FAST COMPUTATION OF GAUSS QUADRATURE NODES AND WEIGHTS ON THE WHOLE REAL LINE

ALEX TOWNSEND*, THOMAS TROGDON†, AND SHEEHAN OLVER‡

Abstract. A fast and accurate algorithm for the computation of Gauss–Hermite and generalized Gauss–Hermite quadrature nodes and weights is presented. The algorithm is based on Newton's method with carefully selected initial guesses for the nodes and a fast evaluation scheme for the associated orthogonal polynomial. In the Gauss–Hermite case the initial guesses and evaluation scheme rely on explicit asymptotic formulas. For generalized Gauss–Hermite, the initial guesses are furnished by sampling a certain equilibrium measure and the associated polynomial evaluated via a Riemann–Hilbert reformulation. In both cases the n-point quadrature rule is computed in $\mathcal{O}(n)$ operations to an accuracy that is close to machine precision. For sufficiently large n, some of the quadrature weights have a value less than the smallest positive normalized floating-point number in double precision and we exploit this fact to achieve a complexity as low as $\mathcal{O}(\sqrt{n})$.

Key words. Gauss quadrature, numerical integration, Hermite polynomials, equilibrum measures, Riemann–Hilbert problems

AMS subject classifications. 65D32, 33C45, 35Q15

1. Introduction.

Numerical quadrature is the approximation of a definite integral of a continuous function f by a weighted linear combination of function evaluations, *i.e.*,

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} w_k f(x_k), \qquad -\infty \le a < b \le \infty,$$

where $\{x_k\}_{k=1}^n$ are the *nodes* and $\{w_k\}_{k=1}^n$ are the *weights* (indexed so that $x_k < x_{k+1}$). An n-point quadrature rule of this form is "Gaussian" if for some nonnegative weight function, denoted by $w(x)$, the approximation

$$\int_a^b w(x)f(x)dx \approx \sum_{k=1}^{n} w_k f(x_k)$$

is exact whenever f is a polynomial of degree $\le 2n-1$. For a fixed weight function this requirement uniquely defines a set of quadrature nodes and weights and the resulting integration scheme is called a *Gauss quadrature rule* [21, Sec. 4.6].

In this paper, we are interested in weight functions of the form $w(x) = e^{-V(x)}$ and integrating functions over the whole real line ($a = -\infty$, $b = \infty$). If $V(x) = x^2$, then $w(x)$ is the classic Hermite weight. More generally, we are concerned with the so-called Freud weights, where $V(x)$ is a real polynomial that grows at infinity. Of particular interest is the case $V(x) = x^{2m}$, $m \ge 1$. For an integer n and a weight function $w(x)$, there are unique sets of Gauss quadrature nodes $\{x_k\}_{k=1}^n$ and weights $\{w_k\}_{k=1}^n$, and it is our goal to compute these two sets to an accuracy of double precision in $\mathcal{O}(n)$ operations.

*Department of Mathematics, Massachusetts Institute of Technology, 77 Massachusetts Avenue Cambridge, MA 02139-4307. (ajt@mit.edu, http://math.mit.edu/∼ajt/)

†Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, 10012-1185. (trogdon@cims.nyu.edu, http://www.cims.nyu.edu/∼trogdon/)

‡The University of Sydney, NSW 2006, Australia (Sheehan.Olver@sydney.edu.au, http://www.maths.usyd.edu.au/u/olver/)

The classic approach for computing Gauss quadrature nodes and weights is the Golub–Welsch algorithm [12], which requires $\mathcal{O}(n^2)$ operations when one is careful and $\mathcal{O}(n^3)$ operations when one is not[1]. However, in recent years several fast algorithms have been developed that require only $\mathcal{O}(n)$ operations. Currently, the state-of-the-art for classic weight functions is Bogaert's algorithm [4] for Gauss–Legendre ($w(x) = 1$, $[a,b] = [-1,1]$), the Hale–Townsend algorithm for Gauss–Jacobi [13] ($w(x) = (1-x)^\alpha(1+x)^\beta$, $[a,b] = [-1,1]$), and the Glaser–Lui–Rokhlin algorithm [11] for Gauss–Laguerre ($w(x) = e^{-x}$, $[a,b] = [0,\infty]$) and Gauss–Hermite ($w(x) = e^{-x^2}$, $[a,b] = [-\infty,\infty]$). In this paper we extend the approach in [13] to a competitive algorithm for computing Gauss–Hermite quadrature nodes and weights. Then, we use Riemann–Hilbert (RH) problems to derive an $\mathcal{O}(n)$ algorithm for generalized Gauss–Hermite quadrature rules. This demostrates, for the first time, that the procedure in [13] can be generalized to nonstandard Gauss quadrature rules.

RH problems are boundary value problems in the complex plane [1, 5, 27]. The use of RH problems is necessary for our general approach. The main idea from [13] that we generalize is that asymptotic formulas for orthogonal polynomials and their derivative can be combined with initial guesses for the Gauss nodes to derive an effective scheme for computing Gauss quadrature nodes and weights. For classical weights, such expansions are known explicitly, but for other weight functions we note that accurate approximations can be calculated numerically via the solution of a parameter-dependent RH problem [5, 20] using nonlinear steepest descent [6, 26].

The paper is structured as follows. In the next section we present an overview of our scheme, which is central to both the computation of Gauss–Hermite nodes and weights and its generalizations. In Section 3 we describe how to compute the Gauss–Hermite nodes and weights in $\mathcal{O}(n)$ operations using Newton's method together with explicit asymptotic formulas. In Section 4 we show how these ideas can be generalized to weights of the form $e^{-V(x)}$ using equilibrium measures and RH problems. Finally, Section 5 describes an application of these methods to barycentric Lagrange interpolation with an appendix analyzing the weighted stability of this interpolation.

2. Overview of approach. For a given weight function $w(x)$, our algorithm for computing the corresponding Gauss quadrature nodes and weights relies on the standard fact that the nodes are precisely the roots of the associated orthogonal polynomial of degree n [10]. That is, if $\phi_0(x), \ldots, \phi_n(x), \ldots,$ is the sequence of orthogonal polynomials that are orthogonal with respect to the inner-product

$$\langle f, g\rangle = \int_{-\infty}^{\infty} w(x) f(x) g(x) dx,$$

then the Gauss nodes $x_1, \ldots, x_n$ satisfy $\phi_n(x_k) = 0$ for $1 \le k \le n$. This is a powerful observation that transforms the abstract notion of a Gauss quadrature rule to n tangible rootfinding problems.

We solve each rootfinding problem $\phi_n(x_k) = 0$ with Newton's method, which needs three pieces of information: (1) A sufficiently close initial guess for x_k, (2) An evaluation scheme for ϕ_n, and (3) An evaluation scheme for ϕ_n'. As there are n rootfinding problems, *i.e.*, $\phi_n(x_k) = 0$ for $1 \le k \le n$, we must solve each one in just $\mathcal{O}(1)$ operations to achieve an overall complexity of $\mathcal{O}(n)$. Here is how we achieve (1), (2), and (3):

[1]In many programming languages, for example MATLAB, the structure of symmetric tridiagonal eigenproblems is not automatically detected or exploited.

(1) Initial guesses. If $w(x) = e^{-x^2}$, then there are explicit asymptotic expansions that approximate the Gauss–Hermite nodes (see Lemmas 3.1 and 3.2). For $n \geq 200$ these provide sufficiently good initial guesses for Newton's method. Unfortunately, for general weights of the form $w(x) = e^{-V(x)}$ explicit asymptotic expansions for the Gauss nodes are not available. Instead, we use equilibrium measures to furnish initial guesses. Roughly speaking, the equilibrium measure describes the asymptotic density of the *Fekete points* [22], which are the global minimizers of the energy functional

$$E(x_1, \ldots, x_n) = \frac{2}{n(n-1)} \sum_{1 \leq i \neq j \leq n} \log |x_i - x_j|^{-1} + \frac{1}{n} \sum_{i=1}^{n} V(x_i)$$

as $n \to \infty$. For $V(x) = x^2$ the Fekete points are the zeros of Hermite polynomials and for general $V(x)$, an asymptotic expansion of the zeros of orthogonal polynomials can be derived in terms of the equilibrium measure [7].

(2) Evaluation of orthogonal polynomial. If $w(x) = e^{-x^2}$, then the degree n Hermite polynomial can be expressed in terms of a parabolic cylinder function, which has a powerful uniform asymptotic expansion involving Airy functions (see Section 3.2). For $n \geq 200$ the resulting asymptotic expansion is accurate to 14-15 digits and requires only $\mathcal{O}(1)$ operations per evaluation. (Note that we are not able to evaluate a Hermite polynomial using the 3-term recurrence [17, (18.9.1)], since that requires $\mathcal{O}(n)$ operations per evaluation.) High-order explicit asymptotic expansions are not available for orthogonal polynomials associated to generalized Hermite weights and instead we use a numerical RH approach (see Section 4.2).

(3) Evaluation of the derivative. In the Gauss–Hermite case, the evaluation of ϕ_n' is achieved by explicit asymptotic expansions. For more general weights we again use a numerical RH approach.

Finally, we also need to compute the Gauss quadrature weights. In both the Gauss–Hermite and generalized quadrature rules, w_k can be expressed in terms of $\phi(x_k)$ and $\phi'(x_k)$; see (3.5) and (4.4).

For sufficiently large n, some of the quadrature weights take a value less than the smallest positive normalized floating-point number in double precision. In such circumstances, these quadrature weights and corresponding nodes do not contribute to the final quadrature estimate when working in double precision (regardless of the function to be integrated). We provide a subsampling scheme so that only the weights and corresponding nodes that contribute to the quadrature estimate are computed. This makes the algorithm far more efficient without reducing the accuracy of the resulting quadrature rule.

Though it is not the focus of this paper, when n is small ($n < 200$) we recommend using Newton's method, as described above, together with polynomial evaluation via the 3-term recurrence [17, (18.9.1)]. We observe this approach to be extremely accurate and is easily applicable to variable precision computations. For very small n, the initial guesses provided by asymptotics or equilibrium measures will not be sufficient to guarantee convergence of Newton's method. At this point, it is reasonable to use the Golub–Welsch algorithm to furnish initial guesses and Newton's method to improve the accuracy of the final nodes and weights.

3. Computing Gauss–Hermite quadrature nodes and weights.

The classical Gauss–Hermite quadrature nodes and weights correspond to the weight function

$w(x) = e^{-x^2}$, which can be used to approximate the following definite integral:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx \approx \sum_{k=1}^{n} w_k f(x_k),$$

where f is a "smooth" function and $\{x_k\}$ and $\{w_k\}$ are the Gauss–Hermite nodes and weights, respectively. The associated sequence of orthogonal polynomials are the Hermite polynomials, denoted by $H_0, H_1, \ldots$, which can be defined via the following 3-term recurrence relation [17, (18.9.1)]:

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x), \qquad n \geq 1, \quad x \in \mathbb{R},$$

where $H_0(x) = 1$ and $H_1(x) = 2x$. The Gauss nodes are the roots of the degree n Hermite polynomial. That is, $H_n(x_k) = 0$ for $1 \leq k \leq n$.

Now, since Hermite polynomials have a reflective symmetry [17, (18.6.1)], *i.e.*, $H_n(-x) = (-1)^n H_n(x)$, the nodes are symmetrically located on the real line. That is, $x_k = -x_{n-k+1}$ for $1 \leq k \leq n$ and if n is odd, $x_{\lceil n/2 \rceil} = 0$. Therefore, we only need to compute the strictly positive nodes since the others can be obtained by symmetry. This will save a factor of roughly 2 in the computational cost of the final algorithm.

In addition, it is known that the Gauss–Hermite nodes satisfy [17, (18.16.16)]

$$-\sqrt{2n+1} < x_1 < \cdots < x_n < \sqrt{2n+1},$$

which means we will only require a fast evaluation scheme for $H_n(x)$ when $0 < x < \sqrt{2n+1}$.

3.1. Initial guesses for Gauss–Hermite quadrature nodes. We use two different asymptotic formulas for the strictly positive Gauss–Hermite nodes. For the majority of the nodes we use the asymptotic approximations derived by Tricomi [23] and given in the following lemma:

LEMMA 3.1 (Tricomi [23]). *Let τ_k be the root of the equation*

$$x - \sin x = \frac{(4\lfloor n/2 \rfloor - 4k + 3)\pi}{4\lfloor n/2 \rfloor + 2\alpha + 2}, \qquad \alpha = \mathrm{mod}(n, 2) - \frac{1}{2}. \tag{3.1}$$

Then, for $n \to \infty$ and $k \geq 0$ fixed, we have

$$x_{k+\lceil n/2 \rceil}^2 = \nu\sigma_k - \frac{1}{3\nu}\left[\frac{5}{4(1-\sigma_k)^2} - \frac{1}{1-\sigma_k} - \frac{1}{4}\right] + \mathcal{O}(n^{-3}), \quad n \to \infty,$$

where $\sigma_k = \cos^2(\tau_k/2)$ and $\nu = 4\lfloor n/2 \rfloor + 2\alpha + 2$.

In order to compute Tricomi's initial guesses we must first calculate τ_k. To achieve this we solve the equation in (3.1) by several steps of Newton's method with an initial guess of $\pi/2$. If we let $f(x) = x - \sin x$, then we note that $f : (0, \pi] \to (0, \pi]$ is twice differentiable, f' is strictly positive, and $f(x)f''(x) > 0$ so that convergence of Newton's method to τ_k is guaranteed regardless of the initial guess.

Tricomi's initial guesses for the nodes are accurate except for a handful near $\sqrt{2n+1}$, and for these nodes we use the asymptotic approximations derived by Gatteschi [9]:

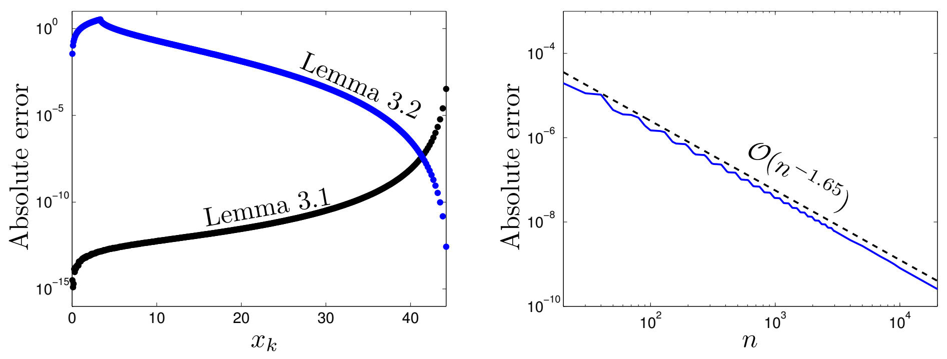

FIG. 3.1. *Left: Absolute error of the initial guesses from Lemma 3.1 (black) and Lemma 3.2 (blue) for $n = 1{,}000$. Right: The maximum error in the initial guesses for $n \leq 20{,}000$ when using Tricomi's initial guesses for $k = 1, \ldots, \lfloor \rho n \rfloor$ and Gatteschi's otherwise, where $\rho = 0.4985$.*

LEMMA 3.2 (Gatteschi [9]). *Let a_m be the mth zero of the Airy function $\mathrm{Ai}(x)$, indexed so that $a_{m+1} < a_m < 0$. Then, for $n \to \infty$ and fixed $k \geq 1$, we have*

$$\begin{aligned} x_{n-k+1}^2 = \nu + 2^{2/3} a_k \nu^{1/3} + \frac{1}{5} 2^{4/3} a_k^2 \nu^{-1/3} + \left(\frac{9}{140} - \frac{12}{175} a_k^3 \right) \nu^{-1} \\ + \left(\frac{16}{1575} a_k + \frac{92}{7875} a_k^4 \right) 2^{2/3} \nu^{-5/3} \\ - \left(\frac{15152}{3031875} a_k^5 + \frac{1088}{121275} a_k^2 \right) 2^{1/3} \nu^{-7/3} + \mathcal{O}(n^{-3}), \end{aligned}$$

where $\nu = 4\lfloor n/2 \rfloor + 2\alpha + 2$ and $\alpha = \mathrm{mod}(n,2) - 1/2$.

In order to compute Gatteschi's initial guesses, the zeros of the Airy function are required. We tabulate the first ten roots of $\mathrm{Ai}(x)$, while the others are computed with the asymptotic formula [17, (9.9.18)] that is observed to be accurate for $m \geq 11$:

$$a_m \approx -s_m^{2/3} \left(1 + \frac{5}{48} s_m^{-2} - \frac{5}{36} s_m^{-4} + \frac{77125}{82944} s_m^{-6} - \frac{108056875}{6967296} s_m^{-8} + \frac{162375596875}{334430208} s_m^{-10} \right),$$

where $s_m = 3\pi(4m-1)/8$.

In practice, we use Tricomi's initial guesses for $k = 0, \ldots, \lfloor \rho n \rfloor$, where $\rho = 0.4985$, and Gatteschi's otherwise. Based on numerical experiments, we have selected $\rho = 0.4985$ because when n is large Tricomi's and Gatteschi's initial guesses have roughly the same error for $x_{\lfloor \rho n \rfloor}$.

In Figure 3.1 (left) we show the absolute error in Tricomi's and Gatteschi's initial guesses for $n = 1{,}000$. It can be seen that Lemma 3.1 provides better initial guesses except when $x_k \approx \sqrt{2n+1}$. In Figure 3.1 (right) we show the absolute error in the initial guesses for $n \leq 20{,}000$ when Tricomi's initial guesses are used for $k = 1, \ldots, \lfloor \rho n \rfloor$ and Gatteschi's otherwise. Interestingly, the observed convergence rate is $\mathcal{O}(n^{-1.65})$, while Lemmas 3.1 and 3.2 only guarantee a rate of $\mathcal{O}(n^{-1.5})$.

For $n \geq 6{,}000$, we observe that the initial guesses become so accurate that just one Newton iteration is required to compute the Gauss–Hermite nodes to double precision.

3.2. Fast evaluation of Hermite polynomials.

The most powerful asymptotic formulas for Hermite polynomials are based on the asymptotics of the parabolic cylinder function. Hermite polynomials satisfy the following relationship [17, (18.15.28)]:

$$H_n(x) = 2^{(\mu^2-1)/4} e^{\mu^2 t^2/2} U\left(-\frac{1}{2}\mu^2, \mu t\sqrt{2}\right),$$

where U is the parabolic cylinder function, $\mu = \sqrt{2n+1}$, and $t = x/\mu$. Moreover, U has the following asymptotic formula that holds as $\mu \to \infty$ in the region $-\mu < x \le \mu$ (equivalently $-1 \le t \le 1$) [17, (12.10.35)]:

$$U\left(-\frac{1}{2}\mu^2, \mu t\sqrt{2}\right) \sim 2\pi^{\frac{1}{2}}\mu^{\frac{1}{3}} g(\mu)\phi(\zeta)\left(\mathrm{Ai}\left(\mu^{\frac{4}{3}}\zeta\right)\sum_{s=0}^{\infty}\frac{A_s(\zeta)}{\mu^{4s}} + \frac{\mathrm{Ai}'\left(\mu^{\frac{4}{3}}\zeta\right)}{\mu^{\frac{8}{3}}}\sum_{s=0}^{\infty}\frac{B_s(\zeta)}{\mu^{4s}}\right) \tag{3.2}$$

where ζ satisfies $2/3(-\zeta)^{3/2} = \frac{1}{2}\cos^{-1} t - \frac{1}{2}t\sqrt{1-t^2}$, and $\phi(\zeta) = (\zeta/(t^2-1))^{1/4}$. Here,

$$g(\mu) = h(\mu)\left(1 + \sum_{s=1}^{\infty}\frac{\phi_s}{(\frac{1}{2}\mu^2)^s}\right), \qquad h(\mu) = 2^{-\frac{1}{4}\mu^2-\frac{1}{4}} e^{-\frac{1}{4}\mu^2} \mu^{\frac{1}{2}\mu^2-\frac{1}{2}},$$

where the coefficients ϕ_s are defined by

$$\Gamma(\tfrac{1}{2}+z) \sim \sqrt{2\pi} e^{-z} z^z \sum_{s=0}^{\infty}\frac{\phi_s}{z^s}.$$

Moreover, in (3.2) we have, for $t = \cos\theta$,

$$A_0(\zeta) = 1, \quad B_0(\zeta) = -(\zeta^6(\cos^3\theta - 6\cos\theta)/24 + 15/144),$$

and A_1, B_1, and higher order terms can be calculated from the recurrence [17, (12.10.42)].

In practice, we truncate the asymptotic formula in (3.2) after four terms. Of course, more (increasingly complicated) terms can be taken, but with just four terms the resulting asymptotic formula is accurate for $n \ge 200$ (see Figure 3.2). Remarkably, despite the rather involved definitions, this asymptotic formula can be evaluated to close to 16 digits of absolute accuracy.

In order to compute the roots of the Hermite polynomial for large n, we scale the parabolic cylinder function so that its absolute maximum is bounded by 1. That is, we actually find the roots of

$$\tilde{U}\left(-\frac{1}{2}\mu^2, \mu t\sqrt{2}\right) = \frac{2^{\frac{1}{4}} U\left(-\frac{1}{2}\mu^2, \mu t\sqrt{2}\right)}{\sqrt{\pi} n^{\frac{1}{4}} g(\mu)}, \qquad \left|\tilde{U}\left(-\frac{1}{2}\mu^2, \mu t\sqrt{2}\right)\right| \le 1. \tag{3.3}$$

Scaling in this way is essential for avoiding numerical overflow issues for large n. We note that $\tilde{U}$ is closely related to the scaled Hermite polynomial described in [11, (32)].

In Figure 3.2 (left) we show the absolute error in the asymptotic formula in (3.2) for evaluating $\tilde{U}$ with four terms and $n = 1{,}000$. The asymptotic formula is only evaluated in the region $x > 0$ since the Gauss–Hermite nodes in $x \le 0$ can be recovered by symmetry (see Section 3). In Figure 3.2 (right) we show the maximum absolute error of (3.2) in the region $x > 0$ for $1 \le n \le 250$. We observe that the absolute error converges to zero like $\mathcal{O}(n^{-59/12})$. This very precise decay rate is expected because (with the scaling in (3.3)) the first neglected term (fifth term) of the asymptotic formula is of magnitude $\mathcal{O}(\mathrm{Ai}(\mu^{\frac{4}{3}}\zeta) n^{\frac{1}{12}} \mu^{-8}) = \mathcal{O}(n^{-59/12})$.

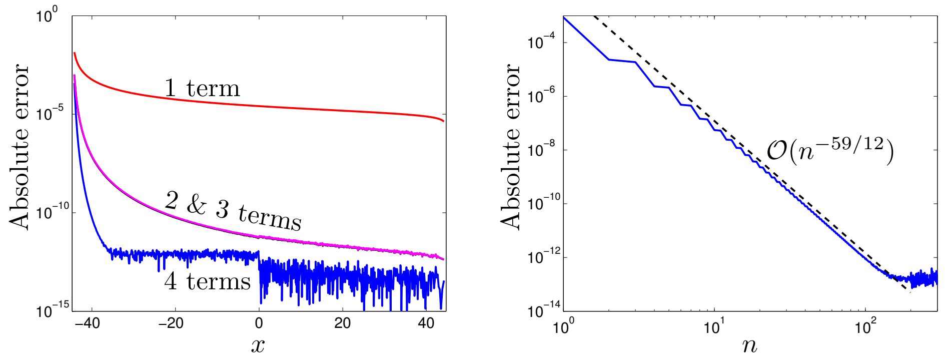

FIG. 3.2. *Left: Error in the asymptotic formula* (3.2) *for* $\tilde{U}$ *with 1 (red), 2 (magenta), 3 (black), and 4 (black) terms when* $n = 1{,}000$. *In our algorithm the expansion is only evaluated at* $x > 0$ *where the asymptotic formula is accurate. Right: Absolute error in the asymptotic formula for* $\tilde{U}$ *when* $1 \leq n \leq 250$. *We observe a convergence rate of* $\mathcal{O}(n^{-59/12})$, *which is consistent with the magnitude of the first neglected term in* (3.2).

3.3. Fast evaluation of the derivative of Hermite polynomials. To evaluate $H_n'(x)$, we have two options. One option is to use the following differentiation relationship [17, (18.9.25)]:

$$H_n'(x) = 2nH_{n-1}(x), \qquad n \geq 1,$$

and then to employ (3.2) corresponding to H_{n-1} rather than H_n. Another option, and the one we employ, is to use the asymptotic formula for U' given by [17, (12.10.36)]

$$U'\left(-\frac{1}{2}\mu^2, \mu t\sqrt{2}\right) \sim \frac{(2\pi)^{\frac{1}{2}}\mu^{\frac{2}{3}}g(\mu)}{\phi(\zeta)}\left(\frac{\mathrm{Ai}\left(\mu^{\frac{4}{3}}\zeta\right)}{\mu^{\frac{4}{3}}}\sum_{s=0}^{\infty}\frac{C_s(\zeta)}{\mu^{4s}} + \mathrm{Ai}'\left(\mu^{\frac{4}{3}}\zeta\right)\sum_{s=0}^{\infty}\frac{D_s(\zeta)}{\mu^{4s}}\right), \tag{3.4}$$

where we have, for $t = \cos\theta$,

$$C_0(\zeta) = \frac{2}{3}\left(\zeta^6(\cos^3\theta + 6\cos\theta)/24 - 7/48\right)\zeta^{-\frac{2}{3}}, \qquad D_0(\zeta) = 1,$$

and higher order terms can be obtained from the recurrence [17, (12.10.44)]. We prefer the latter approach because (3.4) contains exactly the same Airy functions as (3.2) and hence, the expensive (but $\mathcal{O}(1)$) special function evaluations can be reused.

3.4. Newton's method for Gauss–Hermite nodes. We now have all the ingredients to compute Gauss–Hermite nodes using Newton's method. However, we do not perform Newton's method in the usual x-variable, but instead the θ-variable, where

$$t = x/\mu, \qquad t = \cos\theta.$$

This improves the accuracy of the final nodes, particularly those close to $x \approx \mu$, *i.e.*, $t \approx 1$, because it does not require an evaluation of $\cos^{-1}(\,\cdot\,)$ per iteration, which is sensitive to small perturbations in arguments close to 1. Therefore, we take the initial guesses from Section 3.1 and perform a change of variables to obtain initial guesses in the θ-variable. We then proceed with Newton's method performed in this variable. For the kth Gauss–Hermite node one Newton step takes the form:

$$\theta_k^{\text{new}} = \theta_k^{\text{old}} + \frac{\tilde{U}\left(-\frac{1}{2}\mu^2, \sqrt{2}\mu\cos\theta_k^{\text{old}}\right)}{\sqrt{2}\mu\tilde{U}'\left(-\frac{1}{2}\mu^2, \sqrt{2}\mu\cos\theta_k^{\text{old}}\right)\sin\theta_k^{\text{old}}}.$$

If the update, $|\theta_k^{\text{new}} - \theta_k^{\text{old}}|$, is sufficiently small then Newton's method is terminated and the corresponding Gauss–Hermite node is calculated via $x_k = \mu\cos\theta_k^{\text{new}}$. In practice, we use the same number of iterations for every Gauss node so that Newton's method can be vectorized for a slightly improved computational efficiency.

3.5. Computing Gauss–Hermite weights. Once the Gauss–Hermite nodes have been computed, the Gauss–Hermite weights immediately follow by the simple formula [11, (40)]:

$$w_k = 2e^{-x_k^2}/\tilde{H}_n'^2(x_k),$$

where $\tilde{H}_n$ is the Hermite polynomial scaled so that $|\tilde{H}_n(x)| \leq 1$ for $x > 0$. In terms of parabolic cylinder functions this results in the following formula:

$$w_k = \frac{Ce^{-x_k^2}}{\left(\tilde{U}'\left(-\frac{1}{2}\mu^2, \mu t_k\sqrt{2}\right)\right)^2}, \qquad t_k = x_k/\mu, \tag{3.5}$$

where C is a constant so that $\sum_k w_k = \sqrt{\pi}$.

3.6. Subsampling. The Gauss–Hermite quadrature weights that correspond to nodes far away from 0 are usually very small in magnitude. So much so, that a significant proportion of the quadrature weights are less than `realmin`, *i.e.*, $2^{-1022} \approx 2.23 \times 10^{-308}$, which is the smallest normalized positive floating-point number in double precision. Thus, for any quadrature rule employed in double precision these weights will never contribute to the final approximation of the integral.

We have observed that the only quadrature weights that are larger than `realmin` are $w_{\lfloor n/2\rfloor-M+1}, \ldots, w_{\lfloor n/2\rfloor+M}$ when n is even and $w_{\lfloor n/2\rfloor-M}, \ldots, w_{\lfloor n/2\rfloor+M}$ when n is odd, where $M = \lceil 12.5n^{1/2}\rceil$. Therefore, we only run Newton's method with initial guesses for this subset of the weights. Since $M = \mathcal{O}(n^{1/2})$ the resulting algorithm has a reduced complexity of $\mathcal{O}(n^{1/2})$ operations without sacrificing the accuracy of the resulting quadrature rule. In Table 3.1 the execution time in seconds is given for computing the Gauss–Hermite nodes and weights with and without subsampling. It is observed that a significant saving can be achieved by not computing weights that have a magnitude less than `realmin`. Similar computational savings can be easily achieved by the REC and GLR algorithms (see Section 3.7). We believe it is much harder to avoid the computation of a subset of the nodes and weights in the Golub–Welsch algorithm.

3.7. Numerical results. In this section we compare the algorithm described in this section based on asymptotic formulas (ASY for short) against three other methods for computing Gauss–Hermite quadrature nodes and weights, which we refer to using the acronyms:

REC: This recurrence-based algorithm performs Newton's method with orthogonal polynomial evaluation using a 3-term recurrence, requiring $\mathcal{O}(n^2)$ operations;

n	M	No subsampling	Subsampling
10	5	0.004951	0.005011
100	50	0.005663	0.005432
1,000	361	0.011345	0.009602
10,000	1,212	0.053017	0.027845
100,000	3,848	0.380275	0.049851
1,000,000	12,156	3.635814	0.121961
	$\mathcal{O}(n^{1/2})$	$\mathcal{O}(n)$	$\mathcal{O}(n^{1/2})$

TABLE 3.1

Execution time in seconds for computing Gauss–Hermite nodes and weights with and without subsampling. For the n-point Gauss–Hermite quadrature rule only $\mathcal{O}(\sqrt{n})$ weights contribute to the final quadrature approximation in double precision. Hence, a significant proportion of the computation of Gauss rules on the whole real line can be saved.

GLR: The Glaser–Lui–Rokhlin algorithm solves the associated 2nd-order ordinary differential equation [17, Table 18.8.1] using a predictor-corrector-like marching scheme, requiring $\mathcal{O}(n)$ operations [11];

GW: The Golub–Welsch algorithm solves for the eigenvalues (nodes) and eigenvectors (related to the weights) of the associated Jacobi matrix. The implementation we use for GW here requires $\mathcal{O}(n^3)$ operations as we do not exploit the symmetric tridiagonal structure of the Jacobi matrix [12].

These algorithms have been implemented in MATLAB and the numerical comparisons are performed in that language. For the GLR algorithm we use the MATLAB implementation in the `hermpts` command in Chebfun [8]. The algorithm for the ASY method has also been implemented in the `gausshermite` command in the Fast-GaussQuadrature package [24], which is written in the Julia language [3].

In Figure 3.3 we show the absolute errors in the computed Gauss–Hermite nodes $|x_k - x_k^{quad}|$ (left) and the relative error in the weights $|w_k - w_k^{quad}|/|w_k^{quad}|$ (right), where x_k^{quad} and w_k^{quad} are the nodes and weights computed using REC with quadruple precision. The ASY computed nodes are less accuracy near $x = 0$ but more accurate near $x = \sqrt{2n+1}$, which is caused by the asymptotic formula (3.2) written in terms of the θ-variable. More accurate Gauss–Hermite nodes could be obtained by a hybrid between the nodes computed by GLR and ASY.

In Figure 3.4 we compare the computational timings for the four methods. It can be seen that ASY is about 10 times faster than GLR for large n and ASY takes the lowest execution time out of the four methods when $n \geq 200$. The kink in the ASY timings at $n = 6{,}000$ is caused by one fewer Newton iteration required for convergence when $n \geq 6{,}000$. For $n = 1{,}000{,}000$, ASY requires 3.63 seconds. Subsampling can improve the computational speed of ASY, REC, and GLR (see Section 3.6), but the comparisons between these three methods will stay the same.

4. Computing generalized Gauss–Hermite quadrature nodes and weights.

The generalized Gauss–Hermite quadrature nodes and weights correspond to the weight function $w(x) = e^{-V(x)}$, where $V(x) = x^{2m} + \mathcal{O}(x^{2m-1})$ is a monic polynomial of degree $2m$ with real coefficients. As before, for an integer n, we denote the set of generalized Gauss–Hermite nodes and weights as $\{x_k\}_{k=1}^n$ and $\{w_k\}_{k=1}^n$, respectively.

4.1. Initial guesses for generalized Gauss–Hermite quadrature nodes.

Given a polynomial $V(x) = x^{2m} + \mathcal{O}(x^{2m-1})$ with real coefficients, the equilibrium

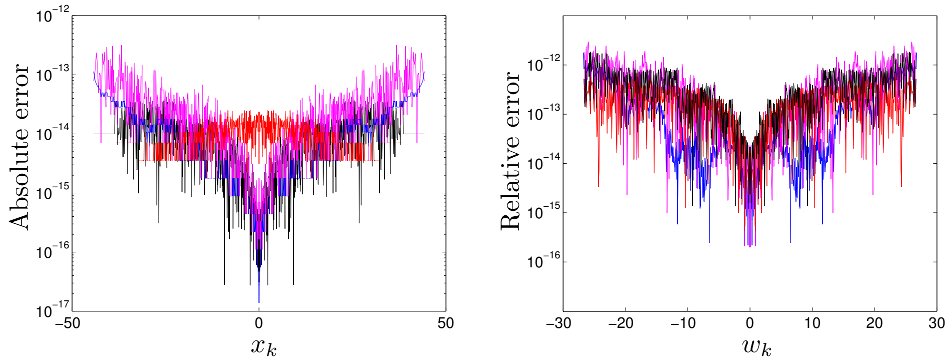

FIG. 3.3. *Left: Absolute error in the computed Gauss–Hermite nodes,* i.e., $|x_k - x_k^{quad}|$*, where* x_k^{quad} *are the nodes obtained using quadruple precision, for ASY (red), REC (black), GLR (blue), and GW (magenta). Right: Relative error in the computed Gauss–Hermite weights,* i.e., $|w_k - w_k^{quad}|/|w_k^{quad}|$*, where* w_k^{quad} *are the weights obtained using quadruple precision for ASY (red), REC (black), GLR (blue), and GW (magenta). Weights corresponding to nodes with absolute magnitude larger than* 27 *are less than the smallest positive normalized floating-point number in double precision and underflow.*

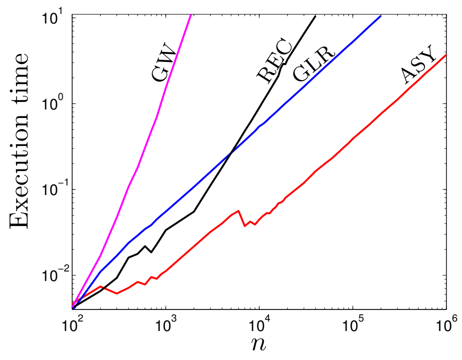

FIG. 3.4. *Comparison of computational times for Gauss–Hermite nodes and weights: ASY (red), REC (black), GLR (blue), and GW (magenta).*

measure $d\mu_{V,n}(x) = \psi_{V,n}(x)dx$ is the unique minimizer of the functional [5]

$$H(\mu) = \iint \log|x-y|^{-1} d\mu(x)d\mu(y) + \frac{1}{n}\int V(xn^{1/(2m)})d\mu(x),$$

among Borel probability measures μ on $\mathbb{R}$, *i.e.*, $\psi_{V,n}(x)dx = \operatorname{argmin}_\mu H(\mu)$. Without loss of generality, we assume $V(0) = V'(0) = 0$ (otherwise perform a change of variables), so that the support of the equilibrium measure $[a,b] \equiv [a_n, b_n]$ converges as $n \to \infty$ to an interval of the form $[-\alpha, \alpha]$, $\alpha > 0$. The zeros $\{\tilde{x}_k\}_{k=1}^n$ of the nth-order polynomial with varying weight $e^{-V(xn^{1/(2m)})}dx$ are distributed according to the equilibrium measure in the sense that the normalized counting measure converges in distribution [5] (see Figure 4.2). That is,

$$\lim_{n\to\infty}\left(\frac{1}{n}\int_a^t \sum_{k=1}^n \delta_{\tilde{x}_k}(x)dx - \int_a^t d\mu_{V,n}(x)\right) = 0, \quad t \in [a,b], \tag{4.1}$$

where $\delta_{\tilde{x}_k}(x)$ is the delta function centered at $\tilde{x}_k$. Therefore, the equilibrium measure can be used to determine the approximate location of the zeros and hence, used to furnish initial guesses for Gauss nodes. It is also known that $\tilde{x}_k \in (a,b)$ [5, Prop. 3.42]. We consider polynomials with respect to the varying weight $e^{-V(xn^{1/(2m)})}dx$ because it is easily seen that the Gauss quadrature nodes with respect to $e^{-V(x)}dx$ satisfy $x_k = \tilde{x}_k n^{1/(2m)}$ for $1 \leq k \leq n$. Furthermore, the weights $\{w_k\}_{k=1}^n$ for $e^{-V(x)}dx$ satisfy $w_k = \tilde{w}_k n^{1/(2m)}$, where $\{\tilde{w}_k\}_{k=1}^n$ are the quadrature weights corresponding to the varying weight function $e^{-nV(xn^{1/(2m)})}$.

Define the function $F_n : [a,b] \to \mathbb{R}$ as

$$F_n(x) = \begin{cases} 0, & x < a, \\ \int_a^x \psi_{V,n}(t)dt, & a \leq x \leq b, \\ 1, & x > b, \end{cases}$$

and the inverse function $G_n : (0,1) \to \mathbb{R}$ as $G_n(y) = \inf_{x\in\mathbb{R}}\{F_n(x) \geq y\}$. Then, for $y \in (0,1)$ the value of $G_n(y)$ lies in the support of $\psi_{V,n}(x)$ and $F_n(G_n(y)) = y$. The following asymptotic formula for the generalized Gauss–Hermite nodes is given in [7]:

$$\left|\tilde{x}_k - G_n\left(\frac{2k-1}{2n} + \frac{1}{2\pi n}\sin^{-1}\left(G_n(k/n)\right)\right)\right| < \frac{C}{n^2\left(\frac{k}{n}(1-\frac{k}{n})\right)^{4/3}},$$

where C is a constant. Therefore, we can furnish Newton's method with the following initial guesses:

$$\tilde{x}_k^{\text{guess}} = G_n\left(\frac{2k-1}{2n} + \frac{1}{2\pi n}\sin^{-1}\left(G_n(k/n)\right)\right).$$

4.1.1. Computing $\boldsymbol{\psi_{V,n}}$ and $\boldsymbol{G_n}$. While the definition of an equilibrium measure is stated as an optimization problem over measures, in the case of smooth V the support of the equilibrium measure $\psi_{V,n}$ is a single interval $[a,b]$ for sufficiently large n. The problem reduces to a simpler optimization over the two parameters a and b, which can be efficiently solved by Newton's method [18]. In particular, when $V(x)$ is a polynomial, the equilibrium measure has the form $\psi_{V,n}(x) = \sqrt{(b-x)(x-a)}p(x)$ for some polynomial $p(x)$ of degree $2m-2$ [5, p. 175]. Once $[a,b]$ is found, $p(x)$ can be calculated in a Chebyshev expansion of the second kind so that

$$\psi_{V,n}(x) = \sqrt{(b-x)(x-a)}\sum_{j=0}^{2m-2}\beta_j U_j\left(M(x)\right), \qquad M(x) = \frac{2x-b-a}{b-a},$$

where U_j is the degree j Chebyshev polynomial of the second kind and the coefficients β_j are determined by

$$V'(x) = \pi(b-a)\sum_{j=0}^{2m-2}\beta_j T_{j+1}\left(M(x)\right).$$

The interval $[a,b]$ is selected so that the zero-th Chebyshev coefficient in the above expansion vanishes. The function $\psi_{V,n}(x)$ is also important for evaluating the associated orthogonal polynomials using a RH reformulation (see Section 4.2).

Once $\psi_{V,n}(x)$ has been computed, we compute G_n by first calculating F_n. For any $x \in [a,b]$, $F_n(x)$ is defined by an indefinite integral of $\psi_{V,n}(x)$ so we rewrite the expansion for $\psi_{V,n}(x)$ in terms of the Chebyshev basis of the first kind using the recurrence [17, (18.9.10)]:

$$\psi_{V,n}\left(M^{-1}(x)\right) = \frac{b-a}{2}\frac{1}{\sqrt{1-x^2}}\sum_{j=0}^{2m}\frac{\beta_{j-2}-\beta_j}{2}T_j(x),$$

where $\beta_{-2} = \beta_{-1} = \beta_{2m-1} = \beta_{2m} = 0$ and $T_j(x) = \cos(j\cos^{-1}x)$ is the degree j Chebyshev polynomial of the first kind. (A factor of $(1-x^2)$ appears in the recurrence [17, (18.9.10)], which results in the $(1-x^2)^{-1/2}$ term above.) This allows the indefinite integral to be easily calculated since T_j satisfies the following relation:

$$\int_{-1}^{y} T_j(t)\frac{dt}{\sqrt{1-t^2}} = \begin{cases} -\frac{\sin(j\cos^{-1}y)}{j}, & j>0, \\ \pi - \cos^{-1}y, & j=0, \end{cases} \qquad y\in[-1,1]. \tag{4.2}$$

Therefore, for any $x\in[a,b]$ we have

$$F_n(x) = \int_a^x \psi_{V,n}(t)dt = \frac{(b-a)^2}{4}\int_{-1}^{M(x)}\sum_{j=0}^{2m}\frac{\beta_{j-2}-\beta_j}{2}T_j(t)\frac{dt}{\sqrt{1-t^2}},$$

where the indefinite integral can be calculated by (4.2). The function $G_n(x)$ can now be computed with Newton's method applied to $F_n(x)$ since evaluation of $F_n(x)$ and its derivative can be computed efficiently and accurately.

4.2. Fast evaulation of generalized Hermite polynomials. Evaluating orthogonal polynomials associated to the weight function $e^{-V(x)}dx$ with $V(x) = x^{2m} + \mathcal{O}(x^{2m-1})$ can be achieved using RH techniques. We do not present the full method here as further details can be found in [19, 20, 25]. At its very essence the method solves the following RH problem:

PROBLEM 4.1. *Find $\Phi : \mathbb{C}\setminus\mathbb{R} \to \mathbb{C}^{2\times 2}$ such that Φ is analytic in $\mathbb{C}\setminus\mathbb{R}$ and satisfies*

$$\lim_{\epsilon\to 0^+}\Phi(x+i\epsilon) = \lim_{\epsilon\to 0^+}\Phi(x-i\epsilon)\begin{bmatrix} 1 & e^{-V(xn^{1/(2m)})} \\ 0 & 1 \end{bmatrix},$$
$$\lim_{z\to\infty}\Phi(z)\begin{bmatrix} z^{-n} & 0 \\ 0 & z^n \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Remarkably, it can be shown that $\Phi(z)$ takes the following form [5]:

$$\Phi(z) = \begin{bmatrix} \pi_n(z) & \mathcal{C}_{\mathbb{R}}[\pi_n(\,\cdot\,)e^{-V(\,\cdot\,n^{1/(2m)})}](z) \\ -2\pi i\gamma_{n-1}\pi_{n-1}(z) & -2\pi i\gamma_{n-1}\mathcal{C}_{\mathbb{R}}[\pi_{n-1}(\,\cdot\,)e^{-V(\,\cdot\,n^{1/2m})}](z) \end{bmatrix},$$
$$\mathcal{C}_{\mathbb{R}}[f](z) = \frac{1}{2\pi i}\int_{\mathbb{R}}\frac{f(s)}{s-z}ds,$$

where $\pi_j(x)$ is the degree j monic orthogonal polynomial associated to the weight function $e^{-V(xn^{1/(2m)})}$ and γ_n is a normalization constant, *i.e.*,

$$\gamma_n = \left[\int_{\mathbb{R}}\pi_n^2(x)e^{-V(xn^{1/(2m)})}dx\right]^{-1}.$$

Initially, $\Phi(z)$ has growth in the first column and decay in the second at infinity and this is not immediately tractable for the numerical method of [19]. Instead, define

$$g(z) = \int \log(z-x)\psi_{V,n}(x)dx.$$

so that g has a branch cut on and to the left of the support of $\psi_{V,n}$. It then follows that

$$e^{ng(z)} = z^n + \mathcal{O}(z^{n-1}) \quad \text{as} \quad z \to \infty$$

is analytic away from the support of $\psi_{V,n}$. The function $g(z)$ has many important properties for the asymptotic analysis of Problem 4.1. Two properties that are important for computation are:

- $e^{ng(z)}$ captures the growth of $\pi_n(z)$ at infinity,
- $e^{ng(z)}$ captures the oscillatory behavior of π_n on the support of $\psi_{V,n}$.

The method in [20] provides a numerical solution to the RH problem and returns a function $U : \Gamma(n) \to \mathbb{C}^{2\times 2}$, where U and $\Gamma(n)$ satisfy

$$\begin{aligned} U|_{\Gamma_{n,j}}(z) &= \sum_{N=0}^{m(n)} \alpha_{j,N} T_N\left(\frac{z-d_{n,j}}{c_{n,j}}\right), \\ \Gamma_{n,j} &= c_{n,j}[-1,1] + d_{n,j}, \\ \Gamma(n) &= \Gamma_{n,1} \cup \cdots \cup \Gamma_{n,N}. \end{aligned}$$

Here, the constants $c_{n,j}$ and $d_{n,j}$ are determined by the deformation of the RH problem and $\alpha_{j,k}$ are matrix-valued constants. There exists a matrix-valued function $N(z) : \mathbb{C} \to \mathbb{C}^{2\times 2}$ such that [20]

$$\Phi(z) \approx \tilde{\Phi}(z) = L_n(\mathcal{C}_{\Gamma(n)}[U](z) + I)N(z)L_n^{-1} \begin{bmatrix} e^{ng(z)} & 0 \\ 0 & e^{-ng(z)} \end{bmatrix}, \tag{4.3}$$

$$\mathcal{C}_{\Gamma(n)}[U](z) = \frac{1}{2\pi i}\int_{\Gamma(n)} \frac{U(s)}{s-z}ds,$$

where I is the 2×2 identity matrix and L_n is a constant diagonal matrix

$$L_n = \begin{bmatrix} e^{-n\ell/2} & 0 \\ 0 & e^{n\ell/2} \end{bmatrix}.$$

The constant ℓ is determined from g. Typically, the error made in this approximation is on the order of machine precision. As described in [20], $m(n)$ is bounded as a function of n. The precise form of $N(z)$ can be deduced from [20, Section 4].

Evaluating $\mathcal{C}_{\Gamma(n)}[U](z)$ requires $\mathcal{O}(1)$ operations because $m(n)$ is bounded as a function of n. The other factors in (4.3) can also be evaluated in $\mathcal{O}(1)$ operations resulting in a method to compute $\Phi(z)$ in $\mathcal{O}(1)$ operations.

4.3. Fast evaluation of the derivative of generalized Hermite polynomials. The function U in (4.3) satisfys the so-called zero-sum condition [26, Definition 3.5]. This implies that differentiation commutes with the Cauchy integral operator $\mathcal{C}_{\Gamma(n)}$. For each j, $U|'_{\Gamma_{n,j}}(z)$ is accurately computed with spectral differentiation. Furthermore, $N'(z)$ can be computed accurately. Define

$$T(z) = L_n^{-1}\tilde{\Phi}(z) = (\mathcal{C}_{\Gamma(n)}[U](z) + I)N(z)L_n^{-1} \begin{bmatrix} e^{ng(z)} & 0 \\ 0 & e^{-ng(z)} \end{bmatrix},$$

and then

$$T'(z) = \Big(\mathcal{C}_{\Gamma(n)}[U'](z)N(z) + (\mathcal{C}_{\Gamma(n)}[U](z)+I)N'(z) \\ +(\mathcal{C}_{\Gamma(n)}[U](z)+I)N'(z)\begin{bmatrix} ng'(z) & 0 \\ 0 & -ng'(z)\end{bmatrix}\Big) L_n^{-1}\begin{bmatrix} e^{ng(z)} & 0 \\ 0 & e^{-ng(z)}\end{bmatrix}.$$

Below, only the $(1,1)$ and $(2,1)$ entries of T and T' are needed.

4.4. Newton's method for generalized Gauss–Hermite nodes. In practice, we use Newton's method to find the zeros of

$$r_n(z) = T_{11}(z)e^{-V(zn^{1/(2m)})/2},$$

which, of course, coincide with the zeros of $\pi_n(z)$. Experiments show that the magnitude of $r_n(z)$ is $\mathcal{O}(1)$ and hence it is more covenient for computation. For the kth node, one step of Newton's method takes the form:

$$\tilde{x}_k^{\text{new}} = \tilde{x}_k^{\text{old}} - \frac{T_{11}(\tilde{x}_k^{\text{old}})e^{-V(\tilde{x}_k^{\text{old}}n^{1/(2m)})}}{[T'_{11}(\tilde{x}_k^{\text{old}}) - n^{1/(2m)}V'(\tilde{x}_k^{\text{old}}n^{1/(2m)})T_{11}(\tilde{x}_k^{\text{old}})]e^{-V(\tilde{x}_k^{\text{old}}n^{1/(2m)})}}.$$

The exponential factors cancel out in this fraction but we leave them there as numerically we observe a small improvement in accuracy by doing so. If the update $|\tilde{x}_k^{\text{new}} - \tilde{x}_k^{\text{old}}|$ is sufficiently small, then we terminate Newton's method and the corresponding generalized Gauss–Hermite node is calculated via $x_k \approx \tilde{x}_k^{\text{new}} n^{1/(2m)}$.

4.5. Calculating generalized Gauss–Hermite weights. Once the zeros $\{\tilde{x}_k\}_{k=1}^n$ of $\pi_n(x)$ are known the quadrature weights are found through the formula [15]

$$w_k = \frac{n^{1/(2m)}}{\gamma_{n-1}\pi_{n-1}(\tilde{x}_k)\pi'_n(\tilde{x}_k)} = -\frac{2\pi n^{1/(2m)}}{T'_{11}(\tilde{x}_k)T_{21}(\tilde{x}_k)}. \tag{4.4}$$

The monic polynomial π_n has an exponentially small amplitude and it is convenient to choose a normalization so that the polynomial is typically $\mathcal{O}(1)$ on the support of $\psi_{V,n}$. We have removed left multiplication by L_n in the definition of $T(z)$ so that $T_{11}(z)$ is a more favorable multiple of π_n.

4.6. Subsampling. For each fix n, V, and $\epsilon > 0$, we can find a threshold parameter $\tau_{n,V,\epsilon}$ such that $|w_k| < \epsilon$ if $k < \tau_{n,V,\epsilon}$ or $k > n - \tau_{n,V,\epsilon}$ where ϵ is less than the smallest positive normalized floating-point number in double precision. A node x_k that satisfies $\tau_{n,V,\epsilon} \le k \le 1 - \tau_{n,V,\epsilon}$ is said to be *non-trivial.* If $V(x) = x^{2m} + \mathcal{O}(x^{2m-1})$, we demonstrate below that $\tau_{n,V,\epsilon} = \mathcal{O}(n^{1-1/(2m)})$ and give a method for choosing a constant c so that $cn^{1-1/(2m)} \le \tau_{n,V,\epsilon}$.

It follows from [5, (7.187), (7.84)], assuming the differentiability of the asymptotic formula, that

$$\pi'_{n-1}(x)\pi_n(x) = (nC_n(x) + \mathcal{O}(1))e^{-2n\operatorname{Re} g(x)}, \quad \gamma_{n-1} \sim \frac{1}{\pi 2^{3/2}}e^{n\ell},$$

where $C_n(x) \le C$ for all n. It also follows that $e^{-2n\operatorname{Re} g(x) + n\ell} = e^{nV(x)}$, see [5, (7.49)]. We assume that $1/C_n(x_k)$ is bounded for all k and n and we have

$$w_k \le \frac{D}{n}e^{-nV(x_k)}.$$

With the chosen scaling, $\psi_V(x) = \lim_{n\to\infty} \psi_{V,n}(x)$ gives the asymptotic density of the nodes x_k, which all lie in a finite interval that contains the origin. Let $R_n > 0$ be the largest value such that $\frac{D}{n} e^{-nV(\pm R_n)} \geq \epsilon$ and $R_n = \mathcal{O}(n^{-1/(2m)})$. Since ψ_V is a continuous density we have

$$\int_{-R_n}^{R_n} \psi_V(x)dx = \mathcal{O}(n^{-1/(2m)}),$$

which is an upper bound on the asymptotic fraction of non-trival nodes. We find in practice that we can take $(nC_n(x) + \mathcal{O}(1))^{-1} \leq 1/n$ and if $V(x) = x^{2m}$ this gives

$$R_n \leq \frac{1}{n^{1/(2m)}} \left(\log \epsilon^{-1} + \log n^{-1} + \log(\pi 2^{3/2})\right)^{1/(2m)}.$$

Then

$$\tau_{n,V,\epsilon} \approx \left\lfloor n \int_{-\infty}^{-R_n} \psi_V(x)dx \right\rfloor,$$

where the right-hand side is actually a lower bound for $\tau_{n,V,\epsilon}$ under the assumptions we have put forth.

4.7. Examples. First, we compare the method described in this section with an implementation of the Stieltjes procedure because it appears to be the current method of choice to compute zeros when $V(x)$ is not quadratic [10]. The Stieltjes procedure is used to compute the coefficients in the 3-term recurrence formula that the orthogonal polynomials satisfy. These coefficients are used to construct an $n \times n$ tridiagonal matrix whose eigenvalues are the zeros of the nth-order polynomial. For exponential weights, it appears that a straightforward implementation of this method has computational complexity that grows like n^3 [25]. In Figure 4.1 we demonstrate the efficiency of the method for large n with $V(x) = x^4$. We also note that our algorithm can be run for $V(x) = x^2$ to compare with the method above and errors on the order of 10^{-14} for the nodes are found. We demonstrate quadrature errors in Figure 5.1 with $V(x) = x^8$.

We can also examine the distribution of the zeros $\{\tilde{x}_k\}$ with a histogram. In Figure 4.2 we show the equilibrium measure density $\psi_{V,n}(x)$ for $V(x) = x^8$ overlayed with a histrogram for 100,000 zeros. Note that $\psi_{V,n}(x)$ is actually independent of n because V is a monomial. It is clear from the figure that the distribution of the zeros approximates $\psi_{V,n}(x)$ in the sense of (4.1).

5. Application to Interpolation.

Everything we have described here can be used for barycentric Lagrange interpolation. We use the second form of the barycentric interpolation formula as discussed in [2]:

$$\mathcal{L}_n[f](x) = \sum_{k=1}^{n} \frac{f(x_k)\lambda_j}{x - x_k} / \sum_{k=1}^{n} \frac{\lambda_k}{x - x_k}, \quad \lambda_k = \frac{c}{\pi_n'(x_k)}, \tag{5.1}$$

for any convenient constant[2] c. It is clear, in light of previous discussion, that we can evaluate $\{x_k\}_{k=1}^n$ and $\{\lambda_k\}_{k=1}^n$ in $\mathcal{O}(n)$ operations. For reasons we discuss in Appendix A, we evaluate

$$\mathcal{L}_n[f](x)e^{-V(x)/2}. \tag{5.2}$$

[2]Often, c is chosen so that $\max_k |\lambda_k| = 1$.

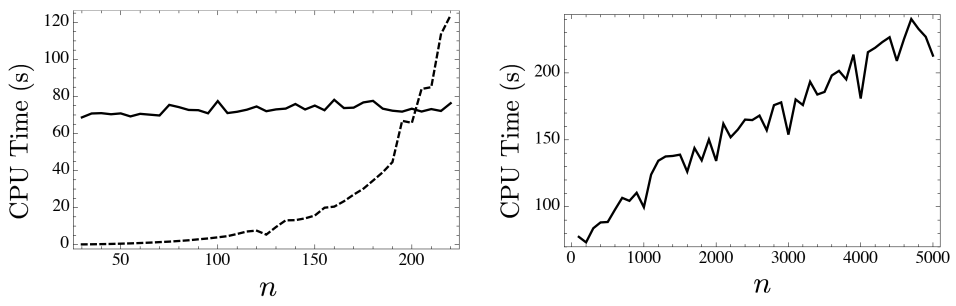

FIG. 4.1. *Left: A comparison of the CPU time required to compute the recurrence coefficients with the Stieltjes procedure (dashed) with the CPU time required to compute the zeros of $\pi_n(x)$ with the approach advocated in this section (solid) when $V(x) = x^4$. We see rapid growth with respect to n for the Stieltjes procedure and linear growth for the Newton's method/RH approach. Right: The CPU time required to compute the zeros of $\pi_n(x)$ with our approach for larger n. There is clear linear growth.*

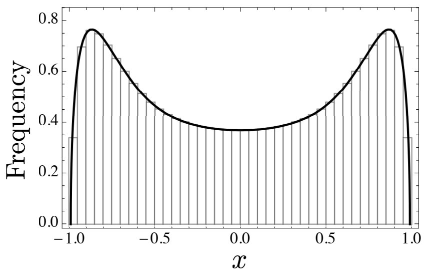

FIG. 4.2. *A histogram for all of the approximated zeros $\{\tilde{x}_k\}$ of $\pi_{100000}(x)$ overlayed with the density $\psi_{V,n}(x)$ when $V(x) = x^8$. The histogram shows that the distribution of the zeros approximates $\psi_{V,n}(x)$ in the sense of* (4.1).

We demonstrate the convergence of an interpolant for $f(x) = e^{\cos(10x)}/(1+25x^2)$ with $V(x) = x^8$ in Figure 5.2.

Conclusions. We have shown the usefulness of a general algorithm for finding quadrature nodes and weights when the associated orthogonal polynomials and their derivatives can be pointwise evaluated in $\mathcal{O}(1)$ operations. The algorithm achieves optimal and state-of-the-art complexity of $\mathcal{O}(n)$ operations to compute n quadrature nodes and weights. In the case of Hermite polynomials the algorithm appears to achieve a smaller constant for this $\mathcal{O}(n)$ term when compared to other existing algorithms. The method extends to quadrature with respect to general weights $e^{-V(x)}$ with the assistance of Riemann–Hilbert problems. We note that while $V(x)$ was chosen to be polynomial here, the extension can be made to entire weights (also known as Erdős weights) such as $V(x) = \cosh x$ [25].

Presumably, extensions of these ideas can be made to the Laguerre weights $w(x) = x^\alpha e^{-V(x)}$ for $x \in (0,\infty)$. Again, we expect the asymptotic expansion to be useful in the classical case of $V(x) = -x$ and the Riemann–Hilbert approach to aid in the general approach. The Riemann–Hilbert approach may also apply to generalized Jacobi-type weights, *i.e.*, $(1-x)^\alpha(1+x)^\beta e^{-V(x)}$ on $[-1,1]$, to extend the work of [13].

The Riemann–Hilbert approach we employed used a numerical method to approximate the polynomials. A possible alternative would be to use asymptotic expansions

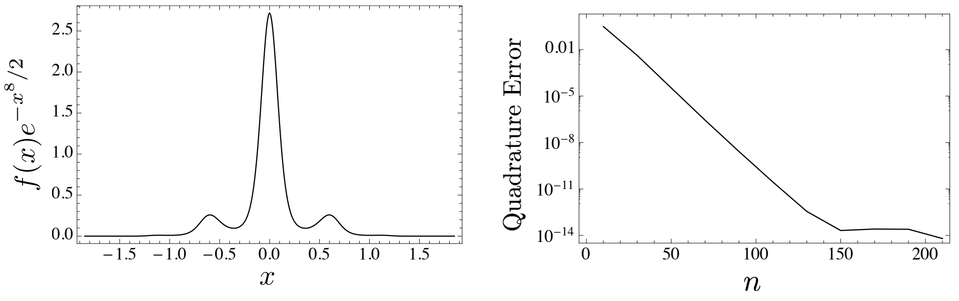

FIG. 5.1. *Left: A plot of $f(x)e^{-V(x)/2}$ for $f(x) = e^{\cos(10x)}/(1+25x^2)$ and $V(x) = x^8$. Right: The convergence of $\sum_{k=1}^{n} f(x_k)w_k$ to the integral of $f(x)e^{-V(x)}$. The result is compared against Clenshaw–Curtis on $[-3,3]$ with 10,000 points.*

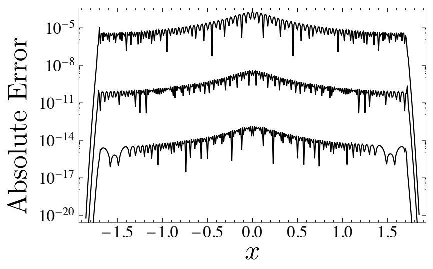

FIG. 5.2. *The absolute error $|\tilde{\mathcal{L}}_n[f](x) - f(x)|e^{-V(x)/2}$ plotted as a function of x for $n = 100, 200, 300$. We see the expected spectral convergence of the approximation. This also validates the accuracy of the barycentric nodes and weights.*

of the polynomials derived from the Riemann–Hilbert approach. An open question remains as to whether a high-order expansion can be computed effectively from the asymptotic Riemann–Hilbert theory.

Acknowledgements. We wish to thank Nick Trefethen for discussing this work with us and to Nick Hale for implementing the Glaser–Lui–Rokhlin algorithm in MATLAB. We acknowledge the generous support of the National Science Foundation through grant NSF-DMS-130318 (TT). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding sources.

Appendix A. Weighted stability of Barycentric interpolation.

In this section we discuss the results of Higham [14] in the context of interpolation on $\mathbb{R}$. Define the condition number for a function f at a point x by

$$\operatorname{cond}(f,x,n) = \lim_{\epsilon\to 0^+} \sup_{|\Delta f| \le \epsilon |f|} \left\{ \left| \frac{\mathcal{L}_n[f](x) - \mathcal{L}_n[f+\Delta f](x)}{\epsilon \mathcal{L}_n[f](x)} \right| \right\}.$$

The main result of [14] is the following theorem.

THEOREM A.1 ([14]). *If u is machine unit roundoff and $\tilde{\mathcal{L}}_n[f](x)$ is the computed value, then*

$$\left| \frac{\mathcal{L}_n[f](x) - \tilde{\mathcal{L}}_n[f](x)}{\mathcal{L}_n[f](x)} \right| \le (3n+4)u \operatorname{cond}(f,x,n) + (3n+2)u \operatorname{cond}(1,x,n) + \mathcal{O}(u^2).$$

This Theorem shows that if $\mathrm{cond}(1, x, n)$ and $\mathrm{cond}(f, x, n)$ do not grow too quickly with respect to n, then the barycentric formula (5.1) is forward stable. On the real line two questions remain:

1. How does $\mathrm{cond}(1, x, n)$ depend on x and n?
2. For what class of functions f is $\mathrm{cond}(f, x, n)$ relatively small?

We begin by noting that $\mathcal{L}_n[\cdot]$ is a linear functional and $\mathcal{L}_n[1](x) = 1$ so that

$$\mathrm{cond}(1, x, n) = \lim_{\epsilon \to 0^+} \sup_{|\Delta f| \le \epsilon} \frac{|\mathcal{L}_n[\Delta f](x)|}{\epsilon} = \sup_{|\Delta f| \le 1} |\mathcal{L}_n[\Delta f](x)| = \sum_{j=1}^{n} |\ell_{j,n}(x)|,$$

where $\ell_{j,n}$ is the Lagrange polynomial of degree n that takes the value of 1 at $\tilde{x}_{j,n}$ and vanishes at all the other nodes. The last equality can be seen by considering $\Delta f(\tilde{x}_{j,n}) = \operatorname{sign} \ell_{j,n}(x)$. In the case of Chebyshev polynomials, this quantity grows like $\log n$, uniformly in $x \in [-1, 1]$. When the domain of interest is the whole real line we must modify the definition slightly as a quick numerical experiment shows that $\mathrm{cond}(1, x, n)$ grows exponentially.

One naïve bound gives

$$\sum_{j=1}^{n} |\ell_{j,n}(x)| e^{-V(x)/2} \le \sum_{j=1}^{n} |\ell_{j,n}(x)| e^{V(\tilde{x}_{j,n})/2 - V(x)/2} =: \Lambda_n(x),$$

where $\Lambda_n(x)$ is the so-called *weighted Lebesgue function* and $\Lambda_n = \sup_{x \in \mathbb{R}} \Lambda_n(x)$ is the *weighted Lebesgue constant* [16]. For our choice of a exponential weighting it follows that $\Lambda_n = \mathcal{O}(n^{1/6})$ [16].

We are led to consider the following weighted relative error:

$$\left| \frac{\mathcal{L}_n[f](x) - \tilde{\mathcal{L}}_n[f](x)}{\mathcal{L}_n[f](x)} \right| e^{-V(x)/2} \\ \le (3n + 4) u \,\mathrm{cond}(f, x, n) e^{-V(x)/2} + C n^{1/6} (3n + 2) u + \mathcal{O}(u^2).$$

Therefore, we achieve forward stability when the errors are damped by the weight $e^{-V(x)/2}$. We interpret this as meaning that the appropriate function space is $L^\infty(\mathbb{R}, e^{-V(x)/2} dx)$ as opposed to $L^\infty([-1, 1])$.

Our final task is to identify a class of functions that is sufficiently regular so that $\mathrm{cond}(f, x, n)$ is well-behaved. By standard approximation theory we can relate the interpolation error to the best approximation error.

$$\| e^{-V(\cdot)/2} (f(\cdot) - \mathcal{L}_n[f](\cdot)) \|_{L^\infty(\mathbb{R})} \le (1 + \Lambda_n) \inf_{\deg P \le n-1} \| e^{-V(\cdot)/2} (f(\cdot) - P(\cdot)) \|_{L^\infty(\mathbb{R})}.$$

For a function that is absolutely continuous and of weighted bounded variation (see [16, Thm. 10.4]), polynomial interpolation converges uniformly on compact sets.

Assuming n is sufficiently large so that $|\mathcal{L}_n[f](x)|^{-1} \le 2|f(x)|^{-1}$ we have

$$\begin{aligned} \mathrm{cond}(f, x, n) &\le \frac{2}{|f(x)|} \sum_{j=1}^{n} |\ell_{j,n}(x)| \epsilon^{-1} |\Delta f(\tilde{x}_{j,n})| \\ &\le \frac{2}{|f(x)|} \sum_{j=1}^{n} |\ell_{j,n}(x)| e^{V(\tilde{x}_{j,n})/2} |\epsilon^{-1} \Delta f(\tilde{x}_{j,n})| e^{-V(\tilde{x}_{j,n})/2} \\ &\le 2 \frac{\| e^{-V(\cdot)/2} f(\cdot) \|_{L^\infty(\mathbb{R})}}{|f(x)|} \Lambda_n(x) e^{V(x)/2} \end{aligned}$$

so that $\mathrm{cond}(f,x,n)e^{-V(x)/2}$ behaves well with respect to both x and n. Not surprisingly, for functions that are well represented by polynomials the weighted barycentric formula (5.2) is forward stable when errors are damped by the exponential function $e^{-V(x)/2}$.

REFERENCES

[1] M. J. Ablowitz and A. S. Fokas, *Complex Variables*, Cambridge University Press, Cambridge, UK, (1997).

[2] J. Berrut and L. Trefethen, *Barycentric Lagrange interpolation*, SIAM Review, 46 (2004), pp. 501–517..

[3] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, *Julia: A Fast Dynamic Language for Technical Computing*, arXiv:1209.5145, (2012).

[4] I. Bogaert, *Iteration-free computation of Gauss–Legendre quadrature nodes and weights*, SIAM J. Sci. Comput., 36 (2014), A1008–A1026.

[5] P. Deift, *Orthogonal Polynomials and Random Matrices: A Riemann–Hilbert Approach*, AMS, 1998.

[6] P. Deift and X. Zhou, *A steepest descent method for oscillatory Riemann–Hilbert problems*, Bull. Amer. Math. Soc., 26 (1992), pp. 119–124.

[7] P. Deift, T. Kriecherbauer, K. T.-R. McLaughlin, S. Venakides, and X. Zhou, *Strong asymptotics of orthogonal polynomials with respect to exponential weights*, Comm. Pure Appl. Math., 52 (1999), pp. 1491–1552.

[8] T. A. Driscoll, N. Hale, and L. N. Trefethen, editors, *Chebfun Guide*, Pafnuty Publications, Oxford, 2014.

[9] L. Gatteschi, *Asymptotics and bounds for the zeros of Laguerre polynomials: a survey*, J. Comput. Appl. Math., 144 (2002), pp. 7–27.

[10] W. Gautschi, *Orthogonal Polynomials: Applications and Computation*, Oxford University Press, (2004).

[11] A. Glaser, X. Liu, and V. Rokhlin, *A fast algorithm for the calculation of the roots of special functions*, SIAM J. Sci. Comput., 29 (2007), pp. 1420–1438.

[12] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.

[13] N. Hale and A. Townsend, *Fast and accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights*, SIAM J. Sci. Comput., 35 (2013), A652–A672.

[14] N. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA J. Numer. Anal., 24 (2004), pp. 547–556.

[15] F. Hildebrand, *Introduction to Numerical Analysis*, Dover, (1989).

[16] D. Lubinsky, *A survey of weighted approximation for exponential weights*, Surveys in Approximation Theory, 3 (2007), pp. 1–105

[17] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, *NIST Handbook of Mathematical Functions*, Cambridge University Press, 2010.

[18] S. Olver, *Computation of equilibrium measures*, J. Approx. Theory, 163 (2011), pp. 1185–1207.

[19] S. Olver, *A general framework for solving Riemann–Hilbert problems numerically*, Numer. Math., 122 (2012), pp. 305–340.

[20] S. Olver and T. Trogdon, *Numerical solution of Riemann–Hilbert problems: random matrix theory and orthogonal polynomials*, Const. Approx., 39 (2014), pp. 101–149.

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd edition, Cambridge University Press, New York, 2007.

[22] E. B. Saff and V. Totik, *Logarithmic Potentials with External Fields*, Springer, New York, (1997).

[23] F. G. Tricomi, *Sul comportamento asintotico dei polinomi di Laguerre*, Ann. Mat. Pura Appl., 28 (1949), pp. 263–289.

[24] A. Townsend, FastGaussQuadrature.jl, `https://github.com/ajt60gaibb/FastGaussQuadrature.jl`, (2014).

[25] T. Trogdon and S. Olver, *A Riemann–Hilbert approach to Jacobi operators and Gaussian quadrature*, submitted, (2013).

[26] T. Trogdon and S. Olver, *Nonlinear steepest descent and the numerical solution of Riemann–Hilbert problems*, Comm. Pure. Appl. Math., to appear.

[27] T. Trogdon, *Riemann–Hilbert Problems, Their Numerical Solution and the Computation of Nonlinear Special Functions*, Ph.D. Thesis, University of Washington, 2013.