We introduce an efficient scheme for the construction of quadrature rules for bandlimited functions. While the scheme is predominantly based on well-known facts about prolate spheroidal wave functions of order zero, it has the asymptotic CPU time estimate $O(n \log n)$ to construct an n-point quadrature rule. Moreover, the size of the "$n \log n$" term in the CPU time estimate is small, so for all practical purposes the CPU time cost is proportional to n. The performance of the algorithm is illustrated by several numerical examples.

A fast procedure for the construction of quadrature formulas for bandlimited functions

A. Gopal[†,⋆], V. Rokhlin[†,‡]
Technical Report YALEU/DCS/TR-1563
July 27, 2022

[⋆] This author was supported in part by AFOSR (grant no. FA9550-19-1-0320).
[‡] This author was supported in part by NSF (grant no. DMS-1952751) and ONR (grant no. N00014-14-1-0797).

[†] Dept. of Mathematics, Yale University, New Haven CT 06511

Keywords: *Prolate Spheroidal Wave Functions, Bandlimited Functions, Quadrature.*

1 Introduction

A function $f : \mathbb{R} \to \mathbb{C}$ is said to be *bandlimited* with *bandlimit* $c > 0$ if there exists $\sigma \in L^2([-1,1])$ such that

$$f(x) = \int_{-1}^{1} e^{ictx} \sigma(t)\, dt, \quad x \in \mathbb{R}. \tag{1}$$

In other words, a function is bandlimited if its Fourier transform is compactly supported. Such functions arise in a variety of applications such as fluid dynamics, signal processing, and scattering theory. Typically, only the restriction of f to some compact domain is of interest, such as an interval.

In this regime, the standard techniques of Chebyshev and Legendre interpolation and quadrature can often be effectively employed. However, polynomials are a suboptimal basis for the space of bandlimited functions, and thus, schemes based on polynomial approximation are inherently suboptimal. Indeed, the optimal basis for bandlimited functions on an interval is actually the prolate spheroidal wavefunctions of order zero (PSWFs). The PSWFs were introduced by Slepian, Landau, and Pollack in the context of information theory in [10, 3, 4, 8, 9], and more recently there have been several advances in numerical algorithms for PSWFs (e.g., [12, 7, 6]).

In this paper, we continue this latter line of work and present an efficient and robust numerical algorithm for computing quadrature rules for bandlimited functions on an interval. For any fixed bandlimit $c > 0$, the algorithm computes an n-point quadrature rule in $O(n \log n)$ floating point operations. The nodes of the computed rules are taken to be the roots of appropriately chosen PSWFs, and the weights are chosen to exactly integrate certain collections of PSWFs. The rules are stable and require substantially fewer nodes to integrate bandlimited functions to a prescribed accuracy when compared to classical polynomial quadrature.

This paper is organized as follows. Section 2 summarizes some standard facts needed in this paper. Section 3 contains various mathematical results and numerical techniques used in our algorithm. We describe our algorithm in Section 4, and the results of several numerical experiments are provided in Section 5. Section 6 contains our conclusions.

2 Preliminaries

2.1 Legendre polynomials and functions

The Legendre polynomials $\{P_k\}_{k=0}^{\infty}$ are defined by the three-term recurrence relation

$$P_{k+1}(x) = \frac{2k+1}{k+1} x P_k(x) - \frac{k}{k+1} P_{k-1}(x), \quad k = 1, 2, \ldots, \tag{2}$$

with the initial conditions

$$P_0(x) = 1, \quad P_1(x) = x. \tag{3}$$

Differentiating (2) yields the recurrence relation

$$P_k'(x) = \frac{2k-1}{k} P_{k-1}(x) + \frac{2k-1}{k} x P_{k-1}'(x) - \frac{k-1}{k} P_{k-2}'(x), \quad k = 2, 3, \dots . \tag{4}$$

The Legendre functions of the second kind $\{Q_k\}_{k=0}^{\infty}$ are also defined by the recurrence relation

$$Q_{k+1}(x) = \frac{2k+1}{k+1} x Q_k(x) - \frac{k}{k+1} Q_{k-1}(x), \quad k = 1, 2, \dots, \tag{5}$$

but with the initial conditions

$$Q_0(x) = \frac{1}{2} \log \frac{1+x}{1-x}, \quad Q_1(x) = \frac{x}{2} \log \frac{1+x}{1-x} - 1. \tag{6}$$

For any $z \in \mathbb{C}$ and integer $k \geq 0$, the identity

$$Q_k(z) = \frac{1}{2} \int_{-1}^{1} \frac{P_k(z)}{z-t} \, dt \tag{7}$$

holds ([1, 8.8.3]).

Remark 2.1. *For any integer $k \geq 0$, P_k and Q_k are independent solutions to the ODE*

$$(1-x^2) y''(x) - 2x y'(x) + k(k+1) y(x) = 0, \quad x \in [-1, 1]. \tag{8}$$

2.2 Prüfer transform

Suppose that the function ψ satisfies a second-order ordinary differential equation of the form

$$p(x) \psi''(x) + q(x) \psi'(x) + r(x) \psi(x) = 0, \quad x \in (-1, 1), \tag{9}$$

where p, q, and r are polynomials of degree 2 and p and r are positive. Given a differentiable, positive function $\gamma : \mathbb{R} \to \mathbb{R}$, we can define the function $\theta : \mathbb{R} \to \mathbb{R}$ by the formula

$$\theta(x) = \arctan\left(\frac{1}{\gamma(x)} \frac{p(x) \psi'(x)}{\psi(x)} \right) + m\pi, \tag{10}$$

where m is an arbitrary integer. Substituting (10) into (9) produces the first-order ODE

$$\theta' = -\frac{\gamma}{p} \sin^2(\theta) - \frac{r}{\gamma} \cos^2(\theta) - \left(\frac{\gamma'}{\gamma} + \frac{q-p'}{p} \right) \frac{\sin(2\theta)}{2}, \quad x \in (-1, 1). \tag{11}$$

From (10), it immediately follows that the roots of ψ are the values of $x \in (-1, 1)$ for which $\theta(x) = (k + 1/2)\pi$ for some integer k; similarly, the roots of ψ' are the values of x for which $\theta(x) = k\pi$ for some integer k.

It is convenient to choose

$$\gamma(x) = \sqrt{r(x)p(x)}. \tag{12}$$

Then, (11) simplifies to

$$\theta' = -\sqrt{\frac{r}{p}} - \frac{r'p - p'r + 2rq}{2rp}\frac{\sin(2\theta)}{2}. \tag{13}$$

Noting in (13) that $d\theta/dx < 0$ for all $x \in (-1, 1)$, we take the reciprocal, which yields

$$\frac{dx}{d\theta} = -\left(\sqrt{\frac{r}{p}} + \frac{r'p - p'r + 2rq}{2rp}\frac{\sin(2\theta)}{2}\right)^{-1}. \tag{14}$$

Observation 2.2. *Given x_0, a root of ψ, the ODE* (14) *can be used to determine all other roots by solving* (14) *subject to the initial condition*

$$x\left(\frac{\pi}{2}\right) = x_0 \tag{15}$$

and evaluating the solution at half-integer multiples of π. Alternatively, if x_0', a root of ψ', is available, the initial condition

$$x\left(0\right) = x_0' \tag{16}$$

can be used instead.

2.3 Taylor series method for ODEs

For any integer $k \geq 0$, differentiating (9) k times produces the recurrence relation

$$\begin{aligned} p\psi^{(k+2)} = &-(kp' + q)\psi^{(k+1)} - \left(\frac{k(k-1)}{2}p'' + kq' + r\right)\psi^{(k)} \\ &- \left(\frac{k(k-1)}{2}q'' + kr'\right)\psi^{(k-1)} - \frac{k(k-1)}{2}r''\psi^{(k-2)}, \end{aligned} \tag{17}$$

where we define $\psi^{(-2)}(x) = 0$ and $\psi^{(-1)}(x) = 0$. Given a point $x_* \in (-1, 1)$ and the values of $\psi(x_*)$ and $\psi'(x_*)$, higher-order derivatives of ψ at x_* can be evaluated using (17). Fixing an integer $P \geq 2$ and a sufficiently small $h > 0$, ψ and ψ' can then be evaluated at $x_* + h$ using the Taylor series

$$\psi(x_* + h) = \sum_{k=0}^{P} \frac{\psi^{(k)}(x_*)}{k!}h^k + O(h^{P+1}), \tag{18}$$

and

$$\psi'(x_* + h) = \sum_{k=0}^{P-1} \frac{\psi^{(k+1)}(x_*)}{k!}h^k + O(h^P). \tag{19}$$

This provides a method for solving the initial value problem (9) with the initial conditions

$$\psi(x_0) = \psi_0, \quad \psi'(x_0) = \psi'_0. \tag{20}$$

To determine the values of ψ and ψ' at the next time step x_1, the recurrence relation (17) is used to compute $\{\psi^{(k)}(x_0)\}_{k=2}^{P}$, and the series (18) and (19) with $x_* = x_1$ and $h = x_1 - x_0$ are used to evaluate $\psi(x_1)$ and $\psi'(x_1)$. This process is repeated for each time step.

2.4 Prolate spheroidal wave functions

Basic properties

For a fixed *bandlimit* $c > 0$, prolate spheroidal wave functions of order zero with bandlimit c (PSWFs) $\{\psi_j\}_{j=0}^{\infty}$ are eigenfunctions of the truncated Fourier transform, defined by the formula

$$F_c[\sigma](x) = \int_{-1}^{1} e^{ictx}\sigma(t)\,dt, \quad x \in [-1, 1]. \tag{21}$$

The operator F_c in (21) is viewed as acting on $L^2([-1,1])$. The PSWFs can be chosen to be real-valued, and we normalize them so that $\|\psi_j\|_2 = 1$ (here, $\|f\|_2$ denotes the $L^2([-1,1])$-norm of f). Moreover, they constitute an orthogonal basis for $L^2([-1,1])$.

Since the PSWFs are analytic on $[-1,1]$, it follows that if ψ_n is expanded into a Legendre series of the form

$$\psi_n(x) = \sum_{k=0}^{\infty} \alpha_k P_k(x), \quad x \in [-1, 1], \tag{22}$$

where P_k is the Legendre polynomial of degree k (see Section 2.1), then the coefficients $\{\alpha_k\}_{k=0}^{\infty}$ decay superalgebraically. Similarly, ψ'_n can also be expanded into the series

$$\psi'_n(x) = \sum_{k=1}^{\infty} \alpha_k P'_k(x), \quad x \in [-1, 1], \tag{23}$$

where $\{P'_k\}_{k=1}^{\infty}$ can be evaluated using (4).

The corresponding eigenvalues, $\{\lambda_j\}_{j=0}^{\infty}$, are simple, and we use the standard ordering convention so that

$$|\lambda_0| \geq |\lambda_1| \geq \ldots \geq 0, \tag{24}$$

which results in ψ_j being odd and even for odd and even j, respectively. For large c, there are about $2c/\pi$ eigenvalues with modulus close to $\sqrt{2\pi/c}$, and the remaining eigenvalues decay superalgebraically with respect to the index (cf. [6, Theorem 2.4]).

The PSWFs are also the eigenfunctions of a differential operator. In particular, the function ψ_n satisfies the equation

$$(1 - x^2)\psi''_n(x) - 2x\psi'_n(x) + (\chi_n - c^2x^2)\psi_n(x) = 0, \quad x \in (-1, 1) \tag{25}$$

for some $\chi_n > 0$. Note that $\{\chi_j\}_{j=1}^{\infty}$ is a positive, strictly increasing sequence.

For a comprehensive overview of the properties of PSWFs, we refer the reader to [6].

Recurrence relations for higher-order derivatives

For any integer $k \geq 0$, differentiating (25) k times yields the recurrence relation

$$\begin{aligned}(1-x^2)\psi_n^{(k+2)}(x) = 2x(1+k)\psi_n^{(k+1)}(x) + (k^2+k+c^2x^2-\chi_n)\psi_n^{(k)}(x) \\ + 2c^2kx\psi_n^{(k-1)}(x) + c^2k(k-1)\psi_n^{(k-2)}(x),\end{aligned} \tag{26}$$

where $\psi_n^{(-1)}(x) = 0$ and $\psi_n^{(-2)}(x) = 0$.

Observation 2.3. *The recurrence relation* (26) *provides a way of evaluating ψ_n and ψ_n' in the vicinity of a point x_* where their values are given, using the approach outlined in Section 2.3. Specifically, for an integer $P \geq 2$,* (26) *is used to evaluate $\{\psi_n^{(k)}(x_*)\}_{k=2}^{P}$, and then* (18) *and* (19) *are used with $\psi = \psi_n$ to evaluate ψ_n and ψ_n' at points near x_*.*

Prüfer transform

The PSWF equation (25) has the Prüfer transform

$$\frac{d\theta}{dx} = -\sqrt{\frac{\chi_n - c^2x^2}{1-x^2}} + \frac{c^2x - 2c^2x^3 + x\chi_n}{2c^2x^4 + 2\chi_n - 2c^2x^2 - 2x^2\chi_n}\sin(2\theta). \tag{27}$$

Then,

$$\theta(x) = \arctan\left(\sqrt{\frac{1-x^2}{\chi_n - c^2x^2}}\frac{\psi_n'(x)}{\psi_n(x)}\right) + m\pi \tag{28}$$

for an arbitrary integer m. The corresponding analog of (14) is

$$\frac{dx}{d\theta} = -\left(\sqrt{\frac{\chi_n - c^2x^2}{1-x^2}} - \frac{c^2x - 2c^2x^3 + x\chi_n}{2c^2x^4 + 2\chi_n - 2c^2x^2 - 2x^2\chi_n}\sin(2\theta)\right)^{-1}. \tag{29}$$

2.5 Sturm bisection

The following elementary results are the basis of the root-finding technique known as Sturm bisection and can be found on pages 227–229 of [11], for example.

Lemma 2.4. *Let $T \in \mathbb{R}^{n\times n}$ be a symmetric tridiagonal matrix, and let $T_r \in \mathbb{R}^{r\times r}$ denote the rth leading principal submatrix of T. Moreover, let p_r denote the characteristic polynomial of T_r, and define $p_{-1}(x) = 0$ and $p_0(x) = 1$. Then,*

$$p_k(x) = (T(r-1,r-1) - x)p_{r-1}(x) - T(r-1,r-2)^2 p_{r-2}(x), \quad r = 2,3,\dots,n, \tag{30}$$

where we have zero-indexed T.

Theorem 2.5. *Let $T \in \mathbb{R}^{n\times n}$ be a symmetric tridiagonal matrix with no zero subdiagonal entries, and let $T_r \in \mathbb{R}^{r\times r}$ denote the rth leading principal submatrix of T. In addition, let p_r denote the characteristic polynomial of T_r and define $p_0(x) = 1$. Let $\rho(x)$ denote the number of sign changes in the sequence*

$$p_0(x), p_1(x), \ldots, p_n(x) \tag{31}$$

with the convention that $p_r(x)$ has the opposite sign of $p_{r-1}(x)$ if $p_r(x) = 0$. Then, $\rho(x)$ equals the number of eigenvalues of T less than x.

3 Analytical and numerical apparatus

3.1 Computing PSWFs

The following theorem states that the coefficients in the Legendre expansion of ψ_n can be obtained by solving an eigenvalue problem. We define the infinite, symmetric tridiagonal matrices

$$A_{\text{even}}(k,k) = 2k(2k+1) + \frac{4k(2k+1)-1}{(4k+3)(4k-1)}c^2 \tag{32}$$

$$A_{\text{even}}(k,k+1) = A_{\text{even}}(k+1,k) = \frac{(2k+2)(2k+1)}{(4k+3)\sqrt{(4k+1)(4k+5)}}c^2, \tag{33}$$

and

$$A_{\text{odd}}(k,k) = (2k+1)(2k+2) + \frac{(4k+2)(2k+2)-1}{(4k+5)(4k+1)}c^2 \tag{34}$$

$$A_{\text{odd}}(k,k+1) = A_{\text{odd}}(k+1,k) = \frac{(2k+3)(2k+2)}{(4k+5)\sqrt{(4k+3)(4k+7)}}c^2 \tag{35}$$

for $k = 0, 1, \ldots$.

The following theorem is the basis for the algorithm in [12] and can be found in a slightly different form as Theorem 4.1 in [12].

Theorem 3.1. *The jth smallest eigenvalue of A_{even} is χ_n, where $n = 2j$. Moreover, the corresponding eigenvector $\beta = \{\beta_k\}_{k=0}^{\infty}$ satisfies*

$$\alpha_k = \begin{cases} \beta_{k/2}\sqrt{k+\frac{1}{2}} & k \text{ is even} \\ 0 & k \text{ is odd} \end{cases}, \quad k = 0, 1, \ldots \tag{36}$$

where $\{\alpha_k\}_{k=0}^{\infty}$ is given in (22). Similarly, the jth smallest eigenvalue of A_{odd} is χ_n, where $n = 2j+1$, and the corresponding eigenvector $\beta = \{\beta_k\}_{k=0}^{\infty}$ satisfies

$$\alpha_k = \begin{cases} 0 & k \text{ is even} \\ \beta_{(k-1)/2}\sqrt{k+\frac{1}{2}} & k \text{ is odd} \end{cases}, \quad k = 0, 1, \ldots. \tag{37}$$

Observation 3.2. *In practice, the matrices A_{even} and A_{odd} must of course be truncated before being diagonalized. We find that to compute ψ_n for a given integer $n \geq 0$ to full quadruple precision accuracy, it suffices to take the $M \times M$ leading principal submatrix of the relevant matrix, where $M = \lceil 1.1c + n + 1000 \rceil$.*

3.2 Computing eigenvalues of the truncated Fourier transform

The following theorem provides a way of computing an eigenvalue of the truncated Fourier transform, if the Legendre series expansion of the respective PSWF is available. It can be found as Theorem 3.27 in [6].

Theorem 3.3. *Suppose that ψ_n is a PSWF with Legendre expansion given by* (22)*. Then,*

$$\lambda_n = \frac{2\alpha_0}{\psi_n(0)} \tag{38}$$

if n is even, and

$$\lambda_n = \frac{2ci\alpha_1}{3\psi_n'(0)} \tag{39}$$

if n is odd.

Observation 3.4. *Theorem 3.3 shows that computing λ_n for large n requires computing small entries of the eigenvector of a tridiagonal matrix. At first glance, it seems that λ_n can only be computed using* (38) *and* (39) *when $|\lambda_n|$ is above the working precision. However, it is shown in [5] that for a certain class of symmetric tridiagonal matrices, which includes the matrices A_{odd} and A_{even}, the entries of the eigenvectors can be computed to high relative accuracy using, for example, Rayleigh quotient iteration. This requires modifying the convergence criterion to track the convergence of the small entries of the eigenvector, in addition to the eigenvalue. This observation is crucial to Algorithm 5 below.*

3.3 Computing eigenvalues of the differential operator

For any integer $n \geq 0$, χ_n can be computed using Theorem 3.1. We let $M = \lceil 1.1c + n + 1000 \rceil$ and compute the matrix $T \in \mathbb{R}^{M \times M}$, consisting of the $M \times M$ leading principal submatrix of A_{even}, if n is even, or A_{odd}, if n is odd.

We then perform Sturm bisection as described below. First, we let $m = \lfloor n/2 \rfloor$, $a = 0$, and $b = (1 + 2n)c$. After this, we compute $\rho(b)$ where ρ is defined in Theorem 2.5, using Lemma 2.4. If $\rho(b) \leq m - 1$, we set b to twice its value, and again check if $\rho(b) \leq m - 1$. This is repeated until $\rho(b) \geq m$.

We then set $d = (a+b)/2$ and evaluate $\rho(d)$. If $\rho(d) \leq m-1$, we set a to d, and if $\rho(d) \geq m$, we set b to d. This procedure is continued until $\rho(a) = m - 1$ and $\rho(b) = m$, and $b - a$ is below the desired accuracy. At this point, we can take $\chi_n = (b + a)/2$.

3.4 Root-finding for special functions

We use a scheme introduced in [2] due to its linear complexity with respect to the number of roots, favorable constants, and ability to attain high accuracy. When an expansion of the form (22) is available, the scheme determines all roots, one at a time, starting with an initial root.

Determining the first root

If n is odd, then 0 is a root of ψ_n. We evaluate $\psi_n'(0)$ using the expansion (23), and then move onto determining subsequent roots.

If n is even, then $\psi_n'(0) = 0$. Thus, a crude approximation to the smallest positive root of ψ_n can be found by solving the ODE (29) subject to the initial condition

$$x(0) = 0 \tag{40}$$

from 0 to $-\pi/2$. In our implementation, we use a second-order Runge–Kutta scheme; any other low-order, single-step scheme could be used. Let $\widetilde{x}^{(0)}$ be the approximation resulting from evaluating the solution at $-\pi/2$. We can now refine it using the standard Newton iteration

$$\widetilde{x}^{(k)} = \widetilde{x}^{(k-1)} - \frac{\psi_n(\widetilde{x}^{(k-1)})}{\psi_n'(\widetilde{x}^{(k-1)})}, \quad k = 1, 2, \ldots. \tag{41}$$

Naturally, this requires evaluating ψ_n and ψ_n' in the vicinity of $\widetilde{x}^{(0)}$. To do this, we first evaluate $\psi_n(0)$ using (22), and then determine higher-order derivatives using the recurrence (26). After this, we use the Taylor series (18) and (19) with $\psi = \psi_n$ and $x_* = 0$ to evaluate ψ_n and ψ_n' in (41). It was determined in [2] that 30 and 60 terms in the Taylor series suffice to get full double and quadruple precision, respectively. After the Newton iteration has converged, the value of ψ_n' at the newly obtained root is determined using its Taylor series.

Determining subsequent roots

When a root x_m is known, as well as the value of $\psi_n'(x_m)$, the next largest root, x_{m+1}, is determined using a two-step procedure. In the first step, (29) is solved subject to the initial condition

$$x\left(\frac{\pi}{2}\right) = x_m \tag{42}$$

from $\pi/2$ to $-\pi/2$. Again, we use a second-order Runge–Kutta method, and $\widetilde{x}^{(0)}$, an approximation to the true root, is obtained by evaluating the solution at $-\pi/2$. In the second step, higher-order derivatives at x_m are obtained using (26). Then, the Newton iteration (41) is used to refine the approximation to the root, where ψ_n and ψ_n' are evaluated using (18) and (19) with $\psi = \psi_n$ and $x_* = x_m$. After x_{m+1} is obtained, $\psi_n'(x_{m+1})$ is evaluated using the Taylor series.

We repeat this procedure to determine all positive roots and then use symmetry to determine negative roots.

3.5 Quadratures for bandlimited functions

It turns out that the Euclidean algorithm for polynomial division has an analog for bandlimited functions. This is summarized by the following result, which is Theorem 6.2 in [12].

Theorem 3.5. *Suppose that $\sigma, \phi : [-1,1] \to \mathbb{C}$ are a pair of C^2-functions such that $\phi(1) \neq 0$ and $\phi(1) \neq 0$, and the functions f and p are defined by the formulas*

$$f(x) = \int_{-1}^{1} \sigma(t) e^{2icxt}\, dt \tag{43}$$

$$p(x) = \int_{-1}^{1} \phi(t) e^{icxt}\, dt. \tag{44}$$

Then there exist two C^1-functions $\eta, \xi : [-1,1] \to \mathbb{C}$ such that

$$f(x) = p(x)q(x) + r(x), \quad x \in \mathbb{R}, \tag{45}$$

with the functions $q, r : [0,1] \to \mathbb{R}$ defined by the formulas

$$q(x) = \int_{-1}^{1} \eta(t) e^{icxt}\, dt \tag{46}$$

$$r(x) = \int_{-1}^{1} \xi(t) e^{icxt}\, dt. \tag{47}$$

It follows from this result that for sufficiently large n, if the roots of ψ_n are chosen as quadrature nodes and the weights are chosen such that the rule integrates $\{\psi_k\}_{k=0}^{n-1}$ exactly, the rule will accurately integrate functions with bandlimit up to $2c$. This is made rigorous by the following theorem which can be found in a slightly different form Theorem 6.3 in [12].

Theorem 3.6. *Suppose $n > 0$ is an integer, and let $\{x_j\}_{j=1}^n$ denote the roots of ψ_n. Suppose that the quadrature rule $\{(w_j, x_j)\}_{j=1}^n$ integrates exactly the first n PSWFs of bandlimit c, $\{\psi_k\}_{k=0}^{n-1}$. Then, we have the following bound for every function $f : [-1,1] \to \mathbb{C}$ that satisfies the conditions of Theorem 3.5:*

$$\left| \sum_{j=1}^{n} w_j f(x_j) - \int_{-1}^{1} f(x)\, dx \right| \leq |\lambda_n| \|\eta\|_2 + \|\xi\|_2 \sum_{k=n}^{\infty} |\lambda_k| \|\psi_k\|_\infty^2 \left(2 + \sum_{j=1}^{n} |w_j| \right), \tag{48}$$

where the functions $\eta, \xi : [-1,1] \to \mathbb{C}$ are defined in Theorem 3.5 with $\phi = \psi_n$ and $\|\psi_k\|_\infty$ denotes the $L^\infty([-1,1])$-norm of ψ_k.

3.6 Quadrature weights

Based on Theorem 3.6, we choose the roots $\{x_j\}_{j=1}^n$ of ψ_n to be the quadrature nodes. We will order the roots so that $\{x_j\}_{j=1}^n$ is increasing. In this and the subsequent section, we describe a scheme for efficiently computing the corresponding quadrature weights $\{w_j\}_{j=1}^n$ such that the rule $\{(w_j, x_j)\}_{j=1}^n$ integrates $\{\psi_k\}_{k=0}^{n-1}$ accurately. For this, we follow the approach in Section 9.4 of [6], which chooses the weights based on a certain partial fractions approximation of the PSWFs.

We first define the functions

$$\varphi_j(x) = \frac{\psi_n(x)}{\psi_n'(x_j)(x - x_j)}, \quad j = 1, 2, \dots, n, \tag{49}$$

and then choose weights

$$w_j = \int_{-1}^{1} \varphi_j(x)\, dx, \quad j = 1, 2, \dots, n. \tag{50}$$

We will use

$$\{(w_j, x_j)\}_{j=1}^n \tag{51}$$

as a quadrature rule.

It turns out that for sufficiently large n, the rule (51) will also integrate $\{\psi_k\}_{k=0}^{n-1}$ to high accuracy. This is summarized in the following theorem, which is found in a slightly differnt form in Theorem 9.13 in [6].

Theorem 3.7. *Let $c > 60$ and $\varepsilon \in (0, 1)$ be fixed, and let $n > 0$ and $0 \le k < n$ be integers, such that*

$$n > \frac{2c}{\pi} + \left(10 + \frac{3}{2}\log c + \frac{1}{2}\log\frac{1}{\varepsilon}\right)\log\left(\frac{c}{2}\right). \tag{52}$$

Then

$$\left| \int_{-1}^{1} \psi_k(x)\, dx - \sum_{j=1}^{n} w_j \psi_k(x_j) \right| < \varepsilon. \tag{53}$$

Remark 3.8. *We note that the choice of weights in (50) has a clear parallel to Gauss–Legendre quadrature. If $\{x_j\}_{j=1}^n$ are the roots of P_n, the corresponding weights in the Gauss–Legendre rule are chosen as the integrals of the Lagrange basis. To be precise,*

$$w_j = \int_{-1}^{1} \ell_j(x)\, dx, \quad j = 1, 2, \dots, n, \tag{54}$$

where

$$\ell_j(x) = \frac{P_n(x)}{P_n'(x_j)(x - x_j)}, \quad j = 1, 2, \dots, n. \tag{55}$$

We note the similarities between (55) and (49).

3.7 Computing the quadrature weights

It follows from Theorem 3.7 that for sufficiently large n, the rule (51) integrates $\{\psi_k\}_{k=0}^{n-1}$ virtually exactly. Let ψ_n have a Legendre expansion given by (22). Using the identity (7), it follows that for $j = 1, 2, \ldots, n$,

$$w_j = \int_{-1}^{1} \varphi_j(x)\,dx = \frac{1}{\psi_n'(x_j)} \int_{-1}^{1} \frac{\psi_j(x)}{x - x_j}\,dx = -\frac{2}{\psi_n'(x_j)} \sum_{k=0}^{\infty} \alpha_k Q_k(x_j) = -\frac{2\Psi_n(x_j)}{\psi_n'(x_j)}, \tag{56}$$

where

$$\Psi_n(x) = \sum_{k=0}^{\infty} \alpha_k Q_k(x), \tag{57}$$

where Q_k denotes the Legendre function of the second kind of index k (see Section 2.1). Since $\{\psi_n'(x_j)\}_{j=1}^{n}$ is determined during the root-finding, computing the weights $\{w_j\}_{j=1}^{n}$ is essentially equivalent to evaluating the series (57) at $x_1, x_2, \ldots, x_n$. An efficient mechanism for this is given in the following theorem, which can be found in a slightly different form as Theorem 9.15 in [6].

Theorem 3.9. *Let Ψ_n be defined as in* (57)*. Then Ψ_n satisfies the ODE*

$$(1 - x^2)\Psi_n''(x) - 2x\Psi_n'(x) + (\chi_n - c^2x^2)\Psi_n(x) = -c^2\alpha_0 x - \frac{c^2\alpha_1}{3}, \quad x \in (-1, 1). \tag{58}$$

Observation 3.10. *Theorem 3.9 offers a straightforward way to evaluate Ψ. We first note that repeated differentiation of* (58) *yields the equations*

$$(1-x^2)\Psi_n'''(x) - 4x\Psi_n''(x) + (\chi_n - c^2x^2 - 2)\Psi_n'(x) - 2c^2x\Psi_n(x) = -c^2\alpha_0^{(n)}, \quad x \in (-1, 1), \tag{59}$$

and for any integer $k \geq 2$

$$\begin{aligned}(1 - x^2)\Psi_n^{(k+2)}(x) &= 2x(1 + k)\Psi_j^{(k+1)}(x) + (k^2 + k + c^2x^2 - \chi_n)\Psi_n^{(k)}(x) \\ &\quad + 2c^2kx\Psi_n^{(k-1)}(x) + c^2k(k-1)\Psi_n^{(k-2)}(x), \quad x \in (-1, 1).\end{aligned} \tag{60}$$

We then let $m = \lfloor n/2+1 \rfloor$, so that x_m is the smallest non-negative root of ψ_n, and determine the value of $\Psi_n(x_m)$ and $\Psi_n'(x_m)$ using (57)*. We then determine the values $\{\Psi_n(x_j)\}_{j=m+1}^{n-4}$ by solving the ODE* (58) *using the Taylor series method described in Section 2.3 and the recurrence relations* (59) *and* (60)*. Again, 30 and 60 terms in the Taylor series suffice to get full accuracy in double and quadruple precision, respectively. The Taylor series method loses accuracy near $x = 1$. As such, we evaluate $\{\Psi_n(x_j)\}_{j=n-3}^{n}$ directly using* (57)*. The values $\{\Psi_n(x_j)\}_{j=1}^{m}$ are evaluated using symmetry.*

4 Algorithm

4.1 Overview of the algorithm

Given a bandlimit $c > 0$ and a sufficiently large integer $n > 0$, our goal is to compute the n-point quadrature rule (51). As intermediate steps, χ_n in (25) and $\{\alpha_j\}_{j=0}^{N}$ in (22) are computed. Moreover, the corresponding eigenvalue λ_n of the operator F_c in (21) is computed to high accuracy, which is useful since it appears in various error bounds (e.g., see Theorems 9.6 and 9.8 in [6]).

Obtaining an approximation to $\chi_{\mathbf{n}}$. We first compute $\widetilde{\chi}_n$ such that

$$|\widetilde{\chi}_n - \chi_n| < \max_{m \neq n} |\widetilde{\chi}_n - \chi_m|. \tag{61}$$

We let $M = \lceil 1.1c + n + 1000 \rceil$ (see Observation 3.2) and compute the leading $M \times M$ leading principal submatrix of A_{even} using (32) and (33) if n is even and A_{odd} using (34) and (35) if n is odd. Since these matrices are tridiagonal, only $O(n)$ entries need to be computed. We then perform Sturm bisection as described in Section 3.3. Terminating the bisection when (61) and $|\widetilde{\chi}_n - \chi_n| < 2^{-40}$ works well in practice. While this requires a total number of operations that scales proportionately to $n \log n$, the associated constant is very small.

Computing $\{\alpha_{\mathbf{j}}\}_{\mathbf{j=0}}^{\mathbf{N}}$. Next, $\widetilde{\chi}_n$ is used as the initial shift for Rayleigh quotient iteration on the leading $(2n+4) \times (2n+4)$ leading principal submatrix of A_{odd} if n is odd and A_{even} if n is even. The standard theory for Rayleigh quotient iteration (cf. [11, Theorem 27.3]) states that convergence to the corresponding eigenvector $\beta = \{\beta_j\}_{j=0}^{2n+3}$ occurs in $O(n)$ operations. It is critical that the Rayleigh quotient iteration should be terminated based on both convergence to the eigenvalue and small entries of the eigenvector (see Observation 3.4 and Algorithm 5).

The coordinates of β provide the coefficients $\{\alpha_j\}_{j=0}^{4n+7}$ in the Legendre expansion (22), and the corresponding eigenvalue is χ_n (see Theorem 3.1). We choose N to be the smallest integer such that $|\alpha_k| < \varepsilon_{\text{mach}}$ for $k > N$ where $\varepsilon_{\text{mach}}$ is machine precision and only keep the coefficients $\{\alpha_j\}_{j=0}^{N}$.

Determining $\{\mathbf{x_j}\}_{\mathbf{j=1}}^{\mathbf{n}}$ and $\{\psi'_{\mathbf{n}}(\mathbf{x_j})\}_{\mathbf{j=1}}^{\mathbf{n}}$. Once $\{\alpha_j\}_{j=0}^{N}$ is determined, the roots $\{x_j\}_{j=1}^{n}$ of ψ_n and the derivative of ψ_n at the roots $\{\psi'_n(x_j)\}_{j=1}^{n}$ are obtained via the algorithm of Section 3.4. As explained in Section 3.4, this entire procedure requires $O(n)$ operations.

Computing $\{\mathbf{w_j}\}_{\mathbf{j=1}}^{\mathbf{n}}$. Once $\{x_j\}_{j=1}^{n}$ and $\{\psi'_n(x_j)\}_{j=1}^{n}$ are available, the quadrature weights are determined via the scheme in Observation 3.10. Again, this requires $O(n)$ operations.

Computing $\lambda_{\mathbf{n}}$. Given $\{\alpha_j\}_{j=0}^{N}$, the eigenvalue λ_n is evaluated in $O(n)$ operations using Theorem 3.3 with $\psi_n(0)$ and $\psi'_n(0)$ evaluated using (22) and (23), respectively. Per Observation 3.4, the result is computed to high relative accuracy.

4.2 Detailed description of the algorithm

Algorithm 1 below computes an approximation $\widetilde{\chi}_n$ to χ_n using Sturm bisection. This is then used in Algorithm 2, which computes the true value of χ_n and $\{\alpha_j\}_{j=0}^{N}$. These quantities are the input of Algorithm 3, which computes $\{x_j\}_{j=1}^{n}$ and $\{\psi'_n(x_j)\}_{j=1}^{n}$. Algorithm 4 computes $\{w_j\}_{j=1}^{n}$, and Algorithm 5 is used to compute λ_n to high relative accuracy.

Algorithm 1 (Compute $\widetilde{\chi}_n$)**.**
Input: c, n
Output: $\widetilde{\chi}_n$

Step 1. *Let* $M = \lceil 1.1c + n + 1000 \rceil$ *and compute the entries on the diagonal and subdiagonal of the* $M \times M$ *leading principal submatrix of the matrix* T, *where* $T = A_{\text{even}}$ *using* (32) *and* (33) *if* n *is even, and* $T = A_{\text{odd}}$ *using* (34) *and* (35) *if* n *is odd.*

Step 2. *Determine an upper bound on* χ_n *using the function* ρ, *as defined in Theorem 2.5:*

- *(a) Set* $a := 0$ *and* $b := (1 + 2n)c$, *and let* $m = \lfloor n/2 \rfloor$.
- *(b) Evaluate* $\rho(b)$ *using Lemma 2.4.*
- *(c) If* $\rho(b) \geq m$, *continue to Step 3. If not, set* $a := b$ *and* $b := 2b$ *and go back to (a).*

Step 3. *Perform bisection on* ρ:

- *(a) Set* $d := (a + b)/2$ *and evaluate* $\rho(d)$ *using Lemma 2.4.*
- *(b) If* $\rho(d) \leq m - 1$, *set* $a := d$. *If* $\rho(d) \geq m$, *then set* $b := d$.
- *(c) If converged, move to Step 4. Otherwise, go back to (b).*

Step 4. *Set* $\widetilde{\chi}_n = (a + b)/2$.

Algorithm 2 (Compute $\{\alpha_j\}_{j=0}^{N}$)**.**
Input: c, n, $\widetilde{\chi}_n$
Output: χ_n, $\{\alpha_j\}_{j=0}^{N}$

Step 1. *Let* $M = 2n + 4$ *and compute the entries on the diagonal and subdiagonal of the* $M \times M$ *leading principal submatrix of the matrix* T, *where* $T = A_{\text{even}}$ *using* (32) *and* (33) *if* n *is even, and* $T = A_{\text{odd}}$ *using* (34) *and* (35) *if* n *is odd.*

Step 2. *Perform Rayleigh quotient iteration on* T *using* $\widetilde{\chi}_n$ *as the initial approximation to the eigenvalue:*

- *(a) Set* $\lambda := \widetilde{\chi}_n$ *and form a random, unit vector* $\beta \in \mathbb{R}^M$.
- *(b) Compute* $x := (T - \lambda I)^{-1}\beta$, *overwrite* $\beta := x/\|x\|$, *and set* $\widetilde{\lambda} := \beta^T T \beta$.
- *(c) If converged (see Observation 3.4), set* $\chi_n = \widetilde{\lambda}$ *and move to Step 3. If not, set* $\lambda := \widetilde{\lambda}$, *and go back to (b).*

Step 3. *Compute* $\{\alpha_j\}_{j=0}^{2M-1}$ *using* (36) *and* (37), *if* n *is even and odd, respectively. Find the smallest* N *such that* $|\alpha_k| < \varepsilon_{\text{mach}}$ *for all* $k > N$, *where* $\varepsilon_{\text{mach}}$ *is machine precision, and return* χ_n *and* $\{\alpha_j\}_{j=0}^{N}$.

Algorithm 3 (Determine $\{x_j\}_{j=1}^n$ and $\{\psi_n'(x_j)\}_{j=1}^n$)**.**
Input: c, n, $\{\alpha_j\}_{j=0}^N$, χ_n
Output: $\{x_j\}_{j=1}^n$, $\{\psi_n'(x_j)\}_{j=1}^n$

Step 1. Set $m = \lfloor \frac{n}{2} \rfloor + 1$*. If* n *is odd, evaluate* $\psi_n'(0)$ *using* (23)*, and then set* $roots(m) = 0$ *and* $ders(m) = \psi_n'(0)$*, and continue to Step 2. If* n *is even, conduct the following procedure:*

- *(a) Evaluate* $\psi_n(0)$ *using* (22)*. Compute* $\psi_n^{(k)}(0)$ *for* $k = 2, 3, \dots, K$ *where* $K = 30$ *in double precision and* $K = 60$ *in quadruple precision, using the recurrence relation* (26) *with the computed value of* $\psi_n(0)$ *and* $\psi_n'(0) = 0$*.*
- *(b) Perform a crude solve on* (29) *subject to the initial condition* $x(0) = 0$ *from* 0 *to* $-\pi/2$ *using a second-order Runge–Kutta scheme, and let* $\widetilde{x}^{(0)}$ *be the computed value of the solution at* $\pi/2$*.*
- *(c) Refine* $\widetilde{x}^{(0)}$ *using the Newton iteration* (41)*. At each iteration,* ψ_n *and* ψ_n' *are evaluated using Taylor series with derivatives computed in (a). Set the obtained root to* $roots(m)$*.*
- *(d) Set* $ders(m)$ *to the value of* $\psi_n'(roots(m))$*, evaluated using the Taylor series.*

Step 2. The remaining non-negative roots can be found through the following procedure:

- *(a) Set* $j := m$*.*
- *(b) Evaluate* $\psi_n^{(k)}(roots(j))$ *for* $k = 2, 3, \dots, K$*, where* $K = 30$ *in double precision and* $K = 60$ *in quadruple precision, using the recurrence relation* (26) *and the facts that* $\psi_n(roots(j)) = 0$ *and* $\psi_n'(roots(j)) = ders(j)$*.*
- *(c) Perform a crude solve on* (29) *subject to the initial condition* $x(\pi/2) = roots(j)$ *from* $\pi/2$ *to* $-\pi/2$ *using a second-order Runge–Kutta scheme, and let* $\widetilde{x}^{(0)}$ *be the value of the resulting solution at* $-\pi/2$*.*
- *(d) Refine* $\widetilde{x}^{(0)}$ *using the Newton iteration* (41)*, where* ψ_n *and* ψ_n' *are evaluated by their Taylor series expansions with derivatives computed in (b). Set the resulting root to* $roots(j+1)$*.*
- *(e) Determine* $ders(j+1)$ *by evaluating* $\psi_n'(roots(j+1))$ *using the Taylor series.*
- *(f) If* $j = n-1$*, continue to Step 3. Otherwise, set* $j := j+1$ *and return back to (b).*

Step 3. For $j = 1, 2, \dots, m-1$*, set* $roots(j) = -roots(n-j+1)$ *and* $ders(j) = (-1)^{n+1} ders(n-j+1)$*.*

Step 4. Set $x_j = roots(j)$ *and note that* $\psi_n'(x_j) = ders(j)$ *for* $j = 1, 2, \dots, n$*.*

Algorithm 4 (Compute $\{w_j\}_{j=1}^n$)**.**
Input: c, n, $\{\alpha_j\}_{j=0}^N$, χ_n, $\{x_j\}_{j=1}^n$, $\{\psi_n'(x_j)\}_{j=1}^n$
Output: $\{w_j\}_{j=1}^n$

Step 1. *Set $m = \lfloor \frac{n}{2} \rfloor + 1$. Evaluate $\Psi_n(x_m)$ and $\Psi_n'(x_m)$ using* (57)*, and set these to vals(m), respectively.*

Step 2. *Next, determine the values of $\Psi_n(x_j)$ and $\Psi_n'(x_j)$ for $j = m+1, m+2, \ldots, n-4$ using the following procedure:*

- *(a) Set $j := m$.*
- *(b) Evaluate $\Psi_n^{(k)}(x_j)$ for $k = 3, 4, \ldots, K$, where $K = 30$ in double precision and $K = 60$ in quadruple precision, using* (58), (59), (60)*, and the initial conditions $\Psi_n(x_j) = vals(j)$ and $\Psi_n'(x_j) = ders(j)$.*
- *(c) Use the derivatives computed in (b) to compute Taylor series for $\Psi_n(x_{j+1})$ and $\Psi_n'(x_{j+1})$ and set these to vals$(j+1)$ and ders$(j+1)$, respectively.*
- *(d) If $j = n-5$, continue to Step 3. If not, set $j := j+1$ and go back to (b).*

1. *For $j = n-3, n-2, n-1, n$, obtain vals(j) by directly evaluating $\Psi_n(x_j)$ using the series* (57)*.*
2. *Set $vals(j) = (-1)^{n+1} vals(n-j+1)$ for $j = 1, 2, \ldots, m-1$.*
3. *Finally, set $w_j = -2vals(j)/\psi_n'(x_j)$ for $j = 1, 2, \ldots, n$.*

Algorithm 5 (Compute λ_n)**.**
Input: c, n, $\{\alpha_j\}_{j=0}^N$
Output: λ_n

Step 1. *If n is even, evaluate $\psi_n(0)$ using* (22)*, and let $\lambda_n = 2\alpha_0/\psi_n(0)$. If n is odd, evaluate $\psi_n'(0)$ using* (23) *and let $\lambda_n = 2ci\alpha_1/(3\psi_n'(0))$.*

5 Numerical experiments

The algorithm described in Section 4 for computing the quadrature rule (51) was implemented in Fortran with the version 11.2.0 GNU Fortran compiler. The following experiments were run on a 2.1 GHz laptop with 16 GB of RAM.

5.1 Experiment 1

In this experiment, we measure the accuracy of and time to compute the quadrature rule (51) as functions of the bandlimit c and the requested accuracy ε. For any $c > 0$, we define the points

$$\omega_k^c = \frac{2kc}{100}, \quad k = 1, 2, \ldots, 100. \tag{62}$$

For any integer $n \geq 0$, we define the error

$$E_n^c = \max_{k=1,2,\ldots,100} \left| \int_{-1}^{1} \cos(\omega_k^c x)\, dx - \sum_{k=1}^{n} w_j^{(n)} \cos(\omega_k^c x_j^{(n)}) \right|, \tag{63}$$

where $\{(w_j^{(n)}, x_j^{(n)})\}_{j=1}^n$ is the n-point rule (51). We also define $n(\varepsilon)$ to be smallest integer n such that

$$|\lambda_n| < \varepsilon, \tag{64}$$

for any $\varepsilon \in (0, 1)$.

For fixed values of c and ε, we compute the $n(\varepsilon)$-point quadrature rule as described in Section 4 and then check $E_{n(\varepsilon)}^c$. The results are shown in Table 1, which has the following the structure. The first and second columns show the values of c and ε, respectively. The third column shows $n(\varepsilon)$ as defined by (64). The fourth column shows $|\lambda_{n(\varepsilon)}|$, computed in double precision. The fifth and sixth columns show the values of $E_{n(\varepsilon)}^c$ when the quadrature rule is computed in double and quadruple precision, respectively. The seventh and eighth columns show the total CPU time taken in seconds t_{total} to compute the quadrature rule in double and quadruple precision, respectively.

The results show that for all choices of parameters such that $\varepsilon > c\varepsilon_{\text{mach}}$, where $\varepsilon_{\text{mach}}$ is machine precision, $E_{n(\varepsilon)}^c < \varepsilon$. When $\varepsilon < c\varepsilon_{\text{mach}}$, $E_{n(\varepsilon)}^c$ is limited to approximately $c\varepsilon_{\text{mach}}$. This is expected since the evaluation of the integrand in (63) has condition number approximately c. Thus, the quadrature rule is capable of integrating functions down to machine precision. Moreover, $|\lambda_n|$ provides a good upper bound for the error when the n-point rule is used (see page 344 in [6] for further discussion).

5.2 Experiment 2

We now fix $c = 10^4$ and examine how E_n^c as defined in (63) and $|\lambda_n|$ change as functions of n. Figure 1 plots the values of E_n^c when the quadrature rule (51) is computed in double (left) and quadruple (right) precision. As is expected from Theorem 3.7, E_n^c starts to decay rapidly when $n \approx 2c/\pi$ and remains at approximately $c\varepsilon_{\text{mach}}$ from then on. Similarly, Figure 2 plots the values of $|\lambda_n|$ for varying values of n in double (left) and quadruple (right) precision. These values also start to decay superalgebraically when $n \approx 2c/\pi$. As discussed in Observation 3.4, we are able to compute $|\lambda_n|$ well below machine precision, down to approximately the square root of the point of underflow.

5.3 Experiment 3

In this experiment, we fix $c = 10^6$ and examine the CPU time required to compute the n-point quadrature rule (51) as a function of n. The results are displayed in Tables 2 and 3 when the computations are done in double and quadruple precision, respectively. Both tables have the following structure. The first column contains the values of n. The second column shows t_{prol},

c	ε	$n(\varepsilon)$	$\lvert\lambda_{n(\varepsilon)}\rvert$	$E^c_{n(\varepsilon)}$ (double)	$E^c_{n(\varepsilon)}$ (quad)	t_{total} (double)	t_{total} (quad)
10^2	10^{-10}	86	0.59988E-10	0.49E-12	0.49E-12	0.554E-03	0.265E-01
10^2	10^{-25}	112	0.33640E-25	0.28E-14	0.56E-28	0.653E-03	0.290E-01
10^2	10^{-50}	147	0.44641E-50	0.14E-13	0.66E-32	0.739E-03	0.332E-01
10^3	10^{-10}	667	0.95582E-10	0.27E-11	0.27E-11	0.238E-02	0.102E+00
10^3	10^{-25}	708	0.97844E-25	0.24E-13	0.32E-28	0.239E-02	0.109E+00
10^3	10^{-50}	768	0.39772E-50	0.12E-13	0.81E-32	0.261E-02	0.117E+00
10^4	10^{-10}	6405	0.57608E-10	0.35E-12	0.18E-12	0.203E-01	0.854E+00
10^4	10^{-25}	6462	0.63792E-25	0.42E-12	0.68E-29	0.204E-01	0.859E+00
10^4	10^{-50}	6548	0.51349E-50	0.15E-12	0.16E-30	0.200E-01	0.877E+00
10^5	10^{-10}	63707	0.71063E-10	0.83E-11	0.33E-13	0.188E+00	0.884E+01
10^5	10^{-25}	63780	0.92981E-25	0.11E-10	0.63E-29	0.191E+00	0.887E+01
10^5	10^{-50}	63893	0.80840E-50	0.44E-11	0.21E-29	0.210E+00	0.933E+01
10^6	10^{-10}	636670	0.79326E-10	0.19E-08	0.15E-12	0.184E+01	0.855E+02
10^6	10^{-25}	636760	0.77413E-25	0.43E-09	0.83E-29	0.184E+01	0.855E+02
10^6	10^{-50}	636900	0.69235E-50	0.29E-10	0.14E-27	0.187E+01	0.858E+02
10^7	10^{-10}	6366252	0.87469E-10	0.42E-08	0.16E-11	0.205E+02	0.939E+03
10^7	10^{-25}	6366358	0.97995E-25	0.20E-08	0.11E-25	0.225E+02	0.100E+04
10^7	10^{-50}	6366525	0.91559E-50	0.83E-10	0.61E-27	0.223E+02	0.945E+03

Table 1: Experiment 1

which is the time required to compute the Legendre expansion for ψ_n using Algorithms 1 and 2. The third column contains t_{roots}, which is the time required to determine the quadrature nodes using Algorithm 3. The fourth column contains t_{weights}, which is the time required to compute the quadrature weights via Algorithm 4. Finally, the fifth column reports t_{total}, which is the total time taken to compute the quadrature rule. The left plot in Figure 3 shows the values of t_{total} plotted as a function of n when the computation is done in double precision. We see that t_{total} scales approximately linearly with n, as expected. The constants are significantly higher when the computation is done in quadruple precision, which is expected since the machine these experiments were run on does not support quadruple precision floating point operations in hardware.

5.4 Experiment 4

We now fix the desired accuracy $\varepsilon = e^{-50}$ and measure the CPU time required to compute the quadrature rule (51) with $n(\varepsilon)$ points (see (64)) as a function of the bandlimit c. The results are displayed in Tables 4 and 5 when the computations are done in double and quadruple precision, respectively. Both tables have the following structure. The first and second columns show the values of c and $n(\varepsilon)$, respectively. The third column contains t_{prol}, which is the time required to compute the Legendre expansion for ψ_n using Algorithms 1 and 2. The fourth column contains t_{roots}, which is the time required to determine the quadrature nodes using Algorithm 3. The

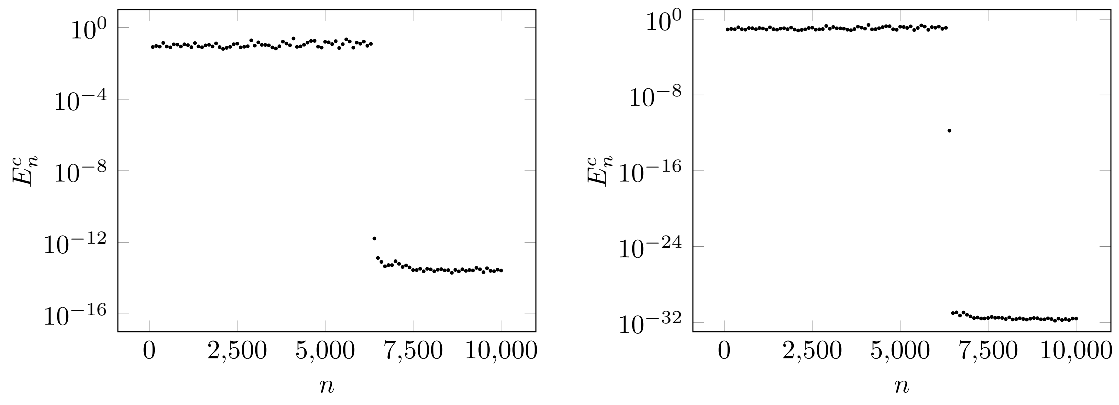


Figure 1: E_n^c in Experiment 2 when computations are done in double (left) and quadruple (right) precision

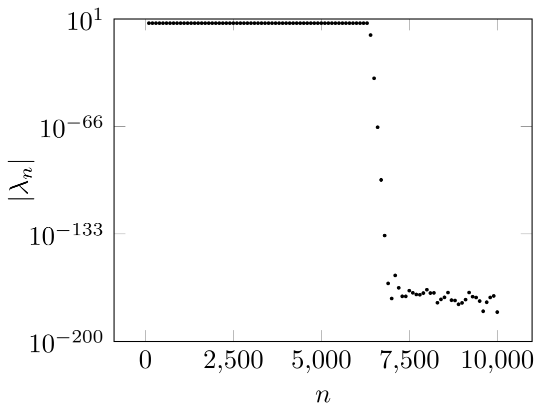
n	t_{prol}	t_{roots}	t_{weights}	t_{total}
500,000	0.763E+00	0.834E+00	0.183E+00	0.178E+01
1,000,000	0.124E+01	0.168E+01	0.337E+00	0.326E+01
2,000,000	0.184E+01	0.338E+01	0.689E+00	0.591E+01
4,000,000	0.404E+01	0.742E+01	0.154E+01	0.130E+02
8,000,000	0.680E+01	0.160E+02	0.318E+01	0.259E+02
16,000,000	0.136E+02	0.296E+02	0.628E+01	0.495E+02

Table 2: Experiment 3 (double precision)

fifth column contains t_{weights}, which is the time required to compute the quadrature weights via Algorithm 4. Finally, the sixth column reports t_{total}, which is the total time taken to compute the quadrature rule. The right plot in Figure 3 shows the values of t_{total} plotted as a function of c when the computation is done in double precision. The data show that t_{total} asymptotically scales approximately linearly with c.

6 Conclusions

In this paper, we describe a fast algorithm for computing quadrature rules for bandlimited functions. For any fixed $c > 0$, the algorithm computes an n-point quadrature rule in $O(n \log n)$ operations. Moreover, the computed quadrature rules are capable of achieving machine precision. The accuracy and speed of the algorithm are demonstrated in several numerical experiments. The approach of this paper can be generalized to design algorithms for computing quadrature rules for a broad class of other special functions.



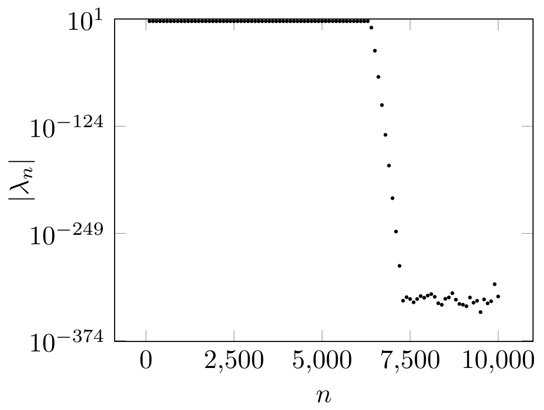


Figure 2: $|\lambda_n|$ for Experiment 2 when computations are done in double (left) and quadruple (right) precision

n	t_{prol}	t_{roots}	t_{weights}	t_{total}
500,000	0.336E+02	0.559E+02	0.110E+02	0.101E+03
1,000,000	0.453E+02	0.114E+03	0.230E+02	0.182E+03
2,000,000	0.545E+02	0.175E+03	0.338E+02	0.264E+03
4,000,000	0.890E+02	0.402E+03	0.774E+02	0.569E+03
8,000,000	0.191E+03	0.828E+03	0.155E+03	0.117E+04
16,000,000	0.375E+03	0.165E+04	0.308E+03	0.234E+04

Table 3: Experiment 3 (quadruple precision)

References

[1] Milton Abramowitz, Irene A Stegun, and Robert H Romer. Handbook of mathematical functions with formulas, graphs, and mathematical tables, 1988.

[2] Andreas Glaser, Xiangtao Liu, and Vladimir Rokhlin. A fast algorithm for the calculation of the roots of special functions. *SIAM Journal on Scientific Computing*, 29(4):1420–1438, 2007.

[3] Henry J Landau and Henry O Pollak. Prolate spheroidal wave functions, Fourier analysis and uncertainty II. *Bell System Technical Journal*, 40(1):65–84, 1961.

[4] Henry J Landau and Henry O Pollak. Prolate spheroidal wave functions, Fourier analysis and uncertainty III. *Bell System Technical Journal*, 41(4):1295–1336, 1962.

[5] Andrei Osipov. Evaluation of small elements of the eigenvectors of certain symmetric tridiagonal matrices with high relative accuracy. *Applied and Computational Harmonic Analysis*, 43(2):173–211, 2017.

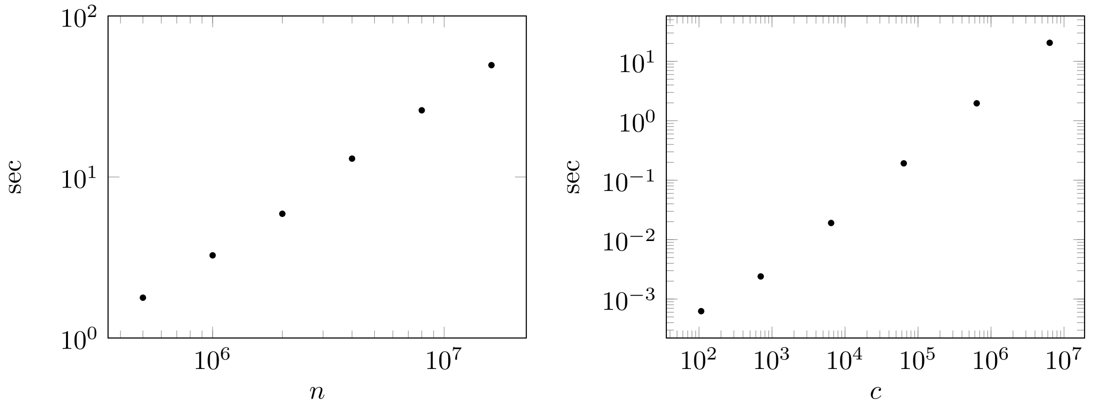


Figure 3: t_{total} for computations done in double precision in Experiments 3 (left) and 4 (right)

c	$n(\varepsilon)$	t_{prol}	t_{roots}	t_{weights}	t_{total}
10^2	107	0.359E-03	0.222E-03	0.440E-04	0.625E-03
10^3	700	0.741E-03	0.138E-02	0.280E-03	0.240E-02
10^4	6450	0.437E-02	0.121E-01	0.256E-02	0.191E-01
10^5	63765	0.456E-01	0.115E+00	0.311E-01	0.192E+00
10^6	636741	0.640E+00	0.110E+01	0.224E+00	0.197E+01
10^7	6366336	0.481E+01	0.134E+02	0.227E+01	0.205E+02

Table 4: Experiment 4 (double precision)

[6] Andrei Osipov, Vladimir Rokhlin, and Hong Xiao. *Prolate Spheroidal Wave Functions of Order Zero*. Springer, 2013.

[7] Yoel Shkolnisky, Mark Tygert, and Vladimir Rokhlin. Approximation of bandlimited functions. *Applied and Computational Harmonic Analysis*, 21(3):413–420, 2006.

[8] David Slepian. Prolate spheroidal wave functions, Fourier analysis and uncertainty IV. *Bell System Technical Journal*, 43(6):3009–3057, 1964.

[9] David Slepian. Prolate spheroidal wave functions, Fourier analysis, and uncertainty V. *Bell System Technical Journal*, 57(5):1371–1430, 1978.

[10] David Slepian and Henry O Pollak. Prolate spheroidal wave functions, Fourier analysis and uncertainty I. *Bell System Technical Journal*, 40(1):43–63, 1961.

[11] Lloyd N Trefethen and David Bau. *Numerical linear algebra*, volume 50. SIAM, 1997.

[12] Hong Xiao, Vladimir Rokhlin, and Norman Yarvin. Prolate spheroidal wavefunctions, quadrature and interpolation. *Inverse problems*, 17(4):805, 2001.

c	$n(\varepsilon)$	t_{prol}	t_{roots}	t_{weights}	t_{total}
10^2	107	0.165E-01	0.113E-01	0.244E-02	0.303E-01
10^3	700	0.427E-01	0.653E-01	0.130E-01	0.121E+00
10^4	6450	0.295E+00	0.632E+00	0.132E+00	0.106E+01
10^5	63765	0.263E+01	0.901E+01	0.170E+01	0.133E+02
10^6	636741	0.377E+02	0.710E+02	0.124E+02	0.121E+03
10^7	6366336	0.371E+03	0.663E+03	0.129E+03	0.116E+04

Table 5: Experiment 4 (quadruple precision)