMPLE 3: B/L V. OURS(Δ)

Mat.	H2(b/l)	H2(Δ)	H3(b/l)	H3(Δ)	P9(b/l)	P9(Δ)	P11(b/l)	P11(Δ)
M1	0.8696	0.7168	0.6888	0.6514	0.5920	0.5643	0.5073	0.5773
M2	2.2499	1.2065	1.3879	0.7985	0.8222	0.6324	0.7582	0.6077
M3	1.1274	1.0053	0.9526	0.8646	0.7365	0.8085	0.7511	0.7238
M4	0.8532	0.7081	0.6808	0.6014	0.7785	0.5467	0.6874	0.7425
M5	0.8542	0.7511	0.7325	0.6768	0.5850	0.6451	0.5033	0.5627

While testing with the heuristic 2 given in [11] however, not all of the test cases converged (marked with a '-'). It is important to note that the tests they performed in their paper were with complex matrices and not real ones.

TABLE XII
RUNTIME (SECONDS) TAKEN FOR EXAMPLE 3 BY [11], HEURISTIC 2

Mat.	H2	H3	P9	P11
M1	0.7705	-	-	-
M2	1.6064	-	0.8409	-
M3	1.1898	-	0.7695	1.0740
M4	-	0.7381	0.6004	0.6841
M5	0.8829	-	0.5960	0.5223

VI. CONCLUSION

In this paper, we proposed two scaled acceleration techniques for the Schulz-type methods for computing generalized inverses. We then performed some numerical experiments to compare our Scale on Δ technique against the baseline methods and a competing technique. We observed that this technique is very particularly efficient for dense matrices but also works for cases of sparse matrices as well. In future, it would be interesting to see if our techniques can be refined to work better with sparse matrices (at the moment it works with about 70% of the cases we tested, by iterations). Another possible future work would be to implement these techniques in a parallel computing environment to reduce some of the computational cost associated with these techniques.

ACKNOWLEDGMENT

The authors would like to thank their colleagues from UMBC that gave valuable feedback regarding this work.

REFERENCES

[1] H. Lipfert, "Mimo ofdm space time coding–spatial multiplexing, increasing performance and spectral efficiency in wireless systems, part i technical basis (technical report)," *Institut für Rundfunktechnik*, 2007.

[2] I. V. Oseledets and E. E. Tyrtyshnikov, "Approximate inversion of matrices in the process of solving a hypersingular integral equation," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, vol. 45, no. 2, pp. 315–326, 2005.

[3] P. S. Stanimirović, S. Chountasis, D. Pappas, and I. Stojanović, "Removal of blur in images based on least squares solutions," *Mathematical Methods in the Applied Sciences*, vol. 36, no. 17, pp. 2280–2296, 2013.

[4] J.-J. Climent, N. Thome, and Y. Wei, "A geometrical approach on generalized inverses by neumann-type series," *Linear algebra and its applications*, vol. 332, pp. 533–540, 2001.

[5] F. Soleymani, P. S. Stanimirović, and F. K. Haghani, "On hyperpower family of iterations for computing outer inverses possessing high efficiencies," *Linear Algebra and its Applications*, vol. 484, pp. 477–495, 2015.

[6] E. Stickel, "On a class of high order methods for inverting matrices," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 67, no. 7, pp. 334–336, 1987.

[7] G. Schulz, "Iterative Berechung der reziproken matrix," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 13, no. 1, pp. 57–59, 1933.

[8] H. Hotelling, "Some new methods in matrix calculation," *The Annals of Mathematical Statistics*, vol. 14, no. 1, pp. 1–34, 1943. [Online]. Available: http://www.jstor.org/stable/2235999

[9] F. K. Haghani and F. Soleymani, "A new high-order stable numerical method for matrix inversion," *The Scientific World Journal*, vol. 2014, 2014.

[10] H.-B. Li, T.-Z. Huang, Y. Zhang, X.-P. Liu, and T.-X. Gu, "Chebyshev-type methods and preconditioning techniques," *Applied Mathematics and Computation*, vol. 218, no. 2, pp. 260–270, 2011.

[11] P. S. Stanimirović, F. Soleymani, and F. K. Haghani, "Computing outer inverses by scaled matrix iterations," *Journal of Computational and Applied Mathematics*, vol. 296, pp. 89–101, 2016.

[12] V. Pan, F. Soleymani, and L. Zhao, "Highly efficient computation of generalized inverse of a matrix," *arXiv preprint arXiv:1604.07893*, 2016.

[13] V. Pan and R. Schreiber, "An improved newton iteration for the generalized inverse of a matrix, with applications," *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 5, pp. 1109–1130, 1991.

[14] A. Ben-Israel and T. N. Greville, *Generalized inverses: theory and applications*. Springer Science & Business Media, 2003, vol. 15.

[15] A. Ben-Israel, "An iterative method for computing the generalized inverse of an arbitrary matrix," *Mathematics of Computation*, pp. 452–455, 1965.

[16] A. Ben-Israel and D. Cohen, "On iterative computation of generalized inverses and associated projections," *SIAM Journal on Numerical Analysis*, vol. 3, no. 3, pp. 410–419, 1966.

[17] L. González and A. Suárez, "Improving approximate inverses based on frobenius norm minimization," *Applied Mathematics and Computation*, vol. 219, no. 17, pp. 9363–9371, 2013.

[18] V. V. Williams, "Breaking the coppersmith-winograd barrier," *E-mail address: jml@ math. tamu. edu*, 2011.