A new representation of generalized averaged Gauss quadrature rules

In Memory of Dirk P. Laurie.

Lothar Reichel

Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA

Miodrag M. Spalević

Department of Mathematics, Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11120 Belgrade 35, Serbia

Abstract

Gauss quadrature rules associated with a nonnegative measure with support on (part of) the real axis find many applications in Scientific Computing. It is important to be able to estimate the quadrature error when replacing an integral by an ℓ-node Gauss quadrature rule in order to choose a suitable number of nodes. A classical approach to estimate this error is to evaluate the associated $(2\ell+1)$-node Gauss–Kronrod rule. However, Gauss–Kronrod rules with $2\ell+1$ real nodes might not exist. The $(2\ell+1)$-node generalized averaged Gauss formula associated with the ℓ-node Gauss rule described in [M. M. Spalević, On generalized averaged Gaussian formulas, Math. Comp., 76 (2007), pp. 1483–1492] is guaranteed to exist and provides an attractive alternative to the $(2\ell+1)$-node Gauss–Kronrod rule. This paper describes a new representation of generalized averaged Gauss formulas that is cheaper to evaluate than the available representation.

Keywords: Gauss quadrature; averaged Gauss rule; generalized averaged Gauss rule

1. Introduction

Let $\mathrm{d}\omega$ be a nonnegative measure with infinitely many points of support on the real axis, and such that all moments $\mu_k = \int x^k \mathrm{d}\omega(x)$, $k = 0, 1, 2, \ldots$, exist. For notational

Email addresses: reichel@math.kent.edu (Lothar Reichel), mspalevic@mas.bg.ac.rs (Miodrag M. Spalević)

simplicity, we will assume that the measure has total mass one. We are interested in approximating integrals of the form

$$\mathcal{I}(f) = \int f(x)\mathrm{d}\omega(x) \tag{1.1}$$

by the ℓ-point Gauss rule

$$\mathcal{G}_\ell(f) = \sum_{k=1}^{\ell} f(x_k^{(\ell)}) w_k^{(\ell)} \tag{1.2}$$

associated with the measure $\mathrm{d}\omega$. This rule is characterized by

$$\mathcal{G}_\ell(f) = \mathcal{I}(f) \qquad \forall\ f \in \mathbb{P}_{2\ell-1},$$

where $\mathbb{P}_{2\ell-1}$ denotes the set of all polynomials of degree at most $2\ell-1$. The nodes $x_k^{(\ell)}$ are known to be distinct and to live in the convex hull of the support of $\mathrm{d}\omega$, and the weights $w_k^{(\ell)}$ are positive; see, e.g., Gautschi [5] or Szegő [17] for properties of Gauss quadrature rules.

It is important to be able to estimate the quadrature error

$$\mathcal{I}(f) - \mathcal{G}_\ell(f) \tag{1.3}$$

to assess whether the number of nodes, ℓ, has been chosen large enough to achieve an approximation of the integral (1.1) of desired accuracy. A classical approach to estimate the error (1.3) is to evaluate the $(2\ell+1)$-node Gauss–Kronrod quadrature rule associated with (1.2), if it exists. This Gauss–Kronrod rule is a quadrature rule of the form

$$\mathcal{K}_{2\ell+1}(f) = \sum_{k=1}^{\ell} f(x_k^{(\ell)}) \widehat{w}_k^{(2\ell+1)} + \sum_{k=\ell+1}^{2\ell+1} f(\widehat{x}_k^{(2\ell+1)}) \widehat{w}_k^{(2\ell+1)},$$

such that the nodes $x_k^{(\ell)}$, $k = 1, 2, \ldots, \ell$, are the nodes in the Gauss rule (1.2), and the Gauss–Kronrod nodes $\widehat{x}_k^{(2\ell+1)}$, $k = \ell+1, \ell+2, \ldots, 2\ell+1$, and the weights $\widehat{w}_k^{(2\ell+1)}$, $k = 1, 2, \ldots, 2\ell+1$, are determined so that

$$\mathcal{K}_{2\ell+1}(f) = \mathcal{I}(f) \qquad \forall\ f \in \mathbb{P}_{3\ell+1}.$$

Generally, the Gauss–Kronrod nodes $\widehat{x}_k^{(2\ell+1)}$, $k = \ell+1, \ell+2, \ldots, 2\ell+1$, are required to be real and to interlace the Gauss nodes. In addition, the Gauss–Kronrod weights $\widehat{w}_k^{(2\ell+1)}$, $k = 1, 2, \ldots, 2\ell+1$, should be positive. Efficient numerical methods for computing the nodes and weights of the Gauss–Kronrod rule $\mathcal{K}_{2\ell+1}$, when the nodes $\widehat{x}_k^{(2\ell+1)}$, $k = \ell+1, \ell+2, \ldots, 2\ell+1$, and weights $\widehat{w}_k^{(2\ell+1)}$, $k = 1, 2, \ldots, 2\ell+1$, satisfy these conditions are described in [1, 11]. The quadrature error (1.3) is approximated by $\mathcal{K}_{2\ell+1}(f) - \mathcal{G}_\ell(f)$.

However, for many measures $\mathrm{d}\omega$, including various Jacobi measures, and for certain numbers of nodes, Gauss–Kronrod rules, whose nodes and weights satisfy the above conditions do not exist; see Notaris [13] for a nice recent survey of Gauss–Kronrod rules and their properties. The non-existence of Gauss–Kronrod rules for important measures prompted Laurie [10] to develop anti-Gauss and averaged rules for the estimation of the error in Gauss rules. The $(\ell+1)$-point anti-Gauss rule, $\widetilde{\mathcal{G}}_{\ell+1}$, associated with the Gauss rule (1.2) is determined by the requirement

$$(\widetilde{\mathcal{G}}_{\ell+1} - \mathcal{I})(f) = -(\mathcal{G}_\ell - \mathcal{I})(f) \qquad \forall\ f \in \mathbb{P}_{2\ell+1}, \tag{1.4}$$

and the associated averaged rule is given by

$$\mathcal{A}_{2\ell+1}(f) = \frac{1}{2}(\mathcal{G}_\ell + \widetilde{\mathcal{G}}_{\ell+1})(f). \tag{1.5}$$

It follows from (1.4) that the degree of precision of $\mathcal{A}_{2\ell+1}$ is at least $2\ell+1$, i.e.,

$$\mathcal{A}_{2\ell+1}(f) = \mathcal{I}(f) \qquad \forall\ f \in \mathbb{P}_{2\ell+1}.$$

It is well known that the Gauss rule (1.2) can be represented by a symmetric tridiagonal matrix

$$T_\ell = \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & & & 0 \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \sqrt{\beta_{\ell-2}} & \alpha_{\ell-2} & \sqrt{\beta_{\ell-1}} \\ 0 & & & \sqrt{\beta_{\ell-1}} & \alpha_{\ell-1} \end{bmatrix} \in \mathbb{R}^{\ell\times\ell}, \tag{1.6}$$

whose entries $\alpha_k \in \mathbb{R}$ and $\beta_k > 0$ are recursion coefficients for the sequence of monic orthogonal polynomials $\{p_k\}_{k=0}^{\infty}$ associate with the measure $\mathrm{d}\omega$,

$$p_{k+1}(t) = (t-\alpha_k)p_k(t) - \beta_k\, p_{k-1}(t), \quad k = 0, 1, \dots\ ,$$

where $p_{-1}(t) \equiv 0$, $p_0(t) \equiv 1$; see, e. g., Gautschi [5] for details. The eigenvalues of T_ℓ are the nodes and the squared first components of the normalized eigenvectors are the weights of $\mathcal{G}_\ell$; see [5, 8]. Recall that the total mass of the measure is assumed to be one; otherwise the formula for the weights differs.

A popular approach to compute the nodes and weights of $\mathcal{G}_\ell$ is furnished by the Golub–Welsch algorithm [9], which requires only $c\ell^2 + \mathcal{O}(\ell)$ arithmetic floating point operations (flops), where $c > 0$ is a fairly small constant independent of ℓ; see also Laurie [12] for a more recent discussion on the computation of nodes and weights. We remark that Glaser et al. [7] describe an algorithm that is faster for certain classical orthogonal polynomials, such as Legendre, Hermite, and Laguerre polynomials, for large values of ℓ. However, in many applications ℓ is not large enough for the latter algorithm to be competitive.

One of the attractions of the anti-Gauss rule $\widetilde{\mathcal{G}}_{\ell+1}$ and the averaged rule $\mathcal{A}_{2\ell+1}$ are their ease of computation: the symmetric tridiagonal matrix $\widetilde{T}_{\ell+1}$ associated with the anti-Gauss rule $\widetilde{\mathcal{G}}_{\ell+1}$ is obtained by multiplying the last off-diagonal elements of the symmetric tridiagonal matrix $T_{\ell+1}$ associated with Gauss rule $\mathcal{G}_{\ell+1}$ by $\sqrt{2}$; see [10]. We denote the tridiagonal matrix so obtained by $\overline{T}_{\ell+1}$. The Golub–Welsch algorithm can be applied to compute the nodes and the weights of the anti-Gauss rule $\widetilde{\mathcal{G}}_{\ell+1}$ in $c\ell^2 + \mathcal{O}(\ell)$ flops. Thus, the computational cost of determining the nodes and weights of both the Gauss rule $\mathcal{G}_\ell$ and the averaged rule $\mathcal{A}_{2\ell+1}$ is $2c\ell^2 + \mathcal{O}(\ell)$ flops.

Spalević [16] observed that the averaged rule can be represented differently from (1.5) and this representation led to the generalized averaged quadrature rule associated with the Gauss rule $\mathcal{G}_\ell$ described below. We first outline the alternate representation of the averaged rule. Introduce the *reverse matrix*

$$T'_\ell = \begin{bmatrix} \alpha_{\ell-1} & \sqrt{\beta_{\ell-1}} & & & 0 \\ \sqrt{\beta_{\ell-1}} & \alpha_{\ell-2} & \sqrt{\beta_{\ell-2}} & & \\ & \ddots & \ddots & \ddots & \\ & & \sqrt{\beta_2} & \alpha_1 & \sqrt{\beta_1} \\ 0 & & & \sqrt{\beta_1} & \alpha_0 \end{bmatrix} \in \mathbb{R}^{\ell\times\ell},$$

which is obtained by reversing the order of the rows and columns of the matrix (1.6), and define the concatenated symmetric tridiagonal matrix

$$\widehat{T}_{2\ell+1} = \begin{bmatrix} T_\ell & \sqrt{\beta_\ell} e_\ell & 0 \\ \sqrt{\beta_\ell} e_\ell^T & \alpha_\ell & \sqrt{\beta_\ell} e_1^T \\ 0 & \sqrt{\beta_\ell} e_1 & T'_\ell \end{bmatrix} \in \mathbb{R}^{(2\ell+1)\times(2\ell+1)}. \tag{1.7}$$

Using results by Peherstofer [15], Spalević [16] showed that the nodes and weights of the averaged rule (1.5) are the eigenvalues and the squared first components of normalized eigenvectors, respectively, of the matrix (1.7). This representation is helpful for showing properties of the averaged rule; see [2, 3] for examples. However, the computation of the nodes and weights of the averaged rule (1.5) by applying the Golub–Welsch algorithms to the matrix (1.7) requires $4c\ell^2 + \mathcal{O}(\ell)$ flops in addition to the $c\ell^2 + \mathcal{O}(\ell)$ flops necessary for the computation of the nodes and weights of the Gauss rule (1.2). Hence, the computation of the nodes and weights of the averaged rule (1.5) by applying the Golub–Welsch algorithm to the matrix (1.7) is more expensive than the approach to compute the nodes and weights of this rule outlined above.

In [16], Spalević also introduced the generalized averaged Gauss rule $\breve{\mathcal{G}}_{2\ell+1}$ with $2\ell+1$ nodes associated with the Gauss rule (1.2). Its nodes and weights are the eigenvalues and the square of the first components of the normalized eigenvectors, respectively, of the matrix

$$\breve{T}_{2\ell+1} = \begin{bmatrix} T_\ell & \sqrt{\beta_\ell} e_\ell & 0 \\ \sqrt{\beta_\ell} e_\ell^T & \alpha_\ell & \sqrt{\beta_{\ell+1}} e_1^T \\ 0 & \sqrt{\beta_{\ell+1}} e_1 & T'_\ell \end{bmatrix} \in \mathbb{R}^{(2\ell+1)\times(2\ell+1)}, \tag{1.8}$$

which differs from the matrix (1.7) in that the elements in positions $(\ell+1,\ell+2)$ and $(\ell+2,\ell+1)$ are replaced by $\sqrt{\beta_{\ell+1}}$. This generalized averaged Gauss rule is exact for at least all polynomials in $\mathbb{P}_{2\ell+2}$ and, thus, generally of higher degree of precision than the averaged Gauss rule (1.5). The generalized averaged Gauss rule $\breve{\mathcal{G}}_{2\ell+1}$ can be used to estimate the quadrature error (1.3) similarly as the averaged Gauss rule (1.5). In this application, one typically uses $\breve{\mathcal{G}}_{2\ell+1}(f)$ as an approximation of the integral (1.1), and then considers $\breve{\mathcal{G}}_{2\ell+1}(f)-\mathcal{G}_\ell(f)$ an estimate of the quadrature error.

A drawback of the rule $\breve{\mathcal{G}}_{2\ell+1}$, compared to the averaged rule (1.5), is that the former is more expensive to compute. The computation of its nodes and weights by the Golub–Welsch algorithm requires $4c\ell^2+\mathcal{O}(\ell)$ flops, just like the computation of the eigenvalues and first components of the eigenvectors of the matrix (1.7). In addition, the nodes and weights of the Gauss rule have to be calculated.

It is the purpose of the present paper to describe a new representation of the generalized averaged Gauss rule $\breve{\mathcal{G}}_{2\ell+1}$ that is analogous to the representation (1.5) and, therefore, is cheaper to compute. This representation is described in Section 2. A computed example with timings is presented in Section 3, and Section 4 contains concluding remarks.

2. A new representation of generalized averaged Gauss rules

The following result provides a decomposition of the generalized averaged quadrature rule $\breve{\mathcal{G}}_{2\ell+1}$ that is analogous to the representation (1.5) of the averaged Gauss rule.

Theorem 1. *The generalized averaged Gauss quadrature rule defined by the tridiagonal matrix* (1.8) *can be expressed as*

$$\breve{\mathcal{G}}_{2\ell+1}=\frac{\beta_{\ell+1}}{\beta_\ell+\beta_{\ell+1}}\mathcal{G}_\ell+\frac{\beta_\ell}{\beta_\ell+\beta_{\ell+1}}\mathcal{G}^*_{\ell+1}, \tag{2.1}$$

where the quadrature rule $\mathcal{G}^*_{\ell+1}$ is determined by the symmetric tridiagonal matrix

$$T^*_{\ell+1}=\begin{bmatrix}
\alpha_0 & \sqrt{\beta_1} & & & & & 0\\
\sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & & & \\
& \ddots & \ddots & \ddots & & & \\
& & \sqrt{\beta_{\ell-2}} & \alpha_{\ell-2} & \sqrt{\beta_{\ell-1}} & & \\
& & & \sqrt{\beta_{\ell-1}} & \alpha_{\ell-1} & \sqrt{\beta_\ell+\beta_{\ell+1}} & \\
0 & & & & \sqrt{\beta_\ell+\beta_{\ell+1}} & \alpha_\ell &
\end{bmatrix}\in\mathbb{R}^{(\ell+1)\times(\ell+1)}. \tag{2.2}$$

It follows that using the representation (2.1)*, one can compute the nodes and weights of the quadrature rule in $2c\ell^2+\mathcal{O}(\ell)$ flops by the Golub–Welsch algorithm. This is the same flop count as required for computing the nodes and weights for the averaged rule* (1.5)*. The flop count includes the computation of the nodes and weights for the Gauss rule* (1.2)*.*

Proof. The nodes of the generalized averaged Gauss quadrature rule $\breve{\mathcal{G}}_{2\ell+1}$, defined by the tridiagonal matrix (1.8), are the zeros of the polynomial (cf. [16])

$$t_{2\ell+1} = p_\ell \cdot F^S_{\ell+1} \tag{2.3}$$

with

$$F^S_{\ell+1} = p_{\ell+1} - \beta_{\ell+1} \cdot p_{\ell-1}, \tag{2.4}$$

where p_j denotes the monic orthogonal polynomial of degree j associated with the measure $\mathrm{d}\omega$. The nodes of the quadrature rule $\mathcal{G}^*_{\ell+1}$ are the zeros of polynomial $F^S_{\ell+1}$ in (2.3), which has the form (2.4); cf. [16, Eq. (2.6) on p. 1487].

We now apply [4, Proposition 1] with $q_{\ell+1} = F^S_{\ell+1}$ and

$$(1+\gamma)\frac{h_\ell}{h_{\ell-1}} = \beta_{\ell+1},$$

where

$$h_\ell = \beta_0\, \beta_1 \cdots \beta_\ell$$

and

$$\gamma = \gamma_\ell := -1 + \frac{\beta_{\ell+1}}{\beta_\ell}. \tag{2.5}$$

We obtain from [4, Eq. (3)] that

$$\left(\mathcal{G}^*_{\ell+1} - \mathcal{I}\right)(x^k) = -\frac{\beta_{\ell+1}}{\beta_\ell}\left(\mathcal{G}_\ell - \mathcal{I}\right)(x^k), \qquad k = 0, 1, \ldots, 2\ell+1,$$

where $\mathcal{I}$ denotes the integral operator (1.1). Equation (2.5) and [4, Eq. (4)] now yield (2.1). □

We remark that among all possible quadrature rules $Q_{2\ell+1}$ in [4, Eq. (4)], the rule $\breve{\mathcal{G}}_{2\ell+1}$ has maximal degree of exactness, $2\ell+2$, for general measures. When the measure $\mathrm{d}\omega$ in (1.1) is even, the degree of exactness increases to $2\ell+3$.

To estimate the error $\mathcal{I}(f) - \mathcal{G}_\ell(f)$, we can use the formula

$$\mathcal{I}(f) - \mathcal{G}_\ell(f) \approx \breve{\mathcal{G}}_{2\ell+1}(f) - \mathcal{G}_\ell(f) = \frac{\beta_\ell}{\beta_\ell + \beta_{\ell+1}}\left(\mathcal{G}^*_{\ell+1}(f) - \mathcal{G}_\ell(f)\right). \tag{2.6}$$

The nodes and weights of the expression $\mathcal{G}^*_{\ell+1} - \mathcal{G}_\ell$ can be computed in $2c\ell^2 + \mathcal{O}(\ell)$ flops by computing the nodes and weights for the quadrature rules $\mathcal{G}^*_{\ell+1}$ and $\mathcal{G}_\ell$ separately, similarly as in the computation of the averaged rule (1.5). In particular, this flop count includes the calculation of the Gauss rule $\mathcal{G}_\ell$.

As the generalized averaged Gauss rule $\breve{G}_{2\ell+1}$ has maximal degree of exactness among all averaged quadrature rules, and its computation requires about the same number of flops as the averaged rule (1.5), we propose that the right-hand side of (2.6) be used for estimating the error in the Gauss rule (1.2).

3. Computed examples

Many illustrations of the performance of generalized averaged Gauss rules for the estimation of the error in the underlying Gauss quadrature rule have been described in [2, 3, 14]. We therefore only present timings for the evaluation of the ℓ-node Gauss rule (1.2) and the $(2\ell+1)$-node generalized averaged Gauss rule when the latter is represented by the matrix (1.8) or the formula (2.1). We use a Matlab implementation `gauss.m` of the Golub–Welsch algorithm made available by Gautschi [6]. This implementation does not fully exploit the structure of the problem and computes the (full) spectral factorization of the tridiagonal matrix to which it is applied. This code requires $\mathcal{O}(\ell^3)$ flops to compute the spectral factorization of the matrix (1.6). All computations are carried out using Matlab version R2016b on a MacBook Pro laptop computer with a 2.5 GHz Intel Core i5 processor and 8GB 1600MHz DDR3 memory. The computations are performed with about 15 significant decimal digits.

Table 1: Ratios of CPU times for computing the Gauss rule (1.2) and the generalized averaged Gauss rule using the representation (1.8), and for computing the Gauss rule (1.2) and the generalized averaged Gauss rule using the representation (2.1) for several values of ℓ. The table shows averages over 1000 timings each of 10^6 runs.

ℓ	$\frac{\text{time for computing the rules (1.2) and (1.8)}}{\text{time for computing the rules (1.2) and (2.1)}}$
20	$\gtrapprox 2.5$
40	$\gtrapprox 1.8$
80	$\gtrapprox 2.6$
160	$\gtrapprox 3.2$

Example 3.1. We compute the nodes and weights of the Gauss rule (1.2) as well as of the generalized averaged Gauss rule by using the Golub–Welsch algorithm as implemented by the Matlab function `gauss.m` in [6]. The representations (1.8) and (2.1) for the generalized averaged Gauss rule are used. The former representation requires that the Golub–Welsch algorithm be applied to the matrix (1.8) of order $2\ell+1$; the latter only demands application of the Golub–Welsch algorithm to the matrix (2.2) of order $\ell+1$ since the Gauss rule (1.2) is computed separately. The flop count for determining the quadrature rules (1.2) and (1.8) is $9c\ell^3 + \mathcal{O}(\ell^2)$ using the code `gauss.m`, while the flop count for computing (1.2) and (2.1) is only $2c\ell^3 + \mathcal{O}(\ell^2)$ using the same code. The ratio of the flop counts therefore is about 4.5 for ℓ large. In addition to the arithmetic operations, the timings include many other things, such as function calls and data access. Table 1 shows the ratio of the computing times for these approaches when $\mathrm{d}\omega(t) \equiv 1$. The variance of the computing times between different runs is fairly large. We therefore report the average ratio of 1000 computations of the quadrature rules 10^6 times each. The symbol $\gtrapprox$ stands for "usually larger than and often fairly close to" the number reported. What is clear from the timings is that

computing the quadrature rules (1.2) and (2.1) is faster than when using (1.2) and the matrix (1.8).

4. Conclusion

A new representation of generalized averaged Gauss rules is derived. It is analogous to the representation of averaged Gauss rules described by Laurie [10]. A numerical example illustrates that it is faster to compute than the representation derived in [16].

Acknowledgmment

The authors would like to thank the referees for carefully reading this manuscript and for their comments. The work of LR was supported in part by NSF grants DMS-1720259 and DMS-1729509, and the work of MMS was supported in part by the Serbian Ministry of Education, Science and Technological Development and Science Fund of the Republic of Serbia.

References

[1] D. Calvetti, G. H. Golub, W. B. Gragg, and L. Reichel, Computation of Gauss–Kronrod quadrature rules, Math. Comp., 69 (2000), pp. 1035–1052.

[2] D. Lj. Djukić, L. Reichel, M. M. Spalević, and J. D. Tomanović, Internality of generalized averaged Gauss rules and their truncations for Bernstein–Szegő weights, Electron. Trans. Numer. Anal., 45 (2016), pp. 405–419.

[3] D. Lj. Djukić, L. Reichel, M. M. Spalević, and J. D. Tomanović, Internality of generalized averaged Gaussian quadrature rules and truncated variants for modified Chebyshev measures of the second kind, J. Comput. Appl. Math., 345 (2019), pp. 70–85.

[4] S. Ehrich, On stratified extensions of Gauss–Laguerre and Gauss–Hermite quadrature formulas, J. Comput. Appl. Math., 140 (2002), pp. 291–299.

[5] W. Gautschi, Orthogonal Polynomials: Approximation and Computation, Oxford University Press, Oxford, 2004.

[6] W. Gautschi, OPQ: A Matlab suite of programs for generating orthogonal polynomials and related quadrature rules, available at https://www.cs.purdue.edu/archives/2002/wxg/codes/OPQ.html

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

[8] G. H. Golub and G. Meurant, Matrices, Moments and Quadrature with Applications, Princeton University Press, Princeton, 2010.

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

[10] D. P. Laurie, Anti-Gaussian quadrature formulas, Math. Comp., 65 (1996), pp. 739–747.

[11] D. P. Laurie, Calculation of Gauss-Kronrod quadrature rules, Math. Comp., 66 (1997), pp. 1133–1145.

[12] D. P. Laurie, Computation of Gauss-type quadrature formulas, J. Comput. Appl. Math., 127 (2001), pp. 201–217.

[13] S. E. Notaris, Gauss–Kronrod quadrature formulae - a survey of fifty years of research, Electron. Trans. Numer. Anal., 45 (2016), pp. 371–404.

[14] L. Reichel, M. M. Spalević, and T. Tang, Generalized averaged Gauss quadrature rules for the approximation of matrix functionals, BIT Numer. Math., 56 (2016), pp. 1045–1067.

[15] F. Peherstorfer, On positive quadrature formulas, in Numerical Integration IV, eds. H. Brass and G. Hämmerlin, Intern. Ser. Numer.Math. # 112, Birkhäuser, Basel, 1993, pp. 297–313.

[16] M. M. Spalević, On generalized averaged Gaussian formulas, Math. Comp., 76 (2007), pp. 1483–1492.

[17] G. Szegő, Orthogonal Polynomials, 4th ed., Amer. Math. Society, Providence, 1975.