We present a non-linear optimization procedure for the design of Generalized Gaussian Quadratures for a fairly broad class of functions. While some of the components of the algorithm have been published previously, we introduce an improved procedure for the determination of an acceptable initial point for the continuation scheme that stabilizes the Newton-type process used to find the quadratures. The resulting procedure never failed when applied to Chebyshev systems (for which the existence and uniqueness of Generalized Gaussian Quadratures are well-known); it also worked for many non-Chebyshev systems, for which the Generalized Gaussian Quadratures are not guaranteed to exist. The performance of the algorithm is illustrated with several numerical examples; some of the presented quadratures integrate efficiently large classes of singular functions.

Non-linear Optimization, Quadrature, and Interpolation

H. Cheng, V. Rokhlin, N. Yarvin
Research Report YALEU/DCS/RR-1169
December 18, 1998

The authors were supported in part by DARPA/AFOSR under Contract F49620/97/1/0011, in part by ONR under Grant N00014-96-1-0188, in part by the AFOSR under Contract F49620-97-C-0052, and in part by the AFOSR under Contract F49620-95-C-0075.
Approved for public release: distribution is unlimited.
Keywords: *Non-linear Optimization, Quadratures, Singular Integrands, Interpolation.*

1 Introduction

Quadrature formulae are one of the most developed areas of computational mathematics. They are used both as a stand-alone numerical tool for the evaluation of integrals, and as an analytical apparatus for the design of interpolation schemes, finite element schemes, etc. Most of the quadrature formulae (at least for functions on $\mathbb{R}^1$) currently in use can be separated into three groups:

1. Gaussian quadratures are the optimal tool for the evaluation of integrals of the form

$$\int_a^b \omega(t) \cdot P(t) dt, \tag{1}$$

where P is a polynomial of t (or a function well-approximated by a polynomial), and ω is a (more or less) arbitrary non-negative function $[a, b] \to \mathbb{R}$. Gaussian quadratures are extremely efficient, mathematically elegant, and easy to obtain (see, for example [3]); whenever applicable, they tend to be the numerical tool of choice.

2. Interpolatory quadrature formulae (Newton-Cotes, etc.) are based on approximating the integrand by some standard function (usually, a polynomial), and integrating the latter. These schemes have the advantage that they (usually) do not prescribe the locations of the nodes; they tend to become numerically unstable for high orders.

3. Miscellaneous special-purpose quadratures ("product integration rules", non-standard Richardson extrapolation, etc.) are normally used when the situation precludes the use of more straightforward techniques.

There appears to exist a class of situations where classical approaches fail to produce rapidly convergent schemes. Specifically, suppose that we wish to integrate functions of the form

$$\sum_{k=0}^{n} \phi_k(x) \cdot s_k(x), \tag{2}$$

where ϕ_k are smooth functions (or polynomials) mapping $[0, 1] \to \mathbb{R}$, and the functions $s_k : [0, 1] \to \mathbb{R}$ are known apriori, and have singularities at $x = 0$. In many situations of interest, the functions s_k have *different* singularities at $x = 0$, and the functions ϕ_k *are not known*; it is only known that the integrand has the form (2), and its values at points on the interval $[0, 1]$ can be evaluated. While efficient quadratures for functions of the form (2) would have obvious applications in the solution of integral equations, in numerical complex analysis, and in several other areas, the authors have failed to find such an apparatus in the literature.

It has been known for about 100 years that Gaussian quadratures admit a drastic generalization, replacing polynomials with fairly general systems of functions (see [11, 12], [2, 8], [6, 7]). The constructions found in (see [11, 12], [2, 8], [6, 7]) do not easily yield numerical algorithms for the design of such quadrature formulae; algorithms of this type were designed (in some cases) in [10, 15], where the resulting quadrature rules are referred to as Generalized Gaussian Quadratures. The approach is based on the observation that the nodes and weights of Gaussian quadratures satisfy systems of non-linear equations, that these equations

have unique solutions, and that when polynomials are replaced with other systems of functions, similar systems of equations are easily constructed. While for functions of the form (2) the resulting equations are non-linear, overdetermined, and non-unique, in the least squares sense they have unique solutions under surprisingly general conditions (see [10, 15]); Newton-type methods converge in this environment, provided a good initial approximation can be found.

As often happens, in the absence of a good initial approximation, the Newton process fails to converge. To some extent, this problem is remedied by the use of continuation techniques, which turn out to be almost always available when designing quadratures for integrands (2). However, yet another problem is frequently encountered: even though *mathematically* the solution of the non-linear problem is unique for all values of the continuation parameter, *numerically* it is not unique at all. Once the (numerical) rank of the Jacobian of an intermediate problem is sufficiently low, the continuation process breaks down; attempts to use globalized search techniques have not been successful.

The final step in the design of a robust scheme for the construction of Generalized Gaussian Quadratures is described in Section 3.3. It finds an initial approximation for which the Jacobian of the system being solved has an acceptably low condition number. While the reasoning behind this step is partly Heuristic, in our experience it works remarkably well. It never failed for a Chebyshev system (see Section 2.1 below); furthermore, it worked for most of the non-Chebyshev systems we tried it on. For a more detailed discussion of our numerical experience, see Section 5 below, where we also present quadratures for functions with *almost* general power singularities at one end (or both ends) of the interval of integration, and with several other types of singularities.

The paper is structured as follows. Section 2 contains mathematical and numerical preliminaries. In Section 3, we build the numerical apparatus to be used in Section 4 to construct the procedure for the determination of nodes and weights of Generalized Gaussian Quadratures. Section 5 contains several examples of quadratures we have obtained. Finally, in Section 6 we outline several possible extensions of this work.

2 Mathematical and Numerical Preliminaries

2.1 Chebyshev systems

Definition 2.1 *A sequence of functions $\phi_1, \ldots, \phi_n$ will be referred to as a* Chebyshev system *on the interval $[a, b]$ if each of them is continuous and the determinant*

$$\begin{vmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) \\ \vdots & & \vdots \\ \phi_n(x_1) & \cdots & \phi_n(x_n) \end{vmatrix} \tag{3}$$

is nonzero for any sequence of points $x_1, \ldots, x_n$ such that $a \leq x_1 < x_2 \ldots < x_n \leq b$.

An alternate definition of a Chebyshev system is that any linear combination of the functions with nonzero coefficients must have no more than n zeros.

A related definition is that of an extended Chebyshev system.

Definition 2.2 *Given a set of functions $\phi_1, \ldots, \phi_n$ which are continuously differentiable on an interval $[a, b]$, and given a sequence of points $x_1, \ldots, x_n$ such that $a \leq x_1 \leq x_2 \leq \ldots \leq x_n \leq b$, let the sequence $m_1, \ldots, m_n$ be defined by the formulae*

$$\begin{array}{rcll} m_1 & = & 0, & \\ m_j & = & 0 & \text{if } j > 1 \text{ and } x_j \neq x_{j-1}, \\ m_j & = & j-1 & \text{if } j > 1 \text{ and } x_j = x_{j-1} = \ldots = x_1, \\ m_j & = & k & \text{if } j > k+1 \text{ and } x_j = x_{j-1} = \ldots = x_{j-k} \neq x_{j-k-1}. \end{array} \tag{4}$$

Let the matrix $C(x_1, \ldots, x_n) = [c_{ij}]$ be defined by the formula

$$c_{ij} = \frac{d^{m_j} \phi_i}{dx^{m_j}}(x_j), \tag{5}$$

in which $\frac{d^0 \phi_i}{dx^0}(x_j)$ is taken to be the function value $\phi_i(x_j)$. Then $\phi_1, \ldots, \phi_n$ will be referred to as an extended Chebyshev system *on $[a, b]$ if the determinant $|C(x_1, \ldots, x_n)|$ is nonzero for all such sequences x_i.*

Remark 2.1 *It is obvious from Definition 2.2 that an extended Chebyshev system is a special case of the Chebyshev system. The additional constraint is that the successive points x_i at which the function is sampled to form the matrix may be identical; in that case, for each duplicated point, the first corresponding column contains the function values, the second column contains the first derivatives of the functions, the third column contains the second derivatives of the functions, and so forth; this matrix must also be nonsingular.*

Examples of Chebyshev and extended Chebyshev systems include the following (additional examples can be found in [7]).

Example 2.1 *The powers $1, x, x^2, \ldots, x^n$ form an extended Chebyshev system on the interval $(-\infty, \infty)$.*

Example 2.2 *The exponentials $e^{-\lambda_1 x}, e^{-\lambda_2 x}, \ldots, e^{-\lambda_n x}$ form an extended Chebyshev system for any $\lambda_1, \ldots, \lambda_n > 0$ on the interval $[0, \infty)$.*

Example 2.3 *The functions $1, \cos x, \sin x, \cos 2x, \sin 2x, \ldots, \cos nx, \sin nx$ form a Chebyshev system on the interval $[0, 2\pi)$.*

2.2 Generalized Gaussian quadratures

The quadrature rules considered in this paper are expressions of the form

$$\sum_{j=1}^{n} w_j \cdot \phi(x_j) \tag{6}$$

where the points $x_j \in \mathbb{R}$ and coefficients $w_j \in \mathbb{R}$ are referred to as the nodes and weights of the quadrature, respectively. They serve as approximations to integrals of the form

$$\int_a^b \phi(x) \; cdot\omega(x) dx \tag{7}$$

where ω has the form

$$\omega(x) = \tilde{\omega}(x) + \sum_{j=1}^{m} \mu_j \cdot \delta(x - \chi_j), \tag{8}$$

with m a non-negative integer, $\tilde{\omega} : [a, b] \to \mathbb{R}$ an integrable non-negative function, $\chi_1, \chi_2, \ldots, \chi_m$ points on the interval $[a, b]$, $\mu_1, \mu_2, \ldots, \mu_m$ positive real coefficients, and δ the Dirac δ-function on $\mathbb{R}$.

Remark 2.2 *Obviously, (8) defines ω to be a linear combination of a non-negative function with a finite collection of δ-functions with positive coefficients. In a mild abuse of terminology, throughout this paper, we will be referring to ω as a non-negative function.*

Quadratures are typically chosen so that the quadrature (6) is equal to the desired integral (7) for some set of functions, commonly polynomials of some fixed order. Of these, the classical Gaussian quadrature rules consist of n nodes and integrate polynomials of order $2n - 1$ exactly. In [10], the notion of a Gaussian quadrature was generalized as follows:

Definition 2.3 *A quadrature formula will be referred to as* Gaussian *with respect to a set of $2n$ functions $\phi_1, \ldots, \phi_{2n} : [a, b] \to \mathbb{R}$ and a weight function $\omega : [a, b] \to \mathbb{R}^+$, if it consists of n weights and nodes, and integrates the functions ϕ_i exactly with the weight function ω for all $i = 1, \ldots, 2n$. The weights and nodes of a Gaussian quadrature will be referred to as Gaussian weights and nodes respectively.*

The following theorem appears to be due to Markov [11, 12]; proofs of it can also be found in [8] and [7] (in a somewhat different form).

Theorem 2.1 *Suppose that the functions $\phi_1, \ldots, \phi_{2n} : [a, b] \to \mathbb{R}$ form a Chebyshev system on $[a, b]$. Suppose in addition that $\omega : [a, b] \to \mathbb{R}$ is defined by (8), and that either*

$$\int_a^b \tilde{\omega}(x)dx > 0, \tag{9}$$

or $m \geq n$ (or both). Then there exists a unique Gaussian quadrature for $\phi_1, \ldots, \phi_{2n}$ on $[a, b]$ with respect to the weight function ω. The weights of this quadrature are positive.

2.3 Quadrature and Interpolation

As is well-known, when Gaussian nodes on the interval $[-1, 1]$ are used for interpolation (for example, via the Lagrange formula), the resulting procedure is numerically stable. Furthermore, the precision obtained via Gaussian (Lagrange) interpolation is almost as high as that obtained via Chebyshev interpolation (see, for example, [4]). Generally, given a weight function ω, the nodes of Gaussian quadratures corresponding to ω lead to interpolation formulae that are stable in an appropriately chosen norm. In this subsection, we formalize this fact for both Gaussian and many Generalized Gaussian quadratures. The analytical tool of this subsection is the following obvious theorem.

Theorem 2.2 *Suppose that the function $\omega : [a,b] \to \mathbb{R}$ is non-negative, and the functions $\phi_1, \phi_2, \ldots, \phi_n : [a,b] \to \mathbb{R}$ are orthonormal with respect to the weight function ω, i.e.*

$$\int_a^b \omega(x) \cdot \phi_j(x) \cdot \phi_i(x) dx = \delta_{ij} \tag{10}$$

for all $i, j = 1, 2, \ldots, n$ (δ_{ij} denotes Kroneker's δ-function). Suppose further that the n-point quadrature rule $x_1, x_2, \ldots, x_n$, $w_1, w_2, \ldots, w_n$, is such that $w_i > 0$ for all $1 \leq i \leq n$. Finally, suppose that

$$\sum_{k=1}^{n} w_k \cdot \phi_i(x_k) \cdot \phi_j(x_k) = \delta_{ij} \tag{11}$$

for all $i, j = 1, 2, \ldots, n$. Then the $n \times n$-matrix A defined by the formula

$$A_{ij} = \sqrt{w_j} \cdot \phi_i(x_j), \tag{12}$$

is orthogonal.

Suppose now that we would like to construct an interpolation formula on the interval $[a,b]$ for functions of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i \cdot \phi_i(x), \tag{13}$$

with $\alpha_1, \alpha_2, \ldots, \alpha_n$ arbitrary real coefficients. In other words, suppose that we are given the values $f_1, f_2, \ldots, f_n$ of a function f at a collection of points $x_1, x_2, \ldots, x_n$, and that it is known that f is defined by the formula (13), but the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$ are not known; we would like to be able to evaluate f at arbitrary points on $[a,b]$. The obvious way to do so is to observe that the values $f_1, f_2, \ldots, f_n$ are linear functions of the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$ (due to (13)); evaluating (13) at the points $x_1, x_2, \ldots, x_n$, we obtain the system of equations

$$f_j = \sum_{i=1}^{n} \alpha_i \cdot \phi_i(x_j), \tag{14}$$

with $j = 1, 2, \ldots, n$. Defining the $n \times n$-matrix B by the formula

$$b_{j,i} = \phi_i(x_j), \tag{15}$$

we rewrite (14) in the form

$$F = B\alpha, \tag{16}$$

with the vectors $\alpha, F \in \mathbb{R}^n$ defined by the formulae

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n), \tag{17}$$

$$F = (f_1, f_2, \ldots, f_n). \tag{18}$$

Now, as long as the matrix B is non-singular, we can evaluate the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$ via the formula

$$\alpha = B^{-1}F, \tag{19}$$

and use (13) to evaluate f at arbitrary points on $[a,b]$. Of course, in actual numerical calculations, it is not sufficient for B to be invertible; its condition number must not be too high. The following observation is the principal purpose of this subsection.

Observation 2.3 *Under the conditions of Theorem 2.2,*

$$A = D \circ B, \tag{20}$$

with D the diagonal matrix defined by the formula

$$D_{i,i} = \sqrt{w_i}, \tag{21}$$

and

$$\alpha = A^* DF \tag{22}$$

(due to the combination of (19) with (20)). In other words, given the table of values $f_1, f_2, \ldots, f_n$ of the function f at the nodes $x_1, x_2, \ldots, x_n$, one obtains the coefficients of the expansion (13) by applying to the vector F the product of two matrices; the first of these matrices is orthogonal, and the second is diagonal; the diagonal elements of the latter are square roots of (positive) weights of the n-point quadrature formula exact for all pairwise products of the functions $\phi_1, \phi_2, \ldots, \phi_n$.

Remark 2.4 *While at first glance the above observation appears to be very limited in its scope (since it relies on the quadrature formula being exact for all pairwise products of the functions $\phi_1, \phi_2, \ldots, \phi_n$), in reality it means that whenever the nodes of a Generalized Gaussian quadrature formula are used as interpolation nodes, the resulting interpolation formula tends to be stable. The reason for this happy coincidence is the fact that the matrix A (see (12) above)) need not be orthogonal for the stability of the interpolation formula; it only needs to be well-conditioned. Thus, as long as the quadrature formula is reasonably accurate for all pairwise products of the functions $\phi_1, \phi_2, \ldots, \phi_n$, the matrix A is close to being orthogonal; therefore, the condition number of A is close to unity, and the interpolation based on the nodes $x_1, x_2, \ldots, x_n$ is stable.*

2.4 Convergence of Newton's method

In this section, we observe that the nodes and the weights of a Gaussian quadrature satisfy a simple system of nonlinear equations. We then prove that the Newton method for this system of equations is always quadratically convergent, provided the functions to be integrated constitute an extended Chebyshev system.

Given a set of functions $\phi_1, \ldots, \phi_{2n}$ and a weight function ω, the Gaussian quadrature is defined by the system of equations

$$\begin{aligned}
\sum_{j=1}^{n} w_j \cdot \phi_1(x_j) &= \int_a^b \phi_1(x) \cdot \omega(x) dx, \\
\sum_{j=1}^{n} w_j \cdot \phi_2(x_j) &= \int_a^b \phi_2(x) \cdot \omega(x) dx, \\
&\vdots \\
\sum_{j=1}^{n} w_j \cdot \phi_{2n}(x_j) &= \int_a^b \phi_{2n}(x) \cdot \omega(x) dx,
\end{aligned} \tag{23}$$

(see Definition 2.3). Let the left hand sides of these equations be denoted by f_1 through f_{2n}. Then each f_i is a function of the weights $w_1, \ldots, w_n$ and nodes $x_1, \ldots, x_n$ of the quadrature. Its partial derivatives are given by the obvious formulae

$$\frac{\partial f_k}{\partial w_i} = \phi_k(x_i), \tag{24}$$

$$\frac{\partial f_k}{\partial x_i} = w_i \cdot \phi_k'(x_i). \tag{25}$$

Thus, the Jacobian matrix of the system (23) is

$$J(x_1, \ldots, x_n, w_1, \ldots, w_n) = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) & w_1\phi_1'(x_1) & \cdots & w_n\phi_1'(x_n) \\ \vdots & & \vdots & \vdots & & \vdots \\ \phi_{2n}(x_1) & \cdots & \phi_{2n}(x_n) & w_1\phi_{2n}'(x_1) & \cdots & w_n\phi_{2n}'(x_n) \end{pmatrix}. \tag{26}$$

Lemma 2.3 *Suppose that the functions $\phi_1, \ldots, \phi_{2n}$ form an extended Chebyshev system. Let the Gaussian quadrature for these functions be denoted by $\hat{w}_i$ and $\hat{x}_i$. Then the determinant of J is nonzero at the point which constitutes the Gaussian quadrature; in other words, $|J(\hat{x}_1, \ldots, \hat{x}_n, \hat{w}_1, \ldots, \hat{w}_n)| \neq 0$.*

Proof. It is immediately obvious from (26) that

$$|J(\hat{x}_1, \ldots, \hat{x}_n, \hat{w}_1, \ldots, \hat{w}_n)| = \hat{w}_1 \cdot \hat{w}_2 \cdot \cdots \cdot \hat{w}_{n-1} \cdot \hat{w}_n \cdot \begin{vmatrix} \phi_1(\hat{x}_1) & \cdots & \phi_1(\hat{x}_n) & \phi_1'(\hat{x}_1) & \cdots & \phi_1'(\hat{x}_n) \\ \vdots & & \vdots & \vdots & & \vdots \\ \phi_{2n}(\hat{x}_1) & \cdots & \phi_{2n}(\hat{x}_n) & \phi_{2n}'(\hat{x}_1) & \cdots & \phi_{2n}'(\hat{x}_n) \end{vmatrix}. \tag{27}$$

If $\phi_1, \ldots, \phi_{2n}$ form an extended Chebyshev system, then by Theorem 2.1, the weights $\hat{w}_1, \ldots, \hat{w}_n$ of the Gaussian quadrature are positive. In addition, by the definition of an extended Chebyshev system, the determinant in the right hand side of (27) is nonzero. Thus

$$|J(\hat{x}_1, \ldots, \hat{x}_n, \hat{w}_1, \ldots, \hat{w}_n)| \neq 0. \tag{28}$$

□

Using the inverse function theorem, we immediately obtain the following corollary:

Corollary 2.4 *Under the conditions of Lemma 2.3, the Gaussian weights and nodes depend continuously on the weight function.*

2.5 Singular value decomposition

The singular value decomposition (SVD) is a ubiquitous tool in numerical analysis, given for the case of real matrices by the following lemma (see, for instance, [13] for more details).

Lemma 2.5 *For any $n \times m$ real matrix A, there exist, for some integer p, an $n \times p$ real matrix U with orthonormal columns, an $m \times p$ real matrix V with orthonormal columns, and a $p \times p$ real diagonal matrix $S = [s_{ij}]$ whose diagonal entries are non-negative, such that $A = U \cdot S \cdot V^*$ and that $s_{ii} \geq s_{i+1,i+1}$ for all $i = 1, \ldots, p-1$.*

The diagonal entries s_{ii} of S are called singular values; the columns of the matrix V are called right singular vectors; the columns of the matrix U are called left singular vectors.

2.6 Singular value decomposition of a sequence of functions

A similar decomposition exists (see [5, 16]) if the columns of the matrix A are replaced by functions:

Theorem 2.6 *Suppose that the functions $\phi_1, \phi_2, \ldots, \phi_n : [a,b] \to \mathbb{R}$ are square integrable. Then there exist a finite orthonormal sequence of functions $u_1, u_2, \ldots, u_p : [a,b] \to \mathbb{R}$, an $n \times p$ matrix $V = [v_{ij}]$ with orthonormal columns, and a sequence $s_1 \geq s_2 \geq \ldots \geq s_p > 0 \in \mathbb{R}$, for some integer p, such that*

$$\phi_j(x) = \sum_{i=1}^{p} u_i(x) s_i v_{ij}, \tag{29}$$

for all $x \in [a,b]$ and all $j = 1, \ldots, n$. The sequence $\{s_i\}$ is uniquely determined by K.

By analogy to the finite-dimensional case, we refer to this factorization as the singular value decomposition. We refer to the functions $\{u_i\}$ as singular functions, to the columns of the matrix V as singular vectors, and to the numbers $\{s_i\}$ as singular values.

A popular application of the Singular Value Decomposition is for the purpose of "compressing" data. Specifically, it often happens that while the total number n of functions is large, almost all of the coefficients s_j in the decomposition (29) are negligibly small. In such cases, (29) is truncated after a small number (say, p_0) of terms, and the resulting expansion

$$\phi_j(x) = \sum_{i=1}^{p_0} u_i(x) \cdot s_i \cdot v_{ij} \tag{30}$$

is viewed as a compact representation of the original family of functions $\phi_1, \phi_2, \ldots, \phi_n$.

The following theorem states that given a sequence of functions on the interval $[a,b]$, their decomposition of the form (30), and a quadrature formula with positive weights on the interval $[a,b]$, the accuracy of the quadrature for the functions $\phi_1, \phi_2, \ldots, \phi_n$ is determined by its accuracy for the singular functions u_j, corresponding to non-trivial singular values. Its proof is an exercise in elementary linear algebra, and is omitted.

Theorem 2.7 *Suppose that under the conditions of Theorem 2.6, ϵ is a positive real number, $1 < p_0 < n$ is an integer, and*

$$\sum_{i=p_0+1}^{n} s_i^2 < \frac{\epsilon^2}{4}. \tag{31}$$

Suppose further that the m-point quadrature formula $\{x_i, w_i\}$ integrates the functions u_i exactly, i.e.

$$\sum_{j=1}^{m} w_j \cdot u_i(x_j) = \int_a^b u_i(x)\, dx \tag{32}$$

for all $i = 1, 2, \ldots, p_0$, and that all of the weights $w_1, \ldots, w_m$ are positive. Then for each $i = 1, 2, \ldots, n$,

$$\left| \sum_{j=1}^{m} w_j \cdot \phi_i(x_j) - \int_a^b \phi_i(x)\, dx \right| < \epsilon \cdot ||\phi_i||_{L^2}. \tag{33}$$

3 Numerical Apparatus

3.1 Continuation method

In order for Newton's method to converge, the starting point provided to it must be close to the desired solution. One scheme for generating such starting points is the continuation method, described below.

Suppose that in addition to the function $F : \mathbb{R}^n \to \mathbb{R}^n$ whose zero is to be found, another function $G : [0,1] \times \mathbb{R}^n \to \mathbb{R}^n$ is available which possesses the following properties:

- 1. For any $x \in \mathbb{R}^n$,
$$G(1,x) = F(x). \tag{34}$$
- 2. The solution of the equation
$$G(0,x) = 0 \tag{35}$$
is known.
- 3. For all $t \in [0,1]$, the equation
$$G(t,x) = 0 \tag{36}$$
has a unique solution x at which the conditions for Newton's method to converge are satisfied.
- 4. The solution x is a continuous function of t.

If these conditions are met, an algorithm for the solution of the equation

$$F(x) = 0 \tag{37}$$

is as follows. Let the points t_i, for $i = 1, \ldots, m$, be defined by the formula $t_i = i/m$. Solve in succession the equations

$$\begin{aligned} G(t_1,x) &= 0, \\ G(t_2,x) &= 0, \\ &\vdots \\ G(t_m,x) &= 0 \end{aligned} \tag{38}$$

using Newton's method, with the starting point for Newton's method for each equation taken to be the solution of the preceding equation. Due to (34), the solution x of the final equation $G(t_m,x) = 0$ is identical to the solution of (37); obviously, for sufficiently large m, Newton's method is guaranteed to converge at each step.

Remark 3.1 *In practice, it is desirable to choose the smallest m for which the above algorithm will work, in order to reduce the computational cost of the scheme. On the other hand, the largest step* $(t_i - t_{i-1})$ *for which the Newton method will converge commonly varies as a function of t. Thus the algorithm described in this paper uses an adaptive version of the scheme.*

3.2 Continuation scheme

The continuation scheme used is as follows. Let the weight functions $\omega : [0,1] \times [a,b] \to \mathbb{R}^+$ be defined by the formula

$$\omega(\alpha, x) = \alpha\omega_1(x) + (1-\alpha)\sum_{j=1}^{n}\delta(x - c_j), \tag{39}$$

where ω_1 is the weight function for which a Gaussian quadrature is desired, δ denotes the Dirac delta function, and the points $c_j \in [a,b]$ are arbitrary distinct points. These weight functions have the following properties:

- 1. With $\alpha = 1$, the weight function is equal to the desired weight function ω_1, due to (39).
- 2. With $\alpha = 0$, the Gaussian weights and nodes are

$$w_j = 1, \tag{40}$$
$$x_j = c_j, \tag{41}$$

  for $j = 1, \ldots, n$, whatever the functions ϕ_i are (since $\omega(0,x) = 0$, unless $x = c_j$ for some $j \in [1,n]$).
- 3. The quadrature weights and nodes depend continuously on α (by Corollary 2.4).

The intermediate problems which the continuation method solves are the Gaussian quadratures relative to the weight functions $\omega(\alpha, *)$. The scheme starts by setting $\alpha = 0$, then increases α in an adaptive manner until $\alpha = 1$, as follows. A current step size is maintained, by which α is incremented after each successful termination of Newton's method. After each unsuccessful termination of Newton's method, the step size is halved and the algorithm restarts from the point yielded by the last successful termination. After a certain number of successful steps, the current step size is doubled. (Experimentally, the current problem was found to be well suited to an aggressive mode of adaption: in the authors' implementation, the initial value of the step size was chosen to be 0.5, and the step size was doubled after two successful terminations of Newton's method.)

3.3 Starting points

The choice of the points c_j was left indefinite above. In exact arithmetic, and applied to a Chebyshev system, the algorithm would converge for any choice of distinct points (see Lemma 2.3). However, the number of steps of the continuation method, and thus the speed of execution, is affected by the choice. More importantly, the numerical stability of the scheme might be compromised due to poor conditioning of the matrix J (see (26)). Indeed, while Lemma 2.3 guarantees that the matrix J is non-singular, it says nothing about its condition number. In addition, we will be applying the algorithm to cases where the conditions of Lemma 2.3 are not satisfied. For these reasons, the following method of choosing the starting points was adopted. The method seeks to create a matrix J that is well-conditioned. It is a pivoted Gram-Schmidt orthogonalization, altered to operate on pairs of vectors:

- 1. Choose a set of points $x_1, x_2, \ldots, x_m$ on the interval of integration $[a, b]$, such that each of the functions $\phi_1, \phi_2, \ldots, \phi_n$, and each of their derivatives, can be interpolated on $[a, b]$ in a well-conditioned manner from values at these points.
- 2. Create a matrix $\tilde{J}$, of the same form as (26), where the points $\{x_j\}$ which determine the columns are the points chosen in step 1. (This matrix thus has $2m$ columns.)
- 3. Perform the following sequence of operations n times:
  - a) choose the point x_j for which the two columns corresponding to x_j have the largest size. (The issue of what "size"to use is discussed below.)
  - b) orthogonalize the remaining columns to both of those two columns.

The points x_j chosen in step (3a) are then the starting points c_j used in the continuation method.

The algorithm as specified above is for exact arithmetic. As with Gram-Schmidt, the algorithm is numerically unstable, but can be stabilized by an additional re-orthogonalization: after step (3a), re-orthogonalize the two new pivot columns to all of the previously chosen pivot columns.

Remark 3.2 *The "size of two columns" that was used for step (3a) was the sum of the norms of the columns, after the second column has been orthogonalized to the first. This poses the obvious danger that one of the two columns chosen might have a small norm, which was covered up by a large norm of its companion. This would render it unsuitable for pivoting; this danger was never realized in our numerical experiments, but if it were, the obvious remedy would be to attempt to change the definition of the "size". The authors have not investigated this issue in detail.*

3.4 Nested Legendre discretizations of finite sequences of functions

In this paper, we will be confronted with finite sequences of functions $\phi_1, \phi_2, \ldots \phi_n$ on the interval $[a, b]$, possessing the following properties:

- 1. The total number n of functions ϕ_i is reasonably large (e.g. 10000).
- 2. The rank of the set $\phi_1, \phi_2, \ldots \phi_n$, is low (e.g. 40), to high precision.
- 3. Each of the functions $\phi_1, \phi_2, \ldots \phi_n$ is analytic on the interval $[a, b]$, except at a finite (small) number of points; $\phi_i \in L^1[a, b]$ for all $i = 1, 2, \ldots, n$.

Now, if we wish to handle (interpolate, integrate, differentiate, etc.) numerically functions of the form

$$\psi(x) = \sum_{i=1}^{n} \alpha_i \cdot \phi_i, \tag{42}$$

often it is not convenient to represent them by collections of coefficients $\alpha_1, \alpha_2, \ldots \alpha_n$. Indeed, if the functions $\phi_1, \phi_2, \ldots \phi_n$ are linearly dependent, the number of coefficients α_i necessary to

represent them in the form (42) might be grossly excessive, compared to the actual complexity of the function to be represented. Furthermore, the coefficients α_i by themselves provide no mechanism for the integration, interpolation, etc. of functions of the form (42); each time such procedures have to be performed, one has to recompute the original functions $\phi_1, \phi_2, \ldots \phi_n$. Since the latter is often expensive or impossible, it is desirable to have a purely numerical procedure for representing sums of the form (42). Preferably, the scheme should use no information about the functions ϕ_i, except for their values at a finite (preferably, not very large) collection of points on $[a, b]$.

When the functions ϕ_i are smooth, a widely used tool for representing them is Chebyshev interpolation: a sufficiently large integer m is chosen, the functions $\phi_1, \phi_2, \ldots \phi_n$ are tabulated at m Chebyshev nodes on $[a, b]$, and obtained at all other points on $[a, b]$ via standard interpolation procedures. While Chebyshev nodes are an extremely good choice, they are not the only one; for example, Gaussian (Legendre) nodes are almost as efficient as the Chebyshev ones when the functions are to be interpolated, and twice as efficient when the functions are to be integrated (see, for example, [4]). When the behavior of the functions ϕ_i is very non-uniform over the interval $[a, b]$, Chebyshev (Gaussian, etc.) interpolation becomes inefficient; for singular functions it is liable to fail completely. In such cases, adaptive Chebyshev interpolation is used, whereby the interval is subdivided into a collection of subintervals, so that on each subinterval, all of the functions ϕ_i are accurately approximated by a Chebyshev expansion of low order; the subdivisions are performed automatically. When some (or all) of the functions ϕ_i have singularities on the interval $[a, b]$, schemes of this type cluster the subintervals near each singularity, until the subinterval nearest to the singularity is so small as to be ignorable for the purposes of the calculations to be performed.

In the first stage of the algorithm we use, we build a nested Chebyshev discretization of the interval [a,b] for each of the functions ϕ_i. In the second stage, all such discretizations are merged to obtain a single discretization by which all of the functions ϕ_i are adequately represented. In the third stage, n *Legendre* nodes are constructed on each of the obtained intervals.

Stage 1

- 1. Choose the precision ϵ and some reasonably large m (in actual computations, we use $m = 16$).

- 2. Construct the m Chebyshev nodes $x_1^{[a,b]}, x_2^{[a,b]}, \ldots, x_m^{[a,b]}$, on the interval $[a, b]$. Evaluate ϕ at the nodes $x_1^{[a,b]}, x_2^{[a,b]}, \ldots, x_m^{[a,b]}$, obtaining the values $\phi_1^{[a,b]}, \phi_2^{[a,b]}, \ldots, \phi_m^{[a,b]}$.

- 3. Subdivide the interval $[a, b]$ into the subintervals $[a, (a+b)/2]$, $[(a+b)/2, b]$. Construct the Chebyshev nodes $x_1^{[a,(a+b)/2]}, x_2^{[a,(a+b)/2]}, \ldots, x_m^{[a,(a+b)/2]}$ on the interval $[a, (a+b)/2]$, and the Chebyshev nodes $x_1^{[(a+b)/2,b]}, x_2^{[(a+b)/2,b]}, \ldots, x_m^{[(a+b)/2,b]}$ on the interval $[(a+b)/2, b]$. Evaluate the function ϕ at the nodes $x_1^{[a,(a+b)/2]}, x_2^{[a,(a+b)/2]}, \ldots, x_m^{[a,(a+b)/2]}, x_1^{[(a+b)/2,b]}, x_2^{[(a+b)/2,b]}, \ldots, x_m^{[(a+b)/2,b]}$, obtaining the values $\phi_1^{[a,(a+b)/2]}, \phi_2^{[a,(a+b)/2]}, \ldots, \phi_m^{[a,(a+b)/2]}, \phi_1^{[(a+b)/2,b]}, \phi_2^{[(a+b)/2,b]}, \ldots, \phi_m^{[(a+b)/2,b]}$, respectively.

- 4. Interpolate the values of the function ϕ from the nodes $x_1^{[a,b]}, x_2^{[a,b]}, \ldots, x_m^{[a,b]}$, on the interval $[a, b]$ to the nodes $x_1^{[a,(a+b)/2]}, x_2^{[a,(a+b)/2]}, \ldots, x_m^{[a,(a+b)/2]}, x_1^{[(a+b)/2,b]}, x_2^{[(a+b)/2,b]}$,

..., $x_m^{[(a+b)/2,b]}$ on the intervals $[a,(a+b)/2]$, $[(a+b)/2,b]$. If the interpolated values agree to the precision ϵ with the values $\phi_1^{[a,(a+b)/2]}$, $\phi_2^{[a,(a+b)/2]}$, ..., $\phi_m^{[a,(a+b)/2]}$, $\phi_1^{[(a+b)/2,b]}$, $\phi_2^{[(a+b)/2,b]}$, ..., $\phi_m^{[(a+b)/2,b]}$ calculated directly in Step 2 above, the algorithm concludes that the function ϕ is adequately resolved by the m Chebyshev nodes on the interval $[a,b]$; otherwise, the procedure is repeated recursively for each of the subintervals $[a,(a+b)/2]$, $[(a+b)/2,b]$.

Stage 2

- Store the ends (left and right) of all subintervals in all subdivisions in a single array a. Sort the elements of a; remove multiple elements in a. The resulting array of points on the interval $[a,b]$ (including the points a,b) is the array of ends of subintervals of the final subdivision.

Stage 3

- Construct an m-point Legendre discretization of each of the subintervals obtained in Stage 2 above.

Remark 3.3 *In the algorithm above, we use Chebyshev discretizations in Stage 1 to construct the subdivision of the interval $[a,b]$; in subsequent calculations we use Legendre discretizations. The reason for this choice is that the interpolations in Stage 1 are carried out more efficiently with Chebyshev discretizations, via the Discrete Cosine Transform and related tools; the Legendre discretizations used subsequently lead to linear interpolation schemes that preserve inner products (see following subsection).*

Remark 3.4 *The scheme of this subsection is a fairly reliable apparatus for the automatic discretization of sets of (more or less) arbitrary user-specified functions. While it is very easy to construct counterexamples in which the algorithm will fail to resolve some (or all) of the input functions, this problem has never been encountered in our practice.*

3.5 Approximation of SVD of a sequence of functions

This section describes a numerical procedure for computing an approximation to the singular value decomposition of a sequence of functions. The algorithm uses quadratures possessing the following property.

Definition 3.1 *We will say that the combination of a quadrature and an interpolation scheme* preserves inner products *on an interval $[a,b]$ if it possesses the following properties.*

- *1. The nodes of the quadrature are identical to the nodes of the interpolation scheme.*
- *2. The function which is output by the interpolation scheme depends in a linear fashion on the values input to the interpolation scheme.*

- *3. The quadrature integrates exactly any product of two interpolated functions; that is, for any two functions $f, g : [a, b] \to \mathbb{R}$ produced by the interpolation scheme, the integral*

$$\int_a^b f(x) \cdot g(x) dx \tag{43}$$

is computed exactly by the quadrature.

Quadratures and interpolation schemes possessing this property include:

Example 3.1 *The combination of a (classical) Gaussian quadrature at Legendre nodes and polynomial interpolation at the same nodes preserves inner products, since polynomial interpolation on n nodes produces an interpolating polynomial of order $n-1$, the product of two such polynomials is a polynomial of order $2n-2$, and a Gaussian quadrature integrates exactly all polynomials up to order $2n-1$.*

Example 3.2 *If an interval is broken into several subintervals, and a quadrature and interpolation scheme preserving inner products is used on each subinterval, then the arrangement as a whole preserves inner products on the original interval. (This follows directly from the definition.)*

Example 3.3 *The combination of the trapezoidal rule on the interval $[0, 2\pi]$, and Fourier interpolation (using the interpolation functions $1, \cos x, \sin x, \cos 2x, \sin 2x, \ldots, \cos nx, \sin nx$) preserves inner products.*

The algorithm described below takes as input a sequence of functions $\phi_1, \phi_2, \ldots, \phi_n : [a, b] \to \mathbb{R}$. It uses as a tool a quadrature and a linear interpolation scheme on the interval $[a, b]$ preserving inner products; the weights and nodes of this quadrature will be denoted by $w_1, \ldots, w_n \in \mathbb{R}$ and $x_1, \ldots, x_n \in [a, b]$ respectively. As will be shown below, the accuracy of the algorithm is then determined by the accuracy to which the interpolation scheme approximates the functions $\phi_1, \phi_2, \ldots, \phi_n$.

The output of the algorithm is a sequence of functions $u_1, \ldots, u_p : [a, b] \to \mathbb{R}$, a sequence of vectors $v_1, \ldots, v_p \in \mathbb{R}^n$, and a sequence of singular values $s_1, \ldots, s_p \in \mathbb{R}$, forming an approximation to the singular value decomposition of $\phi_1, \phi_2, \ldots, \phi_n$.

Description of the algorithm:

- 1. Construct the $n \times m$ matrix $A = [a_{ij}]$ defined by the formula

$$a_{ij} = \phi_j(x_i) \cdot \sqrt{w_i}. \tag{44}$$

- 2. Compute the singular value decomposition of A, to produce the factorization

$$A = U \circ S \circ V^*, \tag{45}$$

where $U = [u_{ij}]$ is an $n \times p$ matrix with orthonormal columns, $V = [v_{ij}]$ is an $m \times p$ matrix with orthonormal columns, and S is a $p \times p$ diagonal matrix whose j'th diagonal entry is s_j.

- 3. Construct the $n \times p$ values $u_k(x_i)$ defined by

$$u_k(x_i) = u_{ik}/\sqrt{w_i}. \tag{46}$$

- 4. For any desired point $x \in [a,b]$, evaluate the functions $u_k : [a,b] \to \mathbb{R}$ using the interpolation scheme on $[a,b]$.

The proof of the following theorem can be found (in a considerably more general form) in [15].

Theorem 3.1 *Suppose that the combination of the quadrature and interpolation scheme with weights and nodes $w_1, \ldots, w_n \in \mathbb{R}$ and $x_1, \ldots, x_n \in [a,b]$, respectively, preserves inner products on $[a,b]$. For any sequence of functions $\phi_1, \phi_2, \ldots, \phi_n : [a,b] \to \mathbb{R}$, let $u_i : [a,b] \to \mathbb{R}$, $v_{ij} \in \mathbb{R}$, and $s_i \in \mathbb{R}$ be defined in (44)-(46), for all $i = 1, \ldots, p$. Then*

- *1. The functions u_i are orthonormal, i.e.*

$$\int_a^b u_i(x)u_k(x)dx = \delta_{ik} \tag{47}$$

for all $i, k = 1, \ldots, p$, with δ_{ik} the Kronecker symbol.

- *2. The columns of V are orthonormal, i.e.*

$$\sum_{j=1}^{n} v_{ij}v_{kj}dx = \delta_{ik} \tag{48}$$

for all $i, k = 1, \ldots, p$.

- *3. The sequence of functions $\tilde{\phi}_1, \tilde{\phi}_2, \ldots, \tilde{\phi}_n : [a,b] \to \mathbb{R}$ defined by*

$$\tilde{\phi}_k(x) = \sum_{j=1}^{p} s_j u_j(x) v_{jk}, \tag{49}$$

is identical to the sequence of functions produced by sampling the functions $\phi_1, \phi_2, \ldots, \phi_n$ at the points $\{x_i\}$, then interpolating with the interpolation scheme on $[a,b]$.

4 Numerical Algorithm

This section describes a numerical algorithm for the evaluation of nodes and weights of generalized Gaussian quadratures. The algorithm's input are a sequence of functions $\phi_1, \ldots, \phi_{2n} : [a,b] \to \mathbb{R}$, and the precision ϵ to which the quadratures are to be calculated; its output is the weights and nodes of the quadrature. The functions ϕ_i are supplied by the user in the form of a subroutine, with input parameters (x,i), and output parameter $\phi_i(x)$. The algorithm uses the components described the preceding section.

- 1. The interval $[a,b]$ is discretized via the scheme described in Subsection 3.4, so that all functions ϕ_1, $\phi_2, \ldots,$ ϕ_n are represented to the precision ϵ.

- 2. All of the functions ϕ_1, $\phi_2, \ldots$, ϕ_n are tabulated at the nodes of the discretization obtained in p. 1 above, and the Singular Value Decomposition is obtained of the sequence of functions ϕ_1, $\phi_2, \ldots$, ϕ_n via the scheme described in Subsection 3.5; we will be denoting the obtained singular values by $\lambda_1, \lambda_2, \ldots$.

- 3. Denoting by k the positive integer number such that $\lambda_{2 \cdot k+1} \leq \epsilon \leq \lambda_{2 \cdot k-1}$, we observe that any quadrature formulae with positive coefficients that integrates the obtained singular functions u_1, $u_2, \ldots$ $u_{2 \cdot k}$ exactly, will integrate all of the functions ϕ_1, $\phi_2, \ldots$, ϕ_n with precision ϵ (see Theorem 2.7 in Subsection 2.5). The remainder of the algorithm is devoted to constructing a k-point quadrature formula that will integrate the functions u_1, $u_2, \ldots$ $u_{2 \cdot k}$ exactly.

- 4. The scheme of Subsection 3.3 is used to find the starting nodes $x_1^0, x_2^0, \ldots, x_k^0$ for the continuation process of Subsection 3.2.

- 5. An adaptive version of the continuation method of Subsection 3.2 is used to obtain the k-point quadrature for the functions u_1, $u_2, \ldots$, $u_{2 \cdot k}$; on each step, the Newton algorithm described in Subsection 2.4 is used to solve the system (23) defining the nodes and roots of the quadrature formula.

Remark 4.1 *We would like to reiterate that the quadrature formulae produced by the procedure of this section do not integrate the user-specified functions ϕ_1, $\phi_2, \ldots$, ϕ_n exactly; instead, they produce approximations to the integrals. Needless to say, the two are indistinguishable, as long as the chosen precision ϵ is less than the machine precision.*

5 Numerical examples

A variety of quadratures were generated via the algorithm of this paper; several of these are presented below to illustrate its performance. In Examples 5.1, 5.2, the calculations were performed in extended precision (Fortran `REAL*16`) arithmetic, to assure full double precision in the obtained result. In Example 5.3, the calculations were performed in double precision, since the accuracy of the quadrature listed in Table 5 is only 9 digits.

Example 5.1 An obvious problem of interest is the integration on an interval of functions that have a singularity at one end of that interval (or at both ends); of particular interest are power and logarithmic singularities. Many techniques have been proposed for dealing with such problems (see, for example, [1]). While some of these approaches are quite effective for some of the singularities, they have the drawback that each of them only deals with one particular singularity. In this example, we present quadrature rules for the integration of functions of the form

$$\sum_{k=0}^{n} (\gamma_k \cdot log(x) + \sum_{j=1}^{m} \beta_{k,j} \cdot x^{\alpha_j}) \cdot P_k(x) \tag{50}$$

where P_k denotes the (normalized) orthogonal polynomial of order k on the interval $[0, 1]$, $\beta_{k,j}$, γ_k are arbitrary real numbers, and α_j are arbitrary real numbers on the interval $[-0.6, 1]$.

Table 1: 16-node quadrature for functions of the form (50), with $\alpha \in [-0.6, 1]$, $N = 4$, and precision 10^{-15}

x_i	w_i
0.1646476245461994E-18	0.2477997131959177E-17
0.2004881755033198E-13	0.1863311166024058E-12
0.4902407997203263E-10	0.3215991324579055E-09
0.1396853977847601E-07	0.6788563189534853E-07
0.9715236454504147E-06	0.3586206403622012E-05
0.2502196135803993E-04	0.7130636866829449E-04
0.3120851149673110E-03	0.6951436010759356E-03
0.2264576163994000E-02	0.3979838127986921E-02
0.1086917746927712E-01	0.1515746778330600E-01
0.3777218640280392E-01	0.4182483334409624E-01
0.1013279037973986E+00	0.8854031057518543E-01
0.2196196157836697E+00	0.1490380907486389E+00
0.3972680999338400E+00	0.2028312538451011E+00
0.6135562966157080E+00	0.2216836945000430E+00
0.8216868417553706E+00	0.1844567448110479E+00
0.9636466562372551E+00	0.9171766188102896E-01

In order to design such quadratures, we choose a reasonably large natural m, construct m Legendre nodes $\alpha_1, \alpha_2, \ldots, \alpha_m$, on the interval $[-0.6, 1]$, and use all functions of the forms

$$P_k(x) \cdot x^{\alpha_j}, \tag{51}$$

$$P_k(x) \cdot log(x) \tag{52}$$

as input functions ϕ_i for the algorithm of the preceding section. The result is a set quadratures for functions of the forms (51), (52). A somewhat involved analytical calculation shows that for sufficiently large m, the obtained quadratures will work for all functions of the form (50), and our numerical experiments show that $m = 100$ insures full double precision accuracy for all $\alpha_j \in [-0.6, 1]$.

In Tables 1 - 5, we list quadrature nodes and weights for $n = 4, 9, 19, 29$. In Tables 1, 3, 4, 5, the number of nodes is chosen to guarantee 15-digit accuracy. In Table 2, the number of nodes is chosen to guarantee 7 digits.

Example 5.2 The quadrature rules in this example are very similar to those in Example 5.1, except here we construct quadrature rules for functions singular at *both* ends of the interval where they are to be integrated. Specifically, integrands have the form

$$\sum_{k=0}^{n}(\sum_{j=1}^{m}(a_{k,j} \cdot (1+x)^{\alpha_j} + b_{k,j} \cdot (1-x)^{\alpha_j}) + c_k \cdot log(1+x) + d_k \cdot log(1-x)) \cdot P_k(x) \tag{53}$$

Table 2: 8-node quadrature for functions of the form (50), with $\alpha \in [-0.6, 1]$, $N = 4$, and precision 10^{-7}

x_i	w_i
0.1312034302206730E-07	0.1393140646786704E-06
0.2793817088002595E-04	0.1549484313499085E-03
0.2038371172070937E-02	0.6673805929140874E-02
0.2702722219647910E-01	0.5430869272244519E-01
0.1343993651970034E+00	0.1694172186704161E+00
0.3682213359901025E+00	0.2898751155944595E+00
0.6792045461791814E+00	0.3076390470455203E+00
0.9309603731369270E+00	0.1719310626051804E+00

Table 3: 19-node quadrature for functions of the form (50), with $\alpha \in [-0.6, 1]$, $N = 9$, and precision 10^{-15}

x_i	w_i
0.1846942465536925E-18	0.2756403589261532E-17
0.1989380701597045E-13	0.1824804592695847E-12
0.4312593909743526E-10	0.2777592139982985E-09
0.1092964737770428E-07	0.5186860611615611E-07
0.6810397860708155E-06	0.2442433440466041E-05
0.1588655973896037E-04	0.4380009969129837E-04
0.1818339165855430E-03	0.3906506115636250E-03
0.1227551979000820E-02	0.2077051291912717E-02
0.5556316902145769E-02	0.7461053476901383E-02
0.1847419717287859E-01	0.1978838865640943E-01
0.4825255045366560E-01	0.4136988974623410E-01
0.1041307630444531E+00	0.7157248041035670E-01
0.1928680775398894E+00	0.1060884317057585E+00
0.3153775090195431E+00	0.1377804712043467E+00
0.4647713088385197E+00	0.1585409276263068E+00
0.6264814981191495E+00	0.1614751848557232E+00
0.7804757620006211E+00	0.1428196856993585E+00
0.9050563637732498E+00	0.1031243266706421E+00
0.9813553783808000E+00	0.4746516336480648E-01

Table 4: 26-node quadrature for functions of the form (50), with $\alpha \in [-0.6, 1]$, $N = 19$, and precision 10^{-15}

x_i	w_i
0.2852686209735951E-20	0.4390385492743041E-19
0.4655349788609637E-15	0.4445881189691443E-14
0.1432147899313873E-11	0.9689649973398580E-11
0.4915792345704672E-09	0.2471786670704959E-08
0.3986884553883893E-07	0.1527652265503579E-06
0.1168849078081257E-05	0.3470933550491954E-05
0.1630549221175312E-04	0.3803166416108812E-04
0.1307331567674635E-03	0.2422240257088061E-03
0.6884061227847875E-03	0.1022568448159836E-02
0.2620448293548410E-02	0.3143745934305781E-02
0.7740029188833982E-02	0.7549238041954824E-02
0.1872452403074940E-01	0.1495112040361046E-01
0.3869460001276389E-01	0.2548756008178511E-01
0.7058074961479188E-01	0.3865021281644121E-01
0.1165353335503884E+00	0.5342389042306681E-01
0.1775282580420220E+00	0.6849323863305738E-01
0.2531447462199369E+00	0.8243302008328313E-01
0.3415558481256653E+00	0.9386320384208941E-01
0.4396281348394975E+00	0.1015733726852001E+00
0.5431447278197111E+00	0.1046214551363520E+00
0.6471126706707170E+00	0.1024074963963311E+00
0.7461308154896283E+00	0.9472049436813551E-01
0.8347900655356778E+00	0.8175595131244442E-01
0.9080759999882411E+00	0.6410309004863602E-01
0.9617441758037388E+00	0.4270384642243640E-01
0.9926478556999123E+00	0.1881261305258270E-01

Table 5: 36-node quadrature for functions of the form (50), with $\alpha \in [-0.6, 1]$, $N = 39$, and precision 10^{-15}

x_i	w_i
0.1174238417413926E-19	0.1769042596381234E-18
0.1422439193737780E-14	0.1318732300270049E-13
0.3350676698582048E-11	0.2181187238172082E-10
0.8987762100979194E-09	0.4306047388907762E-08
0.5804062676082615E-07	0.2097251047066944E-06
0.1381879982602796E-05	0.3830347070073085E-05
0.1599014834456195E-04	0.3447814965093908E-04
0.1086072834052024E-03	0.1843012333973045E-03
0.4939690780979653E-03	0.6658876227138618E-03
0.1653457719227906E-02	0.1785581170381193E-02
0.4371083474213578E-02	0.3817614649487054E-02
0.9635942477742897E-02	0.6885390581283880E-02
0.1847241513238332E-01	0.1094085630140653E-01
0.3179190367214565E-01	0.1581728538518057E-01
0.5030636405050507E-01	0.2129142636454853E-01
0.7449442868952319E-01	0.2712481569656370E-01
0.1045979502135202E+00	0.3308456773919071E-01
0.1406326475828715E+00	0.3895216905306892E-01
0.1824044449022998E+00	0.4452688339606666E-01
0.2295280679235570E+00	0.4962723403902098E-01
0.2814468220422235E+00	0.5409202169130247E-01
0.3374533767644982E+00	0.5778135262022458E-01
0.3967116179369689E+00	0.6057773920656186E-01
0.4582796041927400E+00	0.6238718893653459E-01
0.5211335571597729E+00	0.6314016307782669E-01
0.5841926980689389E+00	0.6279229386348975E-01
0.6463446423449487E+00	0.6132477029316637E-01
0.7064709858680002E+00	0.5874432665998542E-01
0.7634726623238107E+00	0.5508279084756487E-01
0.8162946187294954E+00	0.5039617034177984E-01
0.8639493438008133E+00	0.4476327290202123E-01
0.9055387898384755E+00	0.3828387702474601E-01
0.9402742542357631E+00	0.3107648956468336E-01
0.9674938463383342E+00	0.2327578565976658E-01
0.9866773942995437E+00	0.1503024417658587E-01
0.9974613070359063E+00	0.6508977351752366E-02

Table 6: 22-node quadrature for functions of the form (53), with $\alpha \in [-0.1, 1]$, $N = 4$, and precision 10^{-15}

$\pm x_i$	w_i
0.1666008119316040E+00	0.3286464553329054E+00
0.4736467937561296E+00	0.2782402062916909E+00
0.7129463900017805E+00	0.1977249261400840E+00
0.8687173264995090E+00	0.1158087624474726E+00
0.9515411665787298E+00	0.5425992604604305E-01
0.9862971262509680E+00	0.1943874113675287E-01
0.9972429072629104E+00	0.4979788483749470E-02
0.9996464539418006E+00	0.8238003428108275E-03
0.9999757993153293E+00	0.7462712208720397E-04
0.9999993605804343E+00	0.2746237603563529E-05
0.9999999970230195E+00	0.2041880191195951E-07

where P_k denotes the (normalized) orthogonal polynomial of order k on the interval $[-1, 1]$, $a_{k,j}$, $b_{k,j}$, c_k, d_k are arbitrary real numbers, and α_j are arbitrary real numbers on the interval $[-0.1, 1]$. Quadrature nodes and weights for $n = 4, 9, 19, 39$ are listed in Tables 6, 7, 8, 9 respectively; in all cases, the precision is 10^{-15}.

Example 5.3 In this example, we construct a direct generalization of quadratures constructed in Example 5.1, permitting the integrands to have power and logarithmic singularities *at arbitrary points on the closed half-line to the left of the interval of integration.* Specifically, integrands have the form

$$\sum_{k=0}^{n} (\gamma_k \cdot log(x+h) + \sum_{j=1}^{m} \beta_{k,j} \cdot (x+h)^{\alpha_j}) \cdot P_k(x) \tag{54}$$

where P_k denotes the (normalized) orthogonal polynomial of order k on the interval $[0, 1]$, $\beta_{k,j}$, γ_k are arbitrary real numbers, α_j are arbitrary real numbers on the interval $[-0.65, 1]$, and h is an arbitrary positive real number. In this case, the calculations were conducted in double precision; the 38-node quadrature formula for $n = 19$ is listed in Table 10; its precision is 10^{-9}.

Several observations can be made from the tables 1-8, and from the more detailed numerical experiments we have conducted.

- 1. The algorithm of this paper is always effective for Chebyshev systems; it almost always works for non-Chebyshev ones.
- 2. The scheme does not lose very many digits compared to the machine precision; when the calculations are performed in double precision, the quadratures can be obtained to 11 or 12 digits; the accuracy of quadratures in Tables 1-8 is full double precision; we used extended precision arithmetic in FORTRAN to obtain them.

Table 7: 27-node quadrature for functions of the form (53), with $\alpha \in [-0.1, 1]$, $N = 9$, and precision 10^{-15}

$\pm x_i$	w_i
0.0000000000000000E+00	0.1969765126094452E+00
0.1953889665467211E+00	0.1922287111905558E+00
0.3814298736462841E+00	0.1784269782500965E+00
0.5496484616443740E+00	0.1568677485350913E+00
0.6932613279607421E+00	0.1296176364576521E+00
0.8078808016610349E+00	0.9937321489137896E-01
0.8920478424190657E+00	0.6925317917837661E-01
0.9475053154471952E+00	0.4247396818782292E-01
0.9790448975739819E+00	0.2179872525134398E-01
0.9936444652327659E+00	0.8672220251831163E-02
0.9986936386311707E+00	0.2388475528070173E-02
0.9998477986092101E+00	0.3837648653769931E-03
0.9999927156219827E+00	0.2671422777541431E-04
0.9999999335937359E+00	0.4068798910349743E-06

Table 8: 33-node quadrature for functions of the form (53), with $\alpha \in [-0.1, 1]$, $N = 19$, and precision 10^{-15}

$\pm x_i$	w_i
0.0000000000000000E+00	0.1802406542699465E+00
0.1789856568226836E+00	0.1764865559769247E+00
0.3505713663705831E+00	0.1655482040246752E+00
0.5079970396268890E+00	0.1483733690643724E+00
0.6457344058749438E+00	0.1264620956535221E+00
0.7599840782344723E+00	0.1017484935648103E+00
0.8490304782768580E+00	0.7643386171408831E-01
0.9134021329241244E+00	0.5276203409291129E-01
0.9557717316319267E+00	0.3272086426808218E-01
0.9805181730564275E+00	0.1766845539228831E-01
0.9929045523533901E+00	0.7963812531655223E-02
0.9979798758935006E+00	0.2833884283485953E-02
0.9995837651123616E+00	0.7387521680930171E-03
0.9999445617386989E+00	0.1267394032662049E-03
0.9999960165362139E+00	0.1207609748958691E-04
0.9999998889650372E+00	0.4709227238502033E-06
0.9999999994557687E+00	0.3706639850258617E-08

Table 9: 45-node quadrature for functions of the form (53), with $\alpha \in [-0.1, 1]$, $N = 39$, and precision 10^{-15}

$\pm x_i$	w_i
0.0000000000000000E+00	0.1138212938786054E+00
0.1135283181390291E+00	0.1129431358863252E+00
0.2253080046824045E+00	0.1103317059272695E+00
0.3336364252858657E+00	0.1060558645237672E+00
0.4369024052356911E+00	0.1002294986469973E+00
0.5336306707891807E+00	0.9301028558331059E-01
0.6225248777667337E+00	0.8459812566475355E-01
0.7025089656717720E+00	0.7523338442881639E-01
0.7727667118189729E+00	0.6519506433099722E-01
0.8327794264993337E+00	0.5479889055074179E-01
0.8823615451977041E+00	0.4439489209928996E-01
0.9216930322777481E+00	0.3436308131973152E-01
0.9513451962287941E+00	0.2510376733595393E-01
0.9722913641056944E+00	0.1701539437521317E-01
0.9858845322639776E+00	0.1044852849794223E-01
0.9937724959340503E+00	0.5626146436355554E-02
0.9977200386244100E+00	0.2543352365327656E-02
0.9993454278943935E+00	0.9118380718941661E-03
0.9998636273258416E+00	0.2403706487446808E-03
0.9999815974719829E+00	0.4181929949775085E-04
0.9999986596740707E+00	0.4045883666118617E-05
0.9999999622133619E+00	0.1599158044436823E-06
0.9999999998137450E+00	0.1268296767711113E-08

Table 10: 38-node quadrature for functions of the form (54), with $\alpha \in [-0.65, 1]$, $N = 19$, and precision 10^{-9}

x_i	w_i
0.7629165866352161E-18	0.4643955333268610E-17
0.3799719398931375E-16	0.1132690565299208E-15
0.5684549949701512E-15	0.1423549582265871E-14
0.6085909916179373E-14	0.1371876219104025E-13
0.5277191865393953E-13	0.1094397021531007E-12
0.3900442913791902E-12	0.7534990994077416E-12
0.2535538557277294E-11	0.4603432835276850E-11
0.1481755662897140E-10	0.2545533729683496E-10
0.7911595380511587E-10	0.1293022088581050E-09
0.3907746000477183E-09	0.6102781198001779E-09
0.1803070816493823E-08	0.2700678436986190E-08
0.7833265344260583E-08	0.1128792193586090E-07
0.3224897189563689E-07	0.4482855569803782E-07
0.1264894823726299E-06	0.1700035548631482E-06
0.4747932260937661E-06	0.6182057321480894E-06
0.1711978528765632E-05	0.2163108715557027E-05
0.5948052018171647E-05	0.7302447810573277E-05
0.1995877304286260E-04	0.2382492261847977E-04
0.6475274273537152E-04	0.7511062044871306E-04
0.2029004100170709E-03	0.2279609908900293E-03
0.6109309950274235E-03	0.6592765068003472E-03
0.1747449285439932E-02	0.1781666222619331E-02
0.4661579935095226E-02	0.4378093849756735E-02
0.1135932523990354E-01	0.9537600800370288E-02
0.2491532030262493E-01	0.1820679046524441E-01
0.4902801284057732E-01	0.3060746663786768E-01
0.8713816071641225E-01	0.4600643316091537E-01
0.1415514175271372E+00	0.6292513465068938E-01
0.2128806314974303E+00	0.7951989233968431E-01
0.2998564528132552E+00	0.9391761648476182E-01
0.3994239415560721E+00	0.1044517799613406E+00
0.5070313867113639E+00	0.1098153664961849E+00
0.6170411438386144E+00	0.1091553255900476E+00
0.7232121752054713E+00	0.1021230666276667E+00
0.8192137516286219E+00	0.8888680524875885E-01
0.8991333728333283E+00	0.7010796674100402E-01
0.9579443204807173E+00	0.4688508195206744E-01
0.9919093183441774E+00	0.2069742637648333E-01

- 3. The algorithm of this paper is not very efficient. For example, the quadrature formula in Table 1 took about 2 minutes of CPU time on Ultra SPARC 2; the quadrature in Table 8 took about two hours of CPU time. Of course, extended precision on the UltraSparc is quite inefficient; in double precision, Table 8 took about 4 minutes on to construct. In any event, the quadratures of the type presented in this paper need not be constructed "on the fly"; the nodes and weights can be precomputed and stored. From this point of view, the CPU time requirements of our algorithm are not excessive. Still, its CPU time requirements grow as n^3 for large n, making it unsuitable for the construction of quadratures of very high order.

6 Generalizations and Conclusions

We have constructed a scheme for the design of Generalized Gaussian Quadratures for a fairly broad classes of functions. The results presented here should be viewed as somewhat experimental, since while the algorithm appears to work under quite general conditions, we can only *prove* that it *has to* work for Chebyshev systems.

Several possible extensions of the work suggest themselves.

1. Quadratures of the type designed in this paper can be used within compound quadrature rules, not unlike the classical Gaussian quadratures. In particular, they can be substituted for Gaussian quadratures in the scheme described in Subsection 3.4 above. If the functions to be integrated have (for example) power singularities at the left end of the interval of integration, the quadrature rules in Example 5.1 will eliminate the bunching of nodes near the left end of the interval. In this respect, of particular interest are quadratures of the type found in Example 5.3, since their use will eliminate the bunching of quadrature nodes near the ends of the interval for integrands with power singularities *anywhere on* $\mathbb{R}$ *outside the interval of integration.* Furthermore, one does not have to replace classical Gaussian quadratures with ours on all of the subintervals of a compound rule; it is sufficient to do so only on those subintervals near the ends of the interval of integration. In other situations, different special-purpose Genaralized Gaussian Quadratures might be used. Such adaptive compound rules have been constructed; a paper describing them is in preparationn.

The fact that we can only obtain quadratures

2. While our numerical experiments indicate that the scheme of this paper works under very general conditions, we have only been able to prove that it has to work for Chebyshev systems (see Subsection 2.1 above). This discrepancy seems to indicate that it might be profitable to investigate generalizations of Theorem 2.1 to sets of functions other than Chebyshev systems.

3. By combining Observation 2.3 and Remark 2.4 with results in Sections 3, 4, it is fairly straightforward to construct algorithms for the efficient interpolation of fairly large classes of singular functions. For example, the nodes $x_1, x_2, \ldots, x_{36}$ in Table 5 lead to a stable interpolation formula on the interval $[0, 1]$ for all functions of the form

$$\sum_{k=0}^{n} P_k(x) \cdot \sum_{j=1}^{m} \beta_{k,j} \cdot x^{\alpha_j}, \tag{55}$$

with $-0.3 \leq \alpha_j \leq 1$, $0 \leq k \leq 19$, and the precision of interpolation 10^{-15}. Interpolation schemes of this type are currently under vigorous investigation, and will be reported in the near future.

4. In many situations (especially, in the numerical solution of partial differential equations), it is desirable to have "quadrature" formulae that, in addition evaluating integrals, would evaluate certain pseudodifferential operators, i.e. derivative, Hilbert Transform, derivative of the Hilbert Transform, etc. Clearly, such "quadratures" can not have positive weights, except for the Hilbert Transform. Several such quadratures have been constructed numerically, and the appropriate theory appears to be fairly straightforward; this work will be reported at a later date.

5. While the theory of Gaussian Quadratures in one dimension is extremely simple and well-understood, no similar theory exists in higher dimensions, except for a few scattered results (see, for example, [9],[14]). The approach of this paper is quite different from the classical Gaussian Quadratures, and it appears possible to generalize it (at least, formally) to higher dimensions. While the advantages of such a construction would be significant, our investigation of it is at a very early stage. If successful, it will be reported at a later date.

References

[1] R. BULIRSCH, J. STOER, *Fehlerabschätzungen und Extrapolation mit Rationalen Funktionen bei Verfahren vom Richardson-Typus*, Numerische Mathematik, 6, 413-427 (1964).

[2] F. GANTMACHER AND M. KREIN, *Oscillation matrices and kernels and small oscillations of mechanical systems*, 2nd ed., Gosudarstv. Izdat. Tehn-Teor. Lit., Moscow, 1950 (Russian).

[3] W. GAUTCHI, *On Generating Orthogonal Polynomials*, SIAM Journal on Scientific and Statistical Computing, V. 3, No. 3, 1982.

[4] D. GOTTLIEB, S. ORSZAG, *Numerical Analysis of Spectral Methods*, SIAM, Philadelphia, 1977.

[5] T. HRYCAK, V. ROKHLIN, *An Improved Fast Multipole Algorithm for Potential Fields*, SIAM Journal of Scientific Computing, Vol. 19, No. 6, pp. 1804-1826, 1998.

[6] S. KARLIN, *The Existence of Eigenvalues for Integral Operators*, Trans. Am. Math. Soc. v. 113, pp. 1-17 (1964).

[7] S. KARLIN, AND W. J. STUDDEN, *Tchebycheff Systems with Applications In Analysis And Statistics*, John Wiley (Interscience), New York, 1966.

[8] M. G. KREIN, *The Ideas of P. L. Chebyshev and A. A. Markov in the Theory Of Limiting Values Of Integrals*, American Mathematical Society Translations, Ser. 2, Vol. 12, 1959, pp. 1-122.

[9] J. N. LYNESS, D. JESPERSEN, *Moderate Degree Symmetric Quadrature Rules for the Triangle*, Journal of the Institute of Mathematics and its Applications, 1975, V. 15, pp. 19-32.

[10] J. MA, V. ROKHLIN, AND S. WANDZURA, *Generalized Gaussian Quadratures For Systems of Arbitrary Functions*, SIAM Journal of Numerical Analysis, v. 33, No. 3, pp. 971-996, 1996.

[11] A. A. MARKOV, *On the limiting values of integrals in connection with interpolation*, Zap. Imp. Akad. Nauk. Fiz.-Mat. Otd. (8) 6 (1898), no.5 (Russian), pp. 146-230 of [12].

[12] A. A. MARKOV, *Selected papers on continued fractions and the theory of functions deviating least from zero*, OGIZ, Moscow-Leningrad, 1948 (Russian).

[13] J. STOER, R. BULIRSCH, *Introduction to Numerical Analysis, Second Edition*, Springer-Verlag, 1993.

[14] S. WANDZURA, H. XIAO, *Quadrature Rules on Triangles in* $\mathbb{R}^2$, Yale University Technical Report YALEU/DCS/RR-1168 (1998).

[15] N. YARVIN AND V. ROKHLIN, *Generalized Gaussian Quadratures and Singular Value Decompositions of Integral Operators*, Yale University Technical Report YALEU/DCS/ RR-1109 (1996), to appear in *SIAM Journal on Scientific Computing.*

[16] N. YARVIN AND V. ROKHLIN, *An Improved Fast Multipole Algorithm for Potential Fields on One-Dimensional Structures*, Yale University Technical Report YALEU/DCS/RR-1119 (1997), to appear in *SIAM Journal on Numerical Analysis.*