Preprint ANL/MCS-P5059-0114

A STABLE SCALING OF NEWTON-SCHULZ FOR IMPROVING THE SIGN FUNCTION COMPUTATION OF A HERMITIAN MATRIX

JIE CHEN* AND EDMOND CHOW†

Abstract. The Newton-Schulz iteration is a quadratically convergent, inversion-free method for computing the sign function of a matrix. It is advantageous over other methods for high-performance computing because it is rich in matrix-matrix multiplications. In this paper we propose a variant for Hermitian matrices that improves the initially slow convergence of the iteration. The main idea is to scale the iteration to have steeper derivatives of the mapping function at the origin such that the convergence of the eigenvalues with small magnitudes is accelerated. The scaling is stable based on a backward stability result of Y. Nakatsukasa and N. J. Higham. Generally, the number of iterations is reduced by half compared with standard Newton-Schulz. With proper shifts of the matrix, this number may be further reduced. We demonstrate numerical calculations with matrices of size up to approximately 10^5 on medium-sized computing clusters and also apply the algorithm to electronic structure calculations.

Key words. Matrix sign function, Newton-Schulz iteration, electronic structure calculation

AMS subject classifications. 65F60

1. Introduction.

We are interested in numerically computing the matrix sign function

$$S = \mathrm{sign}(A)$$

for a matrix $A \in \mathbb{C}^{n\times n}$ with no eigenvalues lying on the imaginary axis. The scalar sign function $\mathrm{sign}(z)$ takes value $+1$ when $\Re(z) > 0$ and -1 when $\Re(z) < 0$. In this paper we focus on the Hermitian case, where a simplified definition of the matrix sign function is

$$\mathrm{sign}(A) = U \cdot \mathrm{diag}(\mathrm{sign}(\lambda_1), \ldots, \mathrm{sign}(\lambda_n)) \cdot U^*,$$

with $U^*AU = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ being a diagonalization of A. The Hermitian case appears in several real-life applications, such as lattice quantum chromodynamics [23, 29, 10] and electronic structure calculations [28, 26]. For the latter application, the density matrix ρ of a molecular system with Fermi level μ is the Heaviside function of $\mu I - H$, where H is an approximation to the Hamiltonian. Thus, one can compute ρ as $\frac{1}{2}[\mathrm{sign}(\mu I - H) + I]$. To respect the notational convention in different fields, we recycle the notation ρ, μ, and H for the discussion of matrix functions, where the different meanings are clear in context.

Four iterative methods for computing $\mathrm{sign}(A)$ relevant to the present paper are Newton [12, 11], inverse Newton [11, 21], Halley [21], and Newton-Schulz [12, 11]. Some of these methods are more frequently associated with the computation of the polar decomposition, but since they are akin to the Padé family of approximations of

*Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439 (jiechen@mcs.anl.gov). Work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

†School of Computational Science and Engineering, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 (echow@cc.gatech.edu). Work supported by NSF under grant ACI-1147843.

the sign function [11, Section 5.4], we can use the methods with a slight modification for computing the matrix sign. The iterations admit the form

$$X_{k+1} = f(X_k), \qquad X_0 = A, \tag{1.1}$$

where f corresponds to different mappings. The four methods are

$$\begin{aligned}
\text{Newton:} \quad & f(X) = \frac{1}{2}(X + X^{-1}), \\
\text{Inverse Newton:} \quad & f(X) = 2X(I + X^2)^{-1}, \\
\text{Halley:} \quad & f(X) = X(3I + X^2)(I + 3X^2)^{-1}, \\
\text{Newton-Schulz:} \quad & f(X) = \frac{1}{2}X(3I - X^2).
\end{aligned}$$

It is not hard to see that the iterates produced by the second method are the inverses of the Newton iterates, hence the name "inverse Newton." The first three methods are globally convergent, with the rate being quadratic for the first two and cubic for the third. On the other hand, Newton-Schulz is known to be convergent when $\|I - A^2\| < 1$, for any subordinate matrix norm. When A is Hermitian, the convergence region can be expanded to $\rho(A) < \sqrt{3}$, where ρ means the spectral radius. Hence, in practical use of Newton-Schulz, one may first scale A by $\rho(A)$. We note that in electronic structure calculations, the McWeeny purification method [19, 16] is equivalent to Newton-Schulz.

The fact that spectral information is needed for Newton-Schulz does not necessarily make the method inferior to the other three globally convergent methods. The reason is that scaling becomes a practical need for enhancing the convergence of these methods. Several scalings have been proposed for Newton and inverse Newton; they all share the form

$$X_{k+1} = f(\mu_k X_k).$$

For Newton, the determinantal scaling defines $\mu_k = |\det(X_k)|^{-1/n}$, the spectral scaling defines $\mu_k = \sqrt{\rho(X_k^{-1})/\rho(X_k)}$, and the norm scaling defines $\mu_k = \sqrt{\|X_k^{-1}\|/\|X_k\|}$. The last two scalings are equivalent when A is Hermitian and when $\|\cdot\|$ is the 2-norm; in such a case, they are both optimal. A disadvantage is that these scalings are expensive to compute, because they depend on each iterate X_k. Byers and Xu [2] proposed a suboptimal scaling that requires the computation of the spectral information only once. The scaling sequence reads

$$\mu_0 = 1/\sqrt{\alpha\beta}, \quad \mu_1 = \sqrt{2\sqrt{\alpha\beta}/(\alpha+\beta)}, \quad \mu_k = 1/\sqrt{f(\mu_{k-1})} \text{ for } k = 2, 3, \ldots,$$

where $\alpha = \|A\|_2$, $\beta = \|A^{-1}\|_2^{-1}$ and f is the scalar Newton mapping $f(\mu) = (\mu + \mu^{-1})/2$. By using the Byers and Xu scaling, in exact arithmetic, at most nine iterations are needed for Newton to converge within a tolerance of 10^{-16} for matrices with condition number no greater than 10^{16}. The counterpart scaling approach for inverse Newton is derived by inverting α, β, and f in Byers and Xu.

For the Halley method, Nakatsukasa et al. [21] suggested an optimal scaling

$$X_{k+1} = X_k(a_k I + b_k X_k^2)(I + c_k X_k^2)^{-1}, \quad X_0 = A/\alpha, \tag{1.2}$$

where $\alpha = \|A\|_2$, $\ell_0 = \sigma_{\min}(X_0)$, and

$$a_k = h(\ell_k), \quad b_k = (a_k - 1)^2/4, \quad c_k = a_k + b_k - 1,$$
$$\ell_k = \ell_{k-1}(a_{k-1} + b_{k-1}\ell_{k-1}^2)/(1 + c_{k-1}\ell_{k-1}^2),$$
$$h(\ell) = \sqrt{1+\gamma} + \frac{1}{2}\sqrt{8 - 4\gamma + \frac{8(2-\ell^2)}{\ell^2\sqrt{1+\gamma}}}, \quad \gamma = \sqrt[3]{\frac{4(1-\ell^2)}{\ell^4}}.$$

Here, $\sigma_{\min}$ denotes the smallest singular value. The scaling approach is coined DWH (dynamically weighted Halley). By using such a scaling, in exact arithmetic, at most six iterations are needed for Halley to converge within a tolerance of 10^{-16} for matrices with condition number no greater than 10^{16}.

Despite the above appealing mathematical properties, scalings lead to subtle numerical stability issues. Nakatsukasa and Higham [22] performed a comprehensive analysis on a general fixed-point iteration (1.1). They showed that the iterations are backward stable when the matrix inverse is computed in a mixed backward-forward stable manner and when f does not significantly decrease the size of any singular value relative to the largest one. This result explains the observation that the Newton method with either spectral/norm scaling or Byers-Xu scaling is stable (see also [13] and [2]), but the inverse Newton method with a Byers-Xu-like scaling is generally not. For the Halley method (DWH), a stable implementation is to replace (1.2) by a mathematically equivalent iteration [21]:

$$X_{k+1} = \frac{b_k}{c_k}X_k + \frac{1}{\sqrt{c_k}}\left(a_k - \frac{b_k}{c_k}\right)Q_1Q_2^*, \tag{1.3}$$

where

$$\begin{bmatrix}\sqrt{c_k}X_k \\ I\end{bmatrix} = \begin{bmatrix}Q_1 \\ Q_2\end{bmatrix}R$$

is a thin QR factorization.

Figure 1.1 demonstrates an example of the stability behavior of various scaled iterations. The 20×20 matrix A is defined as

$$A = Q^*DQ, \tag{1.4a}$$

where Q is the orthogonal factor of a Gaussian random matrix and D is a diagonal matrix with

$$\operatorname{diag}(D) = [1, -\epsilon, \epsilon^{17/17}, \epsilon^{16/17}, \ldots, \epsilon^{1/17}, \epsilon^{0/17}]. \tag{1.4b}$$

Here, ϵ is chosen to be 10^{-16} so that the matrix has a condition number 10^{16}. The relative error on the vertical axis is defined as

$$\|A - X_k(H + H^*)/2\|_F/\|A\|_F, \quad \text{where} \quad H = X_k^*A.$$

Such an error metric originates from the polar decomposition, where H is the Hermitian polar factor of A. The error metric accurately measures the preservation of the eigenspace of A in X_k. One sees that Newton with Byers-Xu scaling and Halley (DWH) with the QR implementation (1.3) can reach the level of 10^{-15}, but inverse Newton with Byers-Xu-like scaling and Halley (DWH) with the LU implementation of

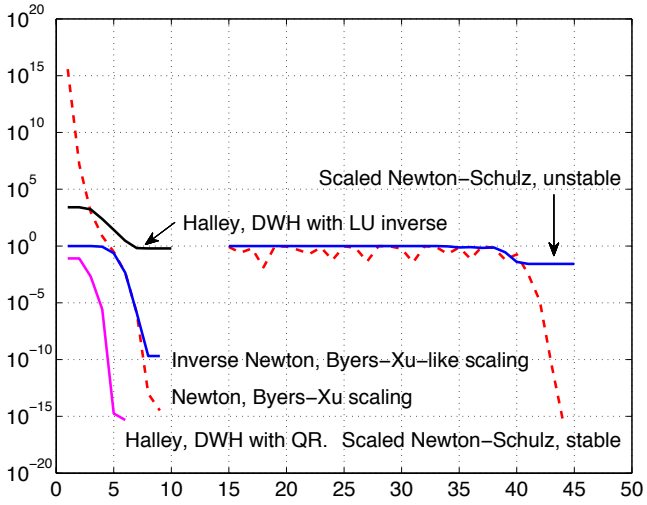


FIG. 1.1. *Convergence history for different scaled iterations. The matrix A is defined in* (1.4).

matrix inverse (1.2) cannot. The figure also shows the results of two scaled Newton-Schulz iterations, the subject of this paper.

Generally, Newton-Schulz requires a few times more iterations to converge than do the other three methods. However, the appeal of Newton-Schulz is that it is inversion-free. The matrix-matrix multiplications therein may be expected to have better parallel scalability than the factorizations used for matrix inversion (e.g., LU and QR), even if the factorizations employ no pivoting. The best communication cost of matrix-matrix multiplications is $\log p$ times smaller than that of LU and QR, where p is the number of processors [1, 5]. Moreover, the arithmetic cost of Newton-Schulz is comparable with that of the other iteration methods, counting the total flops (see Section 4.1). Hence, Newton-Schulz has a greater potential for efficiency in a massively parallel computing setting.

In this paper, we focus on improving the initial convergence of Newton-Schulz, by noting that a drawback of the method is that it requires a large number of iterations before the quadratic convergence is seen. The derivation of the improvement is logically separated in two steps. In the first step, we seek an optimal scaling by analyzing the fixed-point mapping f of Newton-Schulz (Section 2). The key observation is that the initial convergence is governed by the derivative of the mapping at the origin. Thus, the optimal scaling appears in the form

$$X_{k+1} = \frac{1}{2}\alpha_k X_k(3I - \alpha_k^2 X_k^2),$$

where α_k is the smallest magnitude eigenvalue of X_k (Section 3). A recurrence formula can be derived for α_k such that eigenvalue computations are not needed in every iteration. Analysis suggests that the number of iterations for the optimal scaling is reduced by half compared with nonscaling. The same scaling approach was independently proposed by Rubensson [24] in the context of electronic structure calculations. Unfortunately, this scaling is not stable, as demonstrated by the curve annotated as "scaled Newton-Schulz, unstable" in Figure 1.1. The instability stems from the fact that the scaled mapping significantly decreases the largest magnitude

eigenvalue. Hence, we undertake the second step. In Section 4, we modify α_k to stabilize the mapping by sacrificing the optimal derivative at the origin only slightly. The modified iteration is demonstrated by the curve annotated as "scaled Newton-Schulz, stable" in Figure 1.1. We show that in exact arithmetic, at most 44 iterations are needed for the stable, scaled Newton-Schulz to converge within a tolerance of 10^{-16} for matrices with condition number no greater than 10^{16}. We also compare the arithmetic and communication costs with those of the other stable scaling methods and demonstrate the appeal of ours. Practical implementation, including shifting, is discussed in Section 5. Then, we demonstrate large-scale calculations, including in parallel (Section 6), and the application of computing the density matrix in electronic structures (Section 7).

Terminology and notation. In most of this paper, the eigenvalues of A are ordered according to their magnitudes. We use the term "smallest/largest magnitude eigenvalue" of a matrix A to indicate the smallest/largest element of the set $\{|\lambda(A)|\}$, where $\lambda(A)$ denotes the eigenvalues of A. These two elements are denoted as $\lambda_{|\min|}(A)$ and $\lambda_{|\max|}(A)$, respectively. They are not necessarily eigenvalues. Because A is Hermitian, the two elements coincide with the smallest and largest singular values of A, respectively. We note that in some parts of this paper (especially when shifting is concerned), the eigenvalues of A are sorted in their natural order.

Relation to polar decomposition. Much of the theory and analysis of the present paper generalizes to the computation of the polar decomposition of A, where S is the unitary polar factor. In such a case, A does not need to be Hermitian, and the role of eigenvalues (in magnitude) is replaced by that of the singular values. Algorithmically, one needs to change the X_k^2 terms to $X_k^* X_k$; for example, the scaled Newton-Schulz iteration is modified to

$$X_{k+1} = \frac{1}{2}\alpha_k X_k (3I - \alpha_k^2 X_k^* X_k).$$

The mathematical and numerical properties can be established analogously. However, one technically hard (but not critical) aspect to generalize is shifting, which appears to apply only to the sign function but not the polar decomposition.

2. Newton-Schulz. Since A is Hermitian, the convergence of Newton-Schulz is completely characterized by the properties of the scalar mapping

$$f(x) = \frac{1}{2}x(3 - x^2) \tag{2.1}$$

on the real line. Without loss of generality, we assume that $\rho(A) = 1$ and consider only the interval $x \in [-1, 1]$. Figure 2.1 plots f.

Because f is odd, we further restrict our attention to the interval $[0, 1]$. The mapping f on $[0, 1]$ is monotonically increasing and admits $x < f(x)$, except when $x = 0$ or 1. Hence, one intuitive explanation of why the iteration $X_{k+1} = f(X_k)$ converges to the sign of A is that f pushes all the positive eigenvalues of A toward 1 in a monotonic manner (and similarly pushes the negative eigenvalues toward -1). Among all these eigenvalues, the one that converges the most slowly is the eigenvalue closest to the origin. Let this eigenvalue be x_0, and without loss of generality assume that $x_0 > 0$. Then, the initial iteration reduces the condition number from $1/x_0$ to $1/f(x_0)$. When A is ill-conditioned (i.e., $x_0 \approx 0$), the rate of reduction is $f(x_0)/x_0 \approx f'(0)$. Because of the significance of $f'(0)$, one naturally asks what is the optimal mapping

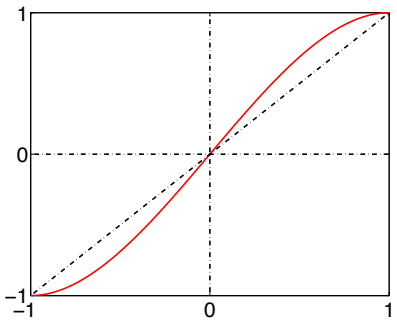


FIG. 2.1. *Mapping f.*

in the sense that the derivative at the origin is maximal. The optimal mapping turns out to be Newton-Schulz, as the following result states.

THEOREM 2.1. *Let P be the set of cubic and odd polynomials that are monotonically increasing on the interval $[0,1]$ and that map this interval to itself. Then,*

$$f = \arg\max_{g \in P} g'(0) \quad \textit{with} \quad f'(0) = \frac{3}{2},$$

where f is defined in (2.1).

Proof. The polynomial must pass the origin because it is odd. It also must pass the point $(1,1)$ because it is monotonically increasing. Then, all such cubic polynomials must have the form $g = ax + (1-a)x^3$. The monotonic increase implies that $a \leq \frac{3}{2}$. Thus, $g'(0) = a$ is maximized when $a = \frac{3}{2}$. ∎

The reduction in the condition number informs only the behavior of the first iteration. Also of interest are the first few iterations. Clearly, the sequence $x_{k+1} = f(x_k)$ generated through the mapping is monotonically increasing and approaching 1. When x_0 is sufficiently small, however, the following result indicates that the first few x_k's are also small. In particular, they depart from 0 at only a linear rate.

THEOREM 2.2. *Let f be the Newton-Schulz mapping* (2.1)*, and define a sequence $x_{k+1} = f(x_k)$ with an initial value $x_0 \in (0,1)$. Then,*

$$\log\left(\frac{x_k}{x_0}\right) < k \log\left(\frac{3}{2}\right) < \log \frac{x_k - f^{-1}(x_k)}{\frac{1}{3}x_0 - \left[f^{-1}(x_k) - \frac{2}{3}x_k\right]} \tag{2.2}$$

whenever

$$\frac{1}{3}x_0 > \left[f^{-1}(x_k) - \frac{2}{3}x_k\right]. \tag{2.3}$$

Proof. We first note that

$$f'(x) = \frac{3}{2}(1 - x^2) > 0 \quad \text{and} \quad f''(x) = -3x < 0,$$

when $0 < x < 1$. Hence, for any k, $x_k < f'(0)x_{k-1}$. Because $f'(0) = \frac{3}{2}$, by induction we have that

$$x_k < \left(\frac{3}{2}\right)^k x_0.$$

This proves the first inequality of (2.2).

Next, we have

$$\left(\frac{3}{2}\right)^k x_0 - x_k = \left(\frac{3}{2}\right)^{k-1}\left(\frac{3}{2}x_0 - x_1\right) + \left(\frac{3}{2}\right)^{k-2}\left(\frac{3}{2}x_1 - x_2\right) + \cdots + \left(\frac{3}{2}\right)^{0}\left(\frac{3}{2}x_{k-1} - x_k\right).$$

Because $f'(0) = \frac{3}{2}$ and f' is decreasing, we have that $\frac{3}{2}x - f(x)$ is positive and is increasing. Then,

$$\begin{aligned}\left(\frac{3}{2}\right)^k x_0 - x_k &< \left[\left(\frac{3}{2}\right)^{k-1} + \cdots + \left(\frac{3}{2}\right)^{0}\right]\left(\frac{3}{2}x_{k-1} - x_k\right) \\ &= \frac{\left(\frac{3}{2}\right)^k - 1}{\frac{3}{2} - 1}\left(\frac{3}{2}f^{-1}(x_k) - x_k\right).\end{aligned}$$

Rearranging terms, we obtain

$$\left\{\left(\frac{3}{2} - 1\right)x_0 - \left(\frac{3}{2}f^{-1}(x_k) - x_k\right)\right\}\left(\frac{3}{2}\right)^k < \frac{3}{2}\left[x_k - f^{-1}(x_k)\right],$$

which proves the second inequality of (2.2). □

To explain the use of Theorem 2.2, we give an example. Consider that the bound (2.2) is with respect to k. Table 2.1 gives the numeric values of (2.2) for $x_0 = 10^{-3}$, where `NA` means the condition (2.3) is invalid. One sees that the bound applies only when x_k is not close to 1 (otherwise (2.3) is invalid); however, whenever it is applicable, the bound for integer k is tight. Thus, we interpret the inequality on the left as

$$\log\left(\frac{x_k}{x_0}\right) \approx k \log\left(\frac{3}{2}\right) \tag{2.4}$$

for small x_k. In other words, x_k grows linearly for the first few k's when the starting value x_0 is sufficiently small. Note that the factor $\log\left(\frac{3}{2}\right)$ is important; we will return to this factor later.

We summarize a few facts in the following theorem, which connects the scalar iteration with the matrix iteration. The validity is clear based on the preceding discussions; hence, the proof is omitted.

THEOREM 2.3. *For a Hermitian matrix A and the Newton-Schulz mapping f defined in* (2.1)*, consider the matrix iteration $X_{k+1} = f(X_k)$, $X_0 = A$, and the scalar iteration $x_{k+1} = f(x_k)$. If the spectral radius of A is* 1 *and x_0 is the smallest magnitude eigenvalue of A, then we have the following.*

1. *x_k is the smallest magnitude eigenvalue of X_k for all k.*
2. *The spectral radius of X_k is* 1 *for all k; hence the condition number of X_k is $1/x_k$.*
3. *$\|X_k - S\|_2 = |x_k - 1|$ for all k.*
4. *$x_k \to 1$ monotonically, and hence $\|X_k - S\|_2 \to 0$ monotonically.*

TABLE 2.1
Numerical values of (2.2) *for* $x_0 = 10^{-3}$.

k	1	2	3
Bound (2.2)	$0.99.. < k < 1.00..$	$1.99.. < k < 2.00..$	$2.99.. < k < 3.00..$
x_k	1.5000e-03	2.2500e-03	3.3750e-03
k	4	5	6
Bound (2.2)	$3.99.. < k < 4.00..$	$4.99.. < k < 5.00..$	$5.99.. < k < 6.00..$
x_k	5.0625e-03	7.5936e-03	1.1390e-02
k	7	8	9
Bound (2.2)	$6.99.. < k < 7.00..$	$7.99.. < k < 8.01..$	$8.99.. < k < 9.03..$
x_k	1.7085e-02	2.5624e-02	3.8428e-02
k	10	11	12
Bound (2.2)	$9.99.. < k < 10.13..$	$10.99.. < k < 11.51..$	$11.98.. < k < 14.51..$
x_k	5.7614e-02	8.6325e-02	1.2917e-01
k	13	14	15
Bound (2.2)	$12.97.. < k <$ NA	$13.94.. < k <$ NA	$14.87.. < k <$ NA
x_k	1.9267e-01	2.8543e-01	4.1652e-01

3. Scaled Newton-Schulz (step 1). In this section, we derive a variant of Newton-Schulz so that the iteration progresses better initially. The optimal variant turns out to be a simple scaling of the Newton-Schulz iterates.

Theorem 2.1 states that the Newton-Schulz mapping f is optimal among all cubic and odd polynomials that map $[0, 1]$ to $[0, 1]$, if in addition the polynomial is required to be increasing. To obtain a polynomial whose derivative at the origin is even larger, we need to sacrifice the monotonicity. Define

$$P' = \{\text{cubic and odd polynomial } g : g([0, 1]) = [0, 1]\}.$$

The polynomials in P' can be parameterized in several ways. Consider $\tilde{g}(x) = ax+bx^3$, where $a > 0$. We crop $\tilde{g}$ in the box $[0, c] \times [0, d]$ and normalize it to obtain

$$g(x) = \frac{1}{d}\tilde{g}(cx) = \frac{ac}{d}x + \frac{bc^3}{d}x^3. \tag{3.1}$$

When d is the maximum of $\tilde{g}$ on $[0, c]$, such polynomials g constitute the set P'.

One can separate the values of b and c into three cases and calculate that

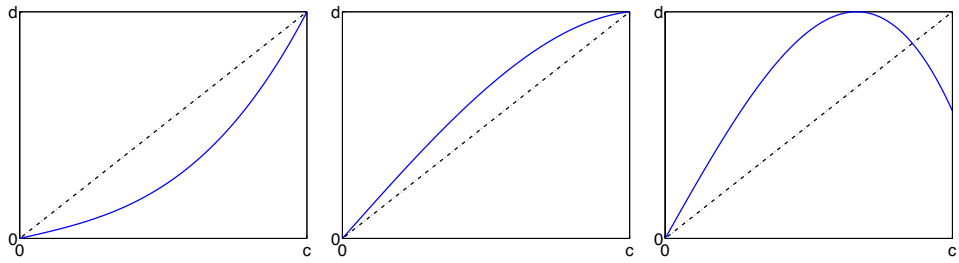
$$d = \begin{cases} ac + bc^3, & \text{in case 1: } b \geq 0 \\ ac + bc^3, & \text{in case 2: } b < 0 \text{ and } c \leq \sqrt{-\dfrac{a}{3b}} \\ \dfrac{2a}{3}\sqrt{-\dfrac{a}{3b}}, & \text{in case 3: } b < 0 \text{ and } c > \sqrt{-\dfrac{a}{3b}}. \end{cases} \tag{3.2}$$

See Figure 3.1 for visual examples of the three cases.

In case 1, for all $x \in [0, 1]$, $g(x) \leq x$; in case 2,

$$g'(0) = \frac{ac}{d} = \frac{ac}{ac + bc^3} \leq \frac{3}{2},$$

because $c \leq \sqrt{-\frac{a}{3b}}$. These two cases do not yield a better polynomial than the



(a) Case 1, $a = 1$, $b = 0.1$ (b) Case 2, $a = 1$, $b = -0.012$ (c) Case 3, $a = 1$, $b = -0.03$

FIG. 3.1. *Function $\tilde{g}$ in different cases of* (3.2). $c = 5$.

standard Newton-Schulz mapping f does. On the other hand, in case 3,

$$g'(0) = \frac{ac}{d} = \frac{3}{2}c \Big/ \sqrt{-\frac{a}{3b}}\,,$$

which is larger than $\frac{3}{2}$. Hence, we further explore this case.

Let $\alpha = c \Big/ \sqrt{-\frac{a}{3b}}$, which is greater than 1. The polynomial in case 3 is simplified to

$$g(x) = \frac{3}{2}\alpha x - \frac{1}{2}\alpha^3 x^3. \tag{3.3}$$

Such a polynomial passes the origin, monotonically increases to the maximum 1, and then decreases until touching $x = 1$. When α becomes larger, the derivative at the origin is steeper, the x value that achieves maximum moves toward the left, and $g(1)$ gets closer to 0. In the extreme case, we solve $g(1) = 0$ and obtain that $\alpha = \sqrt{3}$. Hence, the valid range of α is between 1 and $\sqrt{3}$.

As before, we let $x_0 > 0$ be the smallest magnitude eigenvalue, whereas the largest is 1. We want α to be optimal in the sense that after the mapping g, the condition number of the matrix is maximally reduced. Such an α is obtained by solving $g(x_0) = g(1)$; see Figure 3.2. The solution gives

$$\alpha = \sqrt{\frac{3}{1 + x_0 + x_0^2}}. \tag{3.4}$$

Because such an α yields a mapping g that maps the interval $[x_0, 1]$ to $[g(x_0), 1]$, the smallest magnitude eigenvalue of the matrix is $g(x_0)$ after mapping.

Thus, we apply the optimal mapping g iteratively on the matrix. Note that g keeps changing because the smallest magnitude eigenvalue does so, too. We start with an initial matrix $X_0 = A/\lambda_{|\max|}$ such that the spectral radius of X_0 is 1. Then, its smallest magnitude eigenvalue $x_0 = \lambda_{|\min|}/\lambda_{|\max|}$. The iteration maintains a loop invariant such that the smallest magnitude eigenvalue of X_k is x_k. Specifically, the iteration reads

$$X_{k+1} = \frac{1}{2}\alpha_k X_k (3I - \alpha_k^2 X_k^2), \tag{3.5a}$$

where

$$\alpha_k = \sqrt{\frac{3}{1 + x_k + x_k^2}} \quad \text{and} \quad x_{k+1} = \frac{1}{2}\alpha_k x_k (3 - \alpha_k^2 x_k^2). \tag{3.5b}$$

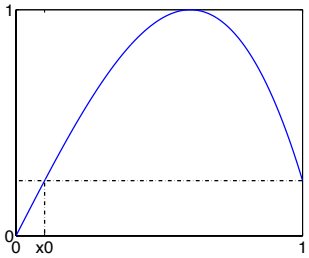


FIG. 3.2. *Optimal mapping g given x_0.*

We call (3.5) the *optimally scaled Newton-Schulz* iteration.

3.1. Analysis. The optimal mapping g in the preceding discussion depends on the smallest magnitude eigenvalue. We may write an eigenvalue-independent mapping to characterize the scaled iteration instead:

$$h(x) = \frac{3}{2}\alpha x - \frac{1}{2}\alpha^3 x^3, \quad \text{where} \quad \alpha(x) = \sqrt{\frac{3}{1+|x|+x^2}}. \tag{3.6}$$

Figure 3.3 plots h on $[-1, 1]$, together with the standard Newton-Schulz mapping f for comparison.

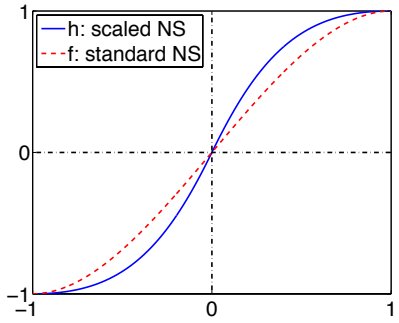


FIG. 3.3. *Mapping h (scaled Newton-Schulz) and f (Newton-Schulz).*

We note that different from the case of f, we shall not treat h as a matrix function and write $X_{k+1} = h(X_k)$, because α takes only a scalar value x as input. By construction, the smallest magnitude eigenvalue maintains its "smallest" property after one iteration. Hence, we define the matrix counterpart of h as

$$\underline{h}(X) = \frac{3}{2}\underline{\alpha} X - \frac{1}{2}\underline{\alpha}^3 X^3, \quad \text{where} \quad \underline{\alpha}(X) = \sqrt{\frac{3}{1+\lambda_{|\min|}(X)+\lambda_{|\min|}(X)^2}}. \tag{3.7}$$

Then, the iteration $X_{k+1} = \underline{h}(X_k)$ is consistent with that of (3.5).

Clearly, the mapping h is monotonically increasing on $[0, 1]$, and it maps this interval to itself. Because $\alpha \geq 1$, we always have $h(x) \geq f(x)$ for $x \in (0, 1)$. Hence, for the same initial value $x_0 = \tilde{x}_0 \in (0, 1)$, the sequence $x_{k+1} = h(x_k)$ is always larger than the sequence $\tilde{x}_{k+1} = f(\tilde{x}_k)$, elementwise. Because $\tilde{x}_k \to 1$ and x_k is bounded by 1, the sequence x_k monotonically increases to the limit 1, the same as does the

sequence $\tilde{x}_k$. Furthermore, x_k is always closer to the limit than is $\tilde{x}_k$. We summarize this result, together with other facts in the following theorem, whose proof is clear based on the foregoing discussion. This theorem connects the matrix iteration with the scalar iteration. One should compare this result with Theorem 2.3.

THEOREM 3.1. *For a Hermitian matrix A and the optimally scaled Newton-Schulz mappings h and $\underline{h}$ defined in* (3.6) *and* (3.7)*, respectively, consider the matrix iteration $X_{k+1} = \underline{h}(X_k)$, $X_0 = A$, and the scalar iteration $x_{k+1} = h(x_k)$. If the spectral radius of A is* 1 *and x_0 is the smallest magnitude eigenvalue of A, then we have the following.*

1. *x_k is the smallest magnitude eigenvalue of X_k for all k.*
2. *$\|X_k - S\|_2 = |x_k - 1|$ for all k.*
3. *$x_k \to 1$ monotonically, and hence $\|X_k - S\|_2 \to 0$ monotonically.*
4. *The spectral radius of X_k converges to* 1*.*

Furthermore, for the iteration $\tilde{x}_{k+1} = f(\tilde{x}_k)$ where $\tilde{x}_0 = x_0$ and where f is the standard Newton-Schulz mapping (2.1)*, we have $\tilde{x}_k < x_k$ for all $k > 0$.*

The significance of the first conclusion of Theorem 3.1 is that the convergence behavior of X_k is completely characterized by that of x_k. Then, we need to focus on only the mapping h. The following theorem states the limiting and the initial behavior of the scaled iterations.

THEOREM 3.2. *Let h be the mapping of the optimally scaled Newton-Schulz iteration* (3.6)*, and define a sequence $x_{k+1} = h(x_k)$ with an initial value $x_0 \in (0, 1)$. Then*

1. *x_k converges to* 1 *quadratically; and*
2. *we have*

$$\log\left(\frac{x_k}{x_0}\right) < k \log\left(\frac{3}{2}\sqrt{3}\right) < \log \frac{x_k - h^{-1}(x_k)}{\left(1 - \frac{2}{3\sqrt{3}}\right) x_0 - \left[h^{-1}(x_k) - \frac{2}{3\sqrt{3}} x_k\right]} \tag{3.8}$$

whenever

$$\left(1 - \frac{2}{3\sqrt{3}}\right) x_0 > \left[h^{-1}(x_k) - \frac{2}{3\sqrt{3}} x_k\right].$$

Proof. We already know that x_k converges to 1 in Theorem 3.1. Because $h(x) - 1 = -\frac{1}{2}(\alpha x + 2)(\alpha x - 1)^2$, we have

$$\lim_{x \to 1} \frac{|h(x) - 1|}{|x - 1|^2} = \left(\lim_{x \to 1} \frac{|\alpha x + 2|}{2}\right) \left(\lim_{x \to 1} \frac{\alpha x - 1}{x - 1}\right)^2 = \frac{3}{2} \left(\lim_{x \to 1} \frac{d\alpha}{dx} + \alpha\right)^2 = \frac{3}{8}, \tag{3.9}$$

where the second equality follows from L'Hospital's rule. This shows that the convergence is quadratic.

The result (3.8) is proved by using the same technique as that for proving (2.2) in Theorem 2.2. To save space, we omit the details here. ∎

Similar to the interpretation of Theorem 2.2, we see that the bound (3.8) is tight. For example, when $x_0 = 10^{-3}$, the numeric values of the bound are given in Table 3.1. Hence, we write

$$\log\left(\frac{x_k}{x_0}\right) \approx k \log\left(\frac{3}{2}\sqrt{3}\right). \tag{3.10}$$

Compare (3.10) with (2.4), which we repeat here by adding a tilde to denote the sequence generated through the f mapping (as we previously did):

$$\log\left(\frac{\tilde{x}_k}{x_0}\right) \approx k \log\left(\frac{3}{2}\right).$$

Because the ratio between $\log\left(\frac{3}{2}\sqrt{3}\right)$ and $\log\left(\frac{3}{2}\right)$ is 2.35.., we can loosely conclude that

$$\tilde{x}_{2k} < x_k < \tilde{x}_{3k}.$$

This means that initially (when k is small), the optimally scaled Newton-Schulz iteration increases the iterate x_k at least twice as fast as does the standard Newton-Schulz iteration.

TABLE 3.1
Numerical values of (3.8) *for* $x_0 = 10^{-3}$.

k	1	2	3
Bound (3.8)	$0.99.. < k < 1.00..$	$1.99.. < k < 2.00..$	$2.99.. < k < 3.03..$
x_k	`2.5968e-03`	`6.7378e-03`	`1.7445e-02`
k	4	5	6
Bound (3.8)	$3.98.. < k < 4.28..$	$4.95.. < k <$ `NA`	$5.88.. < k <$ `NA`
x_k	`4.4914e-02`	`1.1383e-01`	`2.7539e-01`

On closing this section, we remark that in the convergence guarantee of (3.5), x_0 need not be the smallest magnitude eigenvalue of X_0. The following theorem states that (3.5) always converges no matter what value x_0 takes. More amazing is that the quadratic rate of convergence is also maintained. Hence, the price paid for using an arbitrary x_0 is only some more iterations.

THEOREM 3.3. *For the optimally scaled Newton-Schulz iteration* (3.5)*, X_k converges to S quadratically for any $x_0 \in (0, 1)$.*

Proof. We repeat (3.5) in the following for better reading:

$$X_{k+1} = \frac{1}{2}\alpha_k X_k (3I - \alpha_k^2 X_k^2). \tag{3.11}$$

Because the sequence x_k is computed through the fixed-point mapping $x_{k+1} = h(x_k)$, for any $x_0 \in (0, 1)$, x_k converges to 1 quadratically. Hence, α_k also converges to 1 quadratically. We use an auxiliary sequence

$$Y_{k+1} = \frac{1}{2} Y_k (3I - Y_k^2), \quad Y_0 = X_0 \tag{3.12}$$

to gauge the convergence behavior of X_k. Clearly, the sequence Y_k results from the standard Newton-Schulz iteration, and it converges to S quadratically.

Let $V_k = X_k - Y_k$, $W_k = Y_k - S$, and $\epsilon_k = |\alpha_k - 1|$. We have

$$\begin{aligned} \|\alpha_k X_k - Y_k\|_2 &\le \|\alpha_k X_k - \alpha_k Y_k\|_2 + \|\alpha_k Y_k - Y_k\|_2 \\ &= \alpha_k \|V_k\|_2 + \epsilon_k \|Y_k\|_2 \le (1 + \epsilon_k)\|V_k\|_2 + \epsilon_k (1 + \|W_k\|_2). \end{aligned} \tag{3.13}$$

We further let $Z_k = \alpha_k X_k - Y_k$. By noting that X_k and Y_k commute (because both are polynomials of X_0), we subtract (3.12) from (3.11) and obtain

$$\begin{aligned}\|V_{k+1}\|_2 &= \frac{1}{2}\|Z_k(3I - Z_k^2 - 3\alpha_k X_k Y_k)\|_2 \\ &\le \frac{1}{2}\|Z_k\|_2\Big(3\|I - Y_k^2\|_2 + \|Z_k\|_2^2 + 3\|Z_k\|_2\|Y_k\|_2\Big) \\ &\le \frac{1}{2}\|Z_k\|_2\Big(\|Z_k\|_2^2 + 3(1 + \|W_k\|_2)\|Z_k\|_2 + 3(2\|W_k\|_2 + \|W_k\|_2^2)\Big).\end{aligned}$$

With the help of (3.13) we further expand the inequality of $\|V_{k+1}\|_2$ and obtain

$$\begin{aligned}\|V_{k+1}\|_2 \le \Big[3(\epsilon_k + \|W_k\|_2) + O(\epsilon_k^2 + \epsilon_k\|W_k\|_2 + \|W_k\|_2^2)\Big] \cdot \|V_k\|_2 \\ + O(\epsilon_k^2 + \epsilon_k\|W_k\|_2 + \|W_k\|_2^2 + \|V_k\|_2^2).\end{aligned}$$

Note that both ϵ_k and $\|W_k\|_2$ converge to 0 quadratically. Because $\|V_0\|_2 = 0$, by induction we have that $\|V_k\|_2 = O(\epsilon_k + \|W_k\|_2)$. This means that the difference between X_k and Y_k converges to 0 quadratically. Thus, X_k converges to S quadratically. ∎

4. Scaled Newton-Schulz (step 2), stable version. Unfortunately, the optimal scaling (3.5) is numerically unstable at convergence. Stability is measured by the backward error

$$\frac{\|A - X_k(H + H^*)/2\|}{\|A\|} \tag{4.1}$$

of the polar decomposition of A, where H is the Hermitian polar factor $X_k^* A$ and $\|\cdot\|$ is any subordinate matrix norm. The curve annotated as "scaled Newton-Schulz, unstable" in Figure 1.1 shows that the iteration (3.5) cannot decrease the backward error to the level of machine precision. The instability stems from the subtle fact that repeated matrix-matrix multiplications may alter the eigenspace of a matrix. Thus, an alternative stability measure is

$$\frac{\|AX_k - X_k A\|}{\|A\|\|X_k\|},$$

which measures how well X_k commutes with A. This measure leads to the same instability conclusion for (3.5).

A simple but robust fix returns to the derivation of the optimal mapping g in (3.3) and (3.4). Recall that x_0 denotes the smallest magnitude eigenvalue. The optimal scaling factor α is derived by equating $g(x_0)$ with $g(1)$. When x_0 is small, so is $g(x_0)$. Then, such an α forces the largest magnitude eigenvalue 1 to drop to $g(1)$, a substantial decrease. In the analysis of [22], such a decrease is the origin of instability. To circumvent the issue, we set a threshold t that limits the smallest value of $g(1)$. Equating $g(1) = t$ gives

$$\frac{3}{2}\alpha - \frac{1}{2}\alpha^3 - t = 0. \tag{4.2}$$

Clearly, α is a decreasing function of t when $\alpha > 1$. Let $\hat{\alpha}$ solve (4.2), which has only one root on the interval $(1, \sqrt{3})$. Then, the modified scaling factor α reads

$$\alpha = \min\left\{\sqrt{\frac{3}{1 + x_0 + x_0^2}},\ \hat{\alpha}\right\}, \tag{4.3}$$

which is upperbounded by $\hat{\alpha}$.

An appropriate value of the threshold t cannot be too small or too large. If too small, the largest magnitude eigenvalue still undergoes a significant decrease. If too large, $g'(0) = \frac{3}{2}\alpha$ decreases accordingly and undermines the acceleration of Newton-Schulz. Let us consider three candidates: 1, 0.1, and 0.01. The first candidate is equivalent to the standard Newton-Schulz—no scaling. The third candidate yields a backward error curve that stagnates above the level of machine precision. On the other hand, the second candidate survives all the stability tests we have conducted. Thus, we set $t = 0.1$.

One additional benefit of using $t = 0.1$ is that it barely changes the convergence progress of the optimally scaled iteration (3.5). Specifically, we write the new mapping

$$\hat{h}(x) = \frac{3}{2}\alpha x - \frac{1}{2}\alpha^3 x^3, \quad \text{where} \quad \alpha(x) = \min\left\{\sqrt{\frac{3}{1+|x|+x^2}},\ \hat{\alpha}\right\}. \tag{4.4}$$

Comparing $\hat{h}$ with h in (3.6), the curve of $\hat{h}$ is visually indistinguishable from that of h (plotted in Figure 3.3). Moreover, an analogous result to (3.10) is

$$\log\left(\frac{\hat{x}_k}{x_0}\right) \approx k \log\left(\frac{3}{2}\hat{\alpha}\right),$$

where $\{\hat{x}_k\}$ denotes the sequence generated by the $\hat{h}$ mapping. Here, $\hat{\alpha} = 1.69...$ when $t = 0.1$, which makes $\log\left(\frac{3}{2}\hat{\alpha}\right) / \log\left(\frac{3}{2}\right) = 2.30....$ Such a ratio is sufficiently close to the ratio $\log\left(\frac{3}{2}\sqrt{3}\right) / \log\left(\frac{3}{2}\right)$, leading to an almost identical convergence progress compared with the optimally scaled Newton-Schulz.

We summarize the whole computational procedure in Algorithm 1. This algorithm is numerically stable.

Algorithm 1 A Stable, Scaled Newton-Schulz Method for Computing sign(A)

Compute $\lambda_{|\min|}$ and $\lambda_{|\max|}$ of A
Let $X_0 = A/\lambda_{|\max|}$ and $x_0 = \lambda_{|\min|}/\lambda_{|\max|}$.
Solve $\frac{1}{2}\hat{\alpha}(3 - \hat{\alpha}^2) = t$ for $\hat{\alpha}$, where $t = 0.1$.
for $k = 0, 1, \ldots$ *maxiter* **do**
    Compute $\alpha_k = \min\left\{\sqrt{\frac{3}{1+x_k+x_k^2}},\ \hat{\alpha}\right\}$.
    Update $X_{k+1} = \frac{1}{2}\alpha_k X_k(3I - \alpha_k^2 X_k^2)$.
    Symmetrize $X_{k+1} \leftarrow \frac{1}{2}(X_{k+1} + X_{k+1}^*)$ if required.
    Update $x_{k+1} = \frac{1}{2}\alpha_k x_k(3 - \alpha_k^2 x_k^2)$.
    If converged, exit loop.
end for
return X_{k+1}

4.1. Computational costs. We list in Table 4.1 the maximum number of iterations needed for the backward error (4.1) to reach 10^{-16} for different values of the matrix condition number. Four stable iterations are compared: the standard Newton-Schulz; the stable, scaled Newton-Schulz (Algorithm 1); Newton with Byers-Xu scaling; and Halley (DWH) with the QR implementation (1.3). One clearly sees

that Algorithm 1 converges approximately twice as fast as the standard Newton-Schulz, whereas it requires only a few more times of iterations compared with the last two methods.

TABLE 4.1
Iteration counts for various iterations to converge to 10^{-16}.

Condition number	10^2	10^4	10^6	10^8	10^{10}	10^{12}	10^{14}	10^{16}
Newton-Schulz (NS)	17	28	39	51	62	74	85	96
Stable, scaled NS	10	15	19	24	29	34	39	44
Newton, Byers-Xu	6	7	7	8	8	8	9	9
Halley, DWH with QR	4	5	5	5	5	5	6	6

In terms of flop count, Newton-Schulz and scaled Newton-Schulz require $2n^3$ flops per iteration in the Hermitian case (neglecting lower-order terms). Scaled Newton also requires $2n^3$ flops per iteration if the inversion is computed by using a triangular factorization, or $6n^3$ flops if the inversion is computed by using a backward stable bidiagonal reduction-based algorithm [21]. The dynamically weighted Halley iteration implemented using QR factorization requires $(26/3)n^3$ flops per iteration, or $(16/3)n^3$ flops per iteration if implemented carefully with Givens rotations.

In terms of communication volume in a parallel implementation, although matrix multiplication, Cholesky, and QR factorization have asymptotically the same lower bounds, current practical "communication-avoiding" implementations of Cholesky and QR come only within a factor of $\log p$, where p is the number of processors, and may perform many more flops than the standard algorithms in order to attain reduced communication [1, 5]. On the other hand, matrix multiplication attains its communication lower bound with standard Cannon and SUMMA algorithms, and many optimized implementations exist for different computer platforms.

In summary, the main computational difference between the Newton-Schulz algorithms (including Algorithm 1) and Newton and Halley is matrix inversion (or factorizations used for reformulating inversion). With inversion, much faster convergence may be attained, but it comes at the cost of potentially poorer scalability. This is a classic tradeoff. The best algorithm to use in any situation will likely depend on the computer architecture, the number of available processors, the problem size, and the matrix condition number.

5. Practical implementation. A few implementation issues for Algorithm 1 are addressed in this section.

5.1. Stopping criterion. We need a convergence test for line 9 of Algorithm 1. Write $X_k - S = (X_k S - I)S$; then,

$$\|X_k - S\| = \|(X_k S - I)S\| \le \|X_k S - I\|\|S\|.$$

Note that the inequality is an equality in the 2-norm case. When $X_k \approx S$, we obtain the relative error

$$\frac{\|X_k - S\|}{\|S\|} \lessapprox \|X_k^2 - I\|.$$

Hence, it is straightforward to use $\|X_k^2 - I\| \le tol$ as the convergence criterion, where *tol* is the relative tolerance. Any submultiplicative norm can be used, but for

computational convenience we use the Frobenius norm. The computed squared factor X_k^2 is stored and used in the next iteration for updating X_k. The best attainable error is $n\mathbf{u}/2$, where $\mathbf{u}$ is the machine precision [11, Section 5.1].

Note that this stopping criterion verifies only the unitarity of X_k but not the eigenspace. At detected convergence, one may in addition compute the backward error (4.1) to ensure that the computed X_k is stable. Computing (4.1) requires two extra matrix-matrix multiplications.

5.2. Shifting. A useful observation is that shifts of A do not change $\mathrm{sign}(A)$ as long as these shifts do not cause eigenvalues of A to cross the origin. Hence, we can perform a shift to precondition A (that is, to start with a larger x_0) and to improve convergence. Specifically, we first compute four eigenvalues of interest:

$$\lambda_{-\,\min}(A) \le \lambda_{-\,\max}(A) < 0 < \lambda_{+\,\min}(A) \le \lambda_{+\,\max}(A),$$

which are the smallest negative, the largest negative, the smallest positive, and the largest positive eigenvalue of A, respectively, and compute a shift

$$\tau = \frac{\lambda_{-\,\max}(A) + \lambda_{+\,\min}(A)}{2}.$$

Then, we input $A' = A - \tau$ and run Algorithm 1 to obtain $S = \mathrm{sign}(A')$. The two eigenvalues in Algorithm 1 now become

$$\lambda'_{|\,\max\,|} = \max\{|\lambda_{+\,\max}(A) - \tau|, |\lambda_{-\,\min}(A) - \tau|\}$$
$$\lambda'_{|\,\min\,|} = \min\{|\lambda_{-\,\max}(A) - \tau|, |\lambda_{+\,\min}(A) - \tau|\}.$$

For the amount of reduction in iterations caused by the reduction in condition number, see Table 4.1.

5.3. Eigenvalue estimation. The requirement of eigenvalue computation in Algorithm 1 forms several levels of difficulty. Our discussion here is oriented to general eigenvalue techniques and software support. Whereas small-scale eigenvalue problems have a standard solution through orthogonal transformations to the condensed form, large-scale eigenvalue problems are challenging, even if only a few extreme and/or interior eigenvalues are sought. We make no effort to design new eigenvalue techniques; rather, the issue we address here is how practical Algorithm 1 is in a large-scale setting with off-the-shelf software packages. For a comprehensive treatment of the theory and state-of-the-art eigenvalue methods, we refer the readers to Saad's book [25]. Practitioners might develop specialized eigenvalue solvers for their applications.

At the easiest level, no eigenvalue computation is carried out. The largest magnitude eigenvalue can be estimated by using the Gershgorin circle theorem. The theorem ensures that the spectral radius of the scaled matrix is no greater than 1. A drawback of this method is that in some cases the bound provided by the theorem is too pessimistic. On the other hand, the starting value x_0 can be arbitrary, as noted earlier; what is sacrificed is the optimal convergence. If a good estimate of the condition number of A is known a priori, the extra number of iterations may not be too large. Note that if x_0 is not an accurate estimate, shifting is impossible.

At the next level, Gershgorin is replaced by a computation of the largest magnitude eigenvalue. This eigenvalue can be computed by using the Lanczos algorithm. Accelerated by implicit shifts and restarts [14, 27], the Lanczos algorithm converges rapidly when the targeted eigenvalue is not clustered with others. The algorithm has

been implemented in ARPACK [15]. The dominant cost of the calculation is forming matrix-vector products, for which extensive research has been devoted to designing high-performance software as well as linear-time or near-linear-time algorithms for different types of matrices, such as sparse, Toeplitz, or kernel matrices. Because the desired eigenvalue converges from inside the spectrum interval, if this eigenvalue cannot be computed to high accuracy, a correction term must be added to ensure that the spectral radius of the scaled matrix is no greater than 1.

The most difficult level comprises the calculation of all four eigenvalues mentioned in Section 5.2. In particular, shifting is useful and produces the correct result only when the two innermost eigenvalues are computed accurately. Computing these eigenvalues, however, is a well-known challenge in applications. The Lanczos algorithm discussed in the preceding paragraph can be reused, by applying A^{-1} as the operator instead of A. Then, the dominant cost becomes solving linear systems with A. For sparse matrices, a direct method is the most robust; software includes CHOLMOD [4] and SuperLU [17]. For dense matrices arising from kernels, direct solves with high-accuracy off-diagonal compression techniques are gaining popularity [30]. Iterative solvers are in general not as robust as direct solvers but sometimes are highly efficient with a good preconditioner. To avoid being too restricted by the standard form of Lanczos, we also mention other popular methods for computing interior eigenvalues: shift-and-invert [7], Davidson's method [3] and its improvements [20, 9], and polynomial filtering [8], the details of which are not further discussed.

6. Numerical experiments. In this section we conduct numerical experiments with matrices from various sources. Many aspects of Algorithm 1 are examined, including stability, eigenvalue estimates, shifting, and computing environments (including a single desktop with multithreaded Matlab and a computing cluster with MPI). The goal is to demonstrate the stability and efficiency of the proposed algorithm. The symmetrization step (line 7 of Algorithm 1) is enforced.

6.1. Matrices from the Matlab gallery and Higham's toolbox. We empirically verify the numerical stability of Algorithm 1 with applicable matrices from the Matlab gallery and Higham's matrix computation toolbox.[1] The size of the matrices is set to 20×20. If a matrix A is not Hermitian, we modify it by $A \leftarrow (A + A^*)/2$. The condition number of the modified matrices ranges from `1` to `1.1e+19`. When the stopping criterion is set to $\|X_k^2 - I\|_F \leq n\mathbf{u}/2$, the largest backward error (4.1) is `2.3e-15` among all matrices.

6.2. An artificial matrix. We test the effect of inaccurate eigenvalue estimates on the convergence of Algorithm 1. For this, consider a matrix built using the standard 2D Laplacian L:

$$A = \begin{bmatrix} L - c\lambda_{|\min|}(L) & \\ & -2L + 2c\lambda_{|\min|}(L) \end{bmatrix}, \tag{6.1}$$

where $c < 1$ is a tunable parameter. The eigenvalues of such a matrix are known. When $c \to 1$, A is increasingly ill-conditioned.

We use a 20×30 grid to construct L and test with two choices of c: $1 - 10^{-4}$ and $1 - 10^{-8}$. Table 6.1 lists the number of iterations with different estimates of the eigenvalues $\lambda_{|\max|}(A)$ and $\lambda_{|\min|}(A)$ in Algorithm 1. For the estimated eigenvalues, we let $\lambda_{|\max|}(A)$ be twice the exact value. Usually, this eigenvalue is easy to estimate,

[1] `http://www.maths.manchester.ac.uk/~higham/mctoolbox/`

and setting it to be twice as large is sufficiently conservative. On the other hand, for $\lambda_{|\min|}(A)$, we perturb the exact value by a factor of 10 or 100 to simulate highly inaccurate estimates. In the table, the first row always uses the exact eigenvalues. The column "NS" stands for Newton-Schulz and "sNS" stands for the stable, scaled variant. As expected, when the eigenvalue estimates are very crude, the number of iterations significantly increases. Nevertheless, the scaled iteration still requires many fewer iterations than does standard Newton-Schulz. The largest backward error (4.1) is `1.49e-16` among all matrices.

TABLE 6.1

Number of iterations for matrix A in (6.1). *"NS" stands for Newton-Schulz and "sNS" stands for the proposed method (Algorithm 1). "Est.* $\lambda_{|\max|}$*" and "Est.* $\lambda_{|\min|}$*" are perturbed eigenvalues of A, except in the first row, which uses the exact eigenvalues.*

	$c = 1 - 10^{-4}$, cond = `4.86802e+06`				$c = 1 - 10^{-8}$, cond = `4.86802e+10`			
	Est. $\lambda_{\lvert\max\rvert}$	Est. $\lambda_{\lvert\min\rvert}$	NS	sNS	Est. $\lambda_{\lvert\max\rvert}$	Est. $\lambda_{\lvert\min\rvert}$	NS	sNS
1:	`15.8696`	`3.26e-06`	43	21	`15.8696`	`3.26e-10`	66	31
2:	`15.8696`	`1.00e-06`	43	22	`15.8696`	`1.00e-10`	66	32
3:	`15.8696`	`1.00e-04`	43	27	`15.8696`	`1.00e-08`	66	36
4:	`15.8696`	`1.00e-08`	43	27	`15.8696`	`1.00e-12`	66	37
5:	`31.7392`	`3.26e-06`	45	22	`31.7392`	`3.26e-10`	68	32
6:	`31.7392`	`1.00e-06`	45	23	`31.7392`	`1.00e-10`	68	33
7:	`31.7392`	`1.00e-04`	45	27	`31.7392`	`1.00e-08`	68	37
8:	`31.7392`	`1.00e-08`	45	28	`31.7392`	`1.00e-12`	68	38

6.3. PARSEC matrices. We test the use of shifting on the PARSEC collection of matrices arising from density functional theory in quantum mechanics. The matrices can be downloaded from the University of Florida Sparse Matrix Collection.[2] This collection contains matrices of size ranging from several hundreds to a few hundred thousands. The computations are carried out on the Blues computing cluster[3] at Argonne National Laboratory. The machine comprises 310 compute nodes, each of which has 16 Intel Sandy Bridge cores and 64 GB of memory. The compute nodes are connected by QLogic QDR InfiniBand with a fat-tree topology.

We prepared two programs, one in Matlab and the other in C with MPI. The Matlab program runs on one compute node with a maximum of 16 threads by default. It uses the backslash command for solving linear systems and uses the `eigs` command for computing eigenvalues. The C program runs on a large number of compute nodes. It uses SuperLU for solving linear systems and uses PARPACK for computing eigenvalues. The matrix-matrix multiplication is implemented by using ScaLAPACK.

Preliminary investigation of the spectra shows that the inertias of the matrices are highly skewed. We thus center the matrices at 1/3 of the spectrum as a preprocessing (that is, the new origin is located at $\frac{1}{3}\lambda_{+\max} + \frac{2}{3}\lambda_{-\min}$). The cutoff 1/3 is arbitrary; our purpose is to demonstrate calculations with an arbitrary change of the origin.

We perform calculations with the preprocessed matrices, once before shifting and once after shifting. The computed results are listed in Table 6.2. One sees that shifting does help reduce the condition number and the iteration count. The reduction on the matrices `benzene` and `Si34H36` is substantial.

[2] `http://www.cise.ufl.edu/research/sparse/matrices/`

[3] `http://www.lcrc.anl.gov/about/blues`

TABLE 6.2
Computation results of the matrices in the PARSEC collection. The matrices were preprocessed by centering. Inertia a/b means a positive eigenvalues and b negative eigenvalues. "Cond" is the condition number. "Iter" is the number of iterations. "B-Err" is the backward error (4.1).

			Before Shift			After Shift		
Matrix	n	Inertia	Cond	Iter	B-Err	Cond	Iter	B-Err
Si2	769	516/253	2.5e+3	13	4.2e-15	1.3e+3	12	3.8e-15
SiH4	5,041	3,467/1,574	1.4e+4	15	4.7e-15	2.3e+3	13	4.2e-15
benzene	8,219	5,459/2,760	5.4e+6	21	6.6e-15	1.5e+4	15	5.0e-15
Si10H16	17,077	11,575/5,502	3.6e+4	16	5.4e-15	2.5e+4	15	5.4e-15
SiO	33,401	22,620/10,781	3.3e+4	16	8.6e-15	2.3e+4	15	8.5e-15
Ga3As3H12	61,349	61,348/1	2.6e+0	5	6.7e-15	1.1e+0	4	4.4e-16
Si34H36	97,569	65,621/31,948	1.4e+6	20	8.6e-15	6.3e+4	16	7.3e-15

Table 6.3 lists the timings of the calculations for the matrices with optimal shifting. Results on the top part of the table are obtained by running the Matlab program, those in the bottom part by running the C program. As expected, computing the interior eigenvalues $\lambda_{-\max}$ and $\lambda_{+\min}$ is more costly than computing the exterior eigenvalues $\lambda_{-\min}$ and $\lambda_{+\max}$. Nevertheless, compared with the iterations, the time for computing the eigenvalues is generally smaller.

TABLE 6.3
Timing results of the computation of the matrices in Table 6.2. All times are in seconds.

Matrix	n	Parallelism	$\lambda_{-\min}$	$\lambda_{-\max}$	$\lambda_{+\min}$	$\lambda_{+\max}$	sNS
Si2	769	16 threads	8.9e-1	5.5e-1	8.6e-1	3.9e-2	6.1e-1
SiH4	5,041	16 threads	9.9e-2	2.8e+1	5.7e+1	9.0e-2	8.8e+1
benzene	8,219	16 threads	1.8e-1	8.1e+1	5.3e+1	1.6e-1	4.4e+2
Si10H16	17,077	16 threads	6.3e-1	8.1e+2	6.1e+2	5.7e-1	3.4e+3
SiO	33,401	1,024 procs	3.8e+0	8.4e+0	6.7e+0	7.4e-1	2.6e+2
Ga3As3H12	61,349	1,024 procs	5.2e+0	1.3e+2	2.3e+1	2.7e-1	4.5e+2
Si34H36	97,569	2,304 procs	1.2e+1	3.5e+1	6.8e+1	5.3e+0	2.6e+3

7. Application: electronic structure calculation. At every iteration of the Hartree-Fock algorithm, also known as the self-consistent field (SCF) iteration, a spectral projector called the density matrix is computed from the Fock matrix, which is an approximation to the Hamiltonian [28]. The density matrix may be computed in many ways, the most obvious being through an eigenvalue decomposition of the Fock matrix. In this section we demonstrate the use of our scaled Newton-Schulz algorithm for computing the density matrix.

Construction of the Fock matrix is the computationally intensive step in SCF iterations, but it is highly parallel. Computation of the density matrix, however, can be the bottleneck limiting parallel scalability: although the density matrix is not extremely large, it must be efficiently computed by using a large number of processors (which were used earlier to construct the Fock matrix), much like the solution of small coarse-grid problems in parallel multigrid. Algorithms of Newton-Schulz type (called McWeeny purification [19] in the quantum chemistry literature) that are rich in matrix-matrix multiplications are thus attractive in the high parallelism setting.

We generated the core-Hamiltonian matrix for two hydrocarbon molecules: a graphene-like molecule, $C_{384}H_{48}$, and a linear alkane, $C_{418}H_{838}$. The core-Hamiltonian

TABLE 7.1
Hydrocarbon test problems. The dimension of the core-Hamiltonian matrix is the basis size, and n_{occ} is the number of occupied orbitals. The value $\lambda_{|\min|}$ is the smallest magnitude eigenvalue of A after it has been scaled to have unit spectral radius.

	Basis Size	n_{occ}	$\lambda_{\|\min\|}$
Graphene $C_{384}H_{48}$	5,616	1,176	`4.1e-05`
Alkane $C_{418}H_{838}$	10,042	1,673	`2.4e-05`

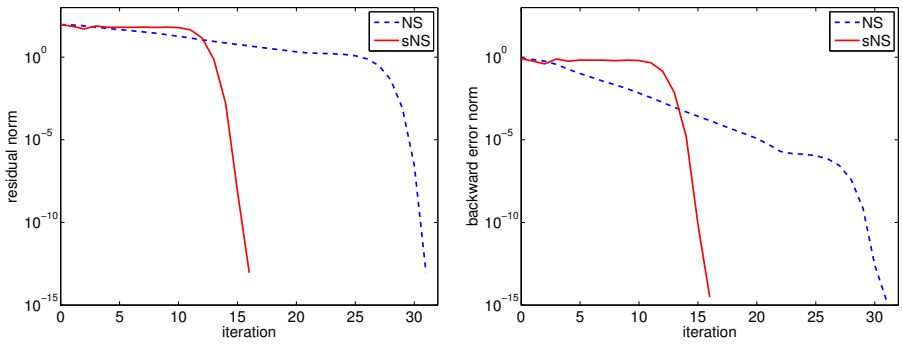
describes the kinetic energy and nuclear attraction of electrons, but not electron-electron interactions, and is often used as the initial guess for the Fock matrix in SCF iterations. Since the Fock matrix itself depends on the density matrix, the core-Hamiltonian initial guess corresponds to a zero initial guess for the density matrix. Here, we compute the density matrix corresponding to these core-Hamiltonians. The elements of the core-Hamiltonian matrices were formed by using the Dunning cc-pVDZ basis set [6] using a standard quantum chemistry code [18]. Table 7.1 shows the resulting matrix dimension (equal to the number of basis functions) for the two test problems.

The density matrix is the spectral projector associated with the n_{occ} lowest eigenvalues and their corresponding eigenvectors, where n_{occ} is the number of occupied orbitals, or half the number of electrons assuming closed-shell orbitals. To compute the density matrix via the sign function, we first compute the sign of

$$A = \mu I - H,$$

where H is the core-Hamiltonian in our case and μ, known as the Fermi level, separates the occupied eigenvalues from the unoccupied eigenvalues. The Fermi level is often known, especially for problems with a large energy band gap. For our tests, we chose the Fermi level to lie exactly between the n_{occ} and $n_{occ}+1$-st eigenvalue of A, sorted in increasing order. This is optimal for the Newton-Schulz method and the scaled variant. In practice, such an exact shift is not known. On the other hand, using the core-Hamiltonian is a kind of worse case, since there is no gap of eigenvalues around μ, as would be the case for more converged Fock matrices in later SCF iterations. We note that once $\text{sign}(A)$ is computed, the density matrix is given by $(\text{sign}(A)+I)/2$. The density matrix can also be computed directly from A by using the McWeeny mapping $g(x) = 3x^2 - 2x^3$, which can also be modified to accelerate convergence as we have in this paper modified the Newton-Schulz mapping.

Figure 7.1 plots the residual norm $\|X_k^2 - I\|_F$ and the backward error (4.1) using the Frobenius norm for Newton-Schulz and for the new scaled method for the alkane test problem. The plots for the graphene test problem are qualitatively the same. (Note that for Newton-Schultz, an initial scaling of the test matrix is used such that the maximum magnitude eigenvalue has magnitude 1.) As we have seen, the scaled method converges in approximately half the number of iterations of the original method. For the original method, convergence in the Frobenius norm is monotone, as every eigenvalue is improved (pushed toward the correct direction) at each iteration. For the scaled method, although it is faster, the situation is different. Here, we observe a "dip" in the first five iterations: an initial decrease, faster than the decrease of standard Newton-Schulz, followed by an increase. In the scaled method, only the eigenvalue closest to zero is guaranteed to improve monotonically; the nonmonotone convergence of other eigenvalues makes the Frobenius norm of the residual converge nonmonotonically.



(a) Residual norm (b) Backward error norm

FIG. 7.1. *Convergence of Newton-Schulz (NS) and scaled Newton-Schulz (sNS) for the alkane* $C_{418}H_{838}$ *core-Hamiltonian.*

8. Conclusion. The multiplication-rich property of Newton-Schulz makes it preferable over other methods for computing the sign function of a Hermitian matrix in the setting of high-performance computing. In this paper, we presented a stable, scaled variant that accelerates the initially slow convergence of the iteration. The proposed algorithm generally converges twice as fast as Newton-Schulz. In exact arithmetic, at most 44 iterations are needed to converge within a tolerance of 10^{-16} for matrices with condition number no greater than 10^{16}. Parallel implementations are straightforward, using existing highly optimized codes for matrix multiplication.

The proposed algorithm requires estimation of the largest magnitude eigenvalue, as does the standard Newton-Schulz method. On the other hand, most of the convergence theory for the proposed algorithm is based on an accurate calculation of the smallest magnitude eigenvalue as well. Nevertheless, we have proved a result stating that the quadratic convergence is maintained even if the smallest magnitude eigenvalue is blindly set. In practice, this value is not arbitrary, but it may be a very crude estimate. Finally, the best performance of the proposed algorithm is achieved by accurately calculating all the following eigenvalues: the two extreme ones and the two straddling the origin, in which case shifting can be applied as a form of preconditioning.

Acknowledgments. We are indebted to Nicholas Higham and Yuji Nakatsukasa for their keen discussions on the stability issue of the optimally scaled Newton-Schulz iteration, which resulted in a robust improvement of the algorithm after the initial submission of the manuscript. We gratefully acknowledge use of the Blues cluster in the Laboratory Computing Resource Center at Argonne National Laboratory.

REFERENCES

[1] G. BALLARD, J. DEMMEL, O. HOLTZ, AND O. SCHWARTZ, *Minimizing communication in numerical linear algebra*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 866–901.

[2] R. BYERS AND H. XU, *A new scaling for Newton's iteration for the polar decomposition and its backward stability*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 822–843.

[3] E. R. DAVIDSON, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87–94.

[4] T. A. DAVIS, *Direct Methods for Sparse Linear Systems*, SIAM, 2006.

[5] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM Journal on Scientific Computing, 34 (2012), pp. A206–A239.
[6] T. H. Dunning Jr, *Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen*, The Journal of Chemical Physics, 90 (1989), p. 1007.
[7] T. Ericsson and A. Ruhe, *The spectral transformation Lánczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems*, Math. Comp., 35 (1980), pp. 1251–1268.
[8] H.-R. Fang and Y. Saad, *A filtered Lanczos procedure for extreme and interior eigenvalue problems*, SIAM J. Sci. Comput., 34 (2012), pp. A2220–A2246.
[9] D. R. Fokkema, G. L. G. Sleijpen, and H. A. V. der, *Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.
[10] A. Frommer, T. Lippert, B. Medeke, and K. Schilling, eds., *Numerical Challenges in Lattice Quantum Chromodynamics*, vol. 15 of Lecture Notes in Computational Science and Engineering, Springer, 2000.
[11] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, 2008.
[12] C. S. Kenney and A. J. Laub, *The matrix sign function*, IEEE Transactions on Automatic Control, 40 (1995), pp. 1330–1348.
[13] A. Kiełbasiński and K. Ziętak, *Numerical behaviour of Higham's scaled method for polar decomposition*, Numerical Algorithms, 32 (2003), pp. 105–140.
[14] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly re-started Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 789–821.
[15] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, 1998.
[16] X.-P. Li, R. W. Nunes, and D. Vanderbilt, *Density-matrix electronic-structure method with linear system-size scaling*, Phys. Rev. B, 47 (1993), pp. 10891–10894.
[17] X. S. Li, *An overview of SuperLU: Algorithms, implementation, and user interface*, ACM Trans. Math. Softw., 31 (2005), pp. 302–325.
[18] V. Lotrich, N. Flocke, M. Ponton, A. Yau, A. Perera, E. Deumens, and R. Bartlett, *Parallel implementation of electronic structure energy, gradient, and Hessian calculations*, The Journal of Chemical Physics, 128 (2008), p. 194104.
[19] R. McWeeny, *Some recent advances in density matrix theory*, Rev. Mod. Phys., 32 (1960), pp. 335–369.
[20] R. B. Morgan and D. S. Scott, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 817–825.
[21] Y. Nakatsukasa, Z. Bai, and F. Gygi, *Optimizing Halley's iteration for computing the matrix polar decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2700–2720.
[22] Y. Nakatsukasa and N. J. Higham, *Backward stability of iterations for computing the polar decomposition*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 460–479.
[23] H. Neuberger, *Exactly massless quarks on the lattice*, Physics Letters B, 417 (1998), pp. 141–144.
[24] E. H. Rubensson, *Nonmonotonic recursive polynomial expansions for linear scaling calculation of the density matrix*, J. Chem. Theory Comput., 7 (2011), pp. 1233–1236.
[25] Y. Saad, *Numerical Methods for Large Eigenvalue Problems, Revised Edition*, SIAM, 2011.
[26] Y. Saad, J. Chelikowsky, and S. Shontz, *Numerical methods for electronic structure calculations of materials*, SIAM Review, 52 (2010), pp. 3–54.
[27] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[28] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*, Dover, 1989.
[29] J. van den Eshof, A. Frommer, T. Lippert, K. Schilling, and H. A. van der Vorst, *Numerical methods for the QCD overlap operator: I. sign-function and error bounds*, Computer Physics Communications, 146 (2002), pp. 203–224.
[30] S. Wang, X. S. Li, J. Xia, Y. Situ, and M. V. D. Hoop, *Efficient scalable algorithms for solving linear systems with hierarchically semiseparable structures.* submitted to SIAM J. Sci. Comput., 2012.