UPPSALA
UNIVERSITET

PROJEKTRAPPORT

Scaling Inverse Factorization Iterations

Authors:
Daniel Berggren
Ismail Aso Abbas
Omar Ghulam Ahmed Malik
Supervisor:
Emanuel Rubensson
Project in Computational Science: Report

February 2022

INSTITUTIONEN FÖR INFORMATIONSTEKNOLOGI

Contents

1 Introduction

In several scientific disciplines, one needs to solve the generalized eigenvalue problem:

$$Fc = \lambda Sc, \tag{1}$$

where F and S are square matrices, c is a column vector, and λ is an eigenvalue — a scalar constant — of the generalized eigenvalue problem (1). The generalized eigenvalue problem can arise in, for example, electronic structure calculations within quantum chemistry. A common way to solve Eq. (1) is to transform to another basis in which S becomes orthonormal; the generalized eigenvalue problem turns into the *standard* eigenvalue problem in the new basis. It is easier to solve the standard eigenvalue problem compared to the generalized one. The structure and hence properties of S determine if and how this transformation can be carried out.

We are particularly interested in the case where S is symmetric positive definite (SPD), meaning that S has strictly positive eigenvalues: such matrices can appear by construction in electronic structure calculations as well as problems in other scientific disciplines. When S is an SPD matrix, then its inverse S^{-1} exists and is also SPD. Since S^{-1} is SPD, it can be factorized as a product of a matrix Z and its transpose Z^T:

$$S^{-1} = ZZ^T. \tag{2}$$

In other words, Z is a factor of S^{-1}, and is alternatively called the *inverse factor* of S; one would like to compute Z directly from S instead of explicitly computing the inverse S^{-1}, a highly costly operation. With Z, one can transform to the new basis where the generalized eigenvalue problem in (1) turns into the standard one

$$\tilde{F}\tilde{c} = \lambda\tilde{c}, \tag{3}$$

where

$$\tilde{F} = Z^T F Z, \quad Z\tilde{c} = c, \quad \text{and} \quad Z^T S Z = I. \tag{4}$$

In other words, when transforming to the new basis, F, c, and S are transformed into $\tilde{F}$, $\tilde{c}$, and I, respectively.

There are several ways to compute the inverse factor Z when S is SPD. The method we were concerned with was originally proposed by Niklasson, and was called *iterative refinement* [1]. With this method, one starts with an approximate inverse factor Z_0 such that $||I - Z_0^T S Z_0|| < 1$. One can then iteratively refine Z_0 to form better

approximations Z_i to the true inverse factor Z, see Fig. 1. Later, Rubensson et. al. showed that Niklasson's iterative refinement can be derived from iterative methods to compute the sign of a particular matrix: one such method uses the Newton-Schulz polynomial [2, 3]. Furthermore, Rubensson developed a scale-and-fold technique to speed up iterations in computing the density matrix — using the McWeeny polynomial — in electronic structure calculations [4]. The Newton-Schulz polynomial has stable fixed points at $x = -1$ and $x = 1$ and an unstable fixed point at $x = 0$. The McWeeny polynomial has stable fixed points at $x = 0$ and $x = 1$ and an unstable fixed point at $x = 0.5$. It can thus be shown that the Newton-Schulz and McWeeny polynomials are equivalent up to scaling and shifting between the domains [-1,1] and [0,1]. Therefore, our goal in this project is to adapt Rubensson's scale-and-fold technique to speed up the Newton-Schulz iterations in computing the matrix sign function, and then utilize the connection between the Newton-Schulz and iterative refinement iterations to speed up the iterative refinement of an approximate inverse factor Z_0. We begin by defining functions of matrices as an extension of the corresponding scalar functions, focusing on diagonalizable matrices.

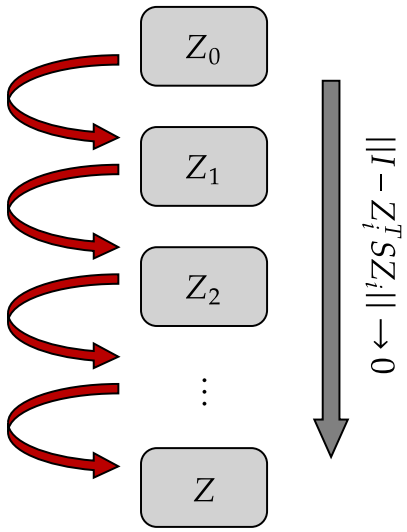


Figure 1: Schematic illustration of the iterative refinement scheme. When the initial error $||I - Z_0^T S Z_0||$ is less than 1, the subsequent errors $||I - Z_i^T S Z_i||$ decrease with each iteration i.

2 Theory

2.1 Functions of matrices

Recall that matrices represent linear transformations from one vector space into another vector space. A linear transformation caused by a matrix $A \in \mathbb{C}^{n\times n}$ can be resolved into a set of simpler, more computationally feasible linear transformations by considering the eigenvalues and eigenvectors of A. Since the eigenvalues are scalars, one can define *matrix functions* — which take A as input — by considering the corresponding *scalar functions* on the eigenvalues. We are particularly interested in the case where A is *diagonalizable*: that is, there exist $Y = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]$ and $D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ such that

$$AY = YD \quad \text{and} \quad A = YDY^{-1}, \tag{5}$$

where λ_i and $\mathbf{y}_i$ are the ith eigenvalue and eigenvector of A, respectively, with $i = 1, 2, \ldots, n$. When A is diagonalizable, one can define the matrix function $f(A)$ as

$$f(A) = Yf(D)Y^{-1}, \quad f(D) = \text{diag}\left(f(\lambda_1), f(\lambda_2), \ldots, f(\lambda_n)\right). \tag{6}$$

In other words, the eigenvalues of $f(A)$ are simply the scalar function $f(\lambda_i)$ evaluated at the eigenvalues of A themselves, and both A as well as $f(A)$ have the *same* eigenvectors when A is diagonalizable. In order for the matrix function $f(A)$ to be defined, $f(\lambda_i)$ must be defined for *all* eigenvalues of A. In the case of this project, we are specifically interested in computing the sign of a matrix using the matrix *sign function*.

2.2 Matrix sign function

The matrix sign function extends from the scalar sign function. For a given complex number z, the scalar sign function is defined as:

$$\text{sign}(z) = \begin{cases} 1, & \text{Re } z > 0, \\ -1, & \text{Re } z < 0. \end{cases} \tag{7}$$

In this project, we are interested in $\text{sign}(A)$ for the case where A is diagonalizable. According to the definition of a matrix function in Eq. (6), applied to our case with the function being the sign function, the sign of A is given by

$$\text{sign}(A) = Y\text{sign}(D)Y^{-1}, \tag{8}$$

$$\text{sign}(D) = \text{diag}(\text{sign}(\lambda_1), \text{sign}(\lambda_2), ..., \text{sign}(\lambda_n)). \tag{9}$$

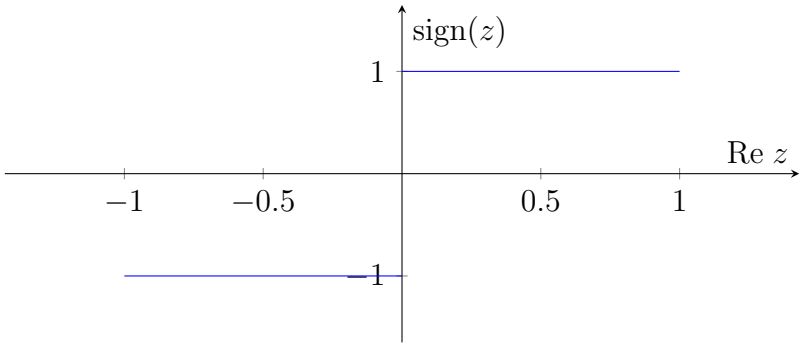


Figure 2: The scalar sign function for $z \in \mathbb{C}$. Sign(z) is undefined on the imaginary axis, i.e Re $z = 0$, where it shows a discontinuous jump from -1 to 1.

Let the dimension of A and D be n, and i = 1,2,..,n. The diagonal of D then consists of the eigenvalues λ_i of A. The negative eigenvalues are varying in an open set on the left-hand side of the imaginary axis. Analogously, the positive eigenvalues are also located in an open set, but on the right-hand side of the imaginary axis. By definition, the eigenvalues of A must not lie on the imaginary axis. This is a necessity, otherwise the sign function cannot be applied on A. Depending on whether the eigenvalues are positive or negative the sign function maps them in the following manner:

$$\text{sign}(\lambda_i) = \begin{cases} 1, & \text{Re } \lambda_i > 0, \\ -1, & \text{Re } \lambda_i < 0. \end{cases} \tag{10}$$

When A has no eigenvalues on the imaginary axis, the sign function of A is thereby defined as

$$\text{sign}(A) = Y \begin{bmatrix} -I & 0 \\ 0 & I \end{bmatrix} Y^{-1}. \tag{11}$$

Note that in the diagonal matrix in Eq. (11) we gathered the mapped eigenvalues in identity matrices. The upper left block, $-I$, corresponds to the negative eigenvalues of A. The lower right block, I, corresponds to the positive eigenvalues.

Commonly in numerical problems, one computes the sign of a matrix iteratively. In the next section, we will discuss two such well-known iterative methods: the Newton iterations and the Newton-Schulz iterations.

2.3 Newton-Schulz iterations

The Newton iterations for the sign matrix, which are given by

$$X_{i+1} = \frac{1}{2}(X_i + X_i^{-1}), \tag{12}$$

require calculation of the inverse of the matrix X_i. To compute the exact inverse is problematic in numerical methods since it increments the computational cost. To avoid explicitly computing matrix inverses, we can instead use the Newton-Schulz iterations from the Newton iterations for the sign matrix by applying the substitution

$$X_i^{-1} \to X_i(2I - X_i^2) \tag{13}$$

into Eq. (12), which give the following relation,

$$\begin{aligned} X_{i+1} &= \frac{1}{2}(X_i + X_i(2I - X_i^2)) = \frac{1}{2}X_i(I + 2I - X_i^2) \\ &= \frac{1}{2}X_i(3I - X_i^2) = \frac{3}{2}X_i - \frac{1}{2}X_i^3. \end{aligned}$$

The Newton-Schulz iterations are hence given by

$$X_{i+1} = \frac{3}{2}X_i - \frac{1}{2}X_i^3. \tag{14}$$

Given an initial guess of X_0, the Newton-Schulz iterations converge when X_0 satisfies

$$||I - X_0^2|| \ < \ 1. \tag{15}$$

For each iteration i in Eq. (14), the eigenvalues of X_i are moved away from 0, towards either 1 or -1. In Fig. 3 we show the mapping of the eigenvalues for iterations $i = 0$ and $i = 3$, which we have arbitrarily chosen. The smallest eigenvalue pair in the eigenspectrum of X_i is denoted by β_i and $-\beta_i$, with $0 < \beta_i \leq 1$. Note that β_i increases with the number of iterations i.

The closer β_0 is to 0, the more ill-conditioned the problem is and the slower Eq. (14) converges. Scaling of X_i and therefore also of β_i can give faster convergence for the Newton-Schulz iterations. To speed up the Newton-Schulz iterations, we apply a scaling parameter to the matrix X_i in Eq. (14). The scaling parameter depends on β_i, see Section 2.5, where we discuss a technique for choosing the optimal scaling parameter. On the other hand, when β_0 is further away from 0 the problem is more well-conditioned. In such a case, the scaling will be weaker. Note that in this project,

all the eigenvalues come in *positive-negative* pairs — which we show in Section 2.6. In other words, we need to consider only the positive half: when the negative half is needed, we simply can set a minus sign in front of the positive half.

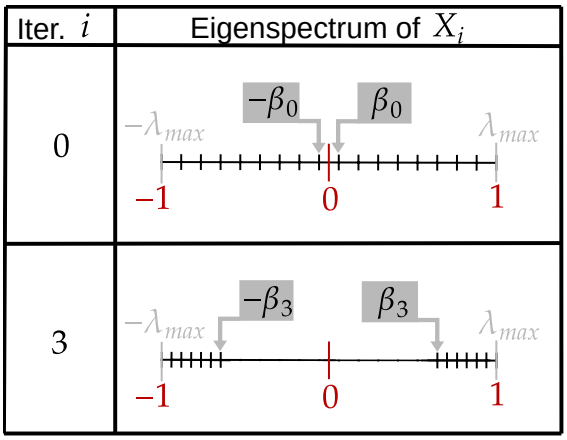


Figure 3: Illustration of how the eigenvalues of X_i are converging towards -1 and 1 after the Newton-Schulz iterations.

When the Newton-Schulz iteration in Eq. (14) is applied to a particular matrix, a useful result is obtained related to the iterative refinement problem, which we discuss in the next section.

2.4 Iterative refinement

We derive an expression for the iterative refinement of an approximate inverse factor Z_0 of a given SPD matrix S. Given an approximate inverse factor Z_0 of an SPD matrix S, then computing the sign of the matrix

$$X_0 = \begin{bmatrix} 0 & Z_0^* S \\ Z_0 & 0 \end{bmatrix} \tag{16}$$

will give us a matrix of the same form where the approximate inverse factor Z_0 is refined into the true inverse factor Z [2]:

$$\operatorname{sign}\left(\begin{bmatrix} 0 & Z_0^* S \\ Z_0 & 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & Z^* S \\ Z & 0 \end{bmatrix}. \tag{17}$$

Eq. (17) can be used to justify that iterative methods for the matrix sign function can be used to refine an approximate inverse factor Z_0 of an SPD matrix S. One such method is the Newton-Schulz iteration, where

$$X_i = \begin{bmatrix} 0 & Z_i^* S \\ Z_i & 0 \end{bmatrix} \tag{18}$$

is used in order to iteratively refine Z_0. We inserted Eq. (18) in Eq. (14) and obtain

$$\begin{bmatrix} 0 & Z_{i+1}^* S \\ Z_{i+1} & 0 \end{bmatrix} = \frac{3}{2} \begin{bmatrix} 0 & Z_i^* S \\ Z_i & 0 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 0 & Z_i^* S Z_i Z_i^* S \\ Z_i Z_i^* S Z_i & 0 \end{bmatrix}, \tag{19}$$

where the lower left blocks of the matrices relate Z_{i+1} to Z_i. As a result, we obtain the iterative refinement of the inverse factor Z_0:

$$Z_{i+1} = \frac{3}{2} Z_i - \frac{1}{2} Z_i Z_i^* S Z_i. \tag{20}$$

The unscaled iterative refinement is shown in Algorithm 1.

Algorithm 1 Unscaled iterative refinement of an approximate inverse factor

Input: Tolerance of the error tol, an SPD matrix S, and approximate inverse factor Z_0 such that $||I - Z_0^* S Z_0|| < 1$.

1: $\delta_0 = I - Z_0^* S Z_0$
2: $i = 0$
3: **while** $||\delta_i||_2 > tol$ **do**
4: $\quad Z_{i+1} = \frac{3}{2} Z_i - \frac{1}{2} Z_i Z_i^* S Z_i$
5: $\quad \delta_{i+1} = I - Z_{i+1}^* S Z_{i+1}$
6: $\quad i = i + 1$
7: **end while**
8: **return** Z_i

2.5 Scaled Newton-Schulz iterations using Rubensson's scale-and-fold technique

As discussed in Section 2.3, the closer β_0 is to 0, the slower the Newton-Schulz iterations in Eq. (14) converges to the sign matrix. Therefore, we scale X_i, and hence scale its eigenvalues — including β_i — before applying the Newton-Schulz iterations:

$$X_{i+1} = \frac{3}{2} \alpha_i X_i - \frac{1}{2} \alpha_i^3 X_i^3. \tag{21}$$

The scale-and-fold technique is used in order to choose the scaling parameter α_i. The method was developed by Rubensson for density matrix purification schemes using the McWeeny polynomial [4]. The McWeeny polynomial has two stable fixed points at 0 and 1, while the Newton-Schulz polynomial has them at -1 and 1. Furthermore, the McWeeny polynomial has an unstable fixed point at 0.5 and the Newton-Schulz polynomial at 0. A stable fixed point is a point where the absolute value of the derivative is *less* than 1 and the point is mapped to itself (i.e $f(a) = a$). In contrast to a stable fixed point, an unstable fixed point has a derivative whose absolute value is *greater* than 1. Note that the interval of interest for these polynomials is determined by their stable fixed points. Since the Newton-Schulz polynomial has a similar shape to the McWeeny polynomial, the scale-and-fold method could also be used to scale the Newton-Schulz iterations.

The idea behind Rubensson's scale-and-fold technique can be summarized in three steps: scaling the eigenvalues into the interval of interest, choosing the optimal scaling factor, and performing the iterations. These steps are covered in the following three sub-sections.

2.5.1 Scale eigenvalues of X_0 into desired interval

At this step the idea is to scale the eigenvalues of X_0 such that they are mapped into the desired interval. The eigenvalue closest to the fixed point is denoted as β_0. For the Newton-Schulz iteration the unstable fixed point is $x = 0$ and the interval of interest is $x \in [-1, 1]$. Hence the eigenvalues of X_0 were scaled as

$$X_0 := \frac{X_0}{\lambda_{max}}. \tag{22}$$

Similarly, β_0 was scaled as

$$\beta_0 := \frac{\beta_0}{\lambda_{max}}. \tag{23}$$

2.5.2 Choosing the optimal scaling factor α_i in each iteration

The polynomial chosen to perform the iterations needs to be investigated in order to find the optimal scaling parameter α_i. We begin with plotting the Newton-Schulz polynomial together with an arbitrary $\pm\lambda_{max}$, α_i and $\pm\ \beta_i$ in the interval $x \in [-2, 2]$:

From Fig. 4 we observe the scaling of eigenvalues before the iteration, thus moving all of the eigenvalues further away from 0 before they are inserted into the Newton-Schulz

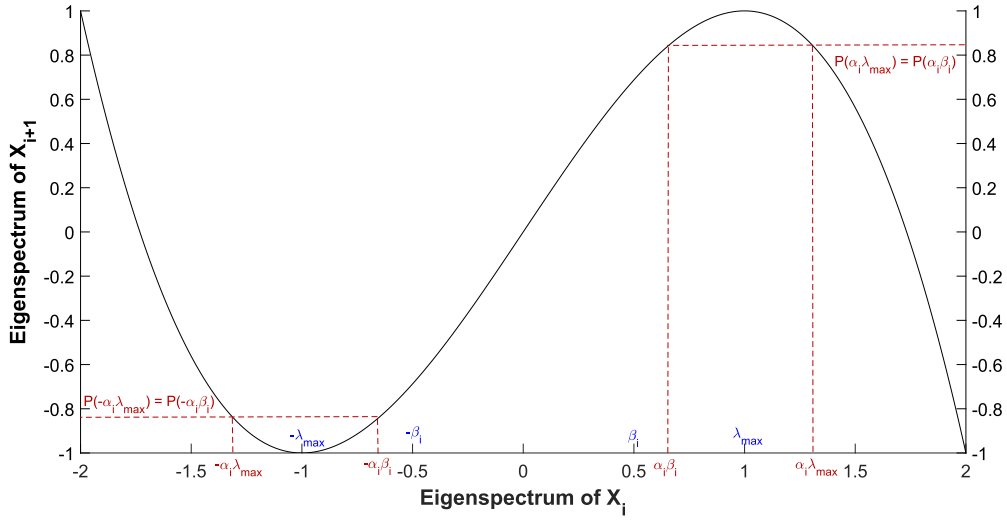


Figure 4: Newton-Schulz iteration: the x-axis represents the eigenspectrum of X_i and the y-axis represents the eigenspectrum of X_{i+1}. Here α_i is the scaling parameter, $\pm\beta_i$ are the closest eigenvalues to 0, $\pm\lambda_{max}$ are the extremal eigenvalues and the index i denotes the ith iteration. The unscaled eigenvalues $\pm\lambda_{max}$ and $\pm\,\beta_i$ are given in blue while the scaled eigenvalues $\pm\alpha_i\lambda_{max}$ and $\pm\,\alpha_i\beta_i$ are given in red.

iteration. Hence, λ_{max} is first scaled to be outside the desired interval, after which it is folded *back* into the interval when the Newton-Schulz iteration is applied. The difference between the scaled and unscaled Newton-Schulz methods for one iteration is illustrated in Fig. 5.

Fig. 4 and Fig. 5 indicate that the scaling needs to be done while considering the following:

1. After each iteration the criterion $P(\alpha_i\lambda_{max}) \geq P(\alpha_i\beta_i)$, should be satisfied in order to not make the scaled perform worse than the unscaled or having an eigenvalue converging to a wrong value: for example, a negative eigenvalue might otherwise converge to +1 instead of -1.

2. The optimal scaling is achieved when α_i is as large as possible and still satisfies $P(\alpha_i\lambda_{max}) \geq P(\alpha_i\beta_i)$.

Hence, the optimal α_i satisfies

$$P(\alpha_i\lambda_{max}) = P(\alpha_i\beta_i). \tag{24}$$

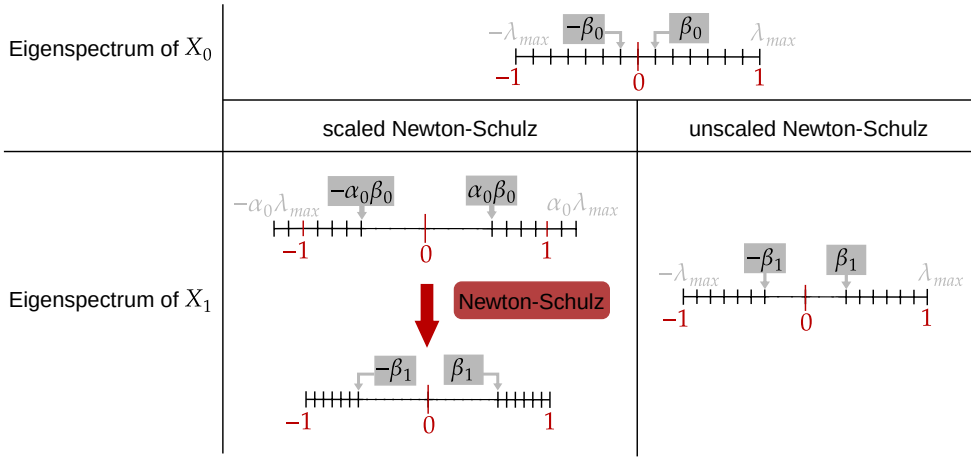


Figure 5: The difference between the scaled and unscaled Newton-Schulz iteration is illustrated, beginning with the initial matrix X_0 and doing one iteration for both cases.

Since we also have that $\lambda_{max} \approx 1$ during each iteration, we can obtain α_i as a function of β_i by solving Eq. (24) for α_i using the upper bound for λ_{max}. We therefore obtain the following expression for α_i:

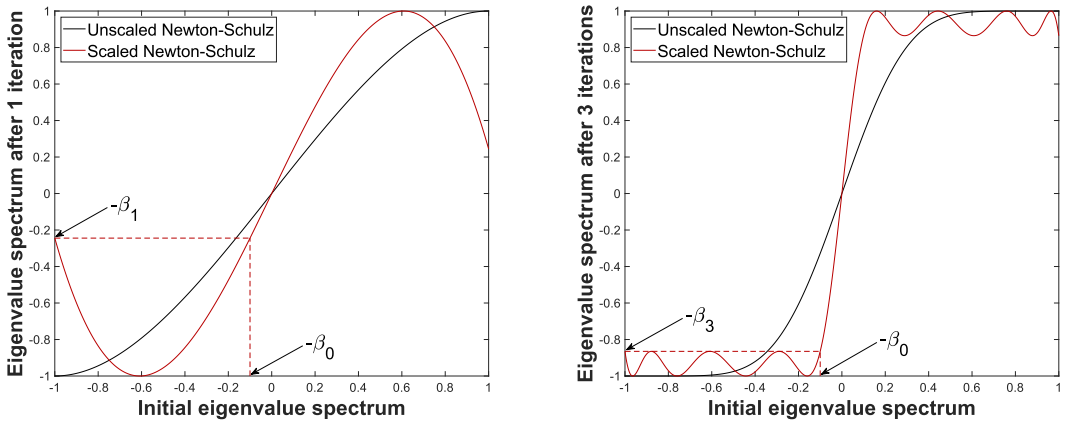
$$\alpha_i = \sqrt{\frac{3}{\beta_i^2 + \beta_i + 1}}. \tag{25}$$

2.5.3 Performing the scaled iterations

After having determined the scaling factor α_i, we can perform the scaled Newton-Schulz iterations. In these iterations α_i is updated according to Eq. (25). After α_i is updated, the scaled Newton-Schulz iteration in Eq. (21) is then applied. Lastly, β_{i+1} is updated through

$$\beta_{i+1} = \frac{3}{2}\alpha_i\beta_i - \frac{1}{2}\alpha_i^3\beta_i^3, \tag{26}$$

which is obtained by making the substitution $X_i \to \beta_i$ in Eq. (21). In order to tell if there is any speed up using the scaled Newton-Schulz iteration compared to the unscaled one, we compared the scaled Newton-Schulz with the unscaled one at two different iterations. From Fig. 6 we can clearly see that the repeated application of the scaled Newton-Schulz iterations approaches the desired sign function faster than the unscaled iterations.



(a) 1 iteration (b) 3 iterations

Figure 6: The red curve corresponds to the scaled Newton-Schulz iterations, and the black one to the unscaled iterations. The x-axis is the initial eigenspectrum of X while the eigenspectrum at the specific iteration is given by the y-axis. β_i is the closest eigenvalue to 0 at the ith iteration.

It is important to understand the scale-and-fold technique in order to develop a scaled method that iteratively refines Z_0, which is discussed in the next section.

2.6 Scaled iterative refinement

After obtaining an expression to iteratively refine an approximate inverse factor Z_0 of a matrix S, we want to speed up its computation. This is accomplished in a same manner as for the unscaled iterative refinement, namely by inserting Eq. (18) in Eq. (21), which reads as follows,

$$\begin{bmatrix} 0 & Z_{i+1}^* S \\ Z_{i+1} & 0 \end{bmatrix} = \frac{3}{2}\alpha_i \begin{bmatrix} 0 & Z_i^* S \\ Z_i & 0 \end{bmatrix} - \frac{1}{2}\alpha_i^3 \begin{bmatrix} 0 & Z_i^* S Z_i Z_i^* S \\ Z_i Z_i^* S Z_i & 0 \end{bmatrix}. \tag{27}$$

We use the same logic as the iterative refinement without the scaling factor α_i, and hence obtain an expression for the scaled iterative refinement by considering only the bottom-left block of Eq. (27):

$$Z_{i+1} = \frac{3}{2}\alpha_i Z_i - \frac{1}{2}\alpha_i^3 Z_i Z_i^* S Z_i. \tag{28}$$

Since the scaled iterative refinement is obtained from the scaled Newton-Schulz iterations, the optimal scaling factor α_i can be obtained in the same way as for the

scaled Newton-Schulz iterations. In order for the scaled iterative refinement to be successfully implemented, we also need to compute the closest eigenvalue of X_0 to 0, β_0, *without* constructing X_0 itself. In order to do so, we first note that the square of X_0 has a block-diagonal form:

$$X_0^2 = \begin{bmatrix} Z_0^* S Z_0 & 0 \\ 0 & Z_0 Z_0^* S \end{bmatrix}. \tag{29}$$

It is well known that the eigenvalues of a block-diagonal matrix are simply the list of eigenvalues of the individual blocks along the diagonal, and also that $Z_0^* S Z_0$ and $Z_0 Z_0^* S$ — i.e: the diagonal blocks themselves — have the *same* eigenvalues. As a result, the eigenvalues of X_0^2 are completely determined by the eigenvalues of either of its diagonal blocks; we chose to compute the eigenvalues of the upper-left block in Eq. 29, $Z_0^* S Z_0$. The eigenvalues of X_0 are therefore the square root of the eigenvalues of X_0^2 and hence $Z_0^* S Z_0$. Because S is SPD, it has strictly positive eigenvalues, so the eigenvalues of $Z_0^* S Z_0$ are guaranteed to be positive. If we denote an eigenvalue of X_0, X_0^2, and $Z_0^* S Z_0$ as λ^{X_0}, $\lambda^{X_0^2}$, and $\lambda^{Z_0^* S Z_0}$, respectively, then we have

$$\begin{aligned} \lambda^{Z_0^* S Z_0} = \lambda^{X_0^2} &= \left(\lambda^{X_0}\right)^2 \geq 0 \\ \lambda^{X_0} &= \pm\sqrt{\lambda^{Z_0^* S Z_0}}. \end{aligned} \tag{30}$$

In other words, for every eigenvalue of $Z_0^* S Z_0$, we have *two* corresponding eigenvalues of X_0, which come in *positive-negative* pairs according to Eq. (30). Namely, the maximum and minimum eigenvalues of X_0 are given by

$$\lambda_{max}^{X_0} = \max_{1 \leq h \leq N} \sqrt{\lambda_h^{Z_0^* S Z_0}}, \tag{31}$$

$$\lambda_{min}^{X_0} = -\lambda_{max}^{X_0}, \tag{32}$$

and the eigenvalue pair of X_0 closest to 0 are given by β_0 and $-\beta_0$, where

$$\beta_0 = \min_{1 \leq h \leq N} \sqrt{\lambda_h^{Z_0^* S Z_0}}. \tag{33}$$

With Eqs. (31)-(33), we are therefore able to compute the necessary eigenvalues of X_0 that are required to perform the scale-and-folding technique to accelerate the iterative refinement of Z_i — using only the bottom-left channel of X_i. The scaling factor at each iteration, α_i, is related to β_i in the same way as in Eq. (25). As Z_i is scaled and iterated using Eq. (28), β_i itself is scaled and iterated using the Newton-Schulz polynomial in Eq. (26). Algorithm 2 shows the scaled iterative refinement using Rubensson's scale-and-fold technique.

Algorithm 2 Scaled iterative refinement of an approximate inverse factor

Input: Tolerance of the error *tol*, an SPD matrix S, an approximate inverse factor Z_0 such that $||I - Z_0^* S Z_0|| < 1$, the eigenvalue closest to 0 β_0, and the maximum eigenvalue $\lambda_{max}^{X_0}$.

$\delta_0 = I - Z_0^* S Z_0$
$Z_0 := Z_0 / \lambda_{max}^{X_0}$
$\beta_0 := \beta_0 / \lambda_{max}^{X_0}$
$\alpha_0 = \sqrt{\frac{3}{\beta_0^2 + \beta_0 + 1}}$
while $||\delta_i||_2 > tol$ **do**
    $Z_{i+1} = \frac{3}{2}\alpha_i Z_i - \frac{1}{2}\alpha_i^3 Z_i Z_i^* S Z_i$
    $\delta_{i+1} = I - Z_{i+1}^* S Z_{i+1}$
    $\beta_{i+1} = \frac{3}{2}\alpha_i \beta_i - \frac{1}{2}\alpha_i^3 \beta_i^3$
    $\alpha_{i+1} = \sqrt{\frac{3}{\beta_i^2 + \beta_i + 1}}$
    $i = i + 1$
end while
return Z_i

3 Numerical experiments

To test our scaled iterative refinement, S is chosen to be a tridiagonal, SPD Toeplitz matrix of the form

$$S = \begin{bmatrix} 1 & c & & & & & \\ c & 1 & c & & & & \\ & c & 1 & . & & & \\ & & . & . & . & & \\ & & & . & . & . & \\ & & & & . & . & c \\ & & & & & c & 1 \end{bmatrix}, \quad 0 < c < 0.5. \tag{34}$$

Furthermore, the initial guess for the inverse factor Z_0 is chosen to be the identity matrix, $Z_0 = I$. As we see below, these choices for S and Z_0 allow to control the condition number of the system at will when testing the performance of our scaled iterative refinement. According to Eqs. (16) and (29), X_0 and X_0^2 in this case are given by

$$X_0 = \begin{bmatrix} 0 & S \\ I & 0 \end{bmatrix} \quad \text{and} \quad X_0^2 = \begin{bmatrix} S & 0 \\ 0 & S \end{bmatrix}. \tag{35}$$

The $2N$ eigenvalues of X_0 are thus completely determined by the N eigenvalues of S, as seen in Eq. (30):

$$\lambda_h^{X_0} = \pm\sqrt{\lambda_h^S}, \quad h = 1, 2, \ldots, N. \tag{36}$$

In addition to the eigenvalues, the condition number of X_0 is also completely determined by that of S. The structure of S given by Eq. (34) has an analytic expression for the eigenvalues, [5]:

$$\lambda_h^S = 1 + 2c\cos\left(\frac{h\pi}{N+1}\right), \quad 0 < c < 0.5, \quad h = 1, 2, \ldots, N. \tag{37}$$

After inserting Eq. (37) in Eq. (36), the eigenvalues of X_0 are therefore given by

$$\lambda_h^{X_0} = \pm\sqrt{1 + 2c\cos\left(\frac{h\pi}{N+1}\right)}, \quad 0 < c < 0.5, \quad h = 1, 2, \ldots, N. \tag{38}$$

Using Eq. (38) prevents the need to explicitly compute the eigenvalues of X_0 before starting the iterative refinement. Namely, Eq. (31) shows that the maximum eigenvalue of X_0 is the 1st,

$$\lambda_{max}^{X_0} = \lambda_1^{X_0} = \sqrt{1 + 2c\cos\left(\frac{\pi}{N+1}\right)}, \tag{39}$$

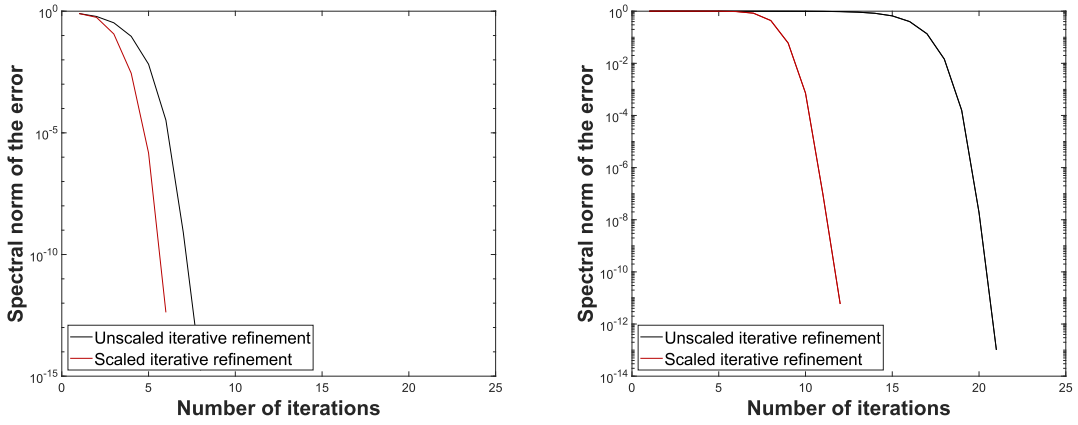
and Eq. (33) shows that β_0 is the Nth eigenvalue,

$$\beta_0 = \lambda_N^{X_0} = \sqrt{1 + 2c\cos\left(\frac{N\pi}{N+1}\right)}. \tag{40}$$

The condition number of S — and hence X_0 — is therefore adjusted by setting c closer to 0.5: this affects the value of β_0. To be able to set larger condition numbers and hence smaller β_0 values, larger system sizes N are tested as well.

4 Results and discussion

To recall, the aim of this project is to investigate if we can achieve a speedup, using Rubensson's scale-and-fold technique, of the Newton-Schulz iteration to iteratively refine an approximate inverse factor Z_0 of a symmetric positive definite matrix S.



(a) $c = 0.4,\ \text{cond}(S) = 9.00,$ $\beta_0 = 0.667$

(b) $c = 0.4999999,\ \text{cond}(S) = 4 \times 10^5,$ $\beta_0 = 0.003$

Figure 7: Error $||I - Z_i^T S Z_i||_2$ decrease during the iterative refinement of Z_i for (a) a well-conditioned S, and (b) an ill-conditioned S, where S is an SPD, tridiagonal toeplitz matrix and Z_0 is the identity matrix. Here $N = 1000$ and $tol = 10^{-10}$.

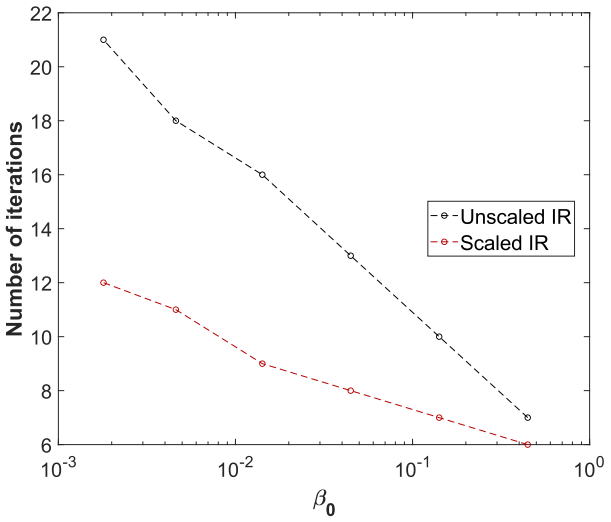


Figure 8: Number of iterations for the unscaled and scaled iterative refinements of Z_i to converge, plotted against size of β_0, the eigenvalue that is closest to 0. Here S is an SPD, tridiagonal toeplitz matrix, Z_0 is the identity matrix, $N = 2000$, $tol = 10^{-6}$, and $c \in \{0.4, 0.49, 0.499, 0.4999, 0.49999, 0.499999\}$.

We present results for a well-conditioned S and compare these to an ill-conditioned S.

Fig. 7 shows how the error in the unscaled and our scaled iterative refinements decreased during the iterations. For well-conditioned systems, our scaled method does not obtain a significant speedup over its unscaled counterpart. However, we observe significant speedup with our scaled method for more ill-conditioned systems. This falls in line with how the scaling parameter α_i is chosen during the iterations. Recall that α_i depends on the closest eigenvalue to 0 — that is, β_i — and is chosen such that β_i is pushed as far away from 0 as possible while ensuring that *all* of X_{i+1}'s positive and negative eigenvalues are bounded by β_{i+1} and $-\beta_{i+1}$, respectively. Consequently, when β_i approaches 1 during the iterations, α_i also approaches 1. Furthermore, the condition number of the system depends on β_0. Well-conditioned systems have a β_0 that is closer to 1 and hence will have a smaller α_0, resulting in a smaller speedup. In contrast, ill-conditioned systems have a β_0 that is closer to 0 and hence will have a larger α_0, resulting in a greater speedup. Additionally, Fig. 8 shows how the number of iterations until convergence changed with respect to the size of β_0 for both unscaled and scaled iterative refinement methods. We see that the scaled iterative refinement saves more iterations as β_0 gets closer to 0 — i.e: as the system becomes more ill-conditioned — and that the number of iterations is decreased by roughly half for highly ill-conditioned systems.

5 Conclusion

In conclusion, the aim of this project is to speed up the iterative refinement of an approximate inverse factor. This is done by first scaling the Newton-Schulz iterations for the sign matrix using Rubensson's scale-and-fold technique. Then, we use a connection between the Newton-Schulz iterations and the iterative refinement to speed up the iterations of the iterative refinement. We compare our iterative refinement method with the corresponding unscaled method. From the obtained results we observe that the scaled iterative refinement needs fewer iterations than the unscaled iterative refinement to converge to the desired tolerance. Furthermore, we can see that more ill-conditioned systems lead to a greater speedup when using the scaled iterative refinement compared to the unscaled counterpart. Specifically, the speedup depends on the condition number of the system; the more ill-conditioned the system, the greater the speedup.

6 References

[1] A. M. N. Niklasson, "Iterative refinement method for the approximate factorization of a matrix inverse," *Phys. Rev. B*, vol. 70, p. 193102, Nov 2004.

[2] E. H. Rubensson, A. G. Artemov, A. Kruchinina, and E. Rudberg, "Localized inverse factorization," *IMA Journal of Numerical Analysis*, vol. 41, pp. 729–763, Apr. 2020.

[3] N. Higham, *Functions of Matrices: Theory and Computation*. Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, 2008.

[4] E. H. Rubensson, "Nonmonotonic Recursive Polynomial Expansions for Linear Scaling Calculation of the Density Matrix," *Journal of Chemical Theory and Computation*, vol. 7, pp. 1233–1236, May 2011. Publisher: American Chemical Society.

[5] S. Noschese, L. Pasquini, and L. Reichel, "Tridiagonal Toeplitz matrices: properties and novel applications," *Numerical Linear Algebra with Applications*, vol. 20, no. 2, pp. 302–326, 2013.