Stable Computation of the Complex Roots of Unity

Stephen R. Tate*†

Department of Computer Sciences
University of North Texas
P. O. Box 13886
Denton, TX 76203–6886

Abstract

In this paper, we show that the problem of computing the complex roots of unity is not as simple as it seems at first. In particular, the formulas given in a standard programmer's reference book [Knuth, *Seminumerical Algorithms*, 1981] are shown to be numerically unstable, giving unacceptably large error for moderate sized sequences. We give alternative formulas, which we show to be superior both by analysis and experiment.

EDICS Number: SP 2.2.1

*Research supported in part by NASA subcontract 550-63 of prime contract NAS5-30428, and performed while the author was at Duke University.

†Phone: (817) 565-4864, Fax: (817) 565-2799, E-mail: `srt@cs.unt.edu`

1 Introduction

In most efficient implementations of the Fast Fourier Transform, tables of the powers of the roots of unity are precomputed so that expensive trigonometric function evaluation can be avoided when computing the transform. In this paper, we consider the problem of precomputing these values. For a good survey of issues arising in computing the Fast Fourier Transform, the reader may refer to the excellent paper by Duhamel and Vetterli [2]. In particular, we show that the method given in a standard reference text [3] is numerically unstable, and can produce very inaccurate values for moderately sized sequences. More importantly, we present an alternative way of calculating the roots of unity, present analysis that shows its superiority over the previous method, and finally give empirical results showing that the benefits of the new method are indeed substantial. In addition, while computation time is not a big issue in the computations that we describe, our methods are computationally simpler than the previous method, so there seems to be no reason to choose the previous method over our new method.

We consider the problem of computing the complex roots of unity $\omega_n^k = e^{2\pi ik/n}$ for $k = 0, 1, \cdots, n-1$, where n is a power of two. It is well known that if the set of values $W_r = e^{2\pi i/2^r}$ are known for $r = 0, 1, \cdots, \log n$, then the entire table of ω_n^k values can be computed by a very simple linear time algorithm (only a single complex multiplication is required per table entry). The accuracy of the ω_n^k values depends entirely on the accuracy of the W_r values, so we concentrate on computing the W_r values.

We can consider computing the real and imaginary parts of $W_r = c_r + is_r$ separately, where

$$c_r = \cos\left(\frac{2\pi}{2^r}\right) \qquad\qquad s_r = \sin\left(\frac{2\pi}{2^r}\right). \tag{1}$$

In a standard programmer's reference book [3, p. 292], Knuth gives the following recurrence equations for computing these sequences.

$$\tilde{c}_{r+1} = \sqrt{\frac{1+\tilde{c}_r}{2}} \qquad\qquad \tilde{s}_{r+1} = \sqrt{\frac{1-\tilde{c}_r}{2}} \tag{2}$$

While the equation for $\tilde{c}_r$ is stable, it should be obvious that since $\tilde{c}_r$ approaches 1 rather quickly, the equation for $\tilde{s}_r$ involves the subtraction of almost equal quantities and thus is very unstable.

In this paper, we propose the following set of equations.

$$\tilde{c}_{r+1} = \frac{1}{2}\sqrt{2(1+\tilde{c}_r)} \qquad\qquad \tilde{s}_{r+1} = \frac{\tilde{s}_r}{\sqrt{2(1+\tilde{c}_r)}} \tag{3}$$

We will show by both analysis and experiment that equations (3) have much better error properties than equations (2). Furthermore, notice that in equations (3) a common factor has been found for both equations, meaning that only one square root needs to be taken when evaluating *both* equations. Therefore, not only are the new equations better in terms of error propagation, but they are also computationally simpler.

2 Analysis

In this section, we will show that the formulas in equation (3) are correct mathematical representations of the series defined in equation (1), and that the formulas in equation (3) are stable.

The correctness of the formulas in equation (3) is obvious from the standard double-angle identities of trigonometry (see for example, [1]), namely

$$\cos\alpha = \sqrt{\frac{1+\cos 2\alpha}{2}} \qquad\qquad \sin\alpha = \frac{\sin 2\alpha}{2\cos\alpha}.$$

Thus it is obvious that if the recurrences could be computed exactly, then they would give the precise answers desired. Unfortunately, since most computers use binary floating point numbers for computations, and since the numbers we are computing are irrational, exact computation is impossible.

In analyzing the stability of the recurrence equations in this paper, we will use u to represent *machine precision*. We assume that all computations are performed to this precision. In other words, if x and y represent binary floating point numbers, then adding these numbers produces an answer $(1+\delta)(x+y)$, where δ represents the relative error introduced by rounding the result to fixed, finite precision, and we know that $|\delta| \le u$.

As an example of the stability analysis, as well as to demonstrate why Knuth's recurrence for s_r is unstable, we will analyze the second recurrence in equation (2). In particular, we consider

$$\tilde{s}_{r+1} = \sqrt{\frac{1-\tilde{c}_r}{2}}.$$

Let δ_c represent the relative error in the approximation $\tilde{c}_r$ (so $\tilde{c}_r = (1+\delta_c)c_r$), let δ_1 represent the relative error introduced in the subtraction, and let δ_2 represent the relative error introduced by taking the square root. We will assume that dividing by two can be done exactly, since this is just a decrement of the floating point exponent.

With these error terms, we can write the value that actually gets computed as

$$\begin{aligned}
\tilde{s}_{r+1} &= (1+\delta_2)\sqrt{\frac{(1-c_r(1+\delta_c))(1+\delta_1)}{2}} \\
&= \left[\frac{1-c_r(1+\delta_c)}{1-c_r}\right]^{1/2}(1+\delta_1)^{1/2}(1+\delta_2)\sqrt{\frac{1-c_r}{2}} \\
&= \left[1-\frac{\delta_c c_r}{1-c_r}\right]^{1/2}(1+\delta_1)^{1/2}(1+\delta_2)\,s_{r+1}.
\end{aligned}$$

If we were proving stability results about this equation, we would next go on to bound the relative error of the above expression. However, since for this equation we only wish to see that the recurrence is *unstable*, all that is required is examining the first term in the expression above. From our definitions (see equation (1)), it fairly easy to see that $\frac{1}{1-c_r} \approx \frac{2^{2r}}{2\pi^2}$, so the error grows rapidly with r. Thus Knuth's method for computing the s_r series is unstable.

We prove that our equations are stable in proof of the following theorem.

Theorem 2.1 If the formulas given in equation (3) are computed using floating point binary with precision $u \leq \frac{1}{10}$, then if we define the error terms $\delta_{c,r}$ and $\delta_{s,r}$ by

$$\tilde{c}_r = (1+\delta_{c,r})c_r \qquad\qquad \tilde{s}_r = (1+\delta_{s,r})s_r,$$

then, for all $r \geq 1$,

$$|\delta_{c,r}| \leq 5u \qquad\qquad |\delta_{s,r}| \leq (1+10u)^r - 1.$$

Note: At first glance, it appears that the error bound for $\delta_{s,r}$ is unacceptable, since it grows exponentially. However, since u is typically very small, the geometric ratio of this sequence is actually very close to one. For example, in the most common form of single precision representation, the mantissa has 24 bits, so we can use $u = 2^{-23}$. Even when $r = 20$ (corresponding to an FFT of over a million elements), the bound in the Theorem 2.1 says that $|\delta_{s,20}| < 2^{-15}$. In fact, with a more rigorous proof, it can be proved that for $u = 2^{-23}$, the error is bounded by $|\delta_{s,20}| < 2^{-17}$. Experimental results (see Section 3) show that in fact the error may be much smaller than shown by this upper bound.

Proof: As above, we will make explicit the rounding errors introduced from computation using finite precision numbers. Consider the computation of $\tilde{c}_{r+1}$ in equation (3). As before, we make the realistic assumption that multiplication and division by 2 can be performed exactly, so these computations add no error to the final result. Thus, the computation of $\tilde{c}_{r+1}$ is represented by

$$\begin{aligned}\tilde{c}_{r+1} &= \frac{1}{2}(1+\delta_2)\sqrt{2(1+\delta_1)(1+(1+\delta_{c,r})c_r)} \\ &= (1+\delta_1)^{1/2}(1+\delta_2)\left(1+\frac{\delta_{c,r}c_r}{1+c_r}\right)^{1/2} c_{r+1},\end{aligned}$$

where δ_1 represents the relative error caused by the addition, δ_2 represents the relative error caused by the square root, and $\delta_{c,r}$ represents the relative error of the approximation $\tilde{c}_r$. Obviously, if we can find a positive value ϵ such that

$$(1-\epsilon) \leq (1+\delta_1)^{1/2}(1+\delta_2)\left(1+\frac{\delta_{c,r}c_r}{1+c_r}\right)^{1/2} \leq (1+\epsilon),$$

then we have proved the bound $|\delta_{c,r+1}| \leq \epsilon$.

We will now introduce an easily verified fact that will be helpful in proving error bounds.

Fact 1: Let α_1 and α_2 be small constants, and define $\alpha_{\max} = \max\{|\alpha_1|, |\alpha_2|\}$. Then

$$(1-\alpha_{\max}) \leq [(1+\alpha_1)(1+\alpha_2)]^{1/2} \leq (1+\alpha_{\max}).$$

Returning to the analysis of the relative error introduced in the computation of $\tilde{c}_{r+1}$, using Fact 1 from above, and the fact that $\frac{c_r}{1+c_r} < \frac{1}{2}$, we can clearly bound the relative error by

$$(1-u)(1-\max(u, \frac{1}{2}|\delta_{c,r}|) \leq (1+\delta_1)^{1/2}(1+\delta_2)\left(1+\frac{\delta_{c,r}c_r}{1+c_r}\right)^{1/2} \leq (1+u)(1+\max(u, \frac{1}{2}|\delta_{c,r}|)).$$

Since $u \leq \frac{1}{10}$, a little manipulation yields

$$(1+u)(1+\max(u, \frac{1}{2}|\delta_{c,r}|) \leq 1 + \frac{21}{10}u + \frac{11}{20}|\delta_{c,r}|.$$

Note that this is a somewhat looser bound than can obtained, but it will suffice for our purposes. An even tighter *lower* bound can be trivially obtained, giving our intermediate result that

$$|\delta_{c,r+1}| \leq \frac{21}{10}u + \frac{11}{20}|\delta_{c,r}|.$$

From this equation, it can easily be shown by induction that for all $r \geq 1$,

$$|\delta_{c,r}| \leq \frac{14}{3}u < 5u,$$

which completes the proof of the first error bound in the theorem statement.

To prove the error bound for the $\tilde{s}_r$ sequence, first notice that the denominator of the recurrence for $\tilde{s}_r$ in equation (3) will have relative error $\delta_{c,r+1}$, since the denominator is only a factor of two different from the value $\tilde{c}_{r+1}$. In other words, we know that

$$\tilde{s}_{r+1} = \frac{\tilde{s}_r}{\sqrt{2(1+\tilde{c}_r)}} = \frac{(1+\delta_{s,r})s_r}{(1+\delta_{c,r+1})\sqrt{2(1+c_r)}} = \frac{1+\delta_{s,r}}{1+\delta_{c,r+1}} s_{r+1}.$$

Now we introduce another easily verified fact to simplify analysis.

Fact 2: If $|\delta| \le \frac{1}{2}$, then

$$(1-2|\delta|)\frac{1}{|x|} \le \frac{1}{(1+\delta)|x|} \le (1+2|\delta|)\frac{1}{|x|}.$$

Using Fact 2 in conjunction with the error bound that we have already proved for $\delta_{c,r+1}$, we get

$$(1-10u)(1-|\delta_{s,r}|)s_{r+1} \le \tilde{s}_{r+1} \le (1+10u)(1+|\delta_{s,r}|)s_{r+1}.$$

In other words,

$$|\delta_{s,r+1}| \le (1+10u)|\delta_{s,r}| + 10u,$$

or (solving this recurrence),

$$|\delta_{s,r+1}| \le (1+10u)^{r+1} - 1,$$

which completes the proof of the second error bound. ∎

3 Empirical Results

In this section, we report on results obtained in some simple implementations of both methods of computing roots of unity. The implementations calculated approximate c_r and s_r series' using single precision floating point operations. To compute the error, we compared these values with the double precision values computed by the library functions $\sin(x)$ and $\cos(x)$. The values computed, as well as the relative error computed in this way, are shown in Table 1.

As can be seen from the table, the error in computing the s_r series via the method found in Knuth's book has significant errors around the 10th term. In fact, at the 13th term, the relative error of Knuth's method is approximately 5 orders of magnitude worse than the new method presented in this paper. These experiments show that the instability of Knuth's method is a very real problem, but one that can be overcome by using equation (3).

	Real Part				Imaginary Part			
	Knuth's Method		Stable Method		Knuth's Method		Stable Method	
r	c_r	Rel. Err.	c_r	Rel. Err.	s_r	Rel. Err.	s_r	Rel. Err.
3	0.707107	1.71e-08	0.707107	1.71e-08	7.07107e-01	1.71e-08	7.07107e-01	1.71e-08
4	0.923880	3.39e-08	0.923880	3.39e-08	3.82683e-01	1.63e-08	3.82683e-01	1.63e-08
5	0.980785	3.05e-08	0.980785	3.05e-08	1.95090e-01	2.21e-07	1.95090e-01	6.78e-08
6	0.995185	7.14e-09	0.995185	7.14e-09	9.80171e-02	7.51e-07	9.80171e-02	6.69e-08
7	0.998795	6.48e-09	0.998795	6.48e-09	4.90677e-02	7.19e-07	4.90677e-02	4.05e-08
8	0.999699	9.67e-10	0.999699	9.67e-10	2.45413e-02	2.68e-06	2.45412e-02	5.61e-08
9	0.999925	1.75e-08	0.999925	1.75e-08	1.22716e-02	1.60e-06	1.22715e-02	6.85e-08
10	0.999981	1.04e-08	0.999981	1.04e-08	6.13517e-03	1.16e-04	6.13588e-03	7.21e-08
11	0.999995	2.58e-09	0.999995	2.58e-09	3.06880e-03	2.75e-04	3.06796e-03	7.30e-08
12	0.999999	1.55e-08	0.999999	1.55e-08	1.53440e-03	2.74e-04	1.53398e-03	3.53e-08
13	1.000000	3.89e-09	1.000000	3.89e-09	7.72040e-04	6.58e-03	7.66990e-04	2.59e-08
14	1.000000	1.39e-08	1.000000	1.39e-08	3.86020e-04	6.58e-03	3.83495e-04	2.36e-08
15	1.000000	1.84e-08	1.000000	1.84e-08	1.72633e-04	9.97e-02	1.91748e-04	4.19e-08
16	1.000000	4.60e-09	1.000000	4.60e-09	0.00000e+00	1.00e+00	9.58738e-05	4.65e-08
17	1.000000	1.15e-09	1.000000	1.15e-09	0.00000e+00	1.00e+00	4.79369e-05	4.77e-08
18	1.000000	2.87e-10	1.000000	2.87e-10	0.00000e+00	1.00e+00	2.39684e-05	4.80e-08
19	1.000000	7.18e-11	1.000000	7.18e-11	0.00000e+00	1.00e+00	1.19842e-05	4.80e-08
20	1.000000	1.80e-11	1.000000	1.80e-11	0.00000e+00	1.00e+00	5.99211e-06	4.81e-08

Table 1: Empirical comparison of stability of the two algorithms

4 Conclusion

We have demonstrated that the method for computing roots of unity given in a standard programmer's reference guide [3] is unstable. The instability is apparent from the analysis, and experimental evidence shows that the instability is indeed a problem for realistically sized data sets.

More importantly, in equation (3) we have given alternative formulas for the computation of the roots of unity. We have shown both by analysis and experiment that the new equations are indeed stable, and provide substantially more accurate results than the previous formulas. In addition, although the complexity of the equations is not a great issue for this problem, the new formulas are computationally simpler than the previous ones. Given these results, it seems that the formulas in equation (3) are the only reasonable way of computing the complex roots of unity.

References

[1] W. H. Beyer, editor. *CRC Standard Mathematical Tables*, 26th Edition. CRC Press, Inc., Boca Raton, FL, 1981.

[2] P. Duhamel and M. Vetterli. "Fast Fourier Transforms: A Tutorial Review and a State of the Art", *Signal Processing*, Vol. 19, 1990, pp. 259–299.

[3] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Second Edition, 1981.