Scaling Bini’s Algorithm for Fast Inversion of Triangular Toeplitz Matrices

Jie Huang† Ting-Zhu Huang‡ Skander Belhaj§
School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P. R. China
University of Tunis El Manar, ENIT-LAMSIN, BP 37, 1002, Tunis, Tunisia

Abstract
In this paper, motivated by Lin, Ching and Ng [Theoretical Computer Science, 315:511-523 (2004)], a scaling version of Bini’s algorithm [SIAM J. Comput., 13:268-276 (1984)] for an approximate inversion of a triangular Toeplitz matrix is proposed. The scaling algorithm introduces a new scale parameter and is mathematically equivalent to the original Bini’s. Its computational cost is about two fast Fourier transforms of $n$-vectors (FFTs($n$)), equal to that of Bini’s. We also improve the accuracy of the proposed approach by embedding the $n$-by-$n$ triangular Toeplitz matrix into an $(n + n_0)$-by-$(n + n_0)$ triangular (banded) Toeplitz matrix, where $n_0$ is a positive integer. The complexity of the resulting revised scaling Bini’s algorithm is about two FFTs($2n$). Several numerical examples are given to illustrate the effectiveness and stability of the proposed methods.

Key words: Bini’s algorithm; Toeplitz matrix; Fast Fourier transform; Inverse; Triangular matrix

AMSC (2010): 65F05; 65F30

1 Introduction
An $n$-by-$n$ matrix $T_n = [t_{j,k}; j, k = 0, 1, \cdots, n-1]$ is said to be Toeplitz if it is constant along its diagonals, i.e., $t_{j,k} = t_{j-k}$. Such matrices arise in a variety of applications in mathematics and engineering, such as signal and image processing [6] and minimum realization problems in control theory; see Bunch [5] and the references therein. In this paper, we focus our attention on fast inversion of an $n$-by-$n$ lower triangular Toeplitz matrix, i.e.,

$$T_n = \begin{pmatrix} t_0 & t_0 & \cdots & \cdots & t_0 \\ t_1 & t_0 & \cdots & \cdots & t_0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ t_{n-1} & \cdots & \cdots & t_1 & t_0 \end{pmatrix},$$

where $t_j, j = 0, 1, \cdots, n-1$ are real with $t_0 \neq 0$.

†This research is supported by NSFC (60973015, 61170311), Chinese Universities Specialized Research Fund for the Doctoral Program (201110185110020) and Sichuan Province Sci. & Tech. Research Project (12ZC1802).

‡E-mail address: jiehuangjh86@gmail.com

§E-mail address: tingzhuhuang@126.com

Skander Belhaj§

E-mail address: skander.belhaj@lamsin.rnu.tn
For fast inversion of triangular Toeplitz matrices, Morf in [13] noted that the divide-and-conquer strategy yields an algorithm using $O(n \log n)$ operations (the same order as a convolution using the fast Fourier transform (FFT)). Then Commenges and Monsion [7] proposed an algorithm requiring $O(n \log n)$ operations for inversion of triangular Toeplitz matrices, more precisely, about 10 fast Fourier transforms (FFTs) of $n$-vectors (FFT$(n)$). After several years of intensive study, the approximate approach using the fast Fourier transform for inversion of triangular Toeplitz matrices has been studied by Bini [2] and Georgiev [9]. Considering triangular Toeplitz matrices versus polynomials [14, 3, 4, 15, 16, 17], many available techniques for polynomial division can be used, such as Knuth, Sieveking-Kung [11] and Bini & Pan [3].

Recently, Lin, Ching and Ng put forward an approximate inversion method for triangular Toeplitz matrices based on trigonometric polynomial interpolation [12]. Moreover, they proposed a revised Bini’s algorithm for a triangular Toeplitz matrix inversion. In this paper, we propose scaling and revised scaling Bini’s methods for fast inversion of triangular Toeplitz matrices. The basic idea is to introduce a new scale parameter used by Lin, Ching and Ng and choose a distinct numerical value. The proposed algorithms are mathematically equivalent to the original ones without scaling, respectively. Therefore, there is not additional computational cost. Several numerical examples are given to illustrate the effectiveness and stability of the proposed algorithms.

The outline of the paper is as follows. In section 2, we provide an improvement on a result of Bini for computing the approximate inversion of a lower triangular Toeplitz matrix and then give a scaling algorithm. We present also the revised version of Bini’s algorithm which brings us back to derive the revised scaling Bini’s Algorithm [12]. Section 3 uses some numerical experiments to demonstrate the efficiency and necessity of this modification. Finally, section 4 draws some conclusions.

2 The scaling version of Bini’s algorithm

2.1 Inversion based on Bini’s method

Before we move into fast algorithms for triangular Toeplitz matrix inversion, we first present a common special, but very important case of Toeplitz matrices: circulant matrix. An $n$-by-$n$ Toeplitz matrix $C_n([C_n]_{jk} = c_{j-k})$ is said to be a circulant if it satisfies $c_{j-k} = c_{n-k}$ for $k = 1, 2, \ldots, n-1$. There are many important properties of a circulant matrix. Of particular importance to us is that all circulant matrices can be diagonalized by the Fourier matrix, and the multiplication of a circulant matrix to a vector can be done in $O(n \log n)$ operations by using FFT [10, 8]. In this event, Bini [2] proposed an approximate approach for fast inversion of triangular Toeplitz matrix $T_n$.

For all $n \geq 1$, let $H_n = [h_{jk}]_{j,k=1}^n$ be the lower shift matrix with ones on the first subdiagonal and zeros elsewhere. We see that

$$T_n = \sum_{j=0}^{n-1} t_j H_n^j.$$

The basic idea of Bini’s algorithm is to use $H_n^{(\varepsilon)} = [h_{jk}^{(\varepsilon)}]_{j,k=1}^n$ to approximate $H_n$, where $h_{jk}^{(\varepsilon)} = h_{jk}$ for $(j, k) \neq (1, n)$ and $h_{1n}^{(\varepsilon)} = \varepsilon^n$, here $\varepsilon^n$ is a small positive number. It follows that $T_n$ can be
approximated by the circulant matrix
\[ T_n^{(\varepsilon)} = \sum_{j=0}^{n-1} t_j (H_n^{(\varepsilon)})^j. \]

Let \( D_n^{(\varepsilon)} = \text{diag}(1, \varepsilon, \cdots, \varepsilon^{n-1}) \), \( D_n = \text{diag}(d) \), where \( d = \sqrt{n} F_n D_n^{(\varepsilon)} [t_j]_{j=0}^{n-1} \), then we have \( (T_n^{(\varepsilon)})^{-1} \) by using the decomposition \( (T_n^{(\varepsilon)})^{-1} = (D_n^{(\varepsilon)})^{-1} F_n D_n^{(\varepsilon)} \), where \( F_n \) is the \( n \)-by-\( n \) Fourier matrix. On the other hand, it is well-known that the inverse of a triangular Toeplitz matrix is also a triangular Toeplitz matrix. Hence, the Bini’s Algorithm for computing the first column \( b^{(\varepsilon)} \) of \( (T_n^{(\varepsilon)})^{-1} \) can be concluded as follows.

**Algorithm 1. Bini’s algorithm**

*Step 0:* Choose \( \varepsilon \in (0, 1) \). Compute \( \tilde{t}_j = t_j \varepsilon^j \), for \( j = 0, 1, \cdots, n - 1 \).

*Step 1:* Compute \( d = (\sqrt{n} F_n)^{\frac{1}{2}} \).

*Step 2:* Compute \( c = [c_j]_{j=0}^{n-1} = [1/d_j]_{j=0}^{n-1} \).

*Step 3:* Compute \( f = (F_n^{2}/\sqrt{n})c \).

*Step 4:* Compute \( b^{(\varepsilon)} = [b_j^{(\varepsilon)}]_{j=0}^{n-1} = [f_j/\varepsilon^j]_{j=0}^{n-1} \).

We note that the computational cost of Bini’s algorithm is about two FFT(\( n \)).

Revised Bini’s algorithm [12, Algorithm 2] was proposed to obtain a faster and more accurate approximate inverse by embedding the \( n \)-by-\( n \) triangular Toeplitz matrix into an \( (n + n_0) \)-by-\( (n + n_0) \) triangular (banded) Toeplitz matrix, where \( n_0 \) is a positive integer. For simplicity, they set \( n_0 = n \) and stated the revised algorithm as follows.

**Algorithm 2. Revised Bini’s algorithm**

*Step 0:* Choose \( \varepsilon \in (0, 1) \). Compute \( \tilde{t}_j = t_j \varepsilon^j \), for \( j = 0, 1, \cdots, n - 1 \), and set \( \tilde{t}_j = 0 \) for \( j = n, n + 1, \cdots, 2n - 1 \).

*Step 1:* Compute \( d = (\sqrt{2n} F_{2n})^{\frac{1}{2}} \).

*Step 2:* Compute \( c = [c_j]_{j=0}^{2n-1} = [1/d_j]_{j=0}^{2n-1} \).

*Step 3:* Compute \( f = (F_{2n}^{2}/\sqrt{2n})c \).

*Step 4:* Compute \( b^{(\varepsilon)} = [b_j^{(\varepsilon)}]_{j=0}^{n-1} = [f_j/\varepsilon^j]_{j=0}^{n-1} \).

Clearly the computational cost of Algorithm 2 is about two FFT(\( 2n \)), about twice that of Algorithm 1.

In Algorithms 1 and 2, special attention should be paid to the choice of parameter \( \varepsilon \). Recall that we use \( H_n^{(\varepsilon)} \) to approximate \( H_n \), so that \( T_n \) is well approximated by \( T_n^{(\varepsilon)} \). Thus, theoretically, the smaller \( \varepsilon^n \), the more accurate the approximate inverse will be. However, notice that if \( \varepsilon^n \) is close to zero, \( D_n^{(\varepsilon)} \) will be very ill-conditioned for large \( n \), the computed vector \( c \) by Step 2 in Algorithms 1 and 2 will be therefore not accurate, consequently, the computed \( b^{(\varepsilon)} \) will not be a meaningful approximation solution. Hence, it is both necessary and important to choose a suitable value of parameter \( \varepsilon \) to balance the two facts. In fact, it has been pointed out in [12] that for a more accurate numerical inverse and a decreased rounding error, \( \varepsilon = (0.5 \times 10^{-8})^{1/n} \) and \( \varepsilon = 10^{-5/n} \) are good choices for Bini’s and revised Bini’s algorithm, respectively, which is consistent with our numerical tests.

### 2.2 Inversion based on scaling Bini’s Algorithm

In this subsection we present our main algorithm. By introducing a simple scale parameter which is crucial in practice based on the Bini’s algorithm for a triangular Toeplitz matrix
inversion, we obtain a faster and more accurate approximate inversion.

Recall that the inverse of triangular Toeplitz matrix is also triangular Toeplitz, i.e.,

$$T_n^{-1} = \begin{pmatrix} t_0 & t_1 & \cdots & t_{n-1} \\ t_1 & t_0 & \cdots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n-1} & \cdots & t_1 & t_0 \end{pmatrix}^{-1} = \begin{pmatrix} b_0 & b_1 & \cdots & b_{n-1} \\ b_1 & b_0 & \cdots & b_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1} & \cdots & b_1 & b_0 \end{pmatrix}.$$ 

Moreover, from [12], for any \( \rho \), define

$$T_n^{(\rho)} = \begin{pmatrix} t_0 & \rho t_1 & \cdots & \rho^{n-1} t_{n-1} \\ \rho t_1 & t_0 & \cdots & \rho^{n-2} t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} t_{n-1} & \cdots & \rho t_1 & t_0 \end{pmatrix},$$

then we have

$$(T_n^{(\rho)})^{-1} = \begin{pmatrix} b_0 & \rho b_1 & \cdots & \rho^{n-1} b_{n-1} \\ \rho b_1 & b_0 & \cdots & b_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} b_{n-1} & \cdots & b_1 & b_0 \end{pmatrix}.$$ 

Using this scaling technique, i.e., \( \rho = 2^{-18/n} \in (0,1) \), the authors in [12] improves the accuracy of trigonometric polynomial interpolation based approximate solution remarkably. Thus with a similar approach, we give a scaling version of Bini’s algorithm, called the scaling Bini’s algorithm.

**Algorithm 3. Scaling Bini’s algorithm**

*Step 0:* Choose a suitable value \( \varepsilon_0 \in (0,1) \). Choose a suitable \( \rho \) and compute \( \hat{t}_j = \varepsilon_0^j t_j \rho^j, j = 0,1,\ldots, n-1 \).

*Step 1:* Compute \( d = (\sqrt{n}F_n)\hat{t} \).

*Step 2:* Compute \( c = [\varepsilon_0 j = 0]^{n-1} = [1/d_j]^{n-1} \).

*Step 3:* Compute \( f = (F_n^*/\sqrt{n})c \).

*Step 4:* Compute \( \hat{b}_j^{n-1} = (f_j/\varepsilon_0 \rho^j) j = 0 \).

We note that the scaling Bini’s algorithm is mathematically equivalent to the original Bini’s by assuming \( \varepsilon = \rho \varepsilon_0 \). Here our contribution is mainly to choose a distinct numerical value of \( \rho \) (numerically we assume \( \varepsilon_0 = \varepsilon \)), to improve the numerical stability of Bini’s algorithm. Thus, the computational cost of Algorithm 3 is the same to that of Algorithm 1, about two FFT\((n)\), equal to that of Bini’s algorithm (Algorithm 1) and half of the revised one (Algorithm 2). Moreover, we have the following two remarks.

**Remark 1** Although the two parameters \( \varepsilon_0 \) and \( \rho \) appear together, their properties and behavior are quite different form both the computational and the theoretical point of view. The parameter \( \varepsilon_0 \) is used to obtain a circulant approximation of \( T_n^{(\rho)} \), while the aim of \( \rho \) is to improve the accuracy of the computed first column of \( T_n^{-1} \) as a scale parameter of Algorithm 1. In addition, we assume \( \varepsilon_0 = (0.5 \times 10^{-8})^{1/n} \) in Algorithm 3 in the rest of paper since it is always a good choice and is not dependent on the choice of \( \rho \) unless it is meaningful.

**Remark 2** It is important to choose a suitable value of parameter \( \rho \) since only a suitable \( \rho \) can improve the accuracy of the computed inverse. Recall that the authors in [12] limited \( \rho \in (0,1) \) to make the parameter \( \rho \) meaningful. Moreover, they noted that \( \rho = 2^{-18/n} \) is a good choice. Here, we note that the parameter \( \rho \) in Algorithm 3 does not have to be limit in \((0,1)\) strictly.
Scaling Bini’s Algorithm for Triangular Toeplitz Inversion

and the computed results can also be improved remarkably with \(\rho: |\rho - 1| < \delta\), and \(\lim_{n \to \infty} \rho = 1\), where \(\delta(< 0.1)\) is a small positive number. Moreover, our numerical tests show that \(\rho = 2^{23/n}\) is a good choice.

To estimate the accuracy of \([\hat{b}_i]_{j=0}^{n-1}\) computed by Algorithm 3, we give a useful lemma and a followed theorem, which can be obtained immediately from [12, Theorem 2].

**Lemma 1** Let \(b^{(\rho, \varepsilon_0)} = [b^{(\rho, \varepsilon_0)}_{j}]_{j=0}^{n-1}\) and \(b^{(\rho)} = [b^{(\rho)}_{j}]_{j=0}^{n-1}\) be the first columns of the approximate inverse of \(T_n^{(\rho)}\) from Bini’s algorithm and the exact inversion, respectively. Then

\[
\hat{b}^{(\rho, \varepsilon_0)} - \hat{b}^{(\rho)} = O(\varepsilon_0^n),
\]

More precisely,

\[
\hat{b}^{(\rho, \varepsilon_0)} - \hat{b}^{(\rho)} = \varepsilon_0^n \sum_{k=1}^{\infty} \varepsilon_0^{(k-1)n} b^{(\rho)}_{j+kn}, \quad j = 0, 1, \ldots, n - 1.
\]

**Theorem 1** Let \(\hat{b} = [\hat{b}_i]_{j=0}^{n-1}\) and \(b = [b_i]_{j=0}^{n-1}\) be the first columns of the approximate inverse of \(T_n\) computed by Algorithm 3 and the exact inversion, respectively. Then

\[
\hat{b}_j - b_j = O(\varepsilon_0^n \rho^n), \quad j = 0, 1, \ldots, n - 1.
\]

**Proof.** From the assumption of \(\hat{b} = [\hat{b}_i]_{j=0}^{n-1}\) and \(b = [b_i]_{j=0}^{n-1}\), we have

\[
\hat{b}_j = b^{(\rho, \varepsilon_0)}_{j} / \rho^j, \quad b_j = b^{(\rho)}_{j} / \rho^j,
\]

for \(j = 0, 1, \ldots, n - 1\). Then Lemma 1 gives

\[
\hat{b}_j - b_j = \varepsilon_0^n \sum_{k=1}^{\infty} \varepsilon_0^{(k-1)n} \rho^k n b_{j+kn} = (\varepsilon_0 \rho)^n \sum_{k=1}^{\infty} (\varepsilon_0 \rho)^{(k-1)n} b_{j+kn},
\]

for \(j = 0, 1, \ldots, n - 1\); the desired result. \(\square\)

From the above result, numerically we see that if \(\rho\) is too small, computing \(1 / \rho^j\) will bring in very large rounding error for large \(j\). However, recall from [12] that if \(\rho\) is too large, then \(b_{n-1} \rho^{n-1}\) may be infinite for large \(n\), which leads to a meaningless approximate inverse. So we have to choose a suitable value of \(\rho\) to balance these two facts. It is worth noting that the optimal value of \(\rho\) for all situations is unlikely to exist. Nonetheless, in many applications, \(\rho_n = 2^{23/n}\) is a good choice for Algorithm 3 to improve the accuracy of the approximate solution to a great extent.

Based on Lin, Ching and Ng [12] techniques, we can further improve the accuracy of our scaling Bini’s algorithm by embedding the \(n\)-by-\(n\) triangular Toeplitz matrix into an \((n + n_0)\)-by-(\(n + n_0\)) triangular (banded) Toeplitz matrix, where \(n_0\) is a positive integer. For simplicity, we set \(n_0 = n\) and give the revised scaling Bini’s Algorithm as follows.

**Algorithm 4. Revised scaling Bini’s algorithm**

Step 0: Choose a suitable value \(\varepsilon_0 \in (0, 1)\). Choose a suitable \(\rho\) and compute \(\tilde{t}_j = \varepsilon_0^j t_j \rho^j, j = 0, 1, \ldots, n - 1\). Set \(\tilde{t}_j = 0\) for \(j = n, n + 1, \ldots, 2n - 1\).

Step 1: Compute \(d = (\sqrt{2nF_{2n}})[\tilde{t}_j]_{j=0}^{2n-1}\).
Step 2: Compute $c = [c_j]^{2n-1}_{j=0} = [1/d_j]^{2n-1}_{j=0}$.
Step 3: Compute $f = (F_{2n}^2/\sqrt{2n})c$.
Step 4: Compute $[\hat{b}_j]^{n-1}_{j=0} = [f_j/(\varepsilon_0 \rho)]^{n-1}_{j=0}$.

We remark that the revised scaling Bini's algorithm is also mathematically equivalent to the revised Bini's algorithm from [12] by replacing $\varepsilon$ with two functional different parameters $\varepsilon_0$ and $\rho$. The computational cost of Algorithm 4 is about two FFT($2n$).

To end this section, we give a simple numerical example to further illustrate the effect of rounding error of the scaling Bini’s algorithm. Let $T_n$ be the lower triangular Toeplitz matrix with the first column given by

$$t_j = 0.5^j, \quad j = 0, 1, \cdots, n - 1.$$

Let $b$ and $\hat{b}$ are the first columns of the exact inverse of $T_n$ and the approximate inversion from Algorithm 3, respectively. We assume $\varepsilon_0 = (0.5 \times 10^{-8})^{1/n}$, $\rho = 2^{23/n}$ (scaling Bini) and $\rho = 1$ (Bini), respectively.

The errors in the numerical results $\hat{b}$ of our scaling Bini’s algorithm for $\rho = 2^{23/4096}$ and $\rho = 1$ are shown in Fig. 1. Having $n$ form 2 to 4096, Fig. 2 depicts the relative errors for $\rho = 2^{23/n}$ and $\rho = 1$. It also shows, for $n > 40$ and $\rho = 2^{23/n}$, the accuracy of the numerical inversion can be improved to a great degree, which is consistent with the theoretical analysis. From Figs. 1 and 2, it is worthwhile stressing that setting $\rho = 2^{23/n}$ instead of 1 only, without any additional computational cost, the approximate inverse is shown to be more stable and efficient.

![Figure 1: log$_{10}$($|\hat{b} - b|$) for $t_j = 0.5^j$, $j = 0, 1, \cdots, 4095$ for $\rho = 2^{23/4096}$ and $\rho = 1$.](image)

3 Numerical experiments

We shall demonstrate the effectiveness of the scaling and revised scaling Bini’s algorithms. In particular, we will compare its performance with Bini’s algorithm, the revised Bini’s algo-
Scaling Bini’s Algorithm for Triangular Toeplitz Inversion

Figure 2: \( \log_{10}(\frac{\|\hat{b} - b\|_1}{\|b\|_1}) \) for \( t_j = 0.5^j, j = 0, 1, \ldots, n, n = 1, 2, \ldots, 4095. \)

algorithm and the algorithm based on trigonometric polynomial interpolation [12]. Seven different sequences of lower triangular Toeplitz matrices are tested. They are

1. \( t_j = \frac{1}{j+1}, j = 0, 1, \ldots, n-1, \)
2. \( t_j = 0.1^j, j = 0, 1, \ldots, n-1, \)
3. \( t_j = 0.5^j, j = 0, 1, \ldots, n-1, \)
4. \( t_j = 0.9^j, j = 0, 1, \ldots, n-1, \)
5. \( t_0 = 1, t_1 = 1/2 \) and \( t_j = 0, j = 2, 3, \ldots, n-1. \)
6. \( t_0 = 1, t_1 = -1/2, t_2 = 1/2 \) and \( t_j = 0, j = 3, \ldots, n-1. \)
7. \( t_0 = 1, t_1 = -1, t_2 = 1/2 \) and \( t_j = 0, j = 3, \ldots, n-1. \)

We note that the test sequence (1) comes from [12], and sequences (2)-(4) are the lower part of the well-known matrices, a class of Toeplitz test matrices:

\[
a_{i-j} = \eta^{|i-j|}, i = 1, \ldots, m; j = 1, \ldots, n,
\]

the parameter \( \eta \in (0, 1) \). Here we choose \( \eta = 0.1, 0.5 \) and 0.9, respectively. Moreover, sequences (5)-(7) which are not diagonally dominant will give some more information.

In the following tables, we show the relative accuracy of the approximate inverse \( \frac{\|\hat{b} - b\|_1}{\|b\|_1} \), where \( \hat{b} \) is the first column of the approximate inverse based on Bini’s algorithm and \( b \) is the inverse computed by the divide-and-conquer approach. The second, third and fourth rows display the accuracy of the computed inverses of Bini’s algorithm, the revised Bini’s algorithm and the algorithm based on trigonometric polynomial interpolation [12, Algorithm 1; with \( \rho = 2^{-18/n} \)], respectively. The fifth and sixth rows displays also the accuracy of the computed inverses of our scaling and revised scaling Bini’s Algorithms, respectively.

From Tables 1 through 7 we see that all approximate inverses of the five methods are very accurate. When not requiring very high order of accuracy, such as in the Gauss-Seidel iteration for Toeplitz systems, Bini’s algorithm is suitable. In some applications, such as in [1], we need an
approximation inversion as accurate as possible. In these cases, the revised, scaling and revised scaling Bini’s algorithms are more preferred. In addition, we remark that our scaling Bini’s algorithm requires two FFT($n$), a slightly lower computational cost, which is equal to that of Bini’s algorithm and about half cost of the revised or revised scaling Bini’s algorithms.

<table>
<thead>
<tr>
<th>$n$</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bini</td>
<td>4.57e-009</td>
<td>3.54e-009</td>
<td>4.73e-009</td>
<td>5.59e-009</td>
<td>7.37e-009</td>
<td>7.71e-009</td>
<td>8.38e-009</td>
</tr>
<tr>
<td>Revised</td>
<td>1.37e-012</td>
<td>1.75e-012</td>
<td>1.96e-012</td>
<td>2.15e-012</td>
<td>3.06e-012</td>
<td>3.55e-012</td>
<td>3.44e-009</td>
</tr>
<tr>
<td>Interpolation</td>
<td>3.00e-011</td>
<td>3.77e-011</td>
<td>4.65e-011</td>
<td>7.49e-011</td>
<td>1.16e-010</td>
<td>1.62e-010</td>
<td>2.24e-010</td>
</tr>
<tr>
<td>Scaling</td>
<td>6.41e-009</td>
<td>7.92e-010</td>
<td>9.85e-011</td>
<td>1.23e-011</td>
<td>1.53e-012</td>
<td>1.92e-013</td>
<td>2.41e-014</td>
</tr>
<tr>
<td>Revised scaling</td>
<td>8.69e-010</td>
<td>1.34e-011</td>
<td>2.01e-013</td>
<td>2.94e-015</td>
<td>1.30e-016</td>
<td>5.97e-017</td>
<td>1.25e-016</td>
</tr>
</tbody>
</table>

Table 1: Accuracy for $t_j = 1/(j + 1)^3$, $j = 0, 1, \cdots, n-1$

<table>
<thead>
<tr>
<th>$n$</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bini</td>
<td>2.71e-009</td>
<td>3.10e-009</td>
<td>3.86e-009</td>
<td>4.24e-009</td>
<td>5.19e-009</td>
<td>5.48e-009</td>
<td>5.21e-009</td>
</tr>
<tr>
<td>Revised</td>
<td>1.15e-012</td>
<td>1.61e-012</td>
<td>1.51e-012</td>
<td>1.87e-012</td>
<td>2.11e-012</td>
<td>1.82e-012</td>
<td>1.67e-012</td>
</tr>
<tr>
<td>Interpolation</td>
<td>1.70e-011</td>
<td>2.68e-011</td>
<td>4.64e-011</td>
<td>5.44e-011</td>
<td>8.69e-011</td>
<td>1.22e-011</td>
<td>1.73e-010</td>
</tr>
<tr>
<td>Scaling</td>
<td>1.06e-015</td>
<td>1.19e-015</td>
<td>9.39e-016</td>
<td>1.18e-015</td>
<td>1.15e-015</td>
<td>1.13e-015</td>
<td>1.16e-015</td>
</tr>
<tr>
<td>Revised scaling</td>
<td>2.95e-017</td>
<td>3.74e-017</td>
<td>3.43e-017</td>
<td>2.92e-017</td>
<td>3.43e-017</td>
<td>3.19e-017</td>
<td>3.42e-017</td>
</tr>
</tbody>
</table>

Table 2: Accuracy for $t_j = 0.1^j$, $j = 0, 1, \cdots, n-1$

<table>
<thead>
<tr>
<th>$n$</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revised</td>
<td>2.61e-012</td>
<td>2.20e-012</td>
<td>1.97e-012</td>
<td>2.45e-012</td>
<td>2.66e-012</td>
<td>2.72e-012</td>
<td>3.11e-012</td>
</tr>
<tr>
<td>Interpolation</td>
<td>1.93e-011</td>
<td>3.39e-011</td>
<td>3.75e-011</td>
<td>6.82e-011</td>
<td>8.61e-011</td>
<td>1.31e-010</td>
<td>2.03e-010</td>
</tr>
<tr>
<td>Scaling</td>
<td>1.27e-015</td>
<td>1.69e-015</td>
<td>2.11e-015</td>
<td>1.95e-015</td>
<td>1.90e-015</td>
<td>1.89e-015</td>
<td>2.28e-015</td>
</tr>
<tr>
<td>Revised scaling</td>
<td>4.95e-017</td>
<td>5.33e-017</td>
<td>1.11e-016</td>
<td>1.16e-016</td>
<td>6.62e-017</td>
<td>1.13e-016</td>
<td>5.86e-017</td>
</tr>
</tbody>
</table>

Table 3: Accuracy for $t_j = 0.5^j$, $j = 0, 1, \cdots, n-1$

4 Conclusions

In this paper we have presented a scaling Bini’s algorithm for approximate inversion of triangular Toeplitz matrices and performed extensive experiments to verify the performance for some triangular Toeplitz matrices. In particular, comparing with the initial Bini’s algorithm, the scaling one improves the accuracy of the numerical solution remarkably without additional computational cost. We also improve the accuracy of the proposed approach by the revised scaling Bini’s algorithm which requires two FFTs of $2n$-vectors.

References

### Table 4: Accuracy for $t_j = 0.9^j$, $j = 0, 1, \ldots, n - 1$

<table>
<thead>
<tr>
<th>$n$</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bini</td>
<td>8.06e-009</td>
<td>6.81e-009</td>
<td>1.41e-008</td>
<td>1.39e-008</td>
<td>2.01e-008</td>
<td>2.31e-008</td>
<td>2.30e-008</td>
</tr>
<tr>
<td>Revised</td>
<td>4.27e-012</td>
<td>6.74e-012</td>
<td>1.02e-011</td>
<td>9.87e-012</td>
<td>8.31e-012</td>
<td>8.55e-012</td>
<td>9.43e-012</td>
</tr>
<tr>
<td>Interpolation</td>
<td>4.50e-011</td>
<td>6.28e-011</td>
<td>1.04e-010</td>
<td>1.43e-010</td>
<td>3.07e-010</td>
<td>3.66e-010</td>
<td>5.01e-010</td>
</tr>
<tr>
<td>Scaling</td>
<td>5.83e-008</td>
<td>8.12e-014</td>
<td>5.17e-015</td>
<td>5.25e-015</td>
<td>6.11e-015</td>
<td>6.08e-015</td>
<td>6.56e-015</td>
</tr>
<tr>
<td>Revised scaling</td>
<td>1.35e-008</td>
<td>2.95e-016</td>
<td>3.54e-016</td>
<td>3.42e-016</td>
<td>3.27e-016</td>
<td>2.33e-016</td>
<td>2.64e-016</td>
</tr>
</tbody>
</table>

### Table 5: Accuracy for $t_0 = 1$, $t_1 = 1/2$, and $t_j = 0$, $j = 2, 3, \ldots, n - 1$

<table>
<thead>
<tr>
<th>$n$</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bini</td>
<td>4.77e-009</td>
<td>6.70e-009</td>
<td>6.08e-009</td>
<td>5.91e-009</td>
<td>7.41e-009</td>
<td>7.01e-009</td>
<td>7.11e-009</td>
</tr>
<tr>
<td>Revised</td>
<td>1.75e-012</td>
<td>1.32e-012</td>
<td>1.83e-012</td>
<td>2.53e-012</td>
<td>2.20e-012</td>
<td>2.31e-012</td>
<td>2.56e-012</td>
</tr>
<tr>
<td>Interpolation</td>
<td>1.28e-006</td>
<td>1.28e-006</td>
<td>1.28e-006</td>
<td>6.65e-009</td>
<td>4.78e-010</td>
<td>4.78e-010</td>
<td>1.99e-011</td>
</tr>
<tr>
<td>Scaling</td>
<td>1.39e-015</td>
<td>1.53e-015</td>
<td>1.66e-015</td>
<td>1.55e-015</td>
<td>1.76e-015</td>
<td>1.64e-015</td>
<td>1.74e-015</td>
</tr>
<tr>
<td>Revised scaling</td>
<td>6.00e-017</td>
<td>6.62e-017</td>
<td>1.07e-016</td>
<td>1.16e-016</td>
<td>1.10e-016</td>
<td>1.20e-016</td>
<td>4.30e-017</td>
</tr>
</tbody>
</table>

### Table 6: Accuracy for $t_0 = 1$, $t_1 = -1/2$, $t_2 = 1/2$ and $t_j = 0$, $j = 3, \ldots, n - 1$

<table>
<thead>
<tr>
<th>$n$</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bini</td>
<td>6.64e-009</td>
<td>4.90e-009</td>
<td>6.40e-009</td>
<td>8.78e-009</td>
<td>1.01e-008</td>
<td>9.10e-009</td>
<td>1.00e-008</td>
</tr>
<tr>
<td>Revised</td>
<td>2.50e-012</td>
<td>2.00e-012</td>
<td>2.95e-012</td>
<td>3.23e-012</td>
<td>2.85e-012</td>
<td>2.86e-012</td>
<td>3.03e-012</td>
</tr>
<tr>
<td>Interpolation</td>
<td>1.25e-006</td>
<td>1.25e-006</td>
<td>1.25e-006</td>
<td>5.84e-009</td>
<td>4.42e-010</td>
<td>4.42e-010</td>
<td>1.25e-010</td>
</tr>
<tr>
<td>Scaling</td>
<td>1.66e-015</td>
<td>1.85e-015</td>
<td>2.05e-015</td>
<td>1.97e-015</td>
<td>2.08e-015</td>
<td>2.05e-015</td>
<td>2.11e-015</td>
</tr>
<tr>
<td>Revised scaling</td>
<td>6.50e-017</td>
<td>9.85e-017</td>
<td>1.01e-016</td>
<td>1.31e-016</td>
<td>1.03e-016</td>
<td>1.25e-016</td>
<td>5.78e-017</td>
</tr>
</tbody>
</table>

### Table 7: Accuracy for $t_0 = 1$, $t_1 = -1/2$, $t_2 = 1/2$ and $t_j = 0$, $j = 3, \ldots, n - 1$

<table>
<thead>
<tr>
<th>$n$</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bini</td>
<td>1.15e-008</td>
<td>8.64e-009</td>
<td>1.24e-008</td>
<td>1.16e-008</td>
<td>1.97e-008</td>
<td>1.97e-008</td>
<td>1.82e-008</td>
</tr>
<tr>
<td>Revised</td>
<td>2.60e-012</td>
<td>2.70e-012</td>
<td>3.63e-012</td>
<td>4.09e-012</td>
<td>3.93e-012</td>
<td>3.84e-012</td>
<td>3.57e-012</td>
</tr>
<tr>
<td>Interpolation</td>
<td>1.49e-006</td>
<td>1.49e-006</td>
<td>1.49e-006</td>
<td>8.47e-009</td>
<td>6.22e-010</td>
<td>6.22e-010</td>
<td>1.79e-010</td>
</tr>
<tr>
<td>Scaling</td>
<td>2.58e-015</td>
<td>2.94e-015</td>
<td>2.44e-015</td>
<td>2.73e-015</td>
<td>3.24e-015</td>
<td>2.79e-015</td>
<td>3.20e-015</td>
</tr>
<tr>
<td>Revised scaling</td>
<td>8.12e-017</td>
<td>1.59e-016</td>
<td>7.62e-017</td>
<td>2.25e-016</td>
<td>1.68e-016</td>
<td>1.84e-016</td>
<td>9.51e-017</td>
</tr>
</tbody>
</table>

Table 4: Accuracy for $t_j = 0.9^j$, $j = 0, 1, \ldots, n - 1$

Table 5: Accuracy for $t_0 = 1, t_1 = 1/2$, and $t_j = 0$, $j = 2, 3, \ldots, n - 1$

Table 6: Accuracy for $t_0 = 1, t_1 = -1/2, t_2 = 1/2$ and $t_j = 0$, $j = 3, \ldots, n - 1$


