Error analysis of some operations involved in the Cooley-Tukey Fast Fourier Transform

Nicolas Brisebarre, Mioara Joldes, Jean-Michel Muller, Ana-Maria Naneş, Joris Picot

Error analysis of some operations involved in the Cooley-Tukey Fast Fourier Transform

Nicolas Brisebarre
CNRS, LIP, Université de Lyon, France
nicolas.brisebarre@ens-lyon.fr

Mioara Joldeş
CNRS, LAAS, Toulouse, France
joldes@laas.fr

Jean-Michel Muller
CNRS, LIP, Université de Lyon, France
jean-michel.muller@ens-lyon.fr

Ana-Maria Naneş
Technical University of Cluj-Napoca, Romania
anamaria.nanes@yahoo.com

Joris Picot
ENS de Lyon, LIP, Université de Lyon, France
joris.picot@ens-lyon.fr

October 23, 2019

keywords: Floating-point arithmetic, Fast Fourier Transform, Rounding error analysis

Abstract

We are interested in obtaining error bounds for the classical Cooley-Tukey FFT algorithm in floating-point arithmetic, for the 2-norm as well as for the infinity norm. For that purpose we also give some results on the relative error of the complex multiplication by a root of unity, and on the largest value that can take the real or imaginary part of one term of the FFT of a vector x, assuming that all terms of x have real and imaginary parts less than some value b.

1 Introduction and notation

The Fast Fourier Transform was introduced in 1965 by Cooley and Tukey in its modern form [4, 5, 22], but can be traced back to Gauss [7]. It is widely used in

digital signal processing [14]. It also plays a central role in fast multiple-precision arithmetic, since it lies at the heart of some of the most efficient big polynomial and big integer multiplication algorithms [20, 13]. Several studies have been devoted to the accuracy of Fast Fourier Transforms and fast algorithms for related transforms such as the DCT [17, 8, 19, 21, 9, 15, 16].

The Discrete Fourier Transform (DFT) $Z = (Z_0, Z_1, \ldots, Z_{N-1})$ of $z = (z_0, z_1, \ldots, z_{N-1}) \in \mathbb{C}^N$ is $(F_\omega z^t)^t$, where

$$F_\omega = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^j & \omega^{2j} & \cdots & \omega^{j(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{pmatrix}$$

and $\omega = e^{-2i\pi/N}$. Note that some authors use the matrix $(1/\sqrt{N}) \cdot F_\omega$ (in order to make the DFT a unitary transformation). In this paper, we will consider the radix-2 FFT algorithm, and we will assume that $N = 2^n$ is a power of 2. More precisely we will assume that the algorithm being implemented is the one described in pseudocode in Fig. 1, and illustrated, in the case $N = 8$ by Fig. 2. The presentation of the Algorithm in Fig. 1 aims at clarity and simplicity: in practice one will seldom implement FFT as presented in that figure, if only to have in-place calculation and consequently save memory, or to parallelize/vectorize the computation. However, if the dependency graph of the operations remains unchanged, our analyses remain valid.

```
/* Radix-2 FFT Algorithm                                          */
/* We assume the values omega[k,j] = exp(-i*j*pi/2^(k-1))
   are precomputed and stored, and reverse(n,j) is the n-bit
   number whose binary representation is the mirror image of
   the n-bit representation of j                                  */
Function reverse(n,j)
    return ∑_{k=0}^{n-1} ((j&(2^n − 1) ≫ k) mod 2) ≪ (n − 1 − k)
end
Function OneStep(x,k,n)
    N = 2^n;
    block_size = 2^k;
    N_blocks = N/block_size ;          /* Number of independent
     order-2^k FFTs */
    for block_number from 0 to N_blocks − 1 do
        first_index = block_number · block_size;
        for  j from 0 to block_size/2 − 1 do
            j1 = j + first_index;
            j2 = j1 + block_size/2;
            y[j1] = x[j1] + omega[k,j] · x[j2];
            y[j2] = x[j1] − omega[k,j] · x[j2];
        end
        return y
    end
end
Function FFT(x, n)
    N = 2^n;
    for j from 0 to N − 1 do
        y[j] = x[reverse(n,j)];
    end
    for k from 1 to n do
        y = OneStep(y,k,n);
    end
    return y
end
```

Figure 1: Pseudocode for the radix-2 FFT algorithm.

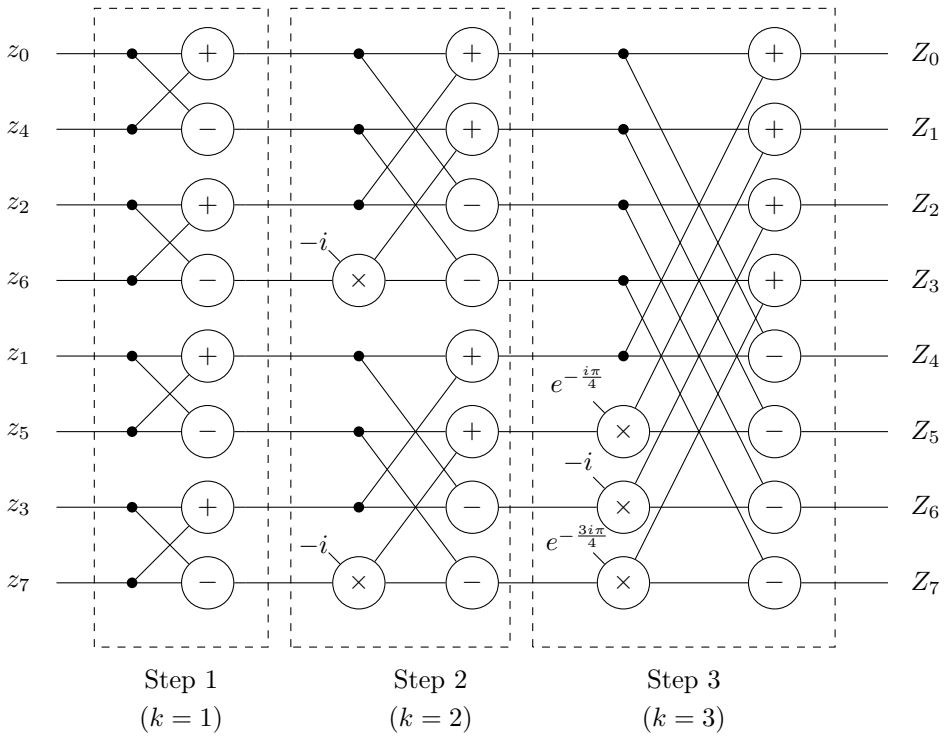

Figure 2: The various computations performed during an 8-point FFT. The first step can be viewed as 4 independent 2-point FFTs, and the first two steps constitute two independent 4-point FFTs.

We assume that we use a radix-2, precision-p, floating-point arithmetic with unbounded exponent range (which implies that the results presented in the paper hold in "real life" floating-point arithmetic provided that overflows and underflows do not occur). If $x \in \mathbb{R}$, define $\mathrm{RN}(x)$ as x rounded to nearest. This is the default rounding mode in IEEE 754 arithmetic [10], so that when the instruction `c = a*b` appears in a program, what is effectively computed is $c = \mathrm{RN}(ab)$. We have, for $x \neq 0$

$$\frac{|x - \mathrm{RN}(x)|}{|x|} \leqslant \frac{u}{1+u} < u, \tag{1}$$

where $u = 2^{-p}$ is called the *rounding unit*. We also have

$$|x - \mathrm{RN}(x)| \leqslant \frac{1}{2}\mathrm{ulp}(x), \tag{2}$$

where the ulp function (ulp is an acronym for *unit in the last place*) is defined as

$$\mathrm{ulp}(x) = \begin{cases} 0 & \text{if } x = 0, \\ 2^{\lfloor \log_2 |x| \rfloor - p + 1} & \text{otherwise.} \end{cases}$$

One easily notices that if $\mathrm{ulp}(x) \cdot 2^{p-1} \leqslant |x| \leqslant \mathrm{ulp}(x) \cdot \left(2^{p-1} + 1/4\right)$, then $\mathrm{RN}(x) = \mathrm{ulp}(x) \cdot 2^{p-1}$, and $|x - \mathrm{RN}(x)| \leqslant \frac{1}{4}\mathrm{ulp}(x)$. This leads us to the definition of another "ulp" function:

$$\mathrm{ulp}^*(x) = \begin{cases} \mathrm{ulp}(x) & \text{if } |x| > \left(2^{p-1} + \frac{1}{4}\right) \cdot \mathrm{ulp}(x), \\ \frac{1}{2}\mathrm{ulp}(x) & \text{otherwise,} \end{cases}$$

for which we also have $|x - \mathrm{RN}(x)| \leqslant \frac{1}{2}\mathrm{ulp}^*(x)$. Of course, ulp^* is almost always equal to ulp, but in the iterative algorithm of Section 5, where we manipulate values that are frequently just above a power of 2, using ulp^* instead of ulp makes a non-negligible difference.

In Section 5 we will also use the notation $\mathrm{RZ}(x)$ for x rounded towards zero. Functions RN (for all inputs) and RZ (for positive inputs) are increasing functions: if $0 \leqslant t_1 \leqslant t_2$ then $\mathrm{RZ}(t_1) \leqslant \mathrm{RZ}(t_2)$ and $\mathrm{RN}(t_1) \leqslant \mathrm{RN}(t_2)$. Note that $\mathrm{ulp}(x)$ and $\mathrm{ulp}^*(x)$ are increasing functions of $|x|$. Hence, if we know a bound B on $|x|$, we can deduce a bound $(1/2)\mathrm{ulp}^*(B) \leqslant (1/2)\mathrm{ulp}(B)$ on $|\mathrm{RN}(x) - x|$.

If $z = (z_0, z_1, \ldots, z_{N-1}) \in \mathbb{C}^N$, we will use the standard notations

$$\|z\|_2 = \sqrt{|z_0|^2 + |z_1|^2 + \cdots + |z_{N-1}|^2} \qquad \text{(2-norm)}$$

and

$$\|z\|_\infty = \max_{i=0,\ldots,N-1} |z_i| \qquad \text{(infinity norm).}$$

We will also use the following notation:

$$\|z\|_\infty^\perp = \max_{i=0,\ldots,N-1} \left\{\max\left(|\Re(z_i)|, |\Im(z_i)|\right)\right\}.$$

We remind the classical inequalities

$$\|z\|_\infty \leqslant \|z\|_2 \leqslant \sqrt{N} \cdot \|z\|_\infty, \tag{3}$$

$$\|z\|_\infty^\perp \leqslant \|z\|_\infty \leqslant \sqrt{2} \cdot \|z\|_\infty^\perp. \tag{4}$$

Choosing which norm should be used for expressing bounds on numerical errors depends much on the problem being dealt with. For expressing the error of the FFT, and assuming z is the input, Z is the exact result and $\hat{Z}$ is the computed result, most authors give a bound on

$$\frac{\|Z - \hat{Z}\|_2}{\|Z\|_2},$$

also called *root mean square* (RMS) *relative error*. The reason is twofold: first, the 2-norm appears naturally in many signal processing applications, and second, the well-known relation $\|Z\|_2 = \sqrt{N} \cdot \|z\|_2$ makes error bounds in terms of 2-norm easier to obtain than error bounds in terms of infinity norms. However, for applications such as the multiplication of big integers (or large polynomials with integer coefficients), we know that the components of the final "exact", theoretical, result of a calculation must be integers. We wish to retrieve these

integers by rounding to the nearest integer the coefficients of the actually computed result. For this strategy to work properly, we must be certain that these coefficients are within distance less than 1/2 from the exact value: in such a case, we need a bound on $\|Z - \hat{Z}\|_\infty^\perp$, given a bound on $\|z\|_\infty^\perp$.

The set of the floating-point numbers is invariant through a multiplication by a power of 2, and if $t = 2^k x$ then $\mathrm{RN}(t) = 2^k \mathrm{RN}(x)$. A consequence of this is that if Z is the exact DFT of z, and if $\hat{Z}$ is the computed value of that DFT using the FFT algorithm in floating-point arithmetic, then if we compute with the same algorithm the DFT of $2^k z$, we will obtain $2^k \hat{Z}$, so that the relative error will be the same. Therefore, if we wish to bound the relative error of the FFT algorithm, it suffices to focus (for instance) on input values z such that $1/2 \leqslant \|z\|_\infty^\perp < 1$.

Let us now consider error bounds proposed in the literature. We start with the results based on the 2-norm.

Just one year after the publication of the seminal paper [4], Gentleman and Sande [6] gave the following result (adapted here to radix-2 FFT and correctly rounded floating-point arithmetic)

Theorem 1 ([6])**.** *Assume radix-2, precision-p arithmetic, with rounding unit $u = 2^{-p}$. Let $\hat{Z}$ be the computed 2^n-point FFT of $z \in \mathbb{C}^{2^n}$, and let Z be the exact value. Then*

$$\frac{\|Z - \hat{Z}\|_2}{\|Z\|_2} \leqslant 8.48 \cdot n \cdot 2^{n/2} \cdot u + \mathcal{O}(u^2).$$

Again adapted to the modern context of correctly rounded arithmetic, and assuming that the real and imaginary parts of the roots of unity are rounded to nearest (i.e. setting the parameter γ of [17] to 1/2) , Theorem 1 of [17] gives

Theorem 2 ([17])**.** *Assume radix-2, precision-p arithmetic, with rounding unit $u = 2^{-p}$. Let $\hat{Z}$ be the computed 2^n-point FFT of $z \in \mathbb{C}^{2^n}$, and let Z be the exact value. Then*

$$\frac{\|Z - \hat{Z}\|_2}{\|Z\|_2} \leqslant \left[n \cdot (4 + \sqrt{2}) - 4\right] \cdot u + \mathcal{O}(u^2).$$

In his book [9], Higham proves the following result.

Theorem 3 ([9])**.** *Assume radix-2, precision-p arithmetic, with rounding unit $u = 2^{-p}$. Assume the roots of unity used in the algorithm are known with error less than or equal to μ. Let $\hat{Z}$ be the computed 2^n-point FFT of $z \in \mathbb{C}^{2^n}$, and let Z be the exact value. Then*

$$\frac{\|Z - \hat{Z}\|_2}{\|Z\|_2} \leqslant \frac{n\eta}{1 - n\eta},$$

where

$$\eta = \mu + \frac{4u}{1 - 4u} \cdot \left(\sqrt{2} + \mu\right).$$

After having proved that the relative error of the naive complex multiplication algorithm is bounded by $\sqrt{5} \cdot u$, Percival [15] deduces the following theorem. The proof of the bound $\sqrt{5} \cdot u$ in [15] turned out to be slightly incorrect: see [2] for a complete proof.

Theorem 4 ([15]). *Assume radix-2, precision-p arithmetic, with rounding unit $u = 2^{-p}$. Assume the roots of unity used in the algorithm are known with error less than or equal to μ. Let $\hat{Z}$ be the computed 2^n-point FFT of $z \in \mathbb{C}^{2^n}$, and let Z be the exact value. Then*

$$\frac{\|Z - \hat{Z}\|_2}{\|Z\|_2} \leqslant (1+u)^n \cdot (1+u\sqrt{5})^n \cdot (1+\mu)^n - 1.$$

Calvetti [3] and Schatzman [19] take a different approach to the roundoff error analysis of the FFT. They try to estimate the statistical distribution of the error on the result. They end up with an error that grows like $\sqrt{n}$. We will not consider that approach here, because we want to obtain sure error bounds.

Tasche and Zeuner [21] perform a worst and average case error analysis of the FFT. From Theorem 4.1 of their paper, one can deduce

Theorem 5 ([21]). *Assume radix-2, precision-p arithmetic, with rounding unit $u = 2^{-p}$. Assume that the roots of unity are precomputed (with rounded to nearest real and imaginary parts) and that the relative error of complex multiplication is bounded by $\rho_\times$. Let $\hat{Z}$ be the computed 2^n-point FFT of $z \in \mathbb{C}^{2^n}$, and let Z be the exact value. Then*

$$\frac{\|Z - \hat{Z}\|_2}{\|Z\|_2} \leqslant \left((n-2) \cdot \frac{\sqrt{2}}{2} + n \right) \cdot u + n \cdot \rho_\times + \mathcal{O}(u^2).$$

Assuming $n \ll 1/u$ (which always holds in practice), and assuming that the real and imaginary parts of the roots of unity are rounded to nearest, the bound given by Theorem 2 is around $(6.41 \cdot n - 5) \cdot u$, the bound given by Theorem 3 is around $6.36 \cdot n \cdot u$, the bound given by Theorem 4 is around $3.94 \cdot n \cdot u$, and (assuming complex multiplication is performed using (11), see Section 2.2), the bound given by Theorem 5 is around $3.71 \cdot n \cdot u$.

Of course, from bounds involving the 2-norm, one can deduce bounds involving the other norms, using (3) and (4). For instance, from

$$\frac{\|Z - \hat{Z}\|_2}{\|Z\|_2} \leqslant B, \tag{5}$$

one can deduce

$$\frac{\|Z - \hat{Z}\|_\infty}{\|Z\|_\infty} \leqslant B\sqrt{N}, \tag{6}$$

or

$$\|Z - \hat{Z}\|_\infty^\perp \leqslant B \cdot N \cdot \sqrt{2} \cdot \|z\|_\infty^\perp. \tag{7}$$

The bounds (6) and (7) are the best that one can deduce from (5). However, they are not necessarily the best that one can deduce from a direct analysis of the algorithm in terms of infinity norms.

Henrici [8, Page 14] gives an analysis based on the infinity norm. He shows the following result.

Theorem 6. *(Theorem 13.1c of [8]) Assume radix-2, precision-p arithmetic, with rounding unit $u = 2^{-p}$. Assume that the roots of unity used in the algorithm are known with error less than or equal to μ, and that the complex arithmetic operations are performed with relative error bounded by the same constant μ. Let $\hat{Z}$ be the computed 2^n-point FFT of $z \in \mathbb{C}^{2^n}$, and let Z be the exact value. Then*

$$\|Z - \hat{Z}\|_\infty \leqslant 2^n \cdot (2n+1) \cdot \mu \cdot \|z\|_\infty + \mathcal{O}(u^2).$$

From Eq. (7) of [23] one can show that if the relative error of complex multiplication is bounded by $\rho_\times$ then

$$\|Z - \hat{Z}\|_\infty \leqslant 2^n \cdot (\rho_\times + 2u) \cdot n \cdot \|z\|_\infty + \mathcal{O}(u^2).$$

If $\rho_\times = 2u$ (which is the case if complex multiplication is performed using (11), see Section 2.2), this gives essentially the same bound as Theorem 6.

For several recent reasons, it is worth reexamining the problem of finding tight error bounds on the computation of Fast Fourier Transforms in floating-point arithmetic:

- first, before Percival's paper [15], the relative error bound $\sqrt{5} \cdot u$ on the naive complex multiplication algorithm was not known;
- the FMA (*Fused Multiply-Add*) instruction, which evaluates expressions of the form $ab+c$ with one rounding only, is now widespread: it is specified by the IEEE-754 Standard on Floating-Point Arithmetic [10], and available in all recent general purpose processors of commercial significance. It was recently shown [11] that with an FMA, the relative error bound on the complex multiplication becomes $2u$;
- when implementing the multiplications by roots of unity that occur in the FFT, we approximate "exact" multiplications $\omega \cdot z$ by "rounded" multiplications of $\hat{\omega}$ by z, where $\hat{\omega}$ approximates ω, and z is a number whose real and imaginary parts are floating-point numbers. Most analyses use a global error bound on these approximations: this simplifies the study and makes it possible to express the error bound on the FFT as a simple expression. However, the approximation error of the multiplication depends much on ω (it is even rather frequently null: values $\omega = \pm 1, \pm i$ are not so rare), and since the values of ω are known in advance, one can try to obtain better, if less simple and elegant, error bounds.

The sequel of the paper is organized as follows. We first bound the relative error that can occur when multiplying a complex number by a root of unity in

floating-point arithmetic. These results are used to analyze Step k of the FFT algorithm in Section 3.1 (similarly to what Percival did in [15], but slightly more accurately and with more details). Then, we derive a relative error bound (for the 2-norm) of the FFT algorithm in Section 3.2. To be able to use (1) or (2) to bound the error of the floating-point operations, we need to bound, as tightly as possible, the largest value that a variable can take at Step k of the FFT algorithm. This is done in Section 4.

Finally, Section 5 gives an iterative algorithm that provides an error bound in terms of the $\|\cdot\|_\infty^\perp$ norm, which is sometimes tighter than what one could deduce from the known bounds in terms of the 2-norm and (7).

2 Relative error of the multiplication of a complex number by a root of unity

In this section, we try to bound as tightly as possible the error resulting from approximating ωz by an adequately chosen multiplication algorithm Alg applied to $\hat{\omega}$ and z. Here, ω is a root of unity, z is a complex number whose real and imaginary parts are floating-point numbers, and $\hat{\omega} = \mathrm{RN}(\Re(\omega)) + i \cdot \mathrm{RN}(\Im(\omega))$. Let us first consider the error resulting from the approximation of ω by $\hat{\omega}$.

2.1 Relative error of the approximation of a complex number in floating-point arithmetic

2.1.1 Case of an arbitrary complex number

Let $z \in \mathbb{C}$, $z = x + iy$, with $x, y \in \mathbb{R}$. The complex number z is approximated by $\hat{z} = \hat{x} + i\hat{y}$, with $\hat{x} = \mathrm{RN}(x)$ and $\hat{y} = \mathrm{RN}(y)$. We will denote $\hat{z} = \mathrm{RN}(z)$. The *componentwise* relative error committed when approximating z by $\hat{z}$ is

$$\max\left\{\left|\frac{x-\hat{x}}{x}\right|, \left|\frac{y-\hat{y}}{y}\right|\right\} \leqslant \frac{u}{1+u} < u.$$

The *normwise* error committed when approximating z by $\hat{z}$ is

$$\left|\frac{z-\hat{z}}{z}\right| \leqslant \frac{u}{1+u} < u. \tag{8}$$

2.1.2 Case of a root of unity

The bound (8) can be significantly improved when the absolute value of z is 1 (which is the case when z is a root of unity). In such a case, since $|x|$ and $|y|$ are less than or equal to 1, we have $|x - \hat{x}| \leqslant u/2$ and $|y - \hat{y}| \leqslant u/2$, so that

$$\left|\frac{z-\hat{z}}{z}\right|^2 = \frac{(x-\hat{x})^2 + (y-\hat{y})^2}{1} \leqslant \frac{u^2}{2},$$

so that

$$\left|\frac{z-\hat{z}}{z}\right| \leqslant \frac{u}{\sqrt{2}}. \tag{9}$$

Of course, one should not forget that the 2^{nd} and 4^{th} roots of 1 are exactly represented in floating-point arithmetic.

For a given floating-point format (i.e., a given precision p) and a given N, one can also compute in advance the largest relative error attained when approximating an N^{th} root of unity. In several cases (especially when N is not too large), this leads to bounds significantly smaller than (9). For instance, if $N = 128$, in single-precision/binary32 arithmetic (i.e., $p = 24$), the largest relative error is $0.500 \cdot u$, which is significantly smaller than $(\sqrt{2}/2) \cdot u \approx 0.707 \cdot u$. Examples are given in Table 1. Table 1 was calculated using very large precision, with careful control of the error, in Maple.

Table 1: Largest relative error attained when approximating an N^{th} root of unity in precision-p, binary, floating-point arithmetic. All errors in this table have been rounded up.

N	2 or 4	8	16	32	128	2048	32768
$p = 24$	0	$0.288 \cdot u$	$0.487 \cdot u$	$0.500 \cdot u$	$0.500 \cdot u$	$0.633 \cdot u$	$0.707 \cdot u$
$p = 53$	0	$0.616 \cdot u$	$0.616 \cdot u$	$0.616 \cdot u$	$0.616 \cdot u$	$0.641 \cdot u$	$0.697 \cdot u$
$p = 113$	0	$0.692 \cdot u$	$0.692 \cdot u$	$0.692 \cdot u$	$0.692 \cdot u$	$0.692 \cdot u$	$0.692 \cdot u$

In the following, Δ_z is a bound on the relative error committed when approximating z by $\hat{z} = \mathrm{RN}(\Re(z)) + i\mathrm{RN}(\Im(z))$.

2.2 Multiplication of a complex number by a root of unity

Normwise relative error bounds on complex multiplication have been derived by Brent et al. [2] for the "naive" algorithm (10), and by Jeannerod et al. [11] assuming an FMA instruction is available, i.e., using (11) or one of the algorithms derived from (11) using symmetries. Let $a + ib$ be a complex number, with a, b floating-point numbers. We wish to evaluate $z = x + iy = (a + ib) \cdot \omega$, where $\omega = c + is$ is a root of unity. Since, in general, c and s are not floating-point numbers, they are approximated by $\hat{c} = \mathrm{RN}(c)$ and $\hat{s} = \mathrm{RN}(s)$. The normwise relative error due to that approximation has been studied in Section 2.1.

Let $\hat{z} = \hat{x} + i\hat{y}$ be the computed product. We consider two cases.

- **Naive multiplication**

$$\begin{cases} \hat{x} &= \mathrm{RN}(\mathrm{RN}(a\hat{c}) - \mathrm{RN}(b\hat{s})), \\ \hat{y} &= \mathrm{RN}(\mathrm{RN}(a\hat{s}) + \mathrm{RN}(b\hat{c})). \end{cases} \tag{10}$$

- **Multiplication with an FMA**

$$\left\{ \begin{array}{rcl} \hat{x} & = & \mathrm{RN}(a\hat{c} - \mathrm{RN}(b\hat{s})), \\ \hat{y} & = & \mathrm{RN}(a\hat{s} + \mathrm{RN}(b\hat{c})). \end{array} \right. \tag{11}$$

Define $z^* = (a+ib)\hat{\omega}$, with $\hat{\omega} = \hat{c} + i\hat{s}$. If the naive multiplication is used then $|z - z^*| \leqslant u\sqrt{5} \cdot |z^*|$ (see [2]), and if multiplication with FMA is used then $|z - z^*| \leqslant 2u \cdot |z^*|$ (see [11]). Note that (11) is interesting only when we want to minimize *normwise* errors (which is the case here). If one wishes to minimize componentwise relative errors (the componentwise relative error is the maximum of the relative error on the real part of the product and the relative error on the imaginary part), there are better solutions (especially if an FMA instruction is available), based on Kahan's algorithm for evaluating expressions of the form $ac - bd$ [11, 12]. In the following, let us define

$$\rho_\times = \left\{ \begin{array}{ll} u\sqrt{5} & \text{if (10) is used,} \\ 2u & \text{if (11) is used.} \end{array} \right.$$

We have

$$|z - \hat{z}| \leqslant |z - z^*| + |z^* - \hat{z}|.$$

Let us first bound $|z - z^*|$. We have,

$$\begin{array}{rcl} |z - z^*| & = & |(a+ib) \cdot [(\hat{c} + i\hat{s}) - (c+is)]| \\ & = & |a+ib| \cdot \left| \frac{(\hat{c}+i\hat{s})-(c+is)}{c+is} \right| \\ & \leqslant & |z| \cdot \Delta_\omega. \end{array}$$

(We remind the reader that $\Delta_\omega = |\omega - \hat{\omega}|/|\omega| = |\omega - \hat{\omega}|$.) We also have $|z^* - \hat{z}| \leqslant \rho_\times \cdot |z^*|$, so that

$$\left| \frac{z^* - \hat{z}}{z} \right| = \left| \frac{z^* - \hat{z}}{z^*} \right| \cdot \left| \frac{z^*}{z} \right| = \left| \frac{z^* - \hat{z}}{z^*} \right| \cdot \left| \frac{\hat{\omega}}{\omega} \right| \leqslant \rho_\times \cdot \left| \frac{\hat{\omega}}{\omega} \right|.$$

Hence $\hat{z}$ approximates z with a relative error bounded by

$$\Delta_\omega + \rho_\times \cdot \left| \frac{\hat{\omega}}{\omega} \right| \leqslant \Delta_\omega + \rho_\times \cdot (1 + \Delta_\omega). \tag{12}$$

The bound (12) is rather tight. For instance, in double precision arithmetic (i.e., $p = 53$), if we use multiplication with an FMA (i.e., (11)), with $\omega = \exp(-7133i\pi/2^{20})$ and $z = 5495961505303309/2^{52} + i \cdot 4506137113525543/2^{42}$, the relative error is $2.455u$, whereas the bound given by (12) is $2.460u$.

3 Bound on the relative error of the FFT for the 2-norm

Let us now use the error bounds on the multiplication by a root of unity, given in Section 2, to bound the relative error of the FFT for the 2-norm. For that purpose, we will first consider Step k of the FFT algorithm.

3.1 Step k of the FFT

This section uses a reasoning presented by Percival [15]. We present it with more detail, and extract more information from it by not uniformly bounding the relative errors of the complex multiplications by roots of unity. At Step k ($k = 1, \ldots, n$) of the $N = 2^n$-point FFT, the 2^k-th roots of unity are used. Define $\Delta_k^{\max}$ as the largest value of $\Delta_\omega = |\omega - \hat{\omega}|$, where ω is a 2^k-th root of unity. Notice that when $k = 1$ or 2, the 2^k-th root of unity are exactly represented, and multiplication by them is errorless. From this and (12), the complex products performed at Step k have a relative error bounded by

$$g_k := \begin{cases} 0 \text{ if } k = 1, 2, \\ \Delta_k^{\max} + \rho_\times \cdot (1 + \Delta_k^{\max}) \text{ otherwise.} \end{cases}$$

For instance, for $k \geqslant 3$,

$$g_k \leqslant \frac{\sqrt{2}}{2} u + \rho_\times \cdot \left(1 + \frac{\sqrt{2}}{2} u \right). \tag{13}$$

Equation (13) can be interesting for obtaining a bound on the final error of the FFT as a closed form, however, one will get tighter bounds by individually computing the terms g_k. Now, Step k of the FFT of order N can be viewed as $N/2$ parallel combinations of the form

$$\begin{pmatrix} z_0 \\ z_1 \end{pmatrix} \to \begin{pmatrix} z_0 + \omega z_1 \\ z_0 - \omega z_1 \end{pmatrix},$$

Let us denote $\widehat{\omega z_1}$ = computed value of ωz_1 (with relative error bounded by g_k).

We have

$$\begin{array}{l} |\widehat{\omega z_1} - \omega z_1| \leqslant |z_1| \cdot g_k, \\ |\mathrm{RN}(z_0 - \widehat{\omega z_1}) - (z_0 - \widehat{\omega z_1})| \leqslant u \cdot |z_0 - \widehat{\omega z_1}|, \\ |\mathrm{RN}(z_0 + \widehat{\omega z_1}) - (z_0 + \widehat{\omega z_1})| \leqslant u \cdot |z_0 + \widehat{\omega z_1}|. \end{array}$$

From which we deduce

$$\begin{array}{rl} |\mathrm{RN}(z_0 + \widehat{\omega z_1}) - (z_0 + \omega z_1)| & \leqslant |\mathrm{RN}(z_0 + \widehat{\omega z_1}) - (z_0 + \widehat{\omega z_1})| + |\widehat{\omega z_1} - \omega z_1| \\ & \leqslant u \cdot |z_0 + \widehat{\omega z_1}| + |z_1| \cdot g_k \\ & \leqslant u \cdot (|z_0 + \omega z_1| + |\widehat{\omega z_1} - \omega z_1|) + |z_1| \cdot g_k, \end{array}$$

which implies

$$|\mathrm{RN}(z_0 + \widehat{\omega z_1}) - (z_0 + \omega z_1)| \leqslant |z_0 + \omega z_1| \cdot u + |z_1| \cdot (g_k + u g_k). \tag{14}$$

Similarly, we have

$$|\mathrm{RN}(z_0 - \widehat{\omega z_1}) - (z_0 - \omega z_1)| \leqslant |z_0 - \omega z_1| \cdot u + |z_1| \cdot (g_k + u g_k). \tag{15}$$

Now, let us notice that

$$|z_0 - \omega z_1|^2 + |z_0 + \omega z_1|^2 = 2z_0\overline{z_0} + 2(\omega z_1)\overline{(\omega z_1)} = 2(|z_0|^2 + |z_1|^2), \qquad (16)$$

a consequence of which is

$$|z_0|^2,\, |z_1|^2 \leqslant \frac{1}{2}\left(|z_0 - \omega z_1|^2 + |z_0 + \omega z_1|^2\right). \qquad (17)$$

By combining (14), (15), and (17), we obtain

$$\begin{aligned}
&|\mathrm{RN}(z_0 + \widehat{\omega z_1}) - (z_0 + \omega z_1)|^2 + |\mathrm{RN}(z_0 - \widehat{\omega z_1}) - (z_0 - \omega z_1)|^2 \\
&\quad \leqslant |z_0 + \omega z_1|^2 \cdot u^2 + \frac{1}{2}\left(|z_0 - \omega z_1|^2 + |z_0 + \omega z_1|^2\right) \cdot (g_k + u g_k)^2 \\
&\quad + |z_0 - \omega z_1|^2 \cdot u^2 + \frac{1}{2}\left(|z_0 - \omega z_1|^2 + |z_0 + \omega z_1|^2\right) \cdot (g_k + u g_k)^2 \\
&\qquad\qquad + 2u|z_1|(g_k + u g_k)\left(|z_0 + \omega z_1| + |z_0 - \omega z_1|\right).
\end{aligned}$$

We will now use the following lemma.

Lemma 1. *For all $t, v \in \mathbb{C}$, we have*

$$(|t + v| + |t - v|)\max(|t|, |v|) \leqslant (|t + v|^2 + |t - v|^2).$$

Proof. Just choose $a = t + v$, $b = t - v$, and notice that

$$\begin{aligned}
&|a|^2 + |b|^2 - (|a| + |b|) \cdot \max\left(\tfrac{|a+b|}{2}, \tfrac{|a-b|}{2}\right) \\
&\geqslant |a|^2 + |b|^2 - (|a| + |b|) \cdot \left(\tfrac{|a|+|b|}{2}\right) \\
&= \tfrac{1}{2}\left(|a| - |b|\right)^2 \geqslant 0.
\end{aligned}$$

□

All this gives

Lemma 2 (Adapted from [15])**.**

$$|\mathrm{RN}(z_0 + \widehat{\omega z_1}) - (z_0 + \omega z_1)|^2 + |\mathrm{RN}(z_0 - \widehat{\omega z_1}) - (z_0 - \omega z_1)|^2 \leqslant \left(|z_0 + \omega z_1|^2 + |z_0 - \omega z_1|^2\right)\Omega_k^2,$$

where

$$\Omega_k := u + g_k(1 + u).$$

3.2 Application to the error of the FFT

In the following, Z is the $N = 2^n$-point (exact) Fourier transform of z, and t_k is the transformation performed at Step k of the FFT algorithm. We denote $z^{(0)} = z$, $z^{(1)} = t_1(z^{(0)})$, $z^{(2)} = t_2(z^{(1)})$, …, $z^{(n)} = t_n(z^{(n-1)}) = Z$ the intermediate exact values of the FFT algorithm, and $\widehat{z^{(1)}}$, $\widehat{z^{(2)}}$, …, $\widehat{z^{(n)}} = \widehat{Z}$ the corresponding computed values. We have $\widehat{z^{(0)}} = z^{(0)}$.

From (16) we easily find that for any y and k, $\|t_k(y)\|_2 = \sqrt{2} \cdot \|y\|_2$.
From Lemma 2, we have

$$\|\widehat{z^{(k)}} - t_k(\widehat{z^{(k-1)}})\|_2 \leqslant \Omega_k \cdot \|t_k(\widehat{z^{(k-1)}})\|_2. \tag{18}$$

Let us show by induction on k the following property

$$\|\widehat{z^{(k)}} - z^{(k)}\|_2 \leqslant \|z^{(k)}\|_2 \cdot \left[\prod_{i=1}^{k}(1+\Omega_i) - 1\right]. \tag{19}$$

For $k = 1$, it is an almost immediate consequence of (18) and the fact that $t_1(\widehat{z^{(0)}}) = t_1(z^{(0)}) = z^{(1)}$. Now, let us assume (19) is true for some k. We have

$$\begin{array}{rcl}
\|\widehat{z^{(k+1)}} - z^{(k+1)}\|_2 & \leqslant & \|\widehat{z^{(k+1)}} - t_{k+1}(\widehat{z^{(k)}})\|_2 + \|t_{k+1}(\widehat{z^{(k)}}) - z^{(k+1)}\|_2 \\
& \leqslant & \|t_{k+1}(\widehat{z^{(k)}})\|_2 \cdot \Omega_{k+1} + \|t_{k+1}(\widehat{z^{(k)}} - z^{(k)})\|_2 \\
& \leqslant & \sqrt{2} \cdot \|\widehat{z^{(k)}}\|_2 \cdot \Omega_{k+1} + \sqrt{2} \cdot \|\widehat{z^{(k)}} - z^{(k)}\|_2 \\
& \leqslant & \sqrt{2} \cdot \Omega_{k+1} \cdot \|z^{(k)}\|_2 \cdot \prod_{i=1}^{k}(1+\Omega_i) + \sqrt{2} \cdot \|z^{(k)}\|_2 \cdot \left[\prod_{i=1}^{k}(1+\Omega_i) - 1\right] \\
& \leqslant & \|z^{(k+1)}\|_2 \cdot \left[\prod_{i=1}^{k+1}(1+\Omega_i) - 1\right].
\end{array}$$

Q.E.D.
This gives

Theorem 7. *Assume radix-2, precision-p arithmetic, with rounding unit $u = 2^{-p}$. Let $\hat{Z}$ be the computed 2^n-point FFT of $z \in \mathbb{C}^{2^n}$, and let Z be the exact value. Then*

$$\|\hat{Z} - Z\|_2 \leqslant \|Z\|_2 \cdot \left(\prod_{i=1}^{n}(1+\Omega_i) - 1\right),$$

with

$$\begin{array}{lcl}
\Omega_k & = & u + g_k(1+u), \\
g_k & = & \left\{ \begin{array}{l} 0 \text{ if } k = 1, 2, \\ \Delta_k^{max} + \rho_\times \cdot (1 + \Delta_k^{max}) \text{ otherwise,} \end{array} \right. \\
\Delta_k^{max} & = & \max_{\{\omega \ 2^k\text{-th root of } 1\}} \Delta_\omega, \\
\Delta_\omega & = & |\hat{\omega} - \omega|, \\
\rho_\times & = & \left\{ \begin{array}{ll} u\sqrt{5} & \text{if (10) is used (naive multiplication),} \\ 2u & \text{if (11) is used (multiplication with FMA).} \end{array} \right.
\end{array}$$

Theorem 7 can be directly used to obtain error bounds, with a preliminary calculation of the terms g_k. As said above, one can get a simpler yet looser bound by noticing that $g_1 = g_2 = 0$ and bounding all other terms g_k by

$$g = \frac{\sqrt{2}}{2}u + \rho_\times \cdot \left(1 + \frac{\sqrt{2}}{2}u\right).$$

This gives the following result, which is essentially the same as Percival's result [15] (our bound is slightly better because we use $g_1 = g_2 = 0$, and when an FMA instruction is available, we know that $\rho_\times = 2u$).

Theorem 8 (Close to Percival's bound [15])**.**

$$\|\hat{Z} - Z\|_2 \leqslant \|Z\|_2 \cdot \left[(1+u)^n (1+g)^{n-2} - 1\right].$$

Of course, this also gives a bound on $\|\hat{Z} - Z\|_\infty$. Tables 2 and 3 compare the various obtained bounds in the case of a 256-point and a 65536-point FFT, respectively.

Table 2: Comparison of the bounds on $\|\hat{Z} - Z\|_2 / \|Z\|_2$ given by Theorems 1, 2, 3 (with $\mu = u\sqrt{2}/2$, the smallest value that always holds), 4 (with the same value of μ), 5, 7 and 8 for a 2^8-point FFT.

	$p = 24$	$p = 53$	$p = 113$
Theorem 1	$1085.44 \cdot u + \mathcal{O}(u^2)$	$1085.44 \cdot u + \mathcal{O}(u^2)$	$1085.44 \cdot u + \mathcal{O}(u^2)$
Theorem 2	$39.31 \cdot u + \mathcal{O}(u^2)$	$39.31 \cdot u + \mathcal{O}(u^2)$	$39.31 \cdot u + \mathcal{O}(u^2)$
Theorem 3 with $\mu = u\sqrt{2}/2$ (does not assume FMA)	$50.92 \cdot u$	$50.92 \cdot u$	$50.92 \cdot u$
Theorem 4 with $\mu = u\sqrt{2}/2$ (does not assume FMA)	$31.55 \cdot u$	$31.55 \cdot u$	$31.55 \cdot u$
Theorem 5 with $\rho_\times = 2u$ (FMA)	$28.24 \cdot u + \mathcal{O}(u^2)$	$28.24 \cdot u + \mathcal{O}(u^2)$	$28.24 \cdot u + \mathcal{O}(u^2)$
Theorem 8 with $\rho_\times = 2u$ (FMA)	$24.25 \cdot u$	$24.25 \cdot u$	$24.25 \cdot u$
Theorem 8 with $\rho_\times = \sqrt{5}u$ (no FMA)	$25.66 \cdot u$	$25.66 \cdot u$	$25.66 \cdot u$
Theorem 7 with calculation of the g_ks with $\rho_\times = 2u$ (FMA)	$22.78 \cdot u$	$23.71 \cdot u$	$24.16 \cdot u$
Theorem 7 with calculation of the g_ks with $\rho_\times = \sqrt{5}u$ (no FMA)	$24.19 \cdot u$	$25.11 \cdot u$	$25.57 \cdot u$

Table 3: Comparison of the bounds on $\|\hat{Z} - Z\|_2/\|Z\|_2$ given by Theorems 1, 2, 3 (with $\mu = u\sqrt{2}/2$, the smallest value that always holds), 4 (with the same value of μ), 5, 7 and 8 for a 2^{16}-point FFT.

	$p = 24$	$p = 53$	$p = 113$
Theorem 1	$34734 \cdot u + \mathcal{O}(u^2)$	$34734 \cdot u + \mathcal{O}(u^2)$	$34734 \cdot u + \mathcal{O}(u^2)$
Theorem 2	$82.62 \cdot u + \mathcal{O}(u^2)$	$82.62 \cdot u + \mathcal{O}(u^2)$	$82.62 \cdot u + \mathcal{O}(u^2)$
Theorem 3 with $\mu = u\sqrt{2}/2$ (does not assume FMA)	$101.83 \cdot u$	$101.83 \cdot u$	$101.83 \cdot u$
Theorem 4 with $\mu = u\sqrt{2}/2$ (does not assume FMA)	$63.10 \cdot u$	$63.10 \cdot u$	$63.10 \cdot u$
Theorem 5 with $\rho_\times = 2u$ (FMA)	$57.89 \cdot u + \mathcal{O}(u^2)$	$57.89 \cdot u + \mathcal{O}(u^2)$	$57.89 \cdot u + \mathcal{O}(u^2)$
Theorem 8 with $\rho_\times = 2u$ (FMA)	$53.90 \cdot u$	$53.90 \cdot u$	$53.90 \cdot u$
Theorem 8 with $\rho_\times = \sqrt{5}u$ (no FMA)	$57.21 \cdot u$	$57.21 \cdot u$	$57.21 \cdot u$
Theorem 7 with calculation of the g_ks with $\rho_\times = 2u$ (FMA)	$52.14 \cdot u$	$53.03 \cdot u$	$53.69 \cdot u$
Theorem 7 with calculation of the g_ks with $\rho_\times = \sqrt{5}u$ (no FMA)	$55.45 \cdot u$	$56.33 \cdot u$	$57.00 \cdot u$

4 Largest values that can occur when computing the FFT of a vector

Assuming that the chosen rounding mode is round-to-nearest, when we perform an arithmetic operation $a \top b$, where a and b are floating-point numbers, what is actually computed is $\mathrm{RN}(a \top b)$. Using (1) and the fact that RN is an increasing function, from a bound M on $|a \top b|$, we can deduce a bound on the rounding error committed when performing that operation:

$$|\mathrm{RN}(a \top b) - (a \top b)| \leqslant \frac{1}{2}\mathrm{ulp}(M) \leqslant u \cdot M.$$

We aim at using that property to bound the rounding errors occurring when performing FFTs. This requires bounding all intermediate values that appear

in the calculation. This is what we deal with in this section.

4.1 A simple bound

Assume that $Z = (Z_0, Z_1, \ldots, Z_{N-1})$ is the order-$N = 2^n$ DFT of $z = (z_0, z_1, \ldots, z_{N-1})$. We have $\|Z\|_2 = \sqrt{N} \cdot \|z\|_2$, therefore

$$\|Z\|_\infty \leqslant \sqrt{N} \cdot \|z\|_2 \leqslant N \cdot \|z\|_\infty.$$

Note that $\|Z\|_\infty$ can be equal to $N \cdot \|z\|_\infty$: just consider $z = (1, 1, 1, \ldots, 1)$, for which we have $Z = (N, 0, 0, 0, \ldots, 0)$. Hence,

$$\|Z\|_\infty^\perp \leqslant \sqrt{2} \cdot N \cdot \|z\|_\infty^\perp. \tag{20}$$

In other words, if the real and imaginary parts of the terms z_i are of absolute value less than b, then the real and imaginary parts of the terms Z_i will be of absolute value less than $B = Nb\sqrt{2}$.

Since the first k steps of an order-N FFT can be viewed as $N/2^k$ independent FFTs of order 2^k, we deduce that the intermediate values computed at Step k have absolute values of the real and imaginary parts less than $2^k b\sqrt{2}$.

As we are going to see, with significantly more involved reasoning, one can replace in (20) the constant $\sqrt{2} \approx 1.414$ by $4/\pi \approx 1.273$.

4.2 Some improvement

Let us now try to obtain a tighter bound.

Let $Z = (Z_0, Z_1, \ldots, Z_{N-1})$ be the DFT of $z = (z_0, z_1, \ldots, z_{N-1})$. We have

$$Z_j = z_0 + \omega^j z_1 + \omega^{2j} z_2 + \cdots + \omega^{(N-1)j} z_{N-1},$$

with $\omega = \exp(-2i\pi/N)$.

Consider a vector z that maximizes the absolute value of the real part of Z_j under the constraints $\|z\|_\infty^\perp \leqslant b$ (the reasoning is straightforwardly similar if we want to maximize the imaginary part). Without loss of generality we assume that the real part of Z_j is nonnegative. Since all the z_k can be chosen independently, we maximize the real part of Z_j by maximizing separately all the terms $\Re(\omega^{jk} z_k) = \cos(2jk\pi/N)\Re(z_k) + \sin(2jk\pi/N)\Im(z_k)$, i.e., by choosing $\Re(z_k) = b \cdot \mathrm{sign}(\cos(2jk\pi/N))$, and $\Im(z_k) = b \cdot \mathrm{sign}(\sin(2jk\pi/N))$, so that the maximum value of $|\Re(Z_j)|$ is $b \cdot S_j$, where

$$S_j = \sum_{k=0}^{N-1} \left(|\cos(2jk\pi/N)| + |\sin(2jk\pi/N)| \right).$$

Hence the bound we are looking for is $K_N = \max_{j=0\ldots N-1} S_j$. Table 4 gives the values of K_N and K_N/N for the first powers of 2.

Let us show that we always have $K_N \leqslant \frac{4}{\pi} \cdot N$ and $4/\pi$ is optimal.

Table 4: First values of K_N and K_N/N. If the terms z_k have the absolute values of their real and imaginary parts less than or equal to b, then the terms Z_k have the absolute values of their real and imaginary parts less than or equal to bK_N, and that bound is attained.

N	K_N	K_N/N
8	$4+4\sqrt{2}=9.6568\cdots$	$1.20710678\cdots$
16	$20.10935\cdots$	$1.2568348730\cdots$
32	$40.6126815\cdots$	$1.269146298451\cdots$
64	$81.421870499\cdots$	$1.272216726561699\cdots$
128	$162.9419354883\cdots$	$1.2729838710026031\cdots$
256	$325.932960826184\cdots$	$1.27317562822728390\cdots$
512	$651.890465652999852\cdots$	$1.273223565728515337\cdots$
1024	$1303.7932031908053959\cdots$	$1.2732355499910208944\cdots$
2048	$2607.592542309575146472\cdots$	$1.273238546049597239488\cdots$
4096	$5215.1881525813276653152\cdots$	$1.2732392950638006995398\cdots$
8192	$10430.37783914351841639453\cdots$	$1.2732394823173240254387 8\cdots$
∞		$4/\pi=1.2732395447351626861\cdots$

Let $m\in\mathbb{N}\setminus\{0\}$, we have

$$\frac{K_m}{m}=\frac{1}{m}\max_{j=0,\ldots,m-1}\left\{\sum_{k=0}^{m-1}\left(|\cos(2jk\pi/m)|+|\sin(2jk\pi/m)|\right)\right\}.$$

We now define, for all $m,j\in\mathbb{N}$, $m\neq 0$,

$$\gamma_m^j=\frac{1}{m}\sum_{k=0}^{m-1}\left(|\cos(2jk\pi/m)|+|\sin(2jk\pi/m)|\right).$$

For all $x\in\mathbb{R}$, let $f(x):=|\cos(x)|+|\sin(x)|$. The function f is $(\pi/2)$-periodic. Since $1\leqslant f(x)\leqslant\sqrt{2}$ for all x, we have, for all $m,j\in\mathbb{N}$, $m\neq 0$,

$$1\leqslant\gamma_m^j\leqslant\sqrt{2},\ \text{hence}\ 1\leqslant\frac{K_m}{m}\leqslant\sqrt{2}. \tag{21}$$

Lemma 3. *For all $m,j\in\mathbb{N}\setminus\{0\}$ such that $(j,m)=1$, we have $\gamma_m^j=\gamma_m^1$.*

Proof. Since j and m are relatively prime, we have

$$\{k \bmod m, k=0,\ldots,m-1\}=\{jk \bmod m, k=0,\ldots,m-1\}.$$

Hence,

$$\gamma_m^j=\frac{1}{m}\sum_{x\in\left\{\frac{jk}{m},0\leqslant k\leqslant m-1\right\}}f(2\pi x)=\frac{1}{m}\sum_{x\in\left\{\frac{k}{m},0\leqslant k\leqslant m-1\right\}}f(2\pi x)=\gamma_m^1.$$

□

Lemma 4. *For all $m, j \in \mathbb{N} \setminus \{0\}$, we have $\gamma_m^j = \gamma_{m/(j,m)}^{j/(j,m)} = \gamma_{m/(j,m)}^1$.*

Proof. Let $d = (j, m)$, $j = dj'$, $m = dm'$ with $(j', m') = 1$, we have for all $k, \ell \in \mathbb{N}$,

$$f\left(\frac{2j\pi}{m}(k + \ell m')\right) = f\left(\frac{2j'k\pi}{m'} + 2j'\ell\pi\right) = f\left(\frac{2j'k\pi}{m'}\right),$$

since f is $(\pi/2)$-periodic. It follows that

$$\gamma_m^j = \frac{1}{m}\sum_{k=0}^{m-1} f\left(\frac{2jk\pi}{m}\right) = \frac{1}{m}\sum_{\ell=0}^{d-1}\sum_{k=0}^{m'-1} f\left(\frac{2j(k+\ell m')\pi}{m}\right) = \frac{d}{m}\sum_{k=0}^{m'-1} f\left(\frac{2j'k\pi}{m'}\right) = \gamma_{m'}^{j'}.$$

The second equality is a consequence of Lemma 3. □

Lemma 5. *The sequence $\left(\gamma_{4m}^1\right)_{m\in\mathbb{N}}$ is increasing and tends to $4/\pi$ as m tends to $+\infty$.*

Proof. We have for all $k, \ell \in \mathbb{N}$,

$$f\left(\frac{2\pi}{4m}(k + \ell m)\right) = f\left(\frac{k\pi}{2m} + \frac{\ell\pi}{2}\right) = f\left(\frac{k\pi}{2m}\right),$$

since f is $(\pi/2)$-periodic. Therefore, for all $m \in \mathbb{N} \setminus \{0\}$, we have

$$\gamma_{4m}^1 = \frac{1}{4m}\sum_{k=0}^{4m-1} f\left(\frac{2k\pi}{4m}\right) = \frac{1}{4m}\sum_{\ell=0}^{3}\sum_{k=0}^{m-1} f\left(\frac{(k+\ell m)\pi}{2m}\right) = \frac{1}{m}\sum_{k=0}^{m-1} f\left(\frac{k\pi}{2m}\right).$$

Hence, $\left(\frac{\pi}{2}\gamma_{4m}^1\right)_{m\in\mathbb{N}}$ is a sequence of Riemann sums which tends to $\int_0^{\pi/2} f(x)\,\mathrm{d}x = 2$ as m tends to ∞. Moreover, since f is concave over $[0, \pi/2]$, this sequence of Riemann sums is increasing [1, Thm 3A]. □

Corollary 1. *For all $n \in \mathbb{N}$, we have*

$$\frac{1}{2^n}K_{2^n} \leqslant \frac{4}{\pi} \text{ and } \lim_{n\to\infty}\frac{1}{2^n}K_{2^n} = \frac{4}{\pi}.$$

Proof. We can assume $n \geqslant 1$ since we know that $K_1 = 1$. It follows from Lemma 4 that there exists $n_0 \geqslant 1$ such that $\max_{j=1,\dots,2^n}\gamma_{2^n}^j = \gamma_{2^{n_0}}^1$. Thus, $K_{2^n}/2^n = \max(\gamma_{2^n}^0, \gamma_{2^{n_0}}^1) = \max(1, \gamma_{2^{n_0}}^1) = \gamma_{2^{n_0}}^1$ thanks to Equation (21). From Lemma 5 and $\gamma_2^1 = 1$, we get $K_{2^n}/2^n = \gamma_{2^{n_0}}^1 \leqslant 4/\pi$. Moreover, by definition of K_{2^n}, we have $\gamma_{2^n}^1 \leqslant K_{2^n}/2^n$. The last two inequalities and Lemma 5 imply $\lim_{n\mapsto\infty} K_{2^n}/2^n = 4/\pi$. □

All this gives the following result.

Theorem 9. *Let $Z = (Z_0, Z_1, \ldots, Z_{N-1})$ be the order-$N = 2^n$ Discrete Fourier Transform of $z = (z_0, z_1, \ldots, z_{N-1})$. We have,*

$$\|Z\|_\infty^\perp \leqslant \frac{4}{\pi} \cdot N \cdot \|z\|_\infty^\perp. \tag{22}$$

and the constant $4/\pi$ in (22) is optimal.

Theorem 9 implies that if the real and imaginary parts of the terms z_i are of absolute value less than or equal to b, then the real and imaginary parts of the terms Z_i are of absolute value less than or equal to

$$\frac{4}{\pi} \cdot N \cdot b, \tag{23}$$

An important consequence, that will be useful in the sequel of this paper is that since the first k steps of an order-N FFT can be viewed as $N/2^k$ independent FFTs of order 2^k, the intermediate values computed at Step k have absolute values of the real and imaginary parts less than $2^{k+2}b/\pi$.

5 Calculation of an error bound for the infinity norm

In the following, we wish to find a bound for $\|Z - \hat{Z}\|_\infty^\perp$, given that $\|z\|_\infty^\perp \leqslant 1$. Note that if the constraint on z becomes $\|z\|_\infty^\perp \leqslant 2^m$, it will suffice to multiply the bound on $\|Z-\hat{Z}\|_\infty^\perp$ by 2^m. Obtaining such bounds is important for implementing Schönhage and Strassen's algorithm for multiplying big integers of large polynomials [20, 13]. One can get such bounds by using Theorem 7 and (7). One might also want to use Henrici's Theorem (Theorem 6): assuming an FMA is used (so that complex multiplication has relative error $2u$), Theorem 6 gives a bound

$$\|\hat{Z} - Z\|_\infty^\perp \leqslant 2^{n+1}(2n+1) \cdot \sqrt{2} \cdot u \cdot \|z\|_\infty^\perp + \mathcal{O}(u^2).$$

Note, however, the "$\mathcal{O}(u^2)$" that does not allow to get sure bounds.

Table 5 compares bounds on $\|\hat{Z} - Z\|_2/\|Z\|_2$ and $\|Z - \hat{Z}\|_\infty^\perp/\|z\|_\infty^\perp$ deduced from Theorem 7 (assuming an FMA instruction is available, and $p = 53$–i.e., double-precision arithmetic) and (7) with bounds deduced from Henrici's theorem.

Before going further, let us give a "bad" case for that norm, i.e., a case for which we obtain a large error.

5.1 A bad case for the infinity norm

It is interesting to build "bad" cases: they help us to know if it is worth trying to improve the error bounds.

We build a bad case as follows. We are interested in having an error as large as possible on the first term of the Fourier transform. Let us quickly explain how

this can be done. We assume that $z_0 \approx z_1 \approx z_2 \approx \cdots \approx z_{N-1} \approx 1$. Denoting $v_i = z_{\text{reverse}(i)}$, where reverse$(i)$ is the integer whose binary representation is the mirror image of the binary representation of i, we are interested in the calculation of the 1st component of the Fourier Transform. The exact result is $z_0 + z_1 + \cdots + z_{N-1}$, and the computed result is

$$\mathrm{RN}\Big\{ \cdots \mathrm{RN}\Big\{ \mathrm{RN}\Big[\mathrm{RN}(v_0{+}v_1){+}\mathrm{RN}(v_2{+}v_3)\Big]{+}\mathrm{RN}\Big[\mathrm{RN}(v_4{+}v_5){+}\mathrm{RN}(v_6{+}v_7)\Big]\Big\}{+}\cdots\Big\} \tag{24}$$

(see the first line in Figure 2). After Step k of the FFT algorithm, the intermediate terms in the sum (24) will be floating-point numbers around 2^k, which means that they will be of the form $2^k + ju2^{k+1}$ (where $j \in \mathbb{N}$) for the terms above 2^k, and $2^k - ju2^k$ for the terms below 2^k. We will assume $0 \leqslant j \leqslant 2^{n+1} \ll 1/u$. Let us explain how we can build a term of the form $2^k + ju2^{k+1}$ from the floating-point addition of two terms around 2^{k-1}, trying to maximize the rounding error. Since we want all rounding errors to be in the same direction (in order to maximize the global error, which will be the sum of all individual rounding errors), we will make sure that all roundings are downwards. We assume that RN is round to nearest *ties to even*: if t is halfway between two consecutive FP numbers, then RN(t) is the one of these two numbers whose significand is *even*, i.e., its rightmost significand bit is a zero (this is the default in the IEEE 754 standard).

Notice that since the distance between two floating-point numbers in the neighborhood of $2^k + ju2^{k+1}$ is $u \cdot 2^{k+1}$, the maximum rounding error that can result from that floating-point addition is $u \cdot 2^k$. Therefore

- if j is even, one easily checks that error $u \cdot 2^k$ is attained when adding the two FP numbers $2^{k-1} + ju2^{k+1}$ and $2^{k-1} + u2^k$. The exact sum is $2^k + \left(j + \frac{1}{2}\right) u2^{k+1}$, the rounded result (thanks to the round ties-to-even rule) is the expected $2^k + ju2^{k+1}$, so that the rounding error is $u \cdot 2^k$. This is illustrated in Fig. 3;

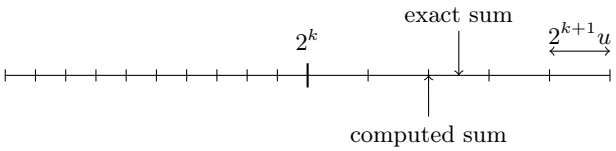

Figure 3: Floating-point addition of $2^{k-1} + ju2^{k+1}$ and $2^{k-1} + u2^k$, where j is even.

- if j is odd, error $u \cdot 2^k$ cannot be attained: if the exact sum was $2^k + ju2^{k+1} \pm u \cdot 2^k$, then (due to the ties-to-even rule) that exact sum would not be rounded to $2^k + ju2^{k+1}$, whose significand is odd. To make sure that the exact sum is not halfway between two FP numbers, it must not be a multiple of $u2^k$, which implies that one of the operands must be below

2^{k-1}. Hence the choice to add $2^{k-1}+u\cdot\left(j+\frac{1}{2}\right)\cdot 2^{k+1}$ and $2^{k-1}-u\cdot 2^{k-1}$. The exact sum is $2^k+ju2^{k+1}+u2^{k-1}$, resulting in a rounded sum equal to the expected 2^k+ju2^{k+1} and a rounding error $u\cdot 2^{k-1}$. This is illustrated in Fig. 4.

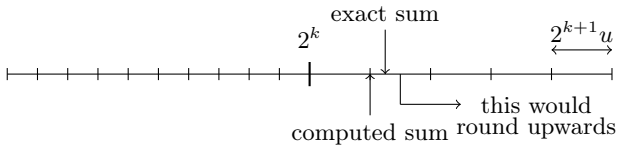

Figure 4: Floating-point addition of $2^{k-1}+u\cdot\left(j+\frac{1}{2}\right)\cdot 2^{k+1}$ and $2^{k-1}-u\cdot 2^{k-1}$, where j is odd.

The reasoning would be similar for building a term of the form 2^k-ju2^k. We can formalize these ideas as follows. Let us call $T_{2^k,\sigma}$ an array of 2^k input values (in "mirror order", i.e., one must choose $z_0=T_{2^k,\sigma}[\text{reverse}(0)]$, $z_1=T_{2^k,\sigma}[\text{reverse}(1)]$, $z_2=T_{2^k,\sigma}[\text{reverse}(2)]$, ... as first elements) for which the *computed sum* that appears in the 1st component of the 2^k-term FFT is equal to $2^k+\sigma$. The number σ will be of the form $j\cdot 2^{k+1}u$ for $\sigma\geqslant 0$, and $-j\cdot 2^k u$ for $\sigma<0$, where j is a small positive integer. We are going to build $T_{2^n,0}$ so that the error when computing the FFT of that array is as large as possible. We denote $T//T'$ as the concatenation of the arrays T and T', we will choose (for $j\geqslant 0$):

$$T_{2^k,j\cdot 2^{k+1}u}=\begin{cases} T_{2^{k-1},(j+\frac{1}{2})\cdot 2^{k+1}u}//T_{2^{k-1},-2^{k-1}u} & \text{if } j \text{ is odd,}\\ T_{2^{k-1},j\cdot 2^{k+1}u}//T_{2^{k-1},2^k u} & \text{if } j \text{ is even,}\end{cases} \tag{25}$$

$$T_{2^k,-j\cdot 2^k u}=\begin{cases} T_{2^{k-1},0}//T_{2^{k-1},-j\cdot 2^k u} & \text{if } j \text{ is odd,}\\ T_{2^{k-1},0}//T_{2^{k-1},(-2j+1)\cdot 2^{k-1}u} & \text{if } j \text{ is even,}\end{cases} \tag{26}$$

with, of course

$$\begin{cases} T_{1,2ju} &= 1+2ju,\\ T_{1,-ju} &= 1-ju.\end{cases}$$

The error on the 1st term of the Fourier Transform will be $C_{n,0}\cdot u$, where the terms $C_{k,m}$ (equal to the error that occurs when performing a 2^k-point FFT with the input values defined by $T_{2^k,m\cdot u}$) are defined by the following recurrence

$$C_{k,m}=\begin{cases} 0 & \text{if } k=0,\\ C_{k-1,m+2^k}+C_{k-1,-2^{k-1}}+2^{k-1} & \text{if } m/2^{k+1} \text{ is odd and } m\geqslant 0,\\ C_{k-1,m}+C_{k-1,2^k}+2^k & \text{if } m/2^{k+1} \text{ is even and } m\geqslant 0,\\ C_{k-1,0}+C_{k-1,m} & \text{if } m/2^k \text{ is odd and } m<0,\\ C_{k-1,0}+C_{k-1,m+2^{k-1}}+2^{k-1} & \text{if } m/2^k \text{ is even and } m<0.\end{cases} \tag{27}$$

Figure 5 illustrates the use of the rules (25) and (26) for building a bad case for $N=8$. Interestingly enough, from the recurrence (27), the GFUN

package [18] makes it possible to find an exact closed formula for $C_{n,0}$. This is based on first computing several terms $C_{n,0}$, say $n \leqslant 24$, and then using GFUN to guess a linear differential equation with polynomial coefficients, satisfied by the generating function $\sum_{n\geqslant 0} C_{n,0}x^n$. This is obviously not always possible, but such an equation exists when $(C_{n,0})_{n\geqslant 0}$ is a P-recursive sequence i.e., it satisfies a linear recurrence with polynomial coefficients in n. It turns out that in our case the guessed generating function is a rational fraction,

$$\frac{2\,x^4 - 2\,x^3 - x^2 + 2\,x}{4\,x^5 - 4\,x^4 + x^3 + 4\,x^2 - 4\,x + 1}.$$

This corresponds to the following guessed linear recurrence

$$\begin{gathered} u_{n+5} - 4u_{n+4} + 4u_{n+3} + u_{n+2} - 4u_{n+1} + 4u_n = 0, \\ u_0 = 0, u_1 = 2, u_2 = 7, u_3 = 18, u_4 = 44, \end{gathered} \tag{28}$$

which has a unique solution, easily obtained by noting that the polynomial

$$x^5 - 4\,x^4 + 4\,x^3 + x^2 - 4\,x + 4$$

is equal to $(x+1)\left(x - e^{i\pi/3}\right)\left(x - e^{-i\pi/3}\right)(x-2)^2$. We now prove

Theorem 10. *The terms $C_{n,0}$ defined by (27) satisfy*

$$C_{n,0} = \frac{1}{27}\cdot 2^n \cdot (15n+14) - \frac{5}{9}\cdot\cos\left(\frac{n\pi}{3}\right) + \frac{1}{9}\cdot\sqrt{3}\cdot\sin\left(\frac{n\pi}{3}\right) + \frac{(-1)^n}{27}.$$

Proof. We have to show that $(C_{n,0})_{n\geqslant 0}$ actually satisfies the guessed recurrence (28). This is done by first showing

$$C_{k,2^{k+1}+j\cdot 2^{k+2}} = C_{k,2^{k+1}} \text{ for all } k, j \geqslant 0, \tag{29}$$

$$C_{k,-2^k+j\cdot 2^{k+1}} = C_{k,-2^k} \text{ for all } k \geqslant 0, j \leqslant 0. \tag{30}$$

We prove (29) by induction on k. For all $j \geqslant 0$, we have $C_{0,2^1+j\cdot 2^2} = 0 = C_{0,2^1}$. Now, from (27), we get

$$C_{k,2^{k+1}} = C_{k-1,2^{k+1}+2^k} + C_{k-1,-2^{k-1}} + 2^{k-1}$$

and

$$C_{k,2^{k+1}+j\cdot 2^{k+2}} = C_{k-1,2^{k+1}+2^k+j\cdot 2^{k+2}} + C_{k-1,-2^{k-1}} + 2^{k-1}$$

and we apply the induction hypothesis to conclude. The proof of (30) is similar to the proof of (29).

Now, to show that $(C_{n,0})_{n\geqslant 0}$ satisfies (28), it suffices to show that for any $k \geqslant 3$, we have

$$C_{k+2,0} - 4C_{k+1,0} + 4C_{k,0} = -C_{k-1,0} + 4C_{k-2,0} - 4C_{k-3,0}.$$

This will be done using (27), (29) and (30). We have

$$
\begin{aligned}
&C_{k+2,0} - 4C_{k+1,0} + 4C_{k,0} \\
=& -3C_{k+1,0} + C_{k+1,2^{k+2}} + 2^{k+2} + 4C_{k,0} \text{ by applying (27) to the term } C_{k+2,0} \\
=& -3\left(C_{k,0} + C_{k,2^{k+1}} + 2^{k+1}\right) + \left(C_{k,2^{k+2}+2^{k+1}} + C_{k,-2^k} + 2^k\right) + 2^{k+2} + 4C_{k,0} \\
=& -3\left(C_{k,0} + C_{k,2^{k+1}} + 2^{k+1}\right) + \left(C_{k,2^{k+1}} + C_{k,-2^k} + 2^k\right) + 2^{k+2} + 4C_{k,0} \text{ by applying (29)} \\
=& C_{k,0} - 2C_{k,2^{k+1}} + C_{k,-2^k} - 2^k \\
=& \left(C_{k-1,0} + C_{k-1,2^k} + 2^k\right) - 2\left(C_{k-1,2^{k+1}+2^k} + C_{k-1,-2^{k-1}} + 2^{k-1}\right) + \left(C_{k-1,0} + C_{k-1,-2^k} - 2^k\right) \\
=& -C_{k-1,0} + 3C_{k-1,0} - C_{k-1,2^k} - 2C_{k-1,-2^{k-1}} + C_{k-1,-2^k} - 2^k \text{ by applying (29)} \\
=& -C_{k-1,0} + 3\left(C_{k-2,0} + C_{k-2,2^{k-1}} + 2^{k-1}\right) - \left(C_{k-2,2^{k-1}} + C_{k-2,-2^{k-2}} + 2^{k-2}\right) \\
& -2\left(C_{k-2,0} + C_{k-2,-2^{k-1}}\right) + \left(C_{k-2,0} + C_{k-2,-2^k+2^{k-2}} + 2^{k-2}\right) - 2^k \\
=& -C_{k-1,0} + 4C_{k-2,0} - 2C_{k-2,0} + 2C_{k-2,2^{k-1}} - 2C_{k-2,-2^{k-1}} + 2^{k-1} \text{ by applying (30)} \\
=& -C_{k-1,0} + 4C_{k-2,0} - 2\left(C_{k-3,0} + C_{k-3,2^{k-2}} + 2^{k-2}\right) + 2\left(C_{k-3,2^{k-2}} + C_{k-3,-2^{k-3}} + 2^{k-3}\right) \\
& -2\left(C_{k-3,0} + C_{k-3,-2^{k-1}+2^{k-3}} + 2^{k-3}\right) + 2^{k-1} \\
=& -C_{k-1,0} + 4C_{k-2,0} - 4C_{k-3,0} \text{ by applying (30).}
\end{aligned}
$$

□

Hence, on the "bad cases" built using the recurrences (25) and (26), the FFT algorithm has an error asymptotically equivalent to $(5/9) \cdot n \cdot 2^n$. The value of $\|z\|_\infty^\perp$ for these bad cases (which is the largest absolute value of an element of the array $T_{N,0}$) is deduced from (25) and (26). It is $1 + M_{n,0} \cdot u$, where the sequence $M_{k,m}$ satisfies

$$
M_{k,m} = \begin{cases}
m & \text{if} \quad k = 0, \\
\max\left(M_{k-1,m+2^k}; M_{k-1,-2^{k-1}}\right) & \text{if} \quad m/2^{k+1} \text{ is odd and } m \geqslant 0, \\
\max\left(M_{k-1,m}; M_{k-1,2^k}\right) & \text{if} \quad m/2^{k+1} \text{ is even and } m \geqslant 0, \\
\max\left(M_{k-1,0}; M_{k-1,m}\right) & \text{if} \quad m/2^k \text{ is odd and } m < 0, \\
\max\left(M_{k-1,0}; M_{k-1,m+2^{k-1}}\right) & \text{if} \quad m/2^k \text{ is even and } m < 0.
\end{cases} \tag{31}
$$

To obtain $M_{k,m}$ from (31), we use numbers of the form $M_{k-1,m'}$, where m' is at most $m + 2^k$, and we continue recursively. Hence the "final" value of m' (when $k = 0$) is at most $2^n + 2^{n-1} + \cdots + 2 = 2N - 2$. Hence, $M_{n,0} \leqslant 2N - 2$. Conversely, that value is effectively attained if we consider the following path of recursive calls:

$$M_{n,0} \to M_{n-1,2^n} \to M_{n-2,2^n+2^{n-1}} \to \cdots \to M_{0,2^n+2^{n-1}+2^{n-2}+\cdots+2}.$$

Therefore, $\|z\|_\infty^\perp = 1 + (2N-2) \cdot u$, so that

$$\frac{\|Z - \hat{Z}\|_\infty^\perp}{\|z\|_\infty^\perp} \sim \frac{5}{9} \cdot \frac{n \cdot 2^n \cdot u}{1 + (2^{n+1} - 2) \cdot u}.$$

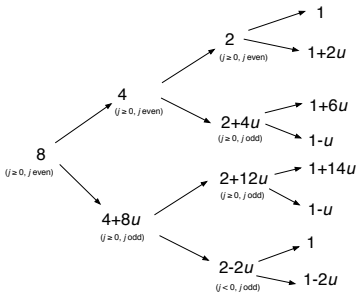

Figure 5: Building a bad case for $N = 8$, according to (25) and (26). The obtained input values are in mirror order: this corresponds to $z_0 = 1$, $z_1 = 1+14u$, $z_2 = 1+6u$, $z_3 = 1$, $z_4 = 1+2u$, $z_5 = 1-u$, $z_6 = 1-u$, and $z_7 = 1-2u$. The absolute error on Z_0, when applying the FFT algorithm on these input values, is $18u$.

Table 5 compares bounds on $\|\hat{Z} - Z\|_2/\|Z\|_2$ and $\|Z - \hat{Z}\|_\infty^\perp/\|z\|_\infty^\perp$ deduced from Theorem 7 (assuming an FMA instruction is available, and $p = 53$–i.e., double-precision arithmetic) and (7) with bounds deduced from Henrici's theorem and the "bad case" we have just built.

5.2 Iterative calculation of an error bound

We assume that the input vector to the FFT algorithm satisfies $\|z\|_\infty^\perp \leqslant 1$. Let us again consider an elementary calculation of the FFT, performed at Step k:

$$\left\{ \begin{array}{rcl} y_1 & = & x_1 + \omega x_2, \\ y_2 & = & x_1 - \omega x_2, \end{array} \right.$$

where x_1, x_2, y_1 and y_2 are "exact" values (i.e., they are obtained from exact calculations, without roundings). We will now iteratively compute error bounds for all the intermediate calculations of the FFT. Denoting x_1^R, x_2^R, y_1^R, y_2^R and ω^R the real parts of x_1, x_2, y_1, y_2 and ω, and x_1^I, x_2^I, y_1^I, y_2^I and ω^I their imaginary parts, we have

$$\left\{ \begin{array}{rcl} y_1^R & = & x_1^R + \omega^R x_2^R - \omega^I x_2^I, \\ y_1^I & = & x_1^I + \omega^I x_2^R + \omega^R x_2^I, \\ y_2^R & = & x_1^R - \omega^R x_2^R + \omega^I x_2^I, \\ y_2^I & = & x_1^I - \omega^I x_2^R - \omega^R x_2^I. \end{array} \right.$$

Now, denote $\widehat{x_1^R}$, $\widehat{x_2^R}$, $\widehat{y_1^R}$, $\widehat{y_2^R}$, $\widehat{x_1^I}$, $\widehat{x_2^I}$, $\widehat{y_1^I}$, and $\widehat{y_2^I}$ as the calculated values of x_1^R, x_2^R, y_1^R, y_2^R, x_1^I, x_2^I, y_1^I, and y_2^I; and denote $\widehat{\omega^R} = \mathrm{RN}(\omega^R)$ and $\widehat{\omega^I} = \mathrm{RN}(\omega^I)$. We assume that an FMA instruction is available, and that $\widehat{y_1^R}$, $\widehat{y_2^R}$, $\widehat{y_1^I}$, and $\widehat{y_2^I}$ are computed as follows

Table 5: Bounds on $\|\hat{Z} - Z\|_2/\|Z\|_2$ and $\|Z - \hat{Z}\|_\infty^\perp/\|z\|_\infty^\perp$ deduced from Theorem 7 (assuming an FMA instruction is available, and $p = 53$–i.e., double-precision arithmetic) and (7), compared with the bound on $\|Z - \hat{Z}\|_\infty^\perp/\|z\|_\infty^\perp$ given by Theorem 6 and the "bad case" presented in this section.

N	Bound on $\|Z - \hat{Z}\|_2/\|Z\|_2$ deduced from Theorem 7 (with FMA)	Bound on $\|Z - \hat{Z}\|_\infty^\perp/\|z\|_\infty^\perp$ deduced from Theorem 7 and (7)	Bound on $\|Z - \hat{Z}\|_\infty^\perp/\|z\|_\infty^\perp$ deduced from Theorem 6	Known bad case for $\|Z - \hat{Z}\|_\infty^\perp/\|z\|_\infty^\perp$ (see Section 5.1)
2^5	$12.85 \cdot u$	$582 \cdot u$	$996 \cdot u + \mathcal{O}(u^2)$	$105 \cdot u$
2^8	$23.71 \cdot u$	$8584 \cdot u$	$12310 \cdot u + \mathcal{O}(u^2)$	$1271 \cdot u$
2^{10}	$30.99 \cdot u$	$44879 \cdot u$	$60823 \cdot u + \mathcal{O}(u^2)$	$6220 \cdot u$
2^{12}	$38.28 \cdot u$	$221720 \cdot u$	$289631 \cdot u + \mathcal{O}(u^2)$	$29430 \cdot u$
2^{14}	$45.63 \cdot u$	$1.058 \times 10^6 \cdot u$	$1.344 \times 10^6 \cdot u + \mathcal{O}(u^2)$	$135927 \cdot u$
2^{16}	$53.03 \cdot u$	$4.915 \times 10^6 \cdot u$	$6.118 \times 10^6 \cdot u + \mathcal{O}(u^2)$	$616524 \cdot u$
2^{18}	$60.43 \cdot u$	$2.240 \times 10^7 \cdot u$	$2.744 \times 10^7 \cdot u + \mathcal{O}(u^2)$	$2.757 \times 10^6 \cdot u$
2^{20}	$67.83 \cdot u$	$1.006 \times 10^8 \cdot u$	$1.216 \times 10^8 \cdot u + \mathcal{O}(u^2)$	$1.219 \times 10^7 \cdot u$

$$\begin{cases} \widehat{y_1^R} &= \mathrm{RN}\left[\widehat{x_1^R} + \mathrm{RN}\left(\widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right)\right)\right], \\ \widehat{y_1^I} &= \mathrm{RN}\left[\widehat{x_1^I} + \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^R} + \mathrm{RN}\left(\widehat{\omega^R}\widehat{x_2^I}\right)\right)\right], \\ \widehat{y_2^R} &= \mathrm{RN}\left[\widehat{x_1^R} - \mathrm{RN}\left(\widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right)\right)\right], \\ \widehat{y_2^I} &= \mathrm{RN}\left[\widehat{x_1^I} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^R} + \mathrm{RN}\left(\widehat{\omega^R}\widehat{x_2^I}\right)\right)\right]. \end{cases} \tag{32}$$

The terms ω^R, ω^I, $\widehat{\omega^R}$ and $\widehat{\omega^I}$ are input-independent, so we assume that the errors

$$\Delta_\omega^R = \left|\widehat{\omega^R} - \omega^R\right| \qquad \text{and} \qquad \Delta_\omega^I = \left|\widehat{\omega^I} - \omega^I\right|$$

are computed in advance. The terms $|x_1^R|$, $|x_1^I|$, $|x_2^R|$, and $|x_2^I|$ are results of a 2^{k-1}-point FFT, hence they are bounded by $\frac{2^{k+1}}{\pi}$ as shown in Section 4.2. However, to take advantage of a significantly smaller bound for small values of k, we will bound $|x_1^R|$, $|x_1^I|$, $|x_2^R|$, and $|x_2^I|$ by β_k, defined as

$$\beta_k = \begin{cases} 2^{k-1} & \text{if } k \leqslant 3, \\ 4 + 4\sqrt{2} & \text{if } k = 4, \\ 2^{k+1}/\pi & \text{otherwise.} \end{cases}$$

Since x_2 is the result of a 2^{k-1}-point FFT, it satisfies (see Section 4.1)

$$\sqrt{\left(x_2^R\right)^2 + \left(x_2^I\right)^2} \leqslant 2^{k-1/2},$$

from which we easily deduce

$$\begin{array}{rcl} \left|x_2^R\omega^R - x_2^I\omega^I\right| & \leqslant & 2^{k-1/2}, \\ \left|x_2^R\omega^I + x_2^I\omega^R\right| & \leqslant & 2^{k-1/2}. \end{array} \tag{33}$$

Also, to take into account the fact that multiplications by ± 1 and $\pm i$ are errorless (these multiplications are frequent in the first steps of the FFT algorithm), define, for a (real) floating-point number t:

$$\mathbb{1}_t = \begin{cases} 1 & \text{if } t = \pm 1, \\ 0 & \text{otherwise,} \end{cases}$$

so that when multiplying some floating-point number v by t, the incurred error is bounded by

$$\frac{1}{2} \cdot (1 - \mathbb{1}_t) \cdot \mathrm{ulp}^*(vt).$$

Define also

$$\mathbb{0}_t = \begin{cases} 1 & \text{if } t = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Let δ_1^R (resp. δ_1^I, δ_2^R, δ_2^I) be a bound on $|x_1^R - \widehat{x_1^R}|$ (resp., $|x_1^I - \widehat{x_1^I}|$, $|x_2^R - \widehat{x_2^R}|$, $|x_2^I - \widehat{x_2^I}|$). From these values we wish to compute η_1^R (resp., η_1^I, η_2^R, η_2^I), bounds on $|y_1^R - \widehat{y_1^R}|$ (resp., $|y_1^I - \widehat{y_1^I}|$, $|y_2^R - \widehat{y_2^R}|$, $|y_2^I - \widehat{y_2^I}|$). Let us detail the calculation of η_1^R. Expressions for η_1^I, η_2^R and η_2^I will be deduced using a straightforward symmetry.

Lemma 6. *The number*

$$\Delta_\omega^R|x_2^R| + \Delta_\omega^I|x_2^I|$$

is less than or equal to

1. $$\Delta_\omega^I \cdot \frac{2^{k+1}}{\pi} + \Delta_\omega^R \cdot 2^k \cdot \sqrt{\frac{1}{2} - \frac{4}{\pi^2}}$$

   if $(\Delta_\omega^R/\Delta_\omega^I)^2 \leqslant \pi^2/8 - 1$*;*

2. $$2^{k-1/2} \cdot \sqrt{(\Delta_\omega^R)^2 + (\Delta_\omega^I)^2}$$

   if $\pi^2/8 - 1 < (\Delta_\omega^R/\Delta_\omega^I)^2 < 1/(\pi^2/8 - 1)$*, and*

3. $$\Delta_\omega^R \cdot \frac{2^{k+1}}{\pi} + \Delta_\omega^I \cdot 2^k \cdot \sqrt{\frac{1}{2} - \frac{4}{\pi^2}}$$

   otherwise.

Proof. The point $(|x_2^R|, |x_2^I|)$ lies in the set

$$\mathcal{A} = \left\{ 0 \leqslant x \leqslant \frac{2^{k+1}}{\pi}, 0 \leqslant y \leqslant \frac{2^{k+1}}{\pi}, x^2 + y^2 \leqslant 2^{2k-1} \right\}.$$

Let $\mathcal{C} = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = 2^{2k-1}\}$ and $(a, b) \in \mathcal{C}$, one equation of the tangent line to $\mathcal{C}$ at (a, b) is

$$\frac{\partial(x^2+y^2)}{\partial x}(a,b)(x-a) + \frac{\partial(x^2+y^2)}{\partial y}(a,b)(y-b) = 0, \text{ i.e., } ax + by = a^2 + b^2.$$

If we denote $\mathcal{D} = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leqslant 2^{2k-1}\}$, for any $(a, b) \in \mathcal{C}$, we have

$$\mathcal{A} \subset \mathcal{D} \subset \{(x, y) \in \mathbb{R}^2 : ax + by \leqslant a^2 + b^2\}. \tag{34}$$

First, we assume $(\Delta_\omega^R/\Delta_\omega^I)^2 \leqslant \pi^2/8 - 1$. Let $(a_0, b_0) = \left(2^k \sqrt{\frac{1}{2} - \frac{4}{\pi^2}}, \frac{2^{k+1}}{\pi}\right) = \left(\frac{2^{k+1}}{\pi}\sqrt{\frac{\pi^2}{8} - 1}, \frac{2^{k+1}}{\pi}\right)$.

For any $(x, y) \in \mathcal{A}$, we have

- either $0 \leqslant x \leqslant a_0$, then
$$\Delta_\omega^R x + \Delta_\omega^I y \leqslant \Delta_\omega^R a_0 + \Delta_\omega^I b_0,$$
- or, we have, from (34), $0 \geqslant a_0(x - a_0) + b_0(y - b_0)$. Now, recall that $a_0/b_0 = \sqrt{\pi^2/8 - 1} \geqslant \Delta_\omega^R/\Delta_\omega^I$ and $x \geqslant a_0$. Therefore, it follows that
$$0 \geqslant \frac{a_0}{b_0}(x - a_0) + (y - b_0) \geqslant \frac{\Delta_\omega^R}{\Delta_\omega^I}(x - a_0) + (y - b_0),$$
i.e.,
$$\Delta_\omega^R x + \Delta_\omega^I y \leqslant \Delta_\omega^R a_0 + \Delta_\omega^I b_0.$$

The case $(\Delta_\omega^R/\Delta_\omega^I)^2 \geqslant 1/(\pi^2/8 - 1)$ follows from symmetry.

Finally, the Cauchy-Schwarz inequality yields, for any $(x, y) \in \mathcal{A}$,

$$\Delta_\omega^R x + \Delta_\omega^I y \leqslant \sqrt{x^2 + y^2}\sqrt{(\Delta_\omega^R)^2 + (\Delta_\omega^I)^2} \leqslant 2^{k-1/2}\sqrt{(\Delta_\omega^R)^2 + (\Delta_\omega^I)^2} \text{ since } (x, y) \in \mathcal{D}.$$

□

In the following, we denote $P(\Delta_\omega^R, \Delta_\omega^I)$ the bound on $\Delta_\omega^I|x_2^I| + \Delta_\omega^R|x_2^R|$ given by Lemma 6. Define

$$A_1^R := \left| \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right) - \omega^I x_2^I \right|.$$

Notice that since $\widehat{x_2^I}$ is a floating-point number within δ_2^I from x_2^I, and since $|x_2^I| \leqslant \beta_k$, we have $|\widehat{x_2^I}| \leqslant \mathrm{RZ}(\beta_k + \delta_2^I)$. We obtain,

$$\begin{aligned} A_1^R &\leqslant \left| \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right) - \widehat{\omega^I}\widehat{x_2^I} \right| + \left| \widehat{\omega^I}\widehat{x_2^I} - \omega^I x_2^I \right| \\ &\leqslant \tfrac{1}{2}(1 - \mathbb{1}_{\widehat{\omega^I}})\mathrm{ulp}^*\left(\widehat{\omega^I}\widehat{x_2^I}\right) + \left| \widehat{\omega^I}\widehat{x_2^I} - \widehat{\omega^I}x_2^I \right| + \left| \widehat{\omega^I}x_2^I - \omega^I x_2^I \right| \qquad (35) \\ &\leqslant \tfrac{1}{2}(1 - \mathbb{1}_{\widehat{\omega^I}})\mathrm{ulp}^*\left(\widehat{\omega^I}\mathrm{RZ}(\beta_k + \delta_2^I)\right) + |\widehat{\omega^I}| \cdot \delta_2^I + \Delta_\omega^I \cdot |x_2^I|. \end{aligned}$$

Define

$$B_1^R := \widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right),$$

and

$$C_1^R := \left|\mathrm{RN}\left(B_1^R\right) - \left(\omega^R x_2^R - \omega^I x_2^I\right)\right|.$$

Note that when $\widehat{\omega^R} = 0$, B_1^R is a floating-point number, hence no error is committed when rounding it. This implies that in all cases the error due to rounding B_1^R is bounded by

$$\frac{1}{2}\left(1 - \mathbb{0}_{\widehat{\omega^R}}\right)\mathrm{ulp}^*(B_1^R).$$

We have,

$$\begin{array}{rcl} C_1^R & \leqslant & \left|\mathrm{RN}\left(\widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right)\right) - \left(\widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right)\right)\right| \\ & & + \left|\left(\widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right)\right) - \left(\omega^R x_2^R - \omega^I x_2^I\right)\right| \\ & \leqslant & \frac{1}{2}\left(1 - \mathbb{0}_{\widehat{\omega^R}}\right)\mathrm{ulp}^*(B_1^R) + \left|\widehat{\omega^R}\widehat{x_2^R} - \omega^R x_2^R\right| \\ & & + \left|\mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right) - \omega^I x_2^I\right|. \end{array} \tag{36}$$

Let us first consider the term $\left|\widehat{\omega^R}\widehat{x_2^R} - \omega^R x_2^R\right|$ in (36). We have

$$\begin{array}{rcl} \left|\widehat{\omega^R}\widehat{x_2^R} - \omega^R x_2^R\right| & \leqslant & \left|\widehat{\omega^R}\widehat{x_2^R} - \widehat{\omega^R} x_2^R\right| + \left|\widehat{\omega^R} x_2^R - \omega^R x_2^R\right| \\ & \leqslant & |\widehat{\omega^R}| \cdot \delta_2^R + \Delta_\omega^R \cdot |x_2^R|. \end{array}$$

Therefore, the sum

$$D_1^R := \left|\widehat{\omega^R}\widehat{x_2^R} - \omega^R x_2^R\right| + \left|\mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right) - \omega^I x_2^I\right|,$$

that appears in (36) is bounded by $A_1^R + |\widehat{\omega^R}| \cdot \delta_2^R + \Delta_\omega^R \cdot |x_2^R|$, which implies from (35) and Lemma 6

$$D_1^R \leqslant \frac{1}{2}(1 - \mathbb{1}_{\widehat{\omega^I}})\mathrm{ulp}^*\left(\widehat{\omega^I}\mathrm{RZ}(\beta_k + \delta_2^I)\right) + |\widehat{\omega^I}| \cdot \delta_2^I + |\widehat{\omega^R}| \cdot \delta_2^R + P(\Delta_\omega^R, \Delta_\omega^I).$$

Finally, in (36), we need to bound the term $|B_1^R|$, in order to obtain a bound on $\frac{1}{2}\mathrm{ulp}^*$ of that value. We have

$$\left|\widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right)\right| \leqslant \left|\widehat{\omega^R}\right| \cdot \mathrm{RZ}(\beta_k + \delta_2^R) + \mathrm{RN}\left(\left|\widehat{\omega^I}\right| \cdot \mathrm{RZ}(\beta_k + \delta_2^I)\right),$$

and we also have, using (33),

$$\left|\widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right)\right| \leqslant \left|\omega^R x_2^R - \omega^I x_2^I\right| + D_1^R \leqslant 2^{k-1/2} + D_1^R.$$

All this gives

$$|B_1^R| \leqslant \min\left\{2^{k-1/2} + D_1^R; \left|\widehat{\omega^R}\right| \cdot \mathrm{RZ}(\beta_k + \delta_2^R) + \mathrm{RN}\left(\left|\widehat{\omega^I}\right| \cdot \mathrm{RZ}(\beta_k + \delta_2^I)\right)\right\}.$$

Now, we have all the elements for obtaining a bound on C_1^R.

$$C_1^R \leqslant \frac{1}{2}\left(1 - \mathbb{0}_{\widehat{\omega^R}}\right)\mathrm{ulp}^*\left(B_1^R\right) + D_1^R.$$

Finally, define

$$Z_1^R := \widehat{x_1^R} + \mathrm{RN}\left(\widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right)\right),$$

so that $\widehat{y_1^R} = \mathrm{RN}(Z_1^R)$. Since $\widehat{x_1^R}$ is a floating-point number less than or equal to $\beta_k + \delta_1^R$, it is less than or equal to $\mathrm{RZ}(\beta_k + \delta_1^R)$. We therefore have

$$Z_1^R \leqslant \mathrm{RZ}(\beta_k + \delta_1^R) + |\mathrm{RN}(B_1^R)|.$$

We can now bound the error on y_1^R:

$$\begin{array}{rcl}
\left|\widehat{y_1^R} - y_1^R\right| & = & \left|\mathrm{RN}(Z_1^R) - \left(x_1^R + \omega^R x_2^R - \omega^I x_2^I\right)\right| \\
& \leqslant & \left|\mathrm{RN}(Z_1^R) - Z_1^R\right| + \left|Z_1^R - \left(x_1^R + \omega^R x_2^R - \omega^I x_2^I\right)\right| \\
& \leqslant & \frac{1}{2}\mathrm{ulp}^*(Z_1^R) + \left|\widehat{x_1^R} - x_1^R\right| \\
& & + \left|\mathrm{RN}\left(\widehat{\omega^R}\widehat{x_2^R} - \mathrm{RN}\left(\widehat{\omega^I}\widehat{x_2^I}\right)\right) - \left(\omega^R x_2^R - \omega^I x_2^I\right)\right| \\
& \leqslant & \frac{1}{2}\mathrm{ulp}^*\left(\mathrm{RZ}(\beta_k + \delta_1^R) + |\mathrm{RN}(B_1^R)|\right) + \delta_1^R + C_1^R.
\end{array}$$

Hence, if we call $\overline{B_1^R}$, $\overline{C_1^R}$, $\overline{D_1^R}$ the bounds we have obtained on $|B_1^R|$, C_1^R and D_1^R, we will choose

$$\eta_1^R = \eta_2^R = \frac{1}{2}\mathrm{ulp}^*\left(\mathrm{RZ}(\beta_k + \delta_1^R) + |\mathrm{RN}(\overline{B_1^R})|\right) + \delta_1^R + \overline{C_1^R}.$$

The same bound straightforwardly applies to $\left|\widehat{y_2^R} - y_2^R\right|$. To deduce a bound on $\left|\widehat{y_1^I} - y_1^I\right|$ and $\left|\widehat{y_2^I} - y_2^I\right|$ it suffices to notice that in (32), one gets line 2 from line 1 and line 4 from line 3 by replacing $\widehat{x_1^R}$ by $\widehat{x_1^I}$, exchanging $\widehat{\omega^R}$ and $\widehat{\omega^I}$, doing the appropriate sign changes, and leaving $\widehat{x_2^R}$ and $\widehat{x_2^I}$ unchanged. Hence, in the final error formulas, one will have to replace δ_1^R by δ_1^I, exchange $\widehat{\omega^R}$ and $\widehat{\omega^I}$, exchange Δ_ω^R and Δ_ω^I, and leave δ_2^R and δ_2^I unchanged. We obtain,

Lemma 7. *We have*

$$\begin{array}{rclcrcl}
\left|\widehat{y_1^R} - y_1^R\right| & \leqslant & \eta_1^R, & \qquad & \left|\widehat{y_2^R} - y_2^R\right| & \leqslant & \eta_2^R, \\
\left|\widehat{y_1^I} - y_1^I\right| & \leqslant & \eta_1^I, & & \left|\widehat{y_2^I} - y_2^I\right| & \leqslant & \eta_2^I,
\end{array}$$

where

$$\eta_1^R = \eta_2^R = \frac{1}{2}\mathrm{ulp}^*\left(\mathrm{RZ}(\beta_k + \delta_1^R) + |\mathrm{RN}(\overline{B_1^R})|\right) + \delta_1^R + \overline{C_1^R}, \tag{37}$$

with

$$\overline{B_1^R} = \min\left\{2^{k-1/2} + \overline{D_1^R}; \left|\widehat{\omega^R}\right| \cdot \mathrm{RZ}(\beta_k + \delta_2^R) + \mathrm{RN}\left(\left|\widehat{\omega^I}\right| \cdot \mathrm{RZ}(\beta_k + \delta_2^I)\right)\right\}.$$

and

$$\overline{C_1^R} = \frac{1}{2}\left(1 - \mathbb{0}_{\widehat{\omega^R}}\right)\mathrm{ulp}^*\left(\overline{B_1^R}\right) + \overline{D_1^R};$$

and

$$\overline{D_1^R} = \frac{1}{2}(1 - \mathbb{1}_{\widehat{\omega^I}})\mathrm{ulp}^*\left(\widehat{\omega^I} \cdot \mathrm{RZ}(\beta_k + \delta_2^I)\right) + |\widehat{\omega^I}| \cdot \delta_2^I + |\widehat{\omega^R}| \cdot \delta_2^R + P(\Delta_\omega^R, \Delta_\omega^I).$$

and

$$\eta_1^I = \eta_2^I = \frac{1}{2}\mathrm{ulp}^*\left(\mathrm{RZ}(\beta_k + \delta_1^I) + |\mathrm{RN}(\overline{B_1^I})|\right) + \delta_1^I + \overline{C_1^I}, \tag{38}$$

with

$$\overline{B_1^I} = \min\left\{2^{k-1/2} + \overline{D_1^I}; \left|\widehat{\omega^I}\right| \cdot \mathrm{RZ}(\beta_k + \delta_2^R) + \mathrm{RN}\left(\left|\widehat{\omega^R}\right| \cdot \mathrm{RZ}(\beta_k + \delta_2^I)\right)\right\}.$$

and

$$\overline{C_1^I} = \frac{1}{2}\left(1 - \mathbb{0}_{\widehat{\omega^I}}\right)\mathrm{ulp}^*\left(\overline{B_1^I}\right) + \overline{D_1^I};$$

and

$$\overline{D_1^I} = \frac{1}{2}(1 - \mathbb{1}_{\widehat{\omega^R}})\mathrm{ulp}^*\left(\widehat{\omega^R} \cdot \mathrm{RZ}(\beta_k + \delta_2^I)\right) + |\widehat{\omega^R}| \cdot \delta_2^I + |\widehat{\omega^I}| \cdot \delta_2^R + P(\Delta_\omega^R, \Delta_\omega^I).$$

We can now use Lemma 7 to "propagate" the error bounds. The structure of the propagation algorithm is exactly the structure of the FFT algorithm. In the pseudcode of Figure 1 we can just replace the two lines
y[j1] = x[j1] + omega[k,j] · x[j2]
y[j2] = x[j1] − omega[k,j] · x[j2]

by the calculation of error bounds on the real and imaginary parts of y[j1] and y[j2] from the error bounds on the real and imaginary parts of x[j1] and x[j2] using Eqs (37) and (38). However, instead of starting from Step $k = 1$ with initial error bounds equal to zero (the input values are assumed exact), one will obtain tighter error bounds by starting from $k = 3$, with initial error bounds equal to the straightforward value $4u$.

The obtained bounds are given in Table 6.

Table 6: Bounds on $\|X - \hat{X}\|_\infty^\perp$, provided by the iterative method. We assume double-precision arithmetic ($p = 53$), and we also assume that an FMA instruction is available. These bounds are for $\|x\|_\infty^\perp \leqslant 1$. To obtain bounds for $\|x\|_\infty^\perp \leqslant 2^m$, it suffices to multiply all the values by 2^m.

N	Bound on $\|X - \hat{X}\|_\infty^\perp$
2^5	$294.21 \cdot u$
2^8	$5757.7 \cdot u$
2^{10}	$36677 \cdot u$
2^{12}	$222685 \cdot u$
2^{14}	$1.321 \times 10^6 \cdot u$
2^{16}	$7.682 \times 10^6 \cdot u$
2^{18}	$4.413 \times 10^7 \cdot u$
2^{20}	$2.519 \times 10^8 \cdot u$

Discussion and conclusion

If we consider error bounds in terms of the 2-norm, in all considered cases, Theorem 7 gives a smaller bound than the previously published ones. When the infinite norm is at stake, using again Theorem 7 and (7) has its interest: in all considered cases (see Table 5), it gives better bounds than Henrici's theorem (Theorem 6), without the annoyance of the $\mathcal{O}(u^2)$ terms. The iterative method developed in Section 5.2 gives a better bound than Theorem 7 and (7) only for reasonably small input size (roughly speaking for $N < 2^{12}$, see Tables 5 and 6). We suspect that this is due to the fact that when we reason in terms of 2-norms we are able to use "exact" relations such as $\|Z\|_2 = \sqrt{N}\|z\|_2$, that have no equivalent in terms of infinite norms.

When one tries to run the FFT algorithm with input values chosen randomly, one obtains errors that are much smaller than the error bounds presented in this paper and the FFT literature. This may make one believe that the bounds are rather loose and need to be significantly improved. The family of examples built in Section 5.1 shows that this is not the case. The obtained errors (for which we have no strong reason to believe that they are worst case errors) are of the same order of magnitude (approximately 8 times smaller) as the bounds given by Theorem 7 and (7), as shown in Table 5.

Finally, the upper bound $\|x\|_\infty^\perp \cdot N \cdot 4/\pi$ on the values that can appear in intermediate calculations, given in Section 4 (and used in Section 5.2), can also be of interest for the designers of hardware implementations of the FFT, since they need to minimize the internal representations of numbers while avoiding overflows.

Acknowledgement

We thank Bruno Salvy for his help in establishing Theorem 10, and the anonymous reviewers whose numerous suggestions have been a great help when revising this paper.
This work is partly supported by the FastRelax grant of the French Agence Nationale de la Recherche.

References

[1] Grahame Bennett and Graham Jameson. Monotonic averages of convex functions. *J. Math. Anal. Appl.*, 252(1):410–430, 2000.

[2] R. P. Brent, C. Percival, and P. Zimmermann. Error bounds on complex floating-point multiplication. *Mathematics of Computation*, 76:1469–1481, 2007.

[3] D. Calvetti. A stochastic roundoff error analysis for the fast Fourier transform. *Mathematics of Computation*, 56(194):755–774, 1991.

[4] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[5] P. Duhamel and M. Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal Process.*, 19:259–299, 1990.

[6] W.M. Gentleman and G. Sande. Fast Fourier transforms–for fun and profit. In *Proc. Fall Joint Computer Conference*, pages 563–578, 1966.

[7] M.T. Heidemann, D.H. Johnson, and C.S. Burrus. Gauss and the history of the fast Fourier transform. *IEEE ASSP Mag.*, pages 14–21, October 1984.

[8] P. Henrici. *Applied and Computational Complex Analysis, Vol. 3.* Wiley, New York, 1986.

[9] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, PA, 2nd edition, 2002.

[10] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic.* IEEE Standard 754-2008, August 2008. Available at `http://ieeexplore.ieee.org/servlet/opac?punumber=4610933`.

[11] Claude-Pierre Jeannerod, Peter Kornerup, Nicolas Louvet, and Jean-Michel Muller. Error bounds on complex floating-point multiplication with an FMA. *Mathematics of Computation*, 86(304):881–898, 2017.

[12] Claude-Pierre Jeannerod, Nicolas Louvet, and Jean-Michel Muller. Further analysis of Kahan's algorithm for the accurate computation of 2×2 determinants. *Mathematics of Computation*, 82(284):2245–2264, October 2013.

[13] D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, Reading, MA, 3rd edition, 1998.

[14] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall Signal Processing Series. Prentice Hall, 2010.

[15] C. Percival. Rapid multiplication modulo the sum and difference of highly composite numbers. *Mathematics of Computation*, 72:387–395, 2002.

[16] G. Plonka and M. Tasche. Fast and numerically stable algorithms for discrete cosine transforms. *Linear Algebra and its Applications*, 394:309–345, 2005.

[17] G.U. Ramos. Roundoff error analysis of the fast Fourier transform. *Mathematics of Computation*, 25(116):757–768, 1971.

[18] B. Salvy and P. Zimmermann. Gfun: A maple package for the manipulation of generating and holonomic functions in one variable. *ACM Trans. Math. Softw.*, 20(2):163–177, June 1994.

[19] J.C. Schatzman. Accuracy of the discrete Fourier transform and the fast Fourier transform. *SIAM J. Sci. Comput.*, 17(5):1150–1166, 1996.

[20] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing (Arch. Elektron. Rechnen)*, 7:281–292, 1971. In German.

[21] Manfred Tasche and Hansmartin Zeuner. Worst and average case roundoff error analysis for fft. *BIT*, 41(3):563–581, Jun 2001.

[22] Charles Van Loan. *Computational frameworks for the fast Fourier transform*, volume 10 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.

[23] Huon Wilson and Uri Keich. Accurate pairwise convolutions of non-negative vectors via FFT. *Comput. Statist. Data Anal.*, 101:300–315, 2016.