A Riemann–Hilbert approach to Jacobi operators and Gaussian quadrature

Thomas Trogdon

trogdon@cims.nyu.edu

Courant Institute of Mathematical Sciences

New York University

251 Mercer St.

New York, NY, 10012, USA

Sheehan Olver

Sheehan.Olver@sydney.edu.au

School of Mathematics and Statistics

The University of Sydney

NSW 2006, Australia

November 22, 2013

Abstract

The computation of the entries of Jacobi operators associated with orthogonal polynomials has important applications in numerical analysis. From truncating the operator to form a Jacobi matrix, one can apply the Golub–Welsh algorithm to compute the Gaussian quadrature weights and nodes. Furthermore, the entries of the Jacobi operator are the coefficients in the three-term recurrence relationship for the polynomials. This provides an efficient method for evaluating the orthogonal polynomials. Here, we present an $\mathcal{O}(N)$ method to compute the first N rows of Jacobi operators from the associated weight. The method exploits the Riemann–Hilbert representation of the polynomials by solving a deformed Riemann–Hilbert problem numerically. We further adapt this computational approach to certain entire weights that are beyond the reach of current asymptotic Riemann–Hilbert techniques.

1 Introduction

Infinite symmetric tridiagonal matrices with positive off-diagonal entries are referred to as Jacobi operators. Such operators have an intimate connection to orthogonal polynomials. Given a positive weight $\mathrm{d}\rho(x) = \omega(x)\,\mathrm{d}x$ on $\mathbb{R}$, ω entire with finite moments, the

Gram–Schmidt procedure produces a sequence of orthogonal polynomials. These polynomials satisfy a three-term recurrence relation, see (1.2). From the coefficients in the recurrence relation, a Jacobi operator

$$
J_\infty(\omega) = \begin{pmatrix}
a_0 & \sqrt{b_1} & & & 0 \\
\sqrt{b_1} & a_1 & \sqrt{b_2} & & \\
& \sqrt{b_2} & a_2 & \sqrt{b_3} & \\
& & \ddots & \ddots & \ddots \\
0 & & & &
\end{pmatrix}
\tag{1.1}
$$

can be constructed. Our aim is to rapidly calculate the entries of this Jacobi operator from the given weight ω.

Computing the Jacobi operator (1.1) is equivalent to computing the coefficients in the three-term recurrence relation (1.2). Gautschi states in [8]:

> The three-term recurrence relation satisfied by orthogonal polynomials is arguably the single most important piece of information for the constructive and computational use of orthogonal polynomials. Apart from its obvious use in generating values of orthogonal polynomials and their derivatives, both within and without the spectrum of the measure, knowledge of the recursion coefficients (i) allows the zeros of orthogonal polynomials to be readily computed as eigenvalues of a symmetric tridiagonal matrix, and with them the all-important Gaussian quadrature rule, (ii) yields immediately the normalization coefficients needed to pass from monic orthogonal to orthonormal polynomials, (iii) opens access to polynomials of the second kind and related continued fractions, and (iv) allows an efficient evaluation of expansions in orthogonal polynomials.

All applications discussed here are merely a consequence of this broad and important fact emphasized by Gautschi.

While the entries of the Jacobi operator can be calculated using the Stieljes procedure (a discretization of the Gram–Schmidt procedure), the complexity to accurately calculate the first N entries is observed to be $\mathcal{O}(N^3)$, see Section 1.1. We develop an alternative to the Stieljes procedure that achieves $\mathcal{O}(N)$ operations: calculate the entries of the Jacobi operator using the formulation of orthogonal polynomials as solutions to Riemann–Hilbert problems (RHPs). Riemann–Hilbert problems are boundary value problems for analytic functions, and over the last 20 years they have played a critical role in the advancement of asymptotic knowledge of orthogonal polynomials [2, 3, 4, 11, 5]. More recently, a numerical approach for solving Riemann–Hilbert problems has been developed [15, 16], which the authors used to numerically calculate the Nth and $(N-1)$st orthogonal polynomials pointwise, allowing for the computation of statistics of random matrices [18], for varying exponential weights of the form $\omega(x; N) = e^{-NV(x)}$. In this paper, we extend this numerical Riemann–Hilbert approach to stably calculate orthogonal polynomials for non-varying weights, as well as the entries of the associated Jacobi operator.

Remark 1.1. The Riemann–Hilbert approach also allows the calculation of $p_N(x)$ pointwise in $\mathcal{O}(1)$ operations, which can be used to achieve an $\mathcal{O}(N)$ algorithm for calculating Gaussian

quadrature rules, *á la* the Glaser–Liu–Rokhlin algorithm [9, 10], whereas the Golub–Welsch algorithm is $\mathcal{O}(N^2)$ operations. While technically an $\mathcal{O}(N)$ algorithm, its applicability is currently limited due to large computational cost of evaluating solutions to RHPs.

Remark 1.2. In many cases one also considers the map from a Jacobi operator to the associated weight. In the case that the elements of a Jacobi operator are unbounded, a self-adjoint extension $\tilde{J}_\infty$ is considered. The spectral measure $\tilde{\rho}$ for $\tilde{J}_\infty$ satisfies

$$\mathbf{e}_1^\mathsf{T}(\tilde{J}_\infty - z)^{-1}\mathbf{e}_1 = \int_{\mathbb{R}} \frac{\mathrm{d}\tilde{\rho}(x)}{x - z}.$$

Here $\mathbf{e}_1$ is the standard basis vector with a one in its first component and zeros elsewhere. It is known that if $\omega(x) = e^{-p(x)}$ for a polynomial p then J_∞ is essentially self-adjoint (*i.e.* $\tilde{J}_\infty$ is unique) and $\omega(x)\,\mathrm{d}x = \mathrm{d}\tilde{\rho}(x)$ [2]. Therefore, from a theoretical standpoint, the map φ from a subset of essentially self-adjoint Jacobi operators to this restricted class of measures is invertible. Viewed in this way, we are concerned with the computation of φ^{-1}, the inverse spectral map.

1.1 Complexity of the Stieljes procedure

The monic polynomials π_n with respect to the weight ω satisfy the three-term recurrence relation

$$\begin{aligned}
\pi_{-1}(x) &= 0, \\
\pi_0(x) &= 1, \\
\pi_{n+1}(x) &= (x - a_n)\pi_n(x) - b_n\pi_{n-1}(x),
\end{aligned} \tag{1.2}$$

where

$$a_n = \langle x\pi_n, \pi_n \rangle \gamma_n \quad \text{and} \quad b_n = \langle x\pi_n, \pi_{n-1} \rangle \gamma_{n-1} \quad \text{for} \quad \langle f, g \rangle = \int f(x)g(x)\omega(x)\,\mathrm{d}x,$$

where γ_n are the normalization constants

$$\gamma_n = \langle \pi_n, \pi_n \rangle^{-1} = \left(\int_{\mathbb{R}} \pi_n^2(x)\omega(x)\,\mathrm{d}x \right)^{-1}.$$

The constants γ_n can be calculated directly from $b_{n,N}$ [8]:

$$\gamma_{n,N}^{-1} = b_n b_{n-1} \cdots b_1 \int \omega(x)\,\mathrm{d}x,$$

though it is convenient to compute them separately from the RHP.

Remark 1.3. In our notation, a_n and b_n are the recurrence coefficients for the monic orthogonal polynomials. The Jacobi operator that follows in (1.1) encodes the recurrence coefficients for the orthonormal polynomials. We also remark that the orthogonal polynomials themselves can be calculated from the recurrence coefficients.

The conventional method for computing recurrence coefficients is the Stieljes procedure, which is a discretized modified Gram–Schmidt method [8]. In this method, one replaces the inner product associated with the weight by an M-point discretization:

$$\langle f, g \rangle \approx \langle f, g \rangle_M = \sum_{j=1}^{M} w_k f(x_k) g(x_k).$$

For example, $\mathbb{R}$ can be mapped to the unit circle $\{|z| = 1\}$ using a Möbius transformation and then the trapezoidal rule with M sample points applied. The three-term recurrence coefficients and the values of the monomial orthogonal polynomial with respect to the discretized inner product can be built up directly:

$$\boldsymbol{p}_0^M = \mathbf{1}, \qquad \boldsymbol{p}_1^M = \boldsymbol{x} - \frac{A_0^M}{\langle \mathbf{1}, \mathbf{1} \rangle_M} \boldsymbol{p}_0^M, \qquad \boldsymbol{p}_{n+1}^M = \boldsymbol{x}\boldsymbol{p}_n^M - \frac{A_n^M}{B_n^M} \boldsymbol{p}_n^M - \frac{B_n^M}{B_{n-1}^M} \boldsymbol{p}_{n-1}^M$$

$$A_n^M = \langle \boldsymbol{x}\boldsymbol{p}_n^M, \boldsymbol{p}_n^M \rangle_M, \quad B_0^M = \langle \mathbf{1}, \mathbf{1} \rangle_M, \qquad B_n^M = \langle \boldsymbol{x}\boldsymbol{p}_n, \boldsymbol{p}_{n-1} \rangle_M,$$

where $\mathbf{1}$ is a vector of M ones, $\boldsymbol{x} = (x_1, \ldots, x_M)^\top$, the product of two vectors is the pointwise product, i.e., $\boldsymbol{x}\boldsymbol{p} := \mathrm{diag}(\boldsymbol{x})\boldsymbol{p}$, and we use the notation $\langle \boldsymbol{f}, \boldsymbol{g} \rangle_M = \boldsymbol{f}^\mathsf{T} \mathrm{diag}(w_1, \ldots, w_M)\boldsymbol{g}$. Noting that B_n^M also equals $\langle \boldsymbol{p}_n^M, \boldsymbol{p}_n^M \rangle_M$, the entries of the Jacobi operator are approximated by

$$a_n \approx a_n^M = \frac{A_n^M}{B_n^M} \qquad \text{and} \qquad b_n \approx b_n^M = \frac{B_n^M}{B_{n-1}^M}.$$

It is shown in [8] that the recurrence coefficients a_n^M, b_n^M obtained from this discrete inner product (via the Stieltjes procedure) converge to a_n, b_n for fixed n as $M \to \infty$. It is stated in [8] that in M must be much larger than n, the order of the polynomial, for a_n^M and b_n^M to accurately approximate a_n and b_n. A significant question is: how much larger must M be? In Figure 1, we show the relative error in calculating the recurrence coefficients of the Hermite weight $\mathrm{e}^{-x^2} \, \mathrm{d}x$ for varying M and n where n is the index of the recurrence coefficient computed. To accurately calculate the first 100 coefficients requires $M \approx 2000$. Experimentally, we observe that we require $M \sim N^2$ to achieve accuracy in calculating the first N recurrence coefficients. Each stage of the Stieljes procedure requires $\mathcal{O}(M)$ operations, hence the total observed complexity is $\mathcal{O}(N^3)$ operations. This compares unfavorably with the $\mathcal{O}(N)$ operations of the RHP approach.

1.2 Varying weights

For stability purposes, it is necessary that we consider weights that depend on a parameter N of the form

$$\mathrm{d}\rho_N(x) = \omega(x; N) \, \mathrm{d}x = \mathrm{e}^{-NV_N(x)} \, \mathrm{d}x \tag{1.3}$$

on $\mathbb{R}$, where $V_N(x) \equiv V(x)$ tends to a nice limit as $N \to \infty$. Here $V_N(x)$ must be entire and have polynomial growth to $+\infty$ as $|x| \to \infty$. For a fixed N, we can stably calculate the nth row for $n \lesssim N$, hence we only consider $n = 0, 1, \ldots, N$. Varying weights of this form are useful in the study of random matrices [2, 18].
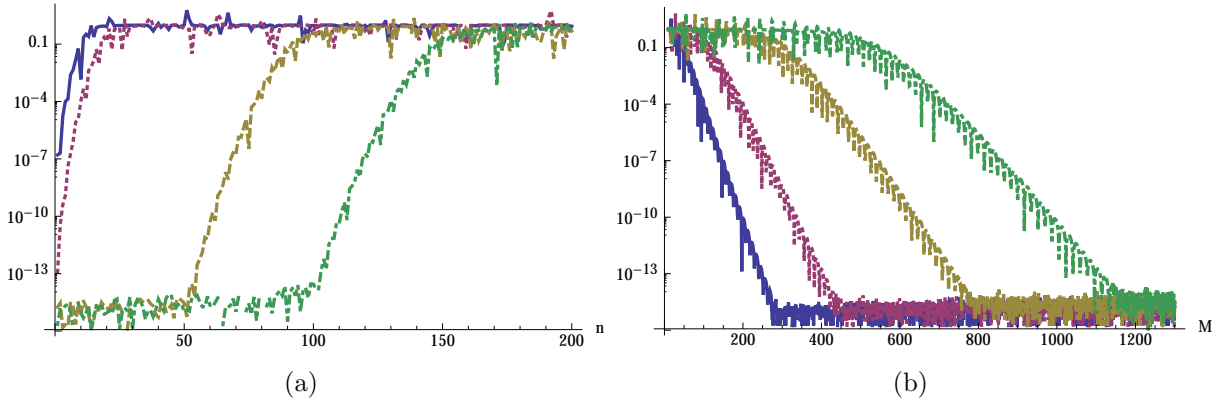
4

Figure 1: Relative error in approximating the Hermite polynomial recurrence coefficients. (a) Varying the number of trapezoidal rule points $M = 50$ (plain), 100 (dotted), 1000 (dashed) and 2000 (dot-dashed). (b) The error in coefficient $n = 10$ (plain), 20 (dotted), 40 (dashed) and 60 (dot-dashed).

For Gaussian quadrature, it is more natural to consider weights of the form $\mathrm{e}^{-Q(x)}\,\mathrm{d}x$. When $Q(x) = qx^m + \mathcal{O}(x^{m-1})$ is a degree m polynomial, a simple change of variables $x \mapsto N^{1/m}x$ is sufficient to put the weight in the required form, where $V_N(x) = N^{-1}Q(N^{1/m}x) \to qx^m$ as $N \to \infty$. When Q is not a polynomial — e.g. an Erdös weight [6] — we do a constant change of variable $x \mapsto \alpha_N x + \beta_N$, but now α_N and β_N are determined numerically in order to control the behaviour of $V_N(x) = N^{-1}Q(\alpha_N x + \beta_N)$, see Section 3.6. Applying this procedure to two entire choices for Q, we observe a slight degradation in computational cost to an estimated count of $\mathcal{O}(N \log^2 N)$ operations to calculate the first N rows of the Jacobi operator, see Appendix A.

Once we calculate the first N rows of the Jacobi operator associated with $\mathrm{d}\rho_N(x) = \mathrm{e}^{-Q(\alpha_N x + \beta_N)}\,\mathrm{d}x$, we can subsequently determine the first N rows of the Jacobi operator of the unscaled weight $\mathrm{d}\rho(x) = \mathrm{e}^{-Q(x)}\,\mathrm{d}x$. First note that if $\pi_{j,N}$ are monic orthogonal polynomials with respect to $\mathrm{d}\rho_N$, then the monic orthogonal polynomials with respect to ρ_N are $\pi_j(x) = \alpha_N^j \pi_{j,N}(\alpha_N^{-1}(x - \beta_N))$. Thus if the entries of the Jacobi operator of ρ_N are given by $a_{n,N}$ and $b_{n,N}$, then a simple calculation verifies that the entries of the original Jacobi operator of ρ are

$$b_n = b_{n,N}\alpha_N^2 \qquad \text{and} \qquad a_n = a_{n,N}\alpha_N + \beta_N. \tag{1.4}$$

The factors in the right-hand sides of (1.4) depend on N while the product does not. It is advantageous to choose N so that $1/c < b_{n,N} < c$ for some $c > 1$ not depending on n or N and the choice $n = N$ is sufficient. The argument for this can be seen by examining the case of the Hermite polynomials where $b_n = n/2$ for $n > 0$. Here $\alpha_{N=n}^2 = n$ so that $b_{n,N=n} = 1/2$ for all n and the growth of b_n is captured exactly by $\alpha_{N=n}^2$. If N is large with respect to n then $b_{n,N} \ll 0$ and the computation can be prone to underflow. Furthermore, errors accumulated in the solution of the RHP are amplified by α_N^2. Our choice for N is one that keeps the "solution" $b_{n,N}$ bounded while keeping the amplification factor α_N^2 minimal.

Remark 1.4. We are implicitly assuming that the equilibrium measure of the scaled potential is supported on a single interval. This is the case when Q is convex, as V_N will also be

5

convex.

1.3 Background: Riemann–Hilbert problems

The method presented here for the computation of recurrence coefficients relies on the numerical solution of a *matrix Riemann–Hilbert problem* (RHP). We refer the reader to [16] for a discussion of the numerical solution of RHPs. In this section we give a brief description of some components of the RHP theory. Given an oriented contour Γ, an RHP consists of finding a sectionally analytic function $\Phi(z) : \mathbb{C} \setminus \Gamma \to \mathbb{C}^{2\times 2}$, depending on n and $\omega(x; N)$, such that

$$\lim_{\substack{z \to x \\ \text{left of } \Gamma}} \Phi(z) = \left(\lim_{\substack{z \to x \\ z \text{ right of } \Gamma}} \Phi(z) \right) G(x; n), \quad G(x; n) : \Gamma \to \mathbb{C}^{2\times 2}.$$

The contour Γ is referred to as the jump contour and G, as the jump matrix. Canonically, RHPs also impose the asymptotic behaviour at ∞: $\lim_{|z|\to\infty} \Phi(z) = I$. For orthogonal polynomials, the initial RHP, Problem 3.1, is reduced to canonical form via a series of transformations described in Section 3.

Of course, the sense in which limits exist needs to be made precise, but this is beyond the scope of this paper, see [2]. We use the notation

$$\Phi^+(x) = \Phi_+(x) = \lim_{\substack{z \to x \\ z \text{ left of } \Gamma}} \Phi(z), \quad \Phi^-(x) = \Phi_-(x) = \lim_{\substack{z \to x \\ z \text{ right of } \Gamma}} \Phi(z),$$

where the $\pm$ may be located in the superscript or subscript. Define

$$\mathcal{C}_\Gamma f(z) = \frac{1}{2\pi i} \int_\Gamma \frac{f(s)}{s - z} ds,$$
$$\mathcal{C}_\Gamma^\pm f(x) = (\mathcal{C}_\Gamma f(x))^\pm.$$

The class RHPs that we consider which satisfy $\lim_{|z|\to\infty} \Phi(z) = I$ have the representation

$$\Phi(z) = I + \mathcal{C}_\Gamma U(z),$$

for a matrix-valued function $U \in L^2(\Gamma)$ [19]. The so-called Plemelj Lemma also holds:

$$\mathcal{C}_\Gamma^+ f - \mathcal{C}_\Gamma^- f = f, \quad f \in L^2(\Gamma).$$

From this it follows that

$$\Phi^+(x) = \Phi^-(x) G(x; n), \quad \lim_{|z|\to\infty} \Phi(z) = I,$$

is equivalent to

$$U - \mathcal{C}_\Gamma^- U \cdot (G - I) = G - I. \tag{1.5}$$

This is a singular integral equation for U and this integral equation is solved numerically with the method in [16].

2 Gaussian Quadrature

In this section we discuss Gaussian quadrature on $\mathbb{R}$. In Gaussian quadrature, one chooses nodes $\{x_j\}_{j=1}^n$ and weights $\{\omega_j\}_{j=1}^n$ such that

$$\int_{\mathbb{R}} p(x)\omega(x)\,\mathrm{d}x = \sum_{j=1}^n p(x_j)\omega_j,$$

for any polynomial $p(x)$ of degree $\leq 2n - 1$.

2.1 Computing nodes and weights

We use the classical Golub–Welsh algorithm to calculate the nodes x_j and weights ω_j from the Jacobi matrix, i.e., $n \times n$ finite section of the Jacobi operator:

$$J_n(\omega) = \begin{pmatrix} a_0 & \sqrt{b_1} & & & & 0 \\ \sqrt{b_1} & a_1 & \sqrt{b_2} & & & \\ & \sqrt{b_2} & a_2 & \sqrt{b_3} & & \\ & & \ddots & \ddots & \ddots & \\ & & & & & \sqrt{b_{n-1}} \\ 0 & & & & \sqrt{b_{n-1}} & a_{n-1} \end{pmatrix}.$$

The eigenvalues of $J_n(\omega)$ are the zeros of $\pi_n(x)$ [8]. For each eigenvalue λ_j of $J_n(\omega)$, an eigenvector is given by

$$v(\lambda_j) = (\pi_0(\lambda_j), \ldots, \pi_{n-1}(\lambda_j))^{\mathsf{T}}.$$

We use $\tilde{v}(\lambda_j)$ to denote the normalized eigenvector, having Euclidean norm one. Let $\mu_0 = \int \omega(x)\,\mathrm{d}x$ and it follows that $\omega_j = \mu_0(\tilde{v}(\lambda_j))_1^2$, i.e. the first component of $\tilde{v}(\lambda_j)$ squared times μ_0 [8]. In practice, μ_0 can be computed with standard quadrature routines, i.e. Clenshaw–Curtis quadrature.

2.2 Example: Hermite

For the Hermite polynomials, $\mu_0 = \beta_0 = \sqrt{\pi}$, $\beta_j = j/2$ for $j > 0$ and $a_j = 0$ always [8]. We use this as a test case to demonstrate the accuracy of our computation. In Figure 2 we show the relative error of the first 700 coefficients computed using the method described below.

2.3 Example: $\omega(x) = e^{-x^8}$

We demonstrate the integration of a meromorphic integrand. In all our examples, we choose to integrate positive functions so that cancellation will not increase our accuracy. See Figure 3 for a demonstration of this.
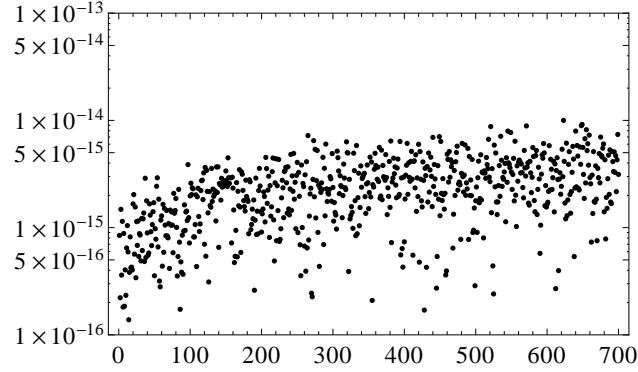
Figure 2: The relative error in b_n for $n = 0, 1, 2, \ldots, 700$. High precision is retained.
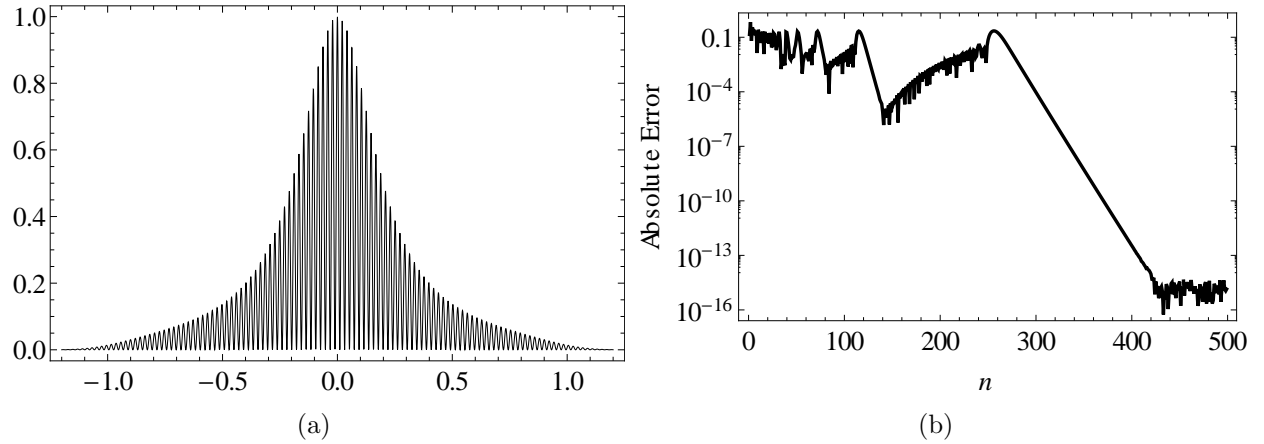

(a)

(b)

Figure 3: (a) The function $f(x)\omega(x)$ for $f(x) = \cos^2(150x)/(25x^2 + 1)$ and $\omega(x) = e^{-x^8}$. (b) The error in the approximation of $\int f(x)\omega(x)\,dx$ with a varying number of nodes and weights. The approximation is compared against an approximation using Clenshaw–Curtis quadrature with 10,000 points.
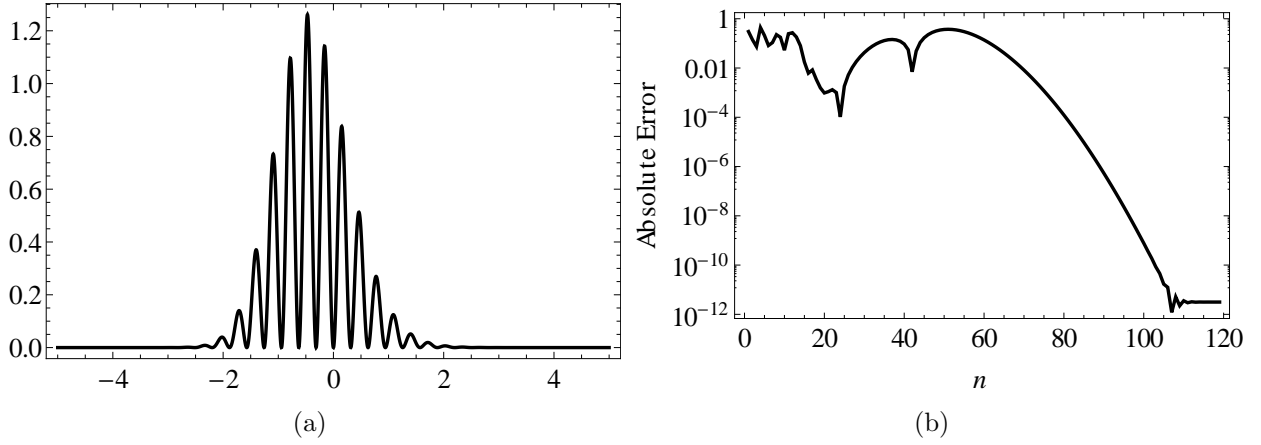
8

Figure 4: (a) The function $f(x)\omega(x)$ for $f(x) = \sin^2(10x)$ and $\omega(x) = e^{-x^2-\sin(x)}$. (b) The error in the approximation of $\int f(x)\omega(x)\,\mathrm{d}x$ with a varying number of nodes and weights. The approximation is compared against an approximation using Clenshaw–Curtis quadrature with 10,000 points.

2.4 Example: $\omega(x) = e^{-x^2-\sin(x)}$

A more exotic choice is $Q(x) = x^2 + \sin(x)$. Even though Q is not a polynomial and it is not convex, numerical experiments demonstrate that the method described below to compute the recurrence coefficients is still effective. We demonstrate using this weight with Gaussian quadrature in Figure 4.

Remark 2.1. The only reason the method would fail on this weight would be if the equilibrium measure (see Section 3) was not supported on a single interval.

2.5 Example: $\omega(x) = e^{-\cosh(x)}$

With the choice $Q(x) = \cosh(x)$, we have super-exponential decay. We include this example to show that the method extends to Q that have super-polynomial growth. Indeed, a modification of the method is needed, see Section 3.6. We demonstrate using this weight with Gaussian quadrature in Figure 5.

3 The numerical RH approach to OPs

Here we present a concise adaptation of the numerical RH approach to computing orthogonal polynomials on $\mathbb{R}$ with analytic exponential weights (originally described in [18]) to calculating *all* orthogonal polynomials of order $n = 0, 1, \ldots, N$. For given n and N, the RHP simultaneously captures the polynomials $\pi_{n,N}(x)$ and $\gamma_{n-1,N}\pi_{n-1,N}(x)$. More precisely, these polynomials can be expressed in terms of the solution of an RHP:
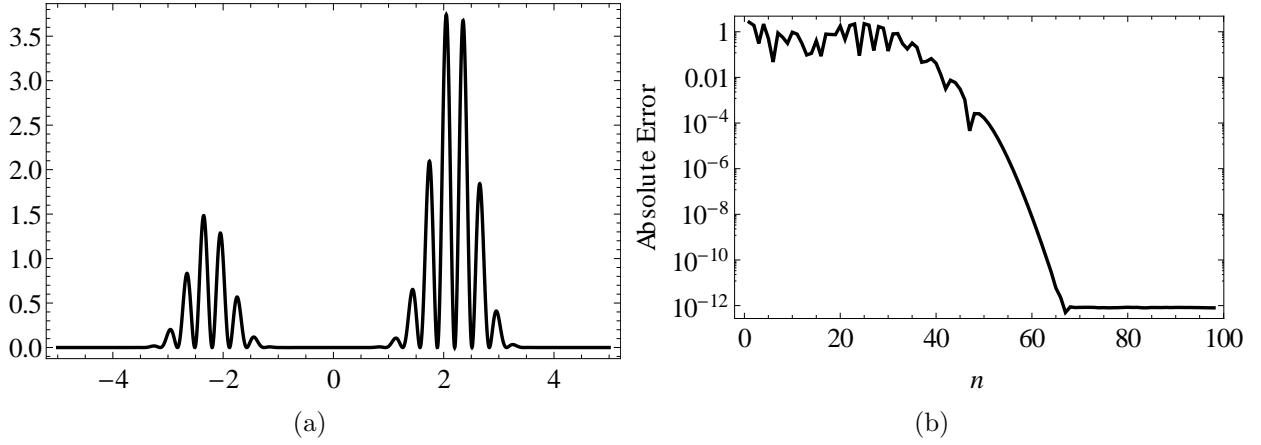
Figure 5: (a) The function $f(x) = 0.1(x^{10} + x^9)\sin^2(10x)e^{-\cosh(x)}$. (b) The error in the approximation of $\int f(x)\,\mathrm{d}x$ with a varying number of nodes and weights. The approximation is compared against an approximation using Clenshaw–Curtis quadrature with 10,000 points.

Problem 3.1. *[7] The function*

$$Y(z) = \begin{pmatrix} \pi_{n,N}(z) & \mathcal{C}_{\mathbb{R}}[\pi_{n,N}e^{-NV}](z) \\ -2\pi i\gamma_{n-1,N}\pi_{n-1,N}(z) & -2\pi i\gamma_{n-1,N}\mathcal{C}_{\mathbb{R}}[\pi_{n-1,N}e^{-NV}](z) \end{pmatrix},$$

where

$$\gamma_{n-1,N} = \left[\int \pi_{n-1,N}^2(x)e^{-NV(x)}\,\mathrm{d}x\right]^{-1},$$

is the unique solution of the RHP on the real line

$$Y^+(z) = Y^-(z)\begin{pmatrix} 1 & e^{-NV(z)} \\ 0 & 1 \end{pmatrix}, \quad z \in \mathbb{R}, \quad Y \sim \begin{pmatrix} z^n & 0 \\ 0 & z^{-n} \end{pmatrix}, \quad z \to \infty.$$

We apply the numerical method described in [16] but conditioning steps are required. Our first task is to remove the growth/decay of Y at ∞. To accomplish this, we must compute the g-function, which has logarithmic growth at infinity so that $e^{\pm ng(z)} \sim z^{\pm n}$, but has special jump properties so that its incorporation into the RHP allows the RHP to behave well in the large-n limit.

The g-function, which depends on n and N, is associated with the equilibrium measure of $V(x)$:

Definition 3.1. The *equilibrium measure* $\mu \equiv \mu_{n,N}$ is the minimizer of

$$\iint \log\frac{1}{|x-y|}\,\mathrm{d}\mu(x)\,\mathrm{d}\mu(y) + \int \frac{N}{n}V(x)\,\mathrm{d}\mu(x).$$

For simplicity, we assume that the equilibrium measure of $V(x)$ is supported on a single interval (a, b). This is certainly true provided V is convex [2]. A numerical method for computing equilibrium measures is presented in [14] and we leave the details to that reference.

10

A discussion of how the computational cost of computing the equilibrium measure varies with N and n is left to Appendix A.

From [2] it follows that $g \equiv g_{n,N}$ is an anti-derivative of the Stieljes transform of μ:

$$g(z) = \int^z \int \frac{\mathrm{d}\mu(x)}{z - x} \, \mathrm{d}z.$$

Therefore, g is analytic off $(-\infty, b)$ and satisfies

$$g_+(z) + g_-(z) = \frac{N}{n} V(z) - \ell, \quad z \in (a, b), \quad \ell \in \mathbb{C},$$
$$g(z) = \log z + \mathcal{O}(z^{-1}) \quad \text{as} \quad z \to \infty,$$
$$g_+(z) - g_-(z) = 2\pi i, \quad z \in (-\infty, a).$$

The method of [14] gives a simple representation of g in terms of the Chebyshev coefficients of V over (a, b) and elementary functions. Numerically, the Chebyshev coefficients of V' are replaced with a numerical approximation via the discrete cosine transform applied to V' evaluated at Chebyshev points, and the g associated with the resulting polynomial is calculated in closed form. The error introduced by this approximation is uniformly small throughout the complex plane, including for z close to or on $\mathbb{R}$.

3.1 Lensing the RHP

We rewrite $Y(z)$ to normalize the behavior at infinity:

$$Y(z) = \begin{pmatrix} e^{\frac{n\ell}{2}} & 0 \\ 0 & e^{-\frac{n\ell}{2}} \end{pmatrix} T(z) \begin{pmatrix} e^{-ng(z)} & 0 \\ 0 & e^{ng(z)} \end{pmatrix} \begin{pmatrix} e^{-\frac{n\ell}{2}} & 0 \\ 0 & e^{\frac{n\ell}{2}} \end{pmatrix}, \tag{3.1}$$

so that $T(z) \sim I$ for large z and $T(z)$ has a jump along the real line, on which it satisfies

$$T_+(z) = T_-(z) \begin{pmatrix} e^{n(g(z)-g_+(z))} & e^{n(g_+(z)+g_-(z)+\ell-\frac{N}{n}V(z))} \\ 0 & e^{n(g_+(z)-g_-(z))} \end{pmatrix}$$

$$= T_- \begin{cases} \begin{pmatrix} 1 & e^{n(g_+(z)+g_-(z)+\ell-\frac{N}{n}V(z))} \\ 0 & 1 \end{pmatrix} & \text{if } z < a, \\[3mm] \begin{pmatrix} e^{n(g_-(z)-g_+(z))} & 1 \\ 0 & e^{n(g_+(z)-g_-(z))} \end{pmatrix} & \text{if } a < z < b, \\[3mm] \begin{pmatrix} 1 & e^{n(2g(z)+\ell-\frac{N}{n}V(z))} \\ 0 & 1 \end{pmatrix} & \text{if } b < z, \end{cases}$$

We appeal to properties of equilibrium measures [2, (7.57)] to assert that
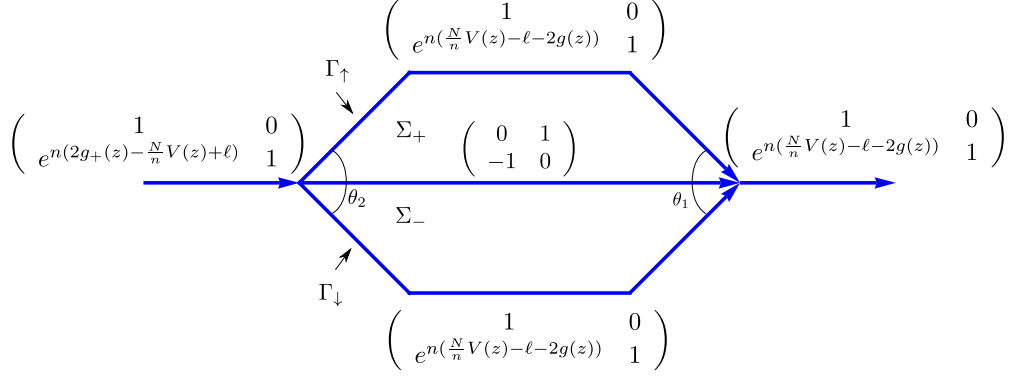
$$g_+(x) + g_-(x) + \ell - \frac{N}{n} V(x) < 0,$$

11

$$\begin{pmatrix} 1 & 0 \\ e^{n(\frac{N}{n}V(z)-\ell-2g(z))} & 1 \end{pmatrix}$$

$\Gamma_\uparrow$

$$\begin{pmatrix} 1 & 0 \\ e^{n(2g_+(z)-\frac{N}{n}V(z)+\ell)} & 1 \end{pmatrix} \qquad \Sigma_+ \qquad \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 \\ e^{n(\frac{N}{n}V(z)-\ell-2g(z))} & 1 \end{pmatrix}$$

θ_2 $\qquad \theta_1$

Σ_-

$\Gamma_\downarrow$

$$\begin{pmatrix} 1 & 0 \\ e^{n(\frac{N}{n}V(z)-\ell-2g(z))} & 1 \end{pmatrix}$$

Figure 6: The jumps of S with the angles θ_1 and θ_2 labelled.

for $x < a$ and $x > b$, thus those contributions of the jump matrix are isolated around a and b. On the other hand, $g_+ - g_-$ is imaginary between a and b [2, pp. 195], hence $e^{\pm n(g_+ - g_-)}$ is oscillatory on (a, b). The RHP is deformed into the complex plane to convert oscillations into exponential decay. A so-called lensing of the RHP is introduced in Figure 6, where we rewrite T as

$$T(z) = S(z) \begin{cases} \begin{pmatrix} 1 & 0 \\ e^{n(\frac{N}{n}V(z)-\ell-2g(z))} & 1 \end{pmatrix}, & \text{if } z \in \Sigma_+, \\[2mm] \begin{pmatrix} 1 & 0 \\ e^{n(\frac{N}{n}V(z)-\ell-2g(z))} & 1 \end{pmatrix}, & \text{if } z \in \Sigma_-, \\[2mm] I, & \text{otherwise.} \end{cases} \tag{3.2}$$

By substituting

$$g_+(z) = \frac{N}{n}V(z) - g_-(z) - \ell,$$

within the support of μ we see that the oscillations have been removed completely from the support:

$$S_+(z) = T_+(z) \begin{pmatrix} 1 & 0 \\ -e^{n(\frac{N}{n}V(z)-\ell-2g_+(z))} & 1 \end{pmatrix} = T_+(z) \begin{pmatrix} 1 & 0 \\ -e^{n(g_-(z)-g_+(z))} & 1 \end{pmatrix}$$

$$= T_-(z) \begin{pmatrix} e^{n(g_-(z)-g_+(z))} & 1 \\ 0 & e^{n(g_+(z)-g_-(z))} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -e^{n(g_-(z)-g_+(z))} & 1 \end{pmatrix}$$

$$= T_-(z) \begin{pmatrix} 0 & 1 \\ -1 & e^{n(g_+(z)-g_-(z))} \end{pmatrix}$$

$$= S_-(z) \begin{pmatrix} 1 & 0 \\ -e^{n(\frac{N}{n}V(z)-\ell-2g_-(z))} & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & e^{n(\frac{N}{n}V(z)-\ell-2g_-(z))} \end{pmatrix}$$

$$= S_-(z) \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

However, we have introduced new jumps on $\Gamma_\uparrow$ and $\Gamma_\downarrow$, on which

$$S_+(z) = T_+(z) = T_-(z) = S_-(z) \begin{pmatrix} 1 & 0 \\ e^{n(\frac{N}{n}V(z)-\ell-2g(z))} & 1 \end{pmatrix}.$$

The choices of θ_1 and θ_2 in Figure 6 are discussed below.

3.2 Removing the connecting jump

The above process has shifted oscillations to exponential decay. However, the goal of these deformations of the RHP is to maintain accuracy of the numerical algorithm for large n. Following the heuristics set out in [17] we must isolate the jumps to neighborhoods of the endpoints a and b by removing the jump along (a, b). We introduce a *global parametrix* that solves the RHP exactly on this single contour. In other words, we require a function which satisfies the following RHP:

$$\mathcal{N}_+(x) = \mathcal{N}_-(x) \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad \text{for } a < x < b \text{ and } \mathcal{N}(\infty) = I.$$

The solution is [2]

$$\mathcal{N}(z) = \frac{1}{2\nu(z)} \begin{pmatrix} 1 & i \\ -i & 1 \end{pmatrix} + \frac{\nu(z)}{2} \begin{pmatrix} 1 & -i \\ i & 1 \end{pmatrix} \quad \text{for } \nu(z) = \left(\frac{z-b}{z-a} \right)^{1/4};$$

i.e., $\nu(z)$ is a solution to

$$\nu_+(x) = i\nu_-(x) \quad \text{for } a < x < b \text{ and } \nu(\infty) = 1.$$

An issue with using $\mathcal{N}(z)$ as a parametrix is that it introduces singularities at a and b, hence we also introduce *local parametrices* to avoid these singularities. Generically, the equilibrium measure $\psi(x)$ has *square-root decay* at the endpoints of its support, and so parametrices based on Airy functions could be employed [2]. However, it is possible that the equilibrium measure has higher-order decay at the endpoints of its support, in which case the local parametrices are given only in terms of a RHP [1]. To avoid this issue, we introduce the *trivially constructed local parametrices* which satisfy the jumps of S in neighborhoods of a and b:

$$P_a(z) = \left(\left\{ \begin{cases} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} & \text{if } \pi - \theta_2 < \arg(z-a) < \pi \\ \begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix} & \text{if } -\pi < \arg(z-a) < -\pi + \theta_2 \\ \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} & \text{if } -\pi + \theta_2 < \arg(z-a) < 0 \\ I & \text{otherwise} \end{cases} \right) \begin{pmatrix} e^{n(\frac{N}{n}V(z)-\ell-2g(z))} & 0 \\ 0 & e^{-n(\frac{N}{n}V(z)-\ell-2g(z))} \end{pmatrix} \right),$$

and

$$P_b(z) = \left(\left\{\begin{array}{ll}\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} & \text{if } \theta_1 < \arg(z-b) < \pi \\ \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix} & \text{if } -\pi < \arg(z-b) < -\theta_1 \\ \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} & \text{if } -\theta_1 < \arg(z-b) < 0 \\ I & \text{otherwise} \end{array}\right.\right) \begin{pmatrix} e^{n(\frac{N}{n}V(z)-\ell-2g(z))} & 0 \\ 0 & e^{-n(\frac{N}{n}V(z)-\ell-2g(z))} \end{pmatrix},$$

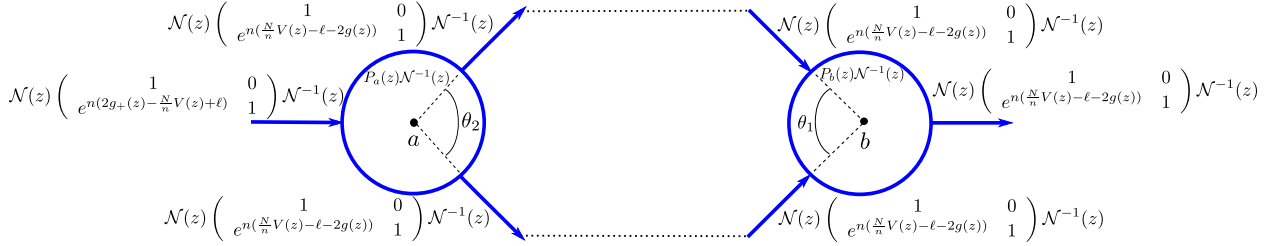where the angles θ_1 and θ_2 are determined in Section 3.3.

Figure 7: The jumps of Φ. We use a counter-clockwise orientation on the circles about a and b.

We write

$$S(z) = \Phi(z) \begin{cases} \mathcal{N}(z) & \text{if } |z-a| > r \text{ and } |z-b| > r, \\ P_b(z) & \text{if } |z-b| < r, \\ P_a(z) & \text{if } |z-a| < r. \end{cases} \tag{3.3}$$

The final RHP for Φ satisfies the jumps depicted in Figure 7. Note that in general r, the radius of the circles in Figure 7, depends on N. We discuss this in more detail in the next section.

In practice, we do not use infinite contours. We truncate contours when the jump matrix is to machine precision the identity matrix and this can be justified following arguments in [17]. In all cases we consider here, after proper deformations the jump matrices are infinitely smooth and are exponentially decaying to the identity matrix for large z.

3.3 Contour scaling and truncation

Evaluating the functions that appear in the jump matrix of the RHP for Φ is not expensive due to the efficiency of the method for computing $g(z)$. This allows us to perform an adaptive truncation of the jump matrices. Before we discuss this, we must discuss the radius of the circles in Figure 7.

In the case of a non-degenerate (generic case) equilibrium measure, $\frac{N}{n}V(z) - \ell - 2g(z) \sim c_a(z-a)^{3/2}$ as $z \to a$ and $\frac{N}{n}V(z) - \ell - 2g \sim c_b(z-b)^{3/2}$ as $z \to b$. In this case, we scale the contours like $n^{-2/3}$:

$$\Omega_n^1 = -n^{-2/3}\Omega^0 + a \quad \text{and} \quad \Omega_n^2 = n^{-2/3}\Omega^0 + b,$$
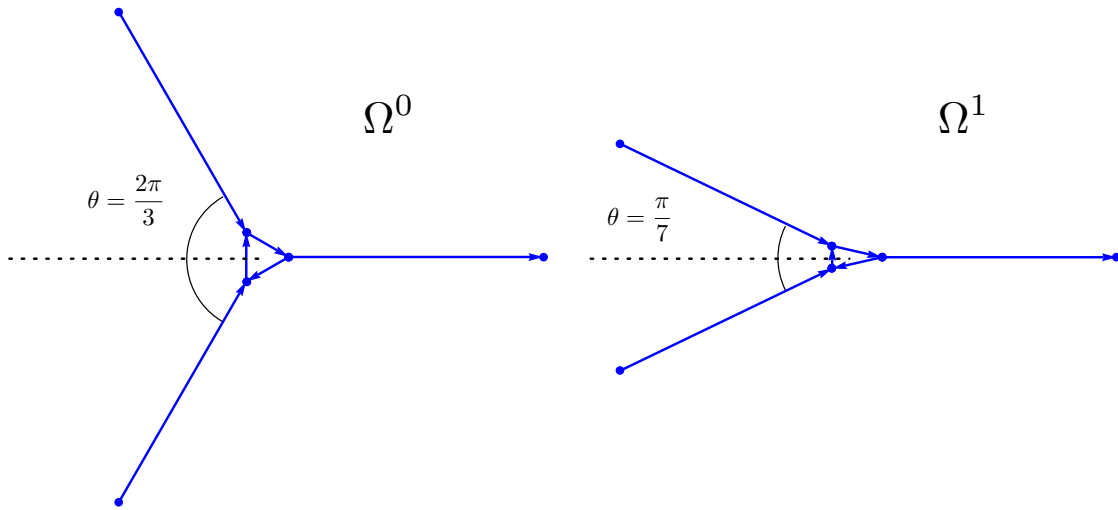
Figure 8: The pre-scaled Ω^0 used for non-degenerate endpoints and the pre-scaled Ω^1 used for first-order degenerate endpoints. These are the contours that are used in practice.

for a fixed domain Ω^0 that is used in practice is depicted in the left graph of Figure 8, and the angle of the contours are chosen to match the direction of steepest descent. This implies that $r \sim n^{-2/3}$. In the first-order degenerate case (e.g., $V(x) = x^2/5 - 4x^3/15 + x^4/20 + 8x/5$), $\frac{N}{n}V(z) - \ell - 2g(z) \sim c_b(z-b)^{7/2}$ as $z \to b$ and we scale like $n^{-2/7}$ (implying $r \sim n^{-2/7}$) at the degenerate endpoint:

$$\Omega_n^1 = -n^{-2/3}\Omega^0 + a \quad \text{and} \quad \Omega_n^2 = n^{-2/7}\Omega^1 + b,$$

where Ω^1 is depicted in the right graph of Figure 8 (the angle is sharper to attach to the new direction of steepest descent). In general, θ_1 and θ_2 in Figure 6 are chosen based on the angles for Ω^0 and Ω^1. Higher order degenerate points can only take the form $\frac{N}{n}V(z) - \ell - 2g(z) \sim c_b(z-b)^{(3+4\lambda)/2}$ for integer λ [1], and can be treated similarly by scaling a suitably constructed contour like $n^{-2/(3+4\lambda)}$.

Each non-self-intersecting component Ω of the contour in Figure 7 is given by a transformation $\Omega = M([-1,1])$ [16]. After truncation, it is often convenient to replace arcs with straight lines. Therefore, we assume that each $M(x) = \alpha x + \beta$ is just an affine transformation. Given a jump matrix G defined on Ω we wish to truncate the contour when $\|G(k) - I\| < \epsilon$.

Algorithm 3.1. *Contour truncation*

1. *Fix $N_{\mathrm{grid}} > 0$ and consider the values*

$$\|G(k_1) - I\|, \|G(k_2) - I\|, \ldots, \|G(k_{M_{\mathrm{grid}}}) - I\|,$$

$$k_j = M\left(2\frac{j}{M_{\mathrm{grid}}} - 1\right), \quad M_{\mathrm{grid}} = \lceil (|\alpha| + 1)N_{\mathrm{grid}} \rceil.$$

Here $\{k_j\}$ represents a uniform grid on Ω that increases in number of points as Ω increases in extent. There is a simple ordering on Ω: we say $s < t$, $s, t \in \Omega$ if and only if $M^{-1}(s) < M^{-1}(t)$.

2. If $\|G(k_1) - I\| \geq \epsilon$ set $S = \{k_1\}$ otherwise $S = \{\}$.

3. For $j = 1, \ldots, M_{\text{grid}} - 1$, if $(\|G(k_j) - I\| - \epsilon)(\|G(k_{j+1}) - I\| - \epsilon) < 0$, indicating a sign change, add $(k_j + k_{j+1})/2$ to the set S.

4. If S has an odd number of elements add $k_{M_{\text{grid}}}$ to S.

5. Order $S = \{t_1, \ldots, t_{2n}\}$ according to $t_j < t_{j+1}$. Replace Ω with the contours $\{\Omega_1, \ldots, \Omega_n\}$ where Ω_j is a straight line that connects t_{2j-1} and t_{2j}.

Provided that N_{grid} is sufficiently large so that at most one zero of $\|G(k_j) - I\| - \epsilon$ lies between successive points k_j and k_{j+1} the algorithm produces a consistent truncation of the contour in the sense that

$$\|G(k) - I\| \leq \sqrt{\frac{\sup \partial_k \|G(k) - I\|^2}{(|\alpha| + 1)N_{\text{grid}}} + \epsilon^2} \text{ for } k \in \Omega \setminus \{\Omega_1 \cup \cdots \cup \Omega_n\}.$$

If $\|G(k) - I\|$ happens to oscillate wildly as n increases then N_{grid} may have to increase with respect to n. In practice, $\|G(k) - I\|$ is seen to either be monotonic or vary slowly. Indeed, this is the purpose of the deformation of an RHP. In both cases, the algorithm produces a good truncation. As a product of this, contours are generally divided up into at most three components after truncation, see Figure 9. The number of mapped Chebyshev points on Ω used to discretize the singular integral equation is used on each of the subdivided contours to ensure enough points. In practice, this procedure ensures accuracy for all n and N.

3.4 Numerically solving the RHP

The RHP for Φ, after scaling and truncation is solved numerically. To avoid too much complication, we only describe the essence of the method [16] that is applied. This method discretizes (1.5) via collocation using an appropriately defined Chebyshev basis. As is described in [18] the scaling and truncation is sufficient to ensure that the number of collocation points needed to compute the solution accurately is bounded (with respect to n). Thus to solve the RHP for Φ the computational cost is $\mathcal{O}(1)$. Once Φ is calculated, we can evaluate orthogonal polynomials pointwise by undoing the sequence of transformations to determine Y. However, we will see that the recurrence coefficients are also calculable directly.

3.5 Computing recurrence coefficients

For a given $V(x)$, we discuss how to accurately obtain $a_{n,N}$ and $b_{n,N}$. For large z, the solution Φ of the RHP in Figure 7 is given by

$$\Phi(z) = T(z)\mathcal{N}^{-1}(z) = \begin{pmatrix} e^{-n\ell/2} & 0 \\ 0 & e^{n\ell/2} \end{pmatrix} Y(z) \begin{pmatrix} e^{n\ell/2} & 0 \\ 0 & e^{-n\ell/2} \end{pmatrix} \begin{pmatrix} e^{ng(z)} & 0 \\ 0 & e^{-ng(z)} \end{pmatrix} \mathcal{N}^{-1}(z).$$
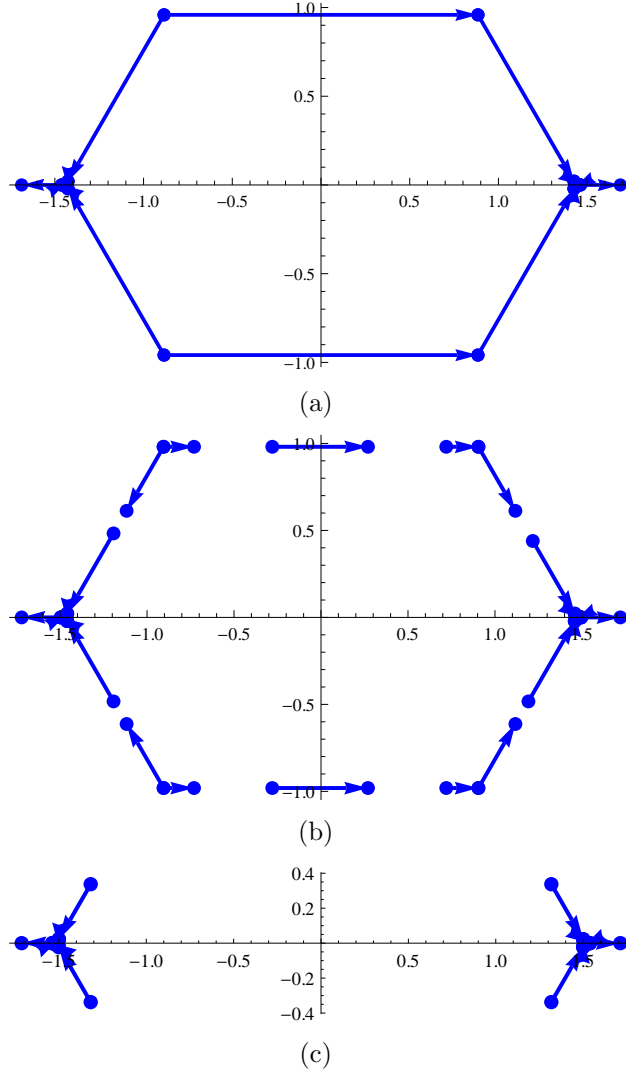
Figure 9: A demonstration of the resulting contours after the application of Algorithm 3.1 with $V(x) = x^8$ and $N = 1$: (a) $n = 20$, (b) $n = 24$, (c) $n = 30$.

For fixed N, we wish to compute the polynomials, orthogonal with respect to $e^{-NV(x)}\,dx$ of order $n = 0, 1, \ldots, N$. From [11] we know that

$$b_{n,N} = (Y_1(n))_{12}(Y_1(n))_{21}, \quad a_{n,N} = \frac{(Y_2(n))_{12}}{(Y_1(n))_{12}} - (Y_1(n))_{22},$$

$$Y(z) = \left(I + \frac{1}{z}Y_1(n) + \frac{1}{z^2}Y_2(n) + \mathcal{O}\left(\frac{1}{z^3}\right) \right) \begin{pmatrix} z^n & 0 \\ 0 & z^{-n} \end{pmatrix}.$$

It is inconvenient to compute $Y_2(n)$ so we look for alternate formulae for $a_{n,N}$ and $b_{n,N}$. From [2, p. 47]

$$a_{n,N} = (Y_1(n))_{11} - (Y_1(n+1))_{11},$$
$$b_{n,N} = (Y_1(n))_{12}(Y_1(n))_{21}.$$

Computing $Y_1(n+1)$ it is more accurate than computing $Y_2(n)$. It is also more efficient when the sequence $a_{j,N}$, $j = 0, 1, 2, \ldots$ is computed with the computation reused at each step.

To compute $Y_1(n)$ we use the formula

$$I + \frac{Y_1(n)}{z} + \mathcal{O}\left(\frac{1}{z^2}\right)$$

$$= \begin{pmatrix} e^{n\ell/2} & 0 \\ 0 & e^{-n\ell/2} \end{pmatrix} \left(I + \frac{\Phi_1(n)}{z} + \mathcal{O}\left(\frac{1}{z^2}\right) \right) \left(I + \frac{\mathcal{N}_1}{z} + \mathcal{O}\left(\frac{1}{z^2}\right) \right)$$

$$\times \left(I + \begin{pmatrix} -ng_1 & 0 \\ 0 & ng_1 \end{pmatrix} \frac{1}{z} + \mathcal{O}\left(\frac{1}{z^2}\right) \right) \begin{pmatrix} e^{-n\ell/2} & 0 \\ 0 & e^{n\ell/2} \end{pmatrix},$$

$$\Phi(z) = I + \frac{\Phi_1(n)}{z} + \mathcal{O}\left(\frac{1}{z^2}\right),$$

$$\mathcal{N}(z) = I + \frac{\mathcal{N}_1(n)}{z} + \mathcal{O}\left(\frac{1}{z^2}\right),$$

$$g(z) = \log z + \frac{g_1}{z} + \mathcal{O}\left(\frac{1}{z^2}\right).$$

Therefore

$$Y_1(n) = \begin{pmatrix} e^{n\ell/2} & 0 \\ 0 & e^{-n\ell/2} \end{pmatrix} \left(\Phi_1(n) + \mathcal{N}_1 + \begin{pmatrix} -ng_1 & 0 \\ 0 & ng_1 \end{pmatrix} \right) \begin{pmatrix} e^{-n\ell/2} & 0 \\ 0 & e^{n\ell/2} \end{pmatrix}$$

Here $\mathcal{N}_1$ can be computed explicitly in terms of the endpoints of the support of the equilibrium measure. Φ_1 is determinable via a contour integral once the RHP for Φ is solved. Recall, the method in [16] returns U such that

$$I + \frac{1}{2\pi i} \int_\Gamma \frac{U(s)}{s - z}\,ds \approx \Phi(z).$$

Then

$$\Phi_1(n) \approx -\frac{1}{2\pi i} \int_\Gamma U(s)\,ds.$$

18

Finally,

$$g_1 = \frac{1}{2\pi i} \int_a^b x \, d\mu(x).$$

When $V(x)$ is polynomial the computational cost to compute any given recurrence coefficient $\mathcal{O}(1)$. When $V(x)$ is entire, but non-polynomial, we obtain conjectures on the computational cost. A full discussion of this is found in Appendix A.

3.6   Regularizing exponential growth

The method in the previous section fails for non-polynomial Q, for example $Q(x) = \cosh(x)$. For polynomial Q, the effect of $Q(x) \mapsto V_N(x)$ is to force the support of the equilibrium measure to be $\mathcal{O}(1)$. Thus, when $Q(x)$ grows faster than any polynomial it is no longer clear a priori what the correct scaling is. We look for functions $\alpha_N > 0$ and β_N such that the equilibrium measure for

$$V_N(x) = N^{-1} Q(\alpha_N x + \beta_N), \quad N > 1,$$

has support $[-1, 1]$. A simple root-finding method can be set up to find α_N and β_N, and if Q symmetric then $\beta_N = 0$. This indicates that the choice $m = \log(N)/\log(\alpha_N)$ ($m = 1$ when $N = 1$) in the previous section is correct choice for exponentially growing potentials.

Remark 3.1. The resulting method is very stable to small perturbations of α_N and β_N. Indeed, any value of α_N or β_N such that the support of the equilibrium measure is $\mathcal{O}(1)$ is an acceptable scaling.

Remark 3.2. The constants α_N and β_N provide the asymptotic behavior of the coefficients of the Jacobi operator for large n of the unscaled Jacobi operator without requiring the solution of any RHP, see [13, Theorem 15.2].

4   Conclusion

We have demonstrated a new, stable and accurate algorithm for computing the entries of Jacobi operators associated with polynomials orthogonal on $\mathbb{R}$ with respect to analytic, exponential weights. Importantly, the computational cost to compute n rows of the Jacobi operator is $\mathcal{O}(N)$ when $V(x)$ is a polynomial. We estimate the computational cost being $\mathcal{O}(N \log^2 N)$ when $V(x)$ is entire for our examples. As a trivial consequence computing the coefficients in the classical three-term recurrence relation for these polynomials allows us to compute the polynomials themselves. Applying the Golub–Welsh algorithm, we compute Gaussian quadrature weights and nodes accurately.

This approach can be extended to measures supported on a single interval, using the RHP derived in [12]. This setting is simplified for fixed weights as the equilibrium measure does not depend on the weight: it is always an arcsine distribution. Another extension is to weights whose equilibrium measure is supported on multiple intervals, see [5]. The numerical approach for equilibrium measures is applicable to the multiple interval setting [14], however, the Newton iteration underlying the determination of the support can have difficulties at transition points, where the number of intervals of support grows or shrinks.

A   Computational cost

The arguments presented in [18] demonstrate that once $g(z)$ is computed, the computational cost required to construct and solve the RHP is bounded. Therefore, it is necessary to understand the computational cost associated with computing the equilibrium measure for varying N and n and when V depends on N. The method in [14] relies on two approximations. The first is the representation of $\frac{N}{n}V'(x)$ in a Chebyshev series. The second is the convergence of Newton's method to determine the support of the equilibrium measure. Thus computational cost behaves like $\mathcal{O}(Kk + k \log k)$ where K is the number of Newton iterations and k is the number of coefficients in the Chebyshev series that are needed to approximate $V'(x)$ to machine precision.

In practice, we see that $K = \mathcal{O}(1)$. By the chosen scalings, the endpoints of the equilibrium measure are $\mathcal{O}(1)$ and an initial guess of $[-1, 1]$ is sufficient. Additionally, one could use an initial guess from a different N or n value as an initial guess to provide a minor improvement in the computation time.

When V is itself an mth order polynomial then it follows that $k = m - 1$ is bounded and the computational cost to compute $g(z)$, construct the RHP and solve the RHP is truly $\mathcal{O}(1)$. This issue is much more delicate when considering, say, $V_N(x) = x^2 + 1/N \sin(\sqrt{N}x)$ (which corresponds to a normalized $V(x) = x^2 + \sin(x)$) or $V_N(x) = N^{-1} \cosh(d_N x)$ (which corresponds to a normalized $V(x) = \cosh(x)$). In the first case, $V_N'''(x)$ is unbounded which forces k to increase with respect to N. In both of these examples, we observe experimentally that $k = \mathcal{O}(\log N)$, implying that the computational cost for $V(x) = x^2 + \sin(x)$ is $\mathcal{O}(N \log^2 N)$. It is also observed that it takes a bounded amount of time to compute d_N and a bounded number of Chebyshev coefficients to compute the equilibrium measure for $V_N(x) = N^{-1} \cosh(d_N x)$ and therefore it is conjectured that the full computation (computing $g(z)$, constructing the RHP and solving the RHP) is $\mathcal{O}(1)$ in this case. Further details are beyond the scope of this paper.

References

[1] T. Claeys, I. Krasovsky, and A. Its. Higher-order analogues of the Tracy–Widom distribution and the Painlevé II hierarchy. *Comm. Pure Appl. Math.*, 63:362–412, 2010.

[2] P. Deift. *Orthogonal Polynomials and Random Matrices: a Riemann–Hilbert Approach.* AMS, 2000.

[3] P. Deift, T. Kriecherbauer, K. T.-R. McLaughlin, S. Venakides, and X. Zhou. Asymptotics for polynomials orthogonal with respect to varying exponential weights. *Internat. Math. Res. Notices*, 16:759–782, 1997.

[4] 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(12):1491–1552, 1999.

[5] P. Deift, T. Kriecherbauer, K. T.-R. McLaughlin, S. Venakides, and X. Zhou. Uniform asymptotics for polynomials orthogonal with respect to varying exponential weights and applications to universality questions in random matrix theory. *Comm. Pure Appl. Math.*, 52(11):1335–1425, 1999.

[6] P. Erdös. On the distribution of roots of orthogonal polynomials. In G. Alexits et al., editor, *Proceedings of the Conference on the Constructive Theory of Functions*, pages 145–150. Akademiai Kiado, Budapest, 1972.

[7] A. S. Fokas, A. R. Its, and A. V. Kitaev. The isomonodromy approach to matric models in 2d quantum gravity. *Comm. Math. Phys.*, 147(2):395–430, 1992.

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

[9] A. Glaser, X. Liu, and V. Rokhlin. A fast algorithm for the calculation of the roots of special functions. *SIAM J. Sci. Comp.*, 29(4):1420–1438, 2007.

[10] N. Hale and A. Townsend. Fast and accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights. *SIAM J. Sci. Comp.*, 35(2):A652–A674, 2013.

[11] A. B. J. Kuijlaars and P. M. J. Tibboel. The asymptotic behaviour of recurrence coefficients for orthogonal polynomials with varying exponential weights. *J. Comput. Appl. Math.*, 233(3):775–785, 2009.

[12] A.B.J. Kuijlaars, K.T.-R. McLaughlin, W. Van Assche, and M. Vanlessen. The riemann–hilbert approach to strong asymptotics for orthogonal polynomials on $[-1, 1]$. *Advances in Mathematics*, 188(2):337–398, 2004.

[13] A. L. Levin and D. S. Lubinsky. *Orthogonal Polynomials for Exponential Weights*, volume 4. Springer, 2001.

[14] S. Olver. Computation of equilibrium measures. *J. Approx. Theory*, 163:1185–1207, 2011.

[15] S. Olver. Numerical solution of Riemann–Hilbert problems: Painlevé II. *Found. Comput. Math.*, 11:153–179, 2011.

[16] S. Olver. A general framework for solving Riemann–Hilbert problems numerically. *Numer. Math.*, 122:305–340, 2012.

[17] S. Olver and T. Trogdon. Nonlinear steepest descent and the numerical solution of Riemann–Hilbert problems. *Comm. Pure Appl. Maths*, 2012. To appear.

[18] S. Olver and T. Trogdon. Numerical solution of Riemann–Hilbert problems: random matrix theory and orthogonal polynomials. *Const. Approx.*, 2013. To appear.

[19] T. Trogdon. *Riemann–Hilbert Problems, Their Numerical Solution and the Computation of Nonlinear Special Functions*. PhD thesis, Univeristy of Washington, 2013.