FILON-CLENSHAW-CURTIS QUADRATURE WITH AUTOMATIC TONE REMOVAL

DAVID MILOVICH

ABSTRACT. We present an adaptive method of computing finite Fourier integrals $\int_a^b f(x)g(x)e^{i\omega x}dx$ for smooth f and smooth nonzero g, intended for the case where g is highly and irregularly oscillatory, but f, $|g|$, and $(\arg(g))'$ are not highly oscillatory. Our method easily extends to real-valued integrals $\int_a^b f(x)\cos h(x)\cos(\omega x)dx$ where h is smooth, with the intention that h is not highly oscillatory but $\cos h$ is. We do not require any special information about f or g, just the ability to evaluate f, g, and g' at arbitrary points. Our basic step per subinterval is to integrate $h(x)e^{i(\omega+m)x}$ where m is the slope of $\arg(g)$ at the subinterval center and $h(x)$ is a Chebyshev polynomial interpolation of $f(x)g(x)e^{-imx}$. Thus, a tone is factored out to improve Chebyshev interpolation accuracy. We also investigate a second-order extension in which a chirp is factored out. An implementation of this paper's algorithms in the Julia programming language is publicly available.

1. INTRODUCTION

1.1. Outline.

(1) This section reviews Filon-Clenshaw-Curtis (FCC) quadrature and briefly introduces our modification of it, tone removal.
(2) Section 2 explains tone removal in more detail and in the context of a hybrid interval-degree adaptive numerical integration algorithm.
(3) Section 3 explains chirp removal, a higher-order extension of tone removal.
(4) Section 4 explains how to adapt tone and chirp removal to real-valued integrands.
(5) Section 5 analyzes our computation of FCC quadrature weights and proves some error bounds.
(6) Section 6 compares the performance of four variations of FCC quadrature: degree adaptive, hybrid interval-degree adaptive, hybrid interval-degree adaptive with tone removal, and hybrid interval-degree adaptive with chirp removal.
(7) Section 7 provides additional information about our software implementation of our algorithms.

1.2. Irregularly Oscillatory Integrands.

Filon-Clenshaw-Curtis (FCC) quadrature is a strategy for computing numerical Fourier integrals

$$\int_{-1}^{1} dx\, e^{i\omega x} f(x) \tag{1}$$

for smooth $f\colon [-1,1]\to\mathbb{C}$. We review the essentials of this method in Subsection 1.4. When f is not highly oscillatory, it is well approximated by Chebyshev interpolation of moderate degree N, and FCC quadrature is very efficient. On the other hand, if f is highly oscillatory, then accurate integration requires either a large degree N or application of a moderate degree N to many small subintervals of $[-1,1]$. Either way, many evaluations of f are required.

Of course, if f can be factored as $f(x)=g(x)e^{i\mu x}$ with μ constant and g not highly oscillatory, then we are back in the case where FCC quadrature performs well, provided we know a good value of μ. The core of our strategy is to divide the domain of integration into subintervals on which f can be factored as above, and to compute μ from the derivative of f at the center of the subinterval. Note that this strategy also

Date: June 10, 2022.
Key words and phrases. Clenshaw-Curtis quadrature, Chebyshev polynomials, oscillatory integrals, automatic differentiation, Julia .
This research was funded and approved for public release by the Defense Threat Reduction Agency under contracts HDTRA1-18-P-0012 and HDTRA1-20-C-0001.

handles non-Fourier oscillatory integrals

$$\int_{-1}^{1} dx\, f(x) \tag{2}$$

as a special case ($\omega = 0$).

Various other modifications of FCC quadrature have been proposed to handle integrands of the form $p(x)e^{i\omega q(x)}$ where p and q are not highly oscillatory. A popular strategy is to use the change of variables $y = q(x)$:

$$\int dx\, e^{i\omega q(x)} f(x) = \int dy\, e^{i\omega y} \frac{p(q^{-1}(y))}{q'(q^{-1}(y))} \tag{3}$$

where q^{-1} is a local inverse. Neighborhoods of stationary points of q are handled using specialized techniques. This is not our strategy for the following reasons.

(1) Our intended application is for families of integrands of a slightly different form: $p(x)q(x)e^{i\omega x}$ for a finite set of values of ω, with p, $|q|$, and $(\arg(q))'$ smooth and not highly oscillatory.
(2) Even though we could rearrange our integrand into the form $p_\omega(x)e^{i\omega q_\omega(x)}$ for each ω, we want our technique to work even in ignorance of any stationary points of $q_\omega(x)$.

1.3. **Motivating Application.** The original motivation for this work was a problem in digital channel simulation [6] where digital simulation of an ionospheric scintillation channel model [3] required computing a covariance tensor whose coordinates were linear combinations of finite Fourier integrals of the form $\int_0^{\pm 1} dx\, e^{2n\pi i x} R(x)$ and $\int_0^{\pm 1} dx\, x e^{2n\pi i x} R(x)$ where $R(x)$ is an autocovariance function of the form

$$\frac{c_1}{\sqrt{q_1(x)}} \exp\left(q_2(x) + \frac{q_3(x)}{q_1(x)}\right) \operatorname{erf}\left(\frac{\sigma c_2 + q_4(x)/q_1(x)}{\sqrt{q_5(x)/q_1(x)}}\right)\Bigg|_{\sigma=-1}^{\sigma=+1} \tag{4}$$

where each c_k is a complex constant and each q_k is a complex quadratic polynomial. For a given channel model, integrals for as many as $\sim 10^3$ values of n and as many as $\sim 10^3$ different functions $R(x)$ were needed. The functions $R(x)$ were typically complex-valued and irregularly and highly oscillatory. However, amplitude $|R(x)|$ and instantaneous angular frequency $\frac{d}{dx}\arg R(x)$ were non-oscillatory.

1.4. **Filon-Clenshaw-Curtis Quadrature.** In outline, the following is the FCC quadrature strategy for computing $\int_{-1}^{1} dx\, e^{i\omega x} f(x)$.

(1) Evaluate f at the Chebyshev nodes $c_k^N = \cos(\pi k/N)$, for $0 \le k \le N$, for some degree N with only small prime factors.
(2) Use a fast cosine transform to compute Chebyshev series coefficients

$$\xi_n^N = \frac{2}{N} {\sum_{k=0}^{N}}'' c_{kn}^N f(c_k^N) \tag{5}$$

in $O(N \log N)$ time.[1] These coefficients encode the Chebyshev polynomial interpolation

$$f(c_k^N) = {\sum_{n=0}^{N}}'' \xi_n^N c_{kn}^N = {\sum_{n=0}^{N}}'' \xi_n^N T_n(c_k^N) \tag{6}$$

where $T_n(\cos\theta) = \cos n\theta$.
(3) For all (real) desired values of ω, evaluate

$$I_N(\vec{\xi}, \omega) = {\sum_{n=0}^{N}}'' \xi_n^N \int_{-1}^{1} dx\, e^{i\omega x} T_n(x). \tag{7}$$

[1]The notation $\sum''$ denotes a sum with half weight given to the first and last summands.

There are many implementations of (3). See [5] for a survey of their stability and efficiency. Importantly, the time complexity of accurately computing $I_N(\vec{\xi},\omega)$ is $O(N)$ and uniform with respect to ω.

To cheaply estimate the error $I_N(\vec{\xi},\omega)$, we compare it to its truncation

$$I_{M,N}(\vec{\xi},\omega) = \sum_{n=0}^{M}{}' \xi_n^N \int_{-1}^{1} dx\, e^{i\omega x} T_n(x) \tag{8}$$

for some $M < N$.[2] A popular but very conservative choice is $M = N/2$. Using degree-M coefficients $(\xi_m^M)_{m\le M}$ in the above truncated sum might appear more appropriate. However, if M divides N, then the condition $\xi_n^N \approx 0$ for $n \in (M,N]$ already implies that $(\xi_m^N)_{m\le M}$ is insensitive to reducing N to M. This is mostly easily seen by comparing cosine transforms of $(\xi_m^M)_{m\le M}$ and $(\xi_m^N)_{m\le M}$:

$$\sum_{m=0}^{M}{}'' c_{mk}^M \xi_m^M = f(c_k^M) = f(c_{kN/M}^N) = \sum_{n=0}^{N}{}'' c_{nkN/M}^N \xi_n^N \approx \sum_{n=0}^{M}{}'' c_{nkN/M}^N \xi_n^N = \sum_{m=0}^{M}{}'' c_{mk}^M \xi_m^N. \tag{9}$$

In practice, even if M does not divide N, if $N-M$ is not too small then smallness of ξ_n^N for $n \in (M,N]$ is strong evidence that $f(\cos\theta)$ is well-approximated by the first $M+1$ terms of its cosine series.

A distinctive feature of Chebyshev interpolation is its progressivity: if N divides N' then the Chebyshev nodes for degree N are a subset of the nodes for degree N'. This reduces the cost of degree-adaptive methods, which typically repeatedly double N until estimated error is tolerable.

2. Tone Removal

Given a desired integral $\int_a^b dx\, e^{i\omega x} f(x)$, we assume that $f\colon [a,b] \to \mathbb{C}$ is smooth and presented as $\alpha\beta$ where β is nonzero and β' can be accurately computed either directly or through automatic differentiation. Our tone removal method is intended for the case where α, $|\beta|$, and $(\arg(\beta))'$ are not highly oscillatory. An example is $\alpha(x) = x^2$ and $\beta(x) = e^{-100ix^2}/(x+i)$. We use β' only to compute the angular velocity of β via $(\arg\beta)' = \Im(\beta'/\beta)$. On any interval S, we can factor $\beta(x)$ as $(\beta(x)e^{-i\nu x})e^{i\nu x}$ where ν is the angular velocity of β at the center of S. If angular velocity does not vary too much on S, then $\beta(x)e^{-i\nu x}$ is not highly oscillatory on S.

Thus, the integral $\int_a^b dx\, e^{i\omega x} f(x)$ is rearranged into $\int_a^b dx\, e^{i\tilde{\omega}x}\tilde{f}(x)$ where $\tilde{\omega} = \omega+\nu$ and $\tilde{f}(x) = f(x)e^{-i\nu x}$. If ν is large and a good approximation of $(\arg(\beta))'$ on $[a,b]$, then this rearrangement is highly advantageous. We can accurately interpolate $\tilde{f}(x)$ from samples at far fewer Chebyshev nodes (that is, using a much smaller N) than would be required to accurately interpolate $f(x)$. If ν is not large or is not a good approximation of $(\arg(\beta))'$ on all of $[a,b]$, then the rearrangement does little harm.

2.1. Hybrid Interval-Degree Adaptive Algorithm.

We use a hybrid adaptive algorithm for computing $\int_a^b dx\, e^{i\omega x} f(x)$ with an outer recursion over subintervals and an inner loop over degree. Fix a finite set Ω of angular frequencies ω, a relative error goal $\delta_{\text{rel}} > 0$, and an absolute error goal $\delta_{\text{abs}} \ge 0$. First, letting $[a,b] = [c-r, c+r]$, we use the change of variables $x = ry + c$ to reduce to the case $[a,b] = [-1,1]$:

$$\int_a^b dx\, e^{i\omega x} f(x) = re^{i\omega c} \int_{-1}^{1} dy\, e^{ir\omega y} f(ry+c). \tag{10}$$

So, assuming $[a,b] = [-1,1]$, our recursive algorithm for computing $\int_{-1}^{1} dx\, e^{i\omega x} f(x)$ first computes $\beta(0)$ and $\beta'(0)$ to obtain ν and then samples $f(x)e^{-i\nu x}$ at the $N+1$ Chebyshev nodes where $N = 8$. After computing the corresponding Chebyshev coefficients $\vec{\xi}$, we compute the FCC estimate

$$\int_{-1}^{1} dx\, e^{i\omega x} f(x) = \int_{-1}^{1} dx\, e^{i(\omega+\nu)y} e^{-i\nu y} f(x) \approx I_N(\vec{\xi}, \omega+\nu) \tag{11}$$

for each ω. To estimate accuracy, we compute the norm Υ_N of $(I_N(\vec{\xi},\omega+\nu))_{\omega\in\Omega}$ and the norm $\Upsilon_{M,N}$ of the discrepancy

$$\Big(I_N(\vec{\xi},\omega+\nu) - I_{M,N}(\vec{\xi},\omega+\nu)\Big)_{\omega\in\Omega} \tag{12}$$

[2]The notation $\sum'$ denotes a sum with half weight given to the first summand.

where $M = 3N/4$. (Our implementation uses the Euclidean norm by default but supports arbitrary user-supplied vector pseudonorms.)

Our acceptance criterion is $\Upsilon_N \leq \max(\delta_{\text{abs}}, \delta_{\text{rel}}\Upsilon_{M,N})$. If it is met, then we output our estimates $(I_N(\vec{\xi}, \omega + \nu))_{\omega\in\Omega}$ and our error estimates $((I_N - I_{M,N})(\vec{\xi}, \omega + \nu))_{\omega\in\Omega}$. If it is not met, then we double N to 16, sample $f(x)e^{-i\nu x}$ at the 8 new Chebyshev nodes $c_1^{16}, c_3^{16}, \ldots, c_{15}^{16}$, and recompute our estimates and error estimates. We repeatedly double N until $\Upsilon_N \leq \max(\delta_{\text{abs}}, \delta_{\text{rel}}\Upsilon_{M,N})$ or $N = 64$. This is the inner loop over degree.

If we complete the inner loop but still reject our estimates, then we divide $[-1, 1]$ into $B = 4$ congruent subintervals and recurse, applying our algorithm to each subinterval with δ_{abs} updated to $\max(\delta_{\text{abs}}, \delta_{\text{rel}}\Upsilon_N)/B$. We output estimates and error estimates summed over the subintervals. We limit the above recursion to depth $D = 10$. That is, at any subinterval of depth 10 in our recursion tree, if we complete the inner loop over degree, then we unconditionally accept our estimates and error estimates for that subinterval.

The inner loop range $\{8, 16, 32, 64\}$, the branching factor $B = 4$, and the maximum recursion depth $D = 10$ are default values of our implementation that can be modified by user. However, we require that the inner loop range be an interval of powers of 2 and that the minimum of this interval must be at least 8.

3. Second-Order Extension: Chirp Removal

If factoring a tone $e^{i\nu x}$ out of $f(x)$ enables accurate numerical integration with fewer evaluations of f, then, for functions f that are very expensive to evaluate, it is tempting to try to factor out a polynomial chirp $e^{ixp(x)}$. We have implemented linear chirp removal where a quadratic Taylor series is used to factor out a linear chirp $e^{ix(\nu+\mu x)}$. However, the additional computational cost of our chirp removal method makes it inferior to tone removal except for integrands that are extremely expensive to evaluate.

Consider the following modification of our tone removal algorithm for estimating $\int_{-1}^{1} dx\, e^{i\omega x} f(x)$ for all ω in some finite set Ω. In addition to computing $\beta(0)$ and $\beta'(0)$ we now also compute $\beta''(0)$ to obtain the linear Taylor approximation $\nu + \mu x$ of $\frac{d}{dx}\arg(\beta)$ at 0. If μ is too large,[3] then we divide $[-1, 1]$ into $B = 4$ congruent subintervals and recurse, unless we reached recursion depth $D = 10$, in which case we fall back to mere tone removal. Otherwise, instead of merely factoring out a tone before sampling at the Chebyshev nodes, we now factor out a linear chirp:

$$\sum_{n=0}^{N}{}'' \xi_n T_n(x) \approx f(x)e^{-ix(\nu+\mu x)}. \tag{13}$$

For each ω, the desired estimate is now

$$\int_{-1}^{1} dx\, e^{i\omega x} f(x) \approx \sum_{n=0}^{N}{}'' \xi_n \int_{-1}^{1} dx\, e^{i(\omega+\nu)x} e^{i\mu x^2/2} T_n(x). \tag{14}$$

For each p in a fixed finite set E,[4] we have precomputed an accurate Chebyshev interpolation

$$\sum_{m=0}^{D_p}{}'' \eta_{p,m} T_m(x) \approx e^{ipx^2}. \tag{15}$$

We choose the least $p \geq \mu$ and set $\kappa = \sqrt{\mu/p}$. For each ω, the desired estimate is now

$$\int_{-1}^{1} dx\, e^{i\omega x} f(x) \approx \sum_{n=0}^{N}{}'' \xi_n \sum_{m=0}^{D_p}{}'' \eta_{p,m} \int_{-1}^{1} dx\, e^{i(\omega+\nu)x} T_m(\kappa x) T_n(x). \tag{16}$$

As described in Subsection 3.1, a dilated Chebyshev polynomial is a linear combination of Chebyshev polynomials of equal or lesser degree and, given κ, we can compute the rearrangement

$$\sum_{m=0}^{D_p}{}'' \eta_{p,m} T_m(\kappa x) = \sum_{m=0}^{D_p}{}'' \eta_{p,m,\kappa} T_m(x) \tag{17}$$

[3] By default, "too large" is $\mu > 2^{9.7}$ in our implementation. The user may optionally increase this threshold up to $2^{16.9}$.

[4] 13 values ranging from $2^{1.5}$ to $2^{16.9}$ in our implementation.

TABLE 1. Chebyshev dilation coefficients.

$\varphi_{n,m}$	$m=0$	$m=1$	$m=2$	$m=3$
$n=0$	1			
$n=1$	κ			
$n=2$	κ^2	κ^2-1		
$n=3$	κ^3	$3\kappa(\kappa^2-1)$		
$n=4$	κ^4	$4\kappa^2(\kappa^2-1)$	$(3\kappa^2-1)(\kappa^2-1)$	
$n=5$	κ^5	$5\kappa^3(\kappa^2-1)$	$5\kappa(2\kappa^2-1)(\kappa^2-1)$	
$n=6$	κ^6	$6\kappa^4(\kappa^2-1)$	$3\kappa^2(5\kappa^2-3)(\kappa^2-1)$	$(10\kappa^4-8\kappa^2+1)(\kappa^2-1)$

in $O(D_p^2)$ time. Next, using the identity $2T_mT_n = T_{m+n} + T_{|m-n|}$, we multiply the Chebyshev series (13) and (17) in $O(ND_p)$ time:

$$\sum_{n=0}^{N}{}'' \xi_n T_n(x) \sum_{m=0}^{D_p}{}'' \eta_{p,m,\kappa} T_m(x) = \sum_{n=0}^{N+D_p}{}'' \zeta_n T_n(x). \tag{18}$$

For each ω, the desired estimate is now

$$\int_{-1}^{1} dx\, e^{i\omega x} f(x) \approx I_{N+D_p}(\vec{\zeta}, \omega+\nu), \tag{19}$$

which we compute in $O(N+D_p)$ time. Total computational complexity is therefore $O(N\log N + ND_p + D_p^2)$. This complexity and the empirical run times in Section 6 recommend against chirp removal except for integrands that are extremely expensive to evaluate.

To obtain an error estimate, we repeat the computations (18) and (19) with the Chebyshev series (13) truncated to degree $M = 3N/4$. Then, just as in our tone removal quadrature algorithm, we decide whether to accept $I_{N+D_p}(\vec{\zeta}, \omega+\nu)$, double N, or recurse to subintervals.

3.1. **Chebyshev Dilation.** Given a scalar κ, the degree-n dilated Chebyshev polynomial $T_n(\kappa x)$, being an even or odd polynomial of degree n, is a linear combination of the Chebyshev polynomials $T_n(x)$, $T_{n-2}(x)$, $T_{n-4}(x)$, ...:

$$T_n(\kappa x) = \sum_{0\le 2m\le n} \varphi_{n,m}(\kappa) T_{n-2m}(x). \tag{20}$$

We compute the above coefficients $\varphi_{n,m} = \varphi_{n,m}(\kappa)$ using the recursion

$$\varphi_{n,m} = \kappa\varphi_{n-1,m} + \kappa\varphi_{n-1,m-1} - \varphi_{n-2,m-1} \quad (\forall m \in [1, \lfloor n/2 \rfloor]); \tag{21}$$

$$\varphi_{2n+1,n} = 2\kappa\varphi_{2n,n} + \kappa\varphi_{2n,n-1} - \varphi_{2n-1,n-1} \quad (\forall n \ge 1); \tag{22}$$

$$\varphi_{2n,n} = \kappa\varphi_{2n-1,n-1} - \varphi_{2n-2,n-1} \quad (\forall n \ge 1); \tag{23}$$

$$\varphi_{n,0} = \kappa\varphi_{n-1,0} \quad (\forall n \ge 1); \tag{24}$$

$$\varphi_{0,0} = 1, \tag{25}$$

which follows from the usual recursion $T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$. Explicit formulas for $0 \le 2m \le n \le 6$ are given in Table 1. We also note that if $\kappa \in [-1,1]$ then $\varphi_{n,m}$ has an integral formula

$$\varphi_{n,m} = \frac{2}{\pi}\int_0^\pi dx\, \cos[n\cos^{-1}(\kappa\cos x)]\cos[(n-2m)x] \tag{26}$$

that implies a uniform bound $|\varphi_{n,m}| \le 2$.

4. Extension to Real Oscillatory Integrals

For real Fourier integrals $\int_a^b dx\, f(x)\cos(\omega x)$ and $\int_a^b dx\, f(x)\sin(\omega x)$ where $f\colon [a,b] \to \mathbb{R}$ is smooth and highly oscillatory, if f can be factored as $\alpha\cos(\gamma)$ or $\alpha\sin(\gamma)$ where α and γ are each real-valued and not

highly oscillatory, then our tone and chirp removal methods are applicable using $\beta = e^{i\gamma}$:

$$\int_a^b dx\, \alpha(x)\cos(\gamma(x))\cos(\omega x) = \frac{1}{2}\sum_{+,-} \Re \int_a^b dx\, e^{\pm i\omega x}\alpha(x)e^{i\gamma(x)} \tag{27}$$

$$\int_a^b dx\, \alpha(x)\cos(\gamma(x))\sin(\omega x) = \frac{1}{2}\sum_{+,-} \pm\Im \int_a^b dx\, e^{\pm i\omega x}\alpha(x)e^{i\gamma(x)} \tag{28}$$

$$\int_a^b dx\, \alpha(x)\sin(\gamma(x))\cos(\omega x) = \frac{1}{2}\sum_{+,-} \Im \int_a^b dx\, e^{\pm i\omega x}\alpha(x)e^{i\gamma(x)} \tag{29}$$

$$\int_a^b dx\, \alpha(x)\sin(\gamma(x))\sin(\omega x) = \frac{1}{2}\sum_{+,-} \mp\Re \int_a^b dx\, e^{\pm i\omega x}\alpha(x)e^{i\gamma(x)}. \tag{30}$$

However, our adaptive algorithm's criterion for recursion to smaller subintervals must be adjusted to take into account that our goal for each $\omega \in \Omega$ is an accurate estimate not of a single complex Fourier integral, but of the real part or imaginary part of the sum of two complex Fourier integrals, a sum that may involve significant cancellation. Our implementation achieves this by using a pseudonorm on $\mathbb{C}^{\Omega\cup-\Omega}$ consisting of a norm (Euclidean by default) on $\mathbb{R}^{\Omega}$ composed with the real or imaginary part operator and with the sum or difference map $\varsigma_{\pm}\colon \mathbb{C}^{\Omega\cup-\Omega} \to \mathbb{C}^{\Omega}$ where $(\varsigma_{\pm}\vec{x})_{\omega} = x_{\omega} \pm x_{-\omega}$.

5. Computation of FCC Quadrature Weights

5.1. **Overview.** The core of our FCC quadrature implementation is an algorithm for accurately computing sequences of weights $\int_{-1}^{1} dx\, e^{i\omega x}T_n(x)$ for $0 \le n \le N$. For this, we numerically solve the three-term recurrence

$$\frac{\omega\tau_{n-1}}{2(n-1)} + \tau_n - \frac{\omega\tau_{n+1}}{2(n+1)} = \frac{2\cos\omega}{1-n^2} \qquad (2 \le n \text{ even}) \tag{31}$$

$$-\frac{\omega\tau_{n-1}}{2(n-1)} + \tau_n + \frac{\omega\tau_{n+1}}{2(n+1)} = \frac{2\sin\omega}{1-n^2} \qquad (3 \le n \text{ odd}) \tag{32}$$

where

$$\tau_{2m} = \int_{-1}^{1} dx\, \cos(\omega x)T_{2m}(x) \tag{33}$$

$$\tau_{2m+1} = \int_{-1}^{1} dx\, \sin(\omega x)T_{2m+1}(x). \tag{34}$$

(This recurrence follows from integration by parts and the identity $T'_{n+1}/(n+1) - T'_{n-1}/(n-1) = 2T_n$.) We use solution method "RR" of [5]: forward recursion where it is numerically stable, for $n \le |\omega|$, and Olver's method [7] where it is numerically stable, for $n > |\omega|$. (The boundary conditions for the forward recursion are $\omega\tau_0 = 2\sin\omega$ and $\tau_0 - \omega\tau_1 = 2\cos\omega$ if $|\omega| \ge 1$.) Our application of Olver's method consists of selecting a padding length P and making the three-term recurrence for $|\omega| < n \le N + P$ a tridiagonal linear system by replacing $\tau_{\lfloor|\omega|\rfloor}$ with its value computed by forward recursion and by approximating τ_{N+P+1} as 0. (If $|\omega| < 1$, we add the boundary condition $4\tau_1 + \omega\tau_2 = 2\sin\omega$.) The resulting tridiagonal system is diagonally dominant and numerically stably solved by the Thomas algorithm in linear time.

5.2. **Padding Length Selection.** How to select the padding length P is not discussed in [5]. We will give strong evidence that $P = 9 + 2\left\lceil(1+\sqrt{74N})/2\right\rceil$ is sufficiently large when using double-precision floating-precision arithmetic, which has unit roundoff 2^{-53}. More precisely, what we will *prove* is that the absolute algorithmic error in our computation of τ_n is at most $2^{-53}\,|\tau_{N+P+1}|$; what we will merely give strong evidence for is that absolute algorithmic error in our computation of τ_n is generically at most $2^{-53}\max(|\tau_n|\,,|\tau_{n+1}|)$ and much smaller when $|\omega| << N$. For comparison, the identity $2xT_n(x) = T_{n-1}(x) + T_{n+1}(x)$ implies

$$d\tau_n/d\omega = (-1)^{n+1}(\tau_{n-1} + \tau_{n+1})/2. \tag{35}$$

In the interesting alternative approach of [4], Olver's method is applied to a three-term recurrence for the integrals $\upsilon_n = \int_{-1}^{1} dx\, e^{i\omega x}T'_n(x)/n$ but υ_{N+P+1} is approximated by an asymptotic expansion instead of by zero. In Experiment 3 of [4], the value of $N + P$ starts at N and is repeatedly multiplied by 3/2 (and

rounded up to the nearest integer) until the last term of their seven-term asymptotic estimate of υ_{N+P+1} is less than 10^{-15}. We prefer our simpler implementation over [4] because the relevant tridiagonal system is very cheap to solve, costing only $O(N)$ elementary arithmetical operations plus the evaluation of $\cos\omega$ and $\sin\omega$.

Towards justifying $P = 9 + 2\left\lceil (1+\sqrt{74N})/2 \right\rceil$, write the three-term recurrence as

$$a_{n-1}\tau_{n-1} + \tau_n - a_{n+1}\tau_{n+1} = b_n \tag{36}$$

where

$$a_n = (-1)^{n+1}\frac{\omega}{2n}; \qquad b_n = \begin{cases} \frac{2\cos\omega}{1-n^2} & : n \text{ even} \\ \frac{2\sin\omega}{1-n^2} & : n \text{ odd} \end{cases}. \tag{37}$$

Letting $s = 1 + \lfloor |\omega| \rfloor \le N \le N+P = t$, we form the tridiagonal system consisting of

$$\hat{\tau}_s - a'_{s+1}\hat{\tau}_{s+1} = b'_s \tag{38}$$

$$a_{n-1}\hat{\tau}_{n-1} + \hat{\tau}_n - a_{n+1}\hat{\tau}_{n+1} = b_n \qquad (s < n < t) \tag{39}$$

$$a_{t-1}\hat{\tau}_{t-1} + \hat{\tau}_t = b_t \tag{40}$$

where $a'_{s+1} = a_{s+1}$ and $b'_s = b_s + a_{s-1}\tau_{s-1}$ if $s > 1$ else $b'_1 = \frac{1}{2}\sin\omega$. Next, we bring the system into upper triangular form

$$\hat{\tau}_n - a'_{n+1}\hat{\tau}_{n+1} = b'_n \qquad (s \le n < t) \tag{41}$$

$$\hat{\tau}_t = b'_t \tag{42}$$

where $a'_n = a_n/(1 + a_{n-2}a'_{n-1})$ for $n \ge s+2$ and $b'_n = (b_n - a_{n-1}b'_{n-1})/(1 + a_{n-2}a'_{n-1})$ for $n \ge s+1$. We then solve the system using back substitution and approximate τ_n by $\hat{\tau}_n$, thus introducing absolute errors according to the recursion

$$|\hat{\tau}_n - \tau_n| = \left|a'_{n+1}\right| |\hat{\tau}_{n+1} - \tau_{n+1}| \tag{43}$$

with boundary condition $\hat{\tau}_{t+1} = 0$. In particular, $|\hat{\tau}_n - \tau_n| = |\tau_{t+1}| \prod_{m=n}^{t} \left|a'_{m+1}\right|$. Our next task is to bound these errors.

Lemma 1. *$|a_m| \le |a'_m| \le 2\,|a_m|$ for all $m \in (s,t]$.*

Proof. Assuming without loss that $\omega \ne 0$, let $c_m = a'_m/2a_m$. It suffices to show that $1/2 \le c_m < 1$. Proceeding by induction on m, the base case $m = s+1$ is clear. For the inductive case where $m \ge s+2$, the recursion $a'_m = a_m/(1 + a_{m-2}a'_{m-1})$ implies

$$\frac{1}{c_m} = \frac{2a_m}{a'_m} = 2(1 + 2a_{m-2}a_{m-1}c_{m-1}) = 2 - \frac{\omega^2 c_{m-1}}{(m-2)(m-1)}. \tag{44}$$

We have $|\omega| < s \le m-2$. Therefore,

$$0 \le c_{m-1} \le 1 \Rightarrow 0 \le \frac{\omega^2 c_{m-1}}{(m-2)(m-1)} < 1 \Rightarrow \frac{1}{2} \le c_m < 1. \tag{45}$$

□

Applying Lemma 1 to the error recursion (43), we have

$$|\hat{\tau}_n - \tau_n| \le \frac{|\omega|}{n+1}\,|\hat{\tau}_{n+1} - \tau_{n+1}| \le |\hat{\tau}_{n+1} - \tau_{n+1}| \tag{46}$$

for each $n \in [s,t]$. Moreover,

$$|\hat{\tau}_n - \tau_n| \le |\tau_{t+1}| \prod_{m=n}^{t} \frac{s}{m+1} \le |\tau_{t+1}| \prod_{m=N}^{N+P} \frac{N}{m+1} = |\tau_{t+1}|\,\frac{N^{P+1}N!}{(N+P+1)!}. \tag{47}$$

Note that if $|\omega| \ll N$, then the above bound exaggerates the absolute error in τ_n by at least a large factor of $(N/|\omega|)^{P+1}$.

Stirling's approximation and $\ln(1+x) = x - x^2/2 + o(x^2)$ are enough to prove the asymptotic estimate

$$\frac{N^{P+1}N!}{(N+P+1)!} \sim \exp\left[-\frac{P^2}{2N}\right] \tag{48}$$

for $P \ll N$. With a little more work, we obtain the following non-asymptotic bound.

Lemma 2.

$$\ln \frac{N^{P+1}N!}{(N+P+1)!} < -\frac{(P+1)(P+2)(N-P-1)}{2N^2}.$$

Proof. Instead of Stirling's approximation, we use Robbins' bounds [8]

$$\frac{1}{12k+1} < \ln(k!) - \left[\left(k+\frac{1}{2}\right)\ln k - k + \ln\sqrt{2\pi}\right] < \frac{1}{12k}. \tag{49}$$

Applying the upper bound to $k = N$ and the lower bound to $k = N+P+1$, we have

$$\ln \frac{N^{P+1}N!}{(N+P+1)!} < P+1 - \left(N+P+\frac{3}{2}\right)\ln\frac{N+P+1}{N} + \frac{12(P+1)+1}{12N(12(N+P+1)+1)} \tag{50}$$

$$< P+1 - \left(N+P+\frac{3}{2}\right)\ln\frac{N+P+1}{N} + \frac{13(P+1)}{(12N)^2}. \tag{51}$$

The estimate $\ln(1+x) > x(1-x/2)$ for $x > 0$ implies

$$P+1 - \left(N+P+\frac{3}{2}\right)\ln\frac{N+P+1}{N} < P+1 - \frac{P+1}{4N^2}(2N+2P+3)(2N-1-P) \tag{52}$$

$$= \frac{P+1}{4N^2}[(2P+3)(P+1) - 2N(P+2)]. \tag{53}$$

Combining this with (51) yields

$$\ln \frac{N^{P+1}N!}{(N+P+1)!} < \frac{P+1}{4N^2}[(2P+3)(P+1) - 2N(P+2) + 13/36] \tag{54}$$

$$< \frac{P+1}{4N^2}[2(P+1)(P+2) - 2N(P+2)] \tag{55}$$

$$= -\frac{(P+1)(P+2)(N-P-1)}{2N^2}.$$

□

Lemma 3. *If $|\omega| < n$ and $p \geq 10 + \sqrt{74n}$, then $\prod_{m=n}^{n+p} |a'_{m+1}| < 2^{-53}$.*

Proof. By Lemma 1, the product $\prod_{m=n}^{n+p} |a'_{m+1}|$ is at most $f(n,p) = \prod_{m=n}^{n+p} \frac{n}{m+1}$. By Lemma 2,

$$\ln f(n,p) < g(n,p) = (p+1)(p+2)(p+1-n)/2n^2. \tag{56}$$

Since $f(n,p)$ is decreasing with respect to p,

$$\ln f(n,p) < g(n, 10+\sqrt{74n}) = h(n) = -37 + \frac{51}{2}\sqrt{\frac{74}{n}} + \frac{1192}{n} + \frac{385}{2n}\sqrt{\frac{74}{n}} + \frac{726}{n^2}. \tag{57}$$

Since h is a decreasing function of n and

$$h(8\times 10^5) = -36.753 < -36.737 = -53\ln 2, \tag{58}$$

the theorem holds for $n \geq 8\times 10^5$. Direct numerical computation verifies the rest. In particular, when $p = 10 + \lceil\sqrt{74n}\rceil$, the maximum value of $\prod_{m=n}^{n+p} \frac{n}{m+1}$ for $n \leq 10^6$ is $2^{-53.14}$ at $n = 296$. □

Lemma 3 implies that if we use some padding P for Olver's method and

$$|\omega| < n \leq n + 10 + \sqrt{74n} \leq n + p \leq N + P, \tag{59}$$

then the approximation error $|\hat{\tau}_n - \tau_n|$ is at most $2^{-53}\,|\hat{\tau}_{n+p+1} - \tau_{n+p+1}|$, which is $2^{-53}\,|\tau_{N+P+1}|$ if also $n+p = N+P$. As discussed earlier, this bound is very loose if $|\omega| \ll N$. For the general case, our goal is to give strong evidence that

$$|\hat{\tau}_n - \tau_n| \leq 2^{-53}\max(|\tau_n|, |\tau_{n+1}|) \tag{60}$$

holds generically.

Therefore, we want to understand if and how $|\tau_n|$ decays as n increases. The three-term recurrence (36) is equivalent to the fixed point equation $\tau_n = (b + A\tau)_n$ where $(Ax)_n = a_{n+1}x_{n+1} - a_{n-1}x_{n-1}$ for $n \geq 2$.

For each $r \geq s$, declare A_r to be the infinite-dimensional operator on vectors of the form $(x_n)_{n\geq r}$ such that $(A_r x)_n = (Ax)_n$ if $n > r$ and $(A_r x)_r = a_{r+1}x_{r+1}$. The l_2 operator norm of A_r satisfies

$$\|A_r\| \leq |a_r| + |a_{r+2}| = \frac{|\omega|\,(r+1)}{r(r+2)} < 1. \tag{61}$$

Therefore,

$$\tau_{\geq r} = (\tau_n)_{n\geq r} = \sum_{k=0}^{\infty} A_r^k c_{\geq r} \tag{62}$$

where $c_n = b_n$ if $n > r$ and $c_r = b_r + a_{r-1}\tau_{r-1}$ if $r > 1$ else $c_1 = b_1'$. In particular,

$$\left|\tau_n - \left[\sum_{k=0}^{n-r-1} A^k b\right]_n\right| = \left|\left[A_r^{n-r}\tau_{\geq r}\right]_n\right| = O\left(\frac{|\omega|\,(r+1)}{r(r+2)}\right)^{n-r}. \tag{63}$$

By induction on k, if $n - 2 \geq k \geq 0$ then

$$(A^k b)_n = \omega^k b_n \left[\prod_{m=2}^{k+1} \frac{2m-1}{n^2-m^2}\right] \cdot \begin{cases} (-1)^{k/2} & : k \text{ even} \\ (-1)^{n+(k-1)/2} & : k \text{ odd} \end{cases}. \tag{64}$$

Therefore, if $r/(r-|\omega|) \ll n - r$, then $|\tau_{n+2}| \leq |\tau_n|$ because

$$n \text{ even } \Rightarrow \tau_n \approx -\frac{2\cos\omega}{n^2-1} - \frac{3\omega\sin\omega}{(n^2-1)(n^2-4)} \tag{65}$$

$$n \text{ odd } \Rightarrow \tau_n \approx -\frac{2\sin\omega}{n^2-1} + \frac{3\omega\cos\omega}{(n^2-1)(n^2-4)}. \tag{66}$$

Empirically, more is true: starting at $n = s$, there is a short transition zone of rapid exponential decay of $|\tau_n|$ followed by an asymptotic regime of decay for even n and decay for odd n according to (65) and (66). Figure 1 illustrates this.

Thus, if $n \in [s, N]$ and P is odd and at least $10 + \sqrt{74n}$, then, choosing $m \in \{n, n+1\}$ such that $N - m$ is even, we have strong evidence that

$$|\hat{\tau}_n - \tau_n| \leq 2^{-53}\,|\tau_{n+P+1}| \leq 2^{-53}\,|\tau_{N+P+1}| \leq 2^{-53}\,|\tau_m| \tag{67}$$

is generically true. Also recall that this bound is larger than the actual absolute error by at least a factor of $(N/|\omega|)^{P+1}$, which is very large if $|\omega| \ll N$.

5.3. **Padding Lengths For Other Precisions.** For a floating-point data format with unit roundoff u other than 2^{-53}, it is not hard to find an analog of the bound $p \geq 10 + \sqrt{74n}$ of Lemma 3. The analog of 74 should be something slightly larger than $-2\ln u$ and the analog of 10 can be determined empirically. In particular, padding length $P = 3 + 2\left\lceil(1 + \sqrt{34N})/2\right\rceil$ is sufficient for single-precision floating-point arithmetic because $u = 2^{-24}$ and if $p = 4 + \lceil\sqrt{34n}\rceil$ then the maximum value of $\prod_{m=n}^{n+p} \frac{n}{m+1}$ for $n \leq 10^6$ is $2^{-24.50}$ at $n = 850$. Similarly, if $u = 2^{-256}$, then padding length $P = 55 + 2\left\lceil\sqrt{356N}/2\right\rceil$ is sufficient because if $p = 55 + \lceil\sqrt{356n}\rceil$ then the maximum value of $\prod_{m=n}^{n+p} \frac{n}{m+1}$ for $n \leq 10^6$ is $2^{-256.22}$ at $n = 4018$.

Figure 2 measures the accuracy of vectors of weights $(\tau_n)_{n\leq N}$ computed using double precision by comparing them to weights computed using 256-bit precision.

6. Empirical Performance

In this section we compare the following four FCC quadrature variants.

(1) Degree: Degree-adaptive[5] FCC quadrature.
(2) Plain: Hybrid[6] adaptive FCC quadrature as in Subsection 2.1 except without tone removal.
(3) Tone: Hybrid adaptive FCC quadrature with tone removal as in Section 2.
(4) Chirp: Hybrid adaptive FCC quadrature with linear chirp removal as in Section 3.

[5]Starting at $N = 8$ and doubling thereafter up to a maximum of 2^{20}.
[6]See Subsection 2.1.

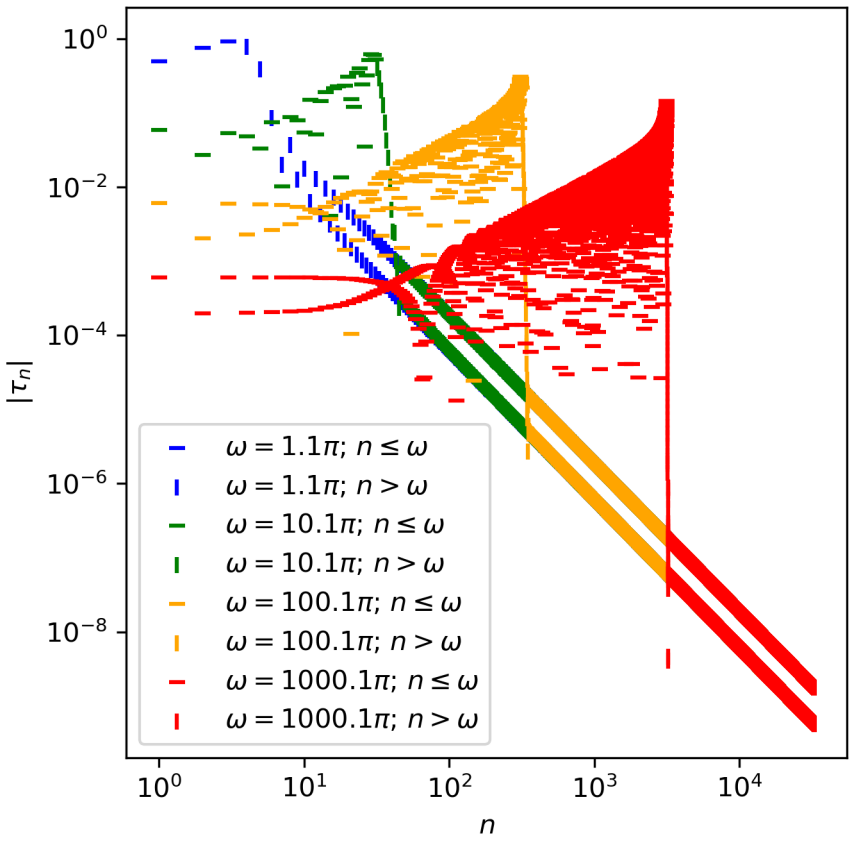

FIGURE 1. Typical relationship between $|\tau_n|$ and n.

TABLE 2. Real-valued integrand factorizations for 7 integrals from [5].

Integral	Integrand Form	$\alpha(x)$	$\gamma(x)$	ω
$I_1 = \int_0^1 dx\ \cos(10x^2)\sin(50x)$	$\alpha(x)\cos\gamma(x)\sin(\omega x)$	1	$10x^2$	50
$I_2 = \int_0^1 dx\ \cos x\cos(40\cos x)$	$\alpha(x)\cos\gamma(x)\cos(\omega x)$	1	$40\cos x$	1
$I_3 = \int_0^1 dx\ \sin x\cos(500(x^2+x))$	$\alpha(x)\cos\gamma(x)\sin(\omega x)$	1	$500(x^2+x)$	1
$I_4 = \int_0^\pi dx\ \cos(30x)\cos(30\cos x)$	$\alpha(x)\cos\gamma(x)\cos(\omega x)$	1	$30\cos x$	30
$I_5 = \int_0^{\pi/2} dx\ \sin x\cos(\cos x)\cos(100\cos x)$	$\alpha(x)\cos\gamma(x)\sin(\omega x)$	$\cos(\cos x)$	$100\cos x$	1
$I_6 = \int_0^2 dx\, e^x \sin(50\cosh x)$	$\alpha(x)\sin\gamma(x)\cos(\omega x)$	e^x	$50\cosh x$	0
$I_7 = \int_0^1 dx\ \cos(47\pi x^2/4)\cos(41\pi x/4)$	$\alpha(x)\cos\gamma(x)\cos(\omega x)$	1	$47\pi x^2/4$	$41\pi/4$

6.1. **Seven Easier Integrals.** We applied our methods to the seven real-valued Fourier integrals of Table 4 of [5].[7] For each of these integrals, the integrand can be factored as in Section 4. Table 2 lists our factorization choices.

[7]We replaced the undefined parameter β of integral I_2 with 1.

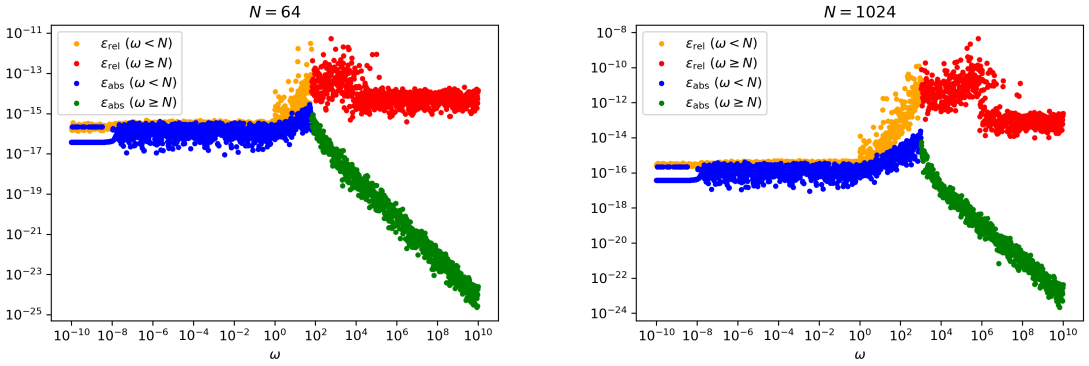

FIGURE 2. Worst relative discrepancy $\varepsilon_{\text{rel}} = \max_{n \leq N} \left| \tau_n^{(53)} / \tau_n^{(256)} - 1 \right|$ and worst absolute discrepancy $\varepsilon_{\text{abs}} = \max_{n \leq N} \left| \tau_n^{(53)} - \tau_n^{(256)} \right|$ between Chebyshev weights computed using double precision and 256-bit precision. (The respective unit roundoffs are 2^{-53} and 2^{-256}.) Each plot shows results for 2,001 values of ω that range from 10^{-10} to 10^{10} with log-uniform spacing. Given an interpolation degree N and angular frequency ω, the vector $(\tau_n)_{n \leq N}$ is computed using forward recursion for $n \leq \omega$ and Olver's method for $n > \omega$. In particular, the discrepancies observed at higher frequencies $\omega \geq N$ are entirely attributable to forward recursion.

TABLE 3. Performance comparison for 7 integrals from [5]. The unit roundoff was 2^{-53} and the relative error goal was 10^{-8}. Timings were performed using a single core of an Intel i7-8750H 2.20GHz CPU and exclude compilation warm up time.

	Integrand Evaluations				Run Time (s)			
Integral	Degree	Plain	Tone	Chirp	Degree	Plain	Tone	Chirp
I_1	65	65	33	9	$10^{-4.2}$	$10^{-4.3}$	$10^{-4.4}$	$10^{-3.5}$
I_2	65	65	33	33	$10^{-4.3}$	$10^{-4.3}$	$10^{-4.3}$	$10^{-2.6}$
I_3	1025	2157	325	9	$10^{-3.5}$	$10^{-3.0}$	$10^{-3.9}$	$10^{-2.1}$
I_4	129	197	197	133	$10^{-4.1}$	$10^{-3.9}$	$10^{-3.9}$	$10^{-2.3}$
I_5	129	261	197	65	$10^{-4.0}$	$10^{-3.9}$	$10^{-3.9}$	$10^{-2.0}$
I_6	257	393	229	65	$10^{-4.0}$	$10^{-4.0}$	$10^{-3.9}$	$10^{-2.0}$
I_7	129	261	197	9	$10^{-4.1}$	$10^{-3.9}$	$10^{-4.0}$	$10^{-2.6}$

Table 3 demonstrates the principle of large efficiency gains through tone or chirp removal. Judging by the integrand evaluation counts, Chirp is best, Tone does about as well as Degree, and Plain is worst. Without tone or chirp removal, restricting the interpolation degrees to $8 \leq N \leq 64$ and dividing the domain into subintervals was typically inferior to simply using a higher interpolation degree. Table 4 verifies that the seven integrals were computed sufficiently accurately by all four methods.

Practically, run time matters more than an evaluation count. For the seven integrands of Table 2, Degree, Plain, and Tone have very similar run times except for one integral where Tone is faster; Chirp is the slowest by a wide margin. Chirp will have a speed advantage over Tone only if the integrand is extremely expensive to evaluate.

6.2. **One Harder Integral.** In contrast to the above seven integrals, if the irregularly oscillatory part of an integrand has much higher typical frequency, then Tone typically has a much greater speed advantage over Plain and Degree. For example, as displayed in Table 5, Tone was 100 times faster than Plain at computing $\int_{12}^{13} dx\, e^{x+ie^x}$ with relative error goal $\delta_{\text{rel}} = 10^{-8}$, using double-precision arithmetic (and $\omega = 0$). The method Degree failed at the same task, estimating its relative error to be $10^{-8.6}$ when it was actually $10^{3.9}$, making its output worse than useless. For Degree, merely increasing working precision to 256 bits did not improve accuracy; merely decreasing δ_{rel} to 10^{-12} improved actual relative error to $10^{-6.6}$ at the cost of a run time 200 times longer than Tone's run time for $\delta_{\text{rel}} = 10^{-8}$ with double-precision arithmetic,

TABLE 4. Accuracy verification for 7 integrals from [5]. The unit roundoff was 2^{-53} and the relative error goal was 10^{-8}. Estimated relative errors were computed as in Subsection 1.4 using $M = 3N/4$. Actual relative errors were computing by comparing to integrals computed using Tone with unit roundoff 2^{-256} and relative error goal $\delta_{\text{rel}} = 10^{-30}$. Nonzero error magnitudes are rounded to the nearest power of 10.

	Estimated Relative Error				Actual Relative Error			
Integral	Degree	Plain	Tone	Chirp	Degree	Plain	Tone	Chirp
I_1	10^{-19}	10^{-19}	10^{-12}	10^{-17}	10^{-15}	10^{-15}	10^{-14}	10^{-15}
I_2	10^{-17}	10^{-17}	10^{-9}	0	10^{-14}	10^{-14}	10^{-13}	10^{-13}
I_3	10^{-17}	10^{-7}	10^{-13}	10^{-14}	10^{-12}	10^{-12}	10^{-13}	10^{-15}
I_4	10^{-15}	10^{-9}	10^{-15}	10^{-9}	10^{-15}	10^{-15}	10^{-15}	10^{-11}
I_5	10^{-9}	10^{-7}	10^{-15}	10^{-12}	10^{-13}	10^{-13}	10^{-13}	10^{-12}
I_6	10^{-17}	10^{-8}	10^{-11}	10^{-8}	10^{-13}	10^{-13}	10^{-13}	10^{-11}
I_7	10^{-19}	10^{-12}	10^{-14}	0	10^{-15}	10^{-15}	10^{-15}	10^{-15}

TABLE 5. Performance and accuracy comparison for the integral $\int_{12}^{13} dx\, e^{x+ie^x}$ including integrand evaluation counts, run times, estimated relative error magnitudes, and actual relative error magnitudes. Timings were performed using a single core of an Intel i7-8750H 2.20GHz CPU and exclude compilation warm up time. Actual errors were computed relative to a value of $i(e^{ie^{12}} - e^{ie^{13}})$ computed with 256-bit precision.

Method	Relative Error Goal	Unit Roundoff	Integrand Evaluations	Run Time (s)	Estimated Relative Error	Actual Relative Error
Plain	10^{-8}	2^{-53}	632053	$10^{-0.9}$	$10^{-5.4}$	$10^{-7.3}$
Plain	10^{-8}	2^{-256}	632053	$10^{1.3}$	$10^{-5.4}$	$10^{-11.5}$
Tone	10^{-8}	2^{-53}	5365	$10^{-2.9}$	$10^{-7.1}$	$10^{-8.8}$
Tone	10^{-8}	2^{-256}	5365	$10^{-0.7}$	$10^{-7.1}$	$10^{-11.5}$
Chirp	10^{-8}	2^{-53}	533	$10^{-0.1}$	$10^{-8.6}$	$10^{-11.2}$
Degree	10^{-8}	2^{-53}	32769	$10^{-1.8}$	$10^{-8.6}$	$10^{3.9}$
Degree	10^{-8}	2^{-256}	32769	$10^{0.4}$	$10^{-8.6}$	$10^{3.9}$
Degree	10^{-12}	2^{-53}	262145	$10^{-0.6}$	$10^{-14.9}$	$10^{-6.6}$
Degree	10^{-12}	2^{-256}	262145	$10^{1.4}$	$10^{-75.8}$	$10^{-67.5}$

which achieved actual relative error $10^{-8.8}$. Using both 256-bit working precision and $\delta_{\text{rel}} = 10^{-12}$ for Degree further improved actual relative error to $10^{-67.5}$ at the cost of increasing run time by another factor of 100. Fundamentally, method Degree suffered from massive cancellation because the Chebyshev series

$$e^{x+ie^x} = \sum_{n=0}^{\infty} c_n T_n(2x - 25) \tag{68}$$

on $[12, 13]$ has an oscillatory coefficient sequence $\vec{c}$ even when restricted to the even terms c_{2n} that contribute to

$$\int_{12}^{13} dx\, e^{x+ie^x} = \sum_{n=0}^{\infty} \frac{c_{2n}}{1 - 4n^2}. \tag{69}$$

The largest coefficient magnitude $|c_{2n}|$ does not occur until $2n$=150,170.

Overall, we recommend Tone as a default method over Degree, Plain, and Chirp.

7. SOFTWARE IMPLEMENTATION

We implemented the algorithms of this paper in the Julia [2] programming language. Our implementation is publicly available as an MIT-licensed Julia package `FCCQuad.jl` at the following URL.

`https://github.com/dkm2/FCCQuad.jl`

7.1. **Supported Data Types.** For the FCC weights computation described in Section 5, we currently support three Julia data types: `Float32`, `Float64`, and `BigFloat`. However, for `BigFloat` we only support the default precision of 256 bits, that is, unit roundoff $u = 2^{-256}$. For each of these data types, the padding length P is as in Section 5.

For numerical integration methods Degree, Plain, and Tone, we currently support the same data types as in the previous paragraph. For the method Chirp, we currently support only Float64.

7.2. **Higher-Degree Automatic Differentiation.** For a univariate scalar-valued analytic functions f, forward-mode automatic differentiation of moderate degree d is both simple and efficient given a type-generic implementation of f in a programming language (for examples, Julia or C++) for which specializations to concrete types can be compiled to machine code. The compiler specializes the implementation of f to a data type representing a ring $K[\varepsilon]/O(\varepsilon^{d+1})$ where K is a field of characteristic zero and the extension of f to this field is defined by the Taylor series expansion

$$f(a + b\varepsilon) = \sum_{n=0}^{d} \frac{f^{(n)}(a)}{n!}(b\varepsilon)^n \tag{70}$$

for all $a \in K$ and $b \in K[\varepsilon]/O(\varepsilon^{d+1})$. If f is an elementary function, there is no more work to be done; libraries implementing the needed primitive operations on $K[\varepsilon]/O(\varepsilon^{d+1})$ are publicly available. One example is the open-source Julia package `TaylorSeries.jl` [1]. On the other hand, if the implementation of f uses, for example, an erf implementation intended only for double-precision floating point, then the user may need to manually extend the definition of f to $K[\varepsilon]/O(\varepsilon^{d+1})$.

Alternatively, if the degree d is moderate, then recursive use of degree-1 automatic differentiation is simpler to implement and only moderately inefficient. For example, working in $L[\varepsilon_2]/O(\varepsilon_2^2)$ where $L = K[\varepsilon_1]/O(\varepsilon_1^2)$, we can extract a second-degree Taylor series:

$$f(x + \varepsilon_1 + \varepsilon_2) = f(x + \varepsilon_1) + f'(x + \varepsilon_1)\varepsilon_2 = f(x) + f'(x)\varepsilon_1 + f'(x)\varepsilon_2 + f''(x)\varepsilon_1\varepsilon_2. \tag{71}$$

Our current implementation uses our own lightweight implementation of $K[\varepsilon]/O(\varepsilon^3)$ for primitive operations and some special functions, including erf, which is not currently supported by `TaylorSeries.jl`. (The popular forward-mode automatic differentiation Julia package `ForwardDiff.jl` supports computing second derivatives. Unfortunately, it currently has poor support for complex numbers.)

Declarations

Ethical Approval and Consent to Participate. Not applicable.

Consent for Publication. Not applicable.

Human and Animal Ethics. Not applicable.

Availability of Supporting Data. The author's implementation of the algorithms of this paper is publicly available at the location given in Section 7. The code used to generate the tables and figures is available upon request to the author.

Competing Interests. The author has no competing interests to declare that are relevant to the content of this article.

Funding. This research was funded by the Defense Threat Reduction Agency (DTRA) under contracts HDTRA1-18-P-0012 and HDTRA1-20-C-0001.

Authors' Contributions. Not applicable.

Acknowledgements. I am grateful to Welkin Sciences and DTRA for giving me the opportunity to tackle the research and design challenges that motivated the algorithms of this paper, and for encouraging me to publish.

References

[1] L. Benet and D.P. Sanders. TaylorSeries.jl: Taylor expansions in one and several variables in Julia. *Journal of Open Source Software*. **4:36** (2019), 1043. Software repository: `https://github.com/JuliaDiff/TaylorSeries.jl`.

[2] J. Bezanson, A. Edelman, S. Karpinski, and V.B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review* **59:1** (2017), 65–98.

[3] R. Dana. *Propagation of RF Signals Through Structured Ionization: The General Model.* Defense Nuclear Agency technical report DNA-TR-90-9 (1991). `https://apps.dtic.mil/docs/citations/ADA234051`.

[4] V. Domínguez, I. Graham, and V. Smyshlyaev. Stability and error estimates for Filon-Clenshaw-Curtis rules for highly-oscillatory integrals. *IMA Journal of Numerical Analysis* **31:4** (2011), 1253–1280. (Preprint available at `http://opus.bath.ac.uk/26655`.)

[5] G.A. Evans and J.R. Webster. A comparison of some methods for the evaluation of highly oscillatory integrals. *Journal of Computational and Applied Mathematics* **112** (1999), 55–69.

[6] D. Milovich, S. Frasier, and B. Sawyer. *The CReG Algorithm.* Welkin Sciences Technical Report WS-R-2020.010 (October 2020).

[7] F.W.J. Olver. Numerical solution of second order linear difference equations. *Journal of Research of the National Bureau of Standards* **71B** (1967) 111–129.

[8] H. Robbins. A Remark on Stirling's Formula. *The American Mathematical Monthly* **62:1** (1955), 26–29.

Email address: `david.milovich@welkinsciences.com`

Welkin Sciences, 2 North Nevada Ave, Suite 1280, Colorado Springs, CO 80903, USA