Image Cover Sheet

CLASSIFICATION	SYSTEM NUMBER
UNCLASSIFIED	143688

TITLE

CQLW - A PACKAGE OF SUBROUTINES FOR POLYNOMIALS ORTHOGONAL WITH RESPECT TO ln\(1/x\), INCLUDING THEORY AND APPLICATIONS

System Number:

Patron Number:

Requester:

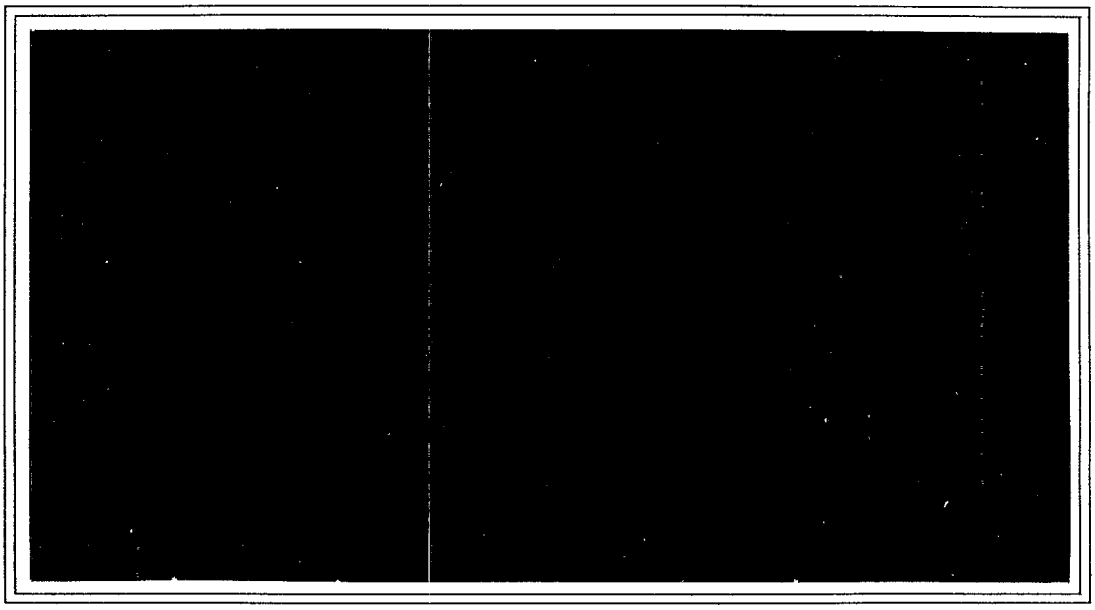
Notes:

DSIS Use only:

Deliver to: RL

Defence Research Establishment Pacific
Centre de recherches pour la défense pacifique

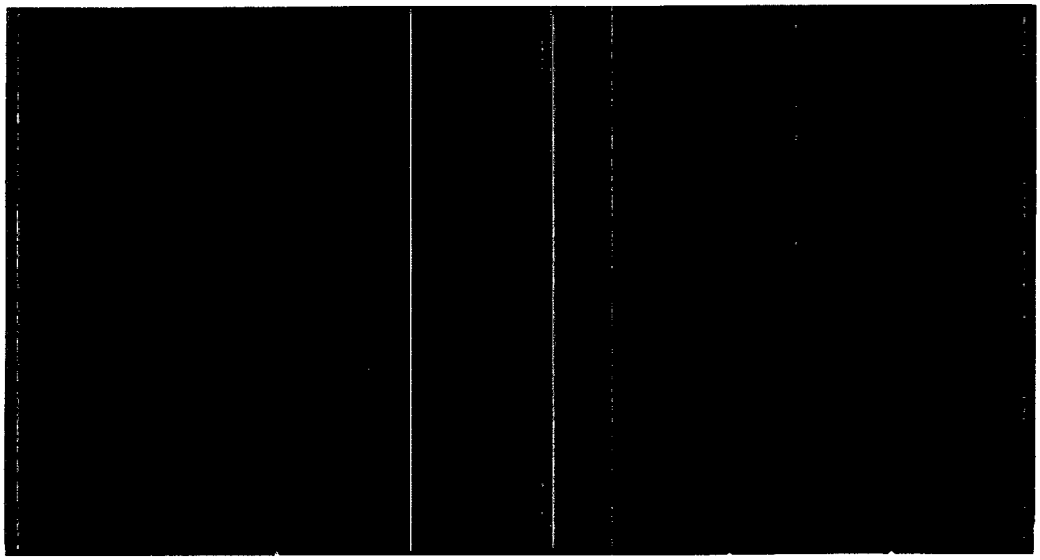
DSIS.



DEFENCE RESEARCH ESTABLISHMENT PACIFIC

Research and Development Branch
Department of National Defence

Canada



Defence Research Establishment Pacific
Centre de recherches pour la défense pacifique

DEFENCE RESEARCH ESTABLISHMENT PACIFIC

CFB Esquimalt, FMO Victoria, B.C. V0S 1B0

DREP Technical Memorandum 94-108

CQLW - A PACKAGE OF SUBROUTINES FOR POLYNOMIALS ORTHOGONAL WITH RESPECT TO ln(1/x), INCLUDING THEORY AND APPLICATIONS

by

T.W. Dawson

June 1994

Approved By:

Chief DREP

Research and Development Branch
Department of National Defence

Canada

Abstract

The main thrust of this document is an improved method for computing polynomials $\{\Lambda_n(x)\}$ orthogonal with respect to the positive weight function $\ln(1/x)$ on the interval $x \in [0,1]$. An implementation of Wheeler's [1] algorithm was recently published in *Numerical Recipes* [2]. However, the modified moments used are prone to underflow on computers with limited precision and floating-point range. The problems can be circumvented by employing scaling, and by working in terms of ratios.

The paper contains a summary of general orthogonal polynomial theory, including a derivation of functional approximation and Gaussian quadrature from a matrix theory viewpoint. Subsequently, a derivation of Wheeler's algorithm, together with the two variations mentioned above, is presented. Appendix A documents a set of FORTRAN routines implementing the theory, including routines for evaluating the recursion coefficients, polynomial values and derivatives, points and weights for Gaussian quadrature, integrating scalar- and vector-valued functions, and approximating functions in series of these orthogonal polynomials. By way of illustration, the first 128 recursion coefficients are tabulated, together with points and weights, for Gaussian quadrature of $\int_0^1 \ln(1/x)\, F(x)\, dx$, of orders $2, 3, 4, 5, 6, 8, 10, 16, 32, 64$ and 128. Appendix B contains MAPLE© source code for working with these polynomials, and indicates how the code can be modified to accommodate other weight functions and intervals. For illustrative purposes, exact values of the first ten recursion coefficients and orthogonal polynomials are given.

Résumé

Ce document est axé sur une méthode améliorée pour calculer les polynômes $\{\Lambda_n(x)\}$, orthogonaux par rapport à la fonction de poids positive $\ln(1/x)$ sur l'intervalle $x \in [0,1]$. Une mise en application de l'algorithme de Wheeler [1] a paru récemment dans *Numerical Recipes* [2]. Néanmoins, les moments modifiés que l'on a utilisés sont enclins au dépassement de capacité négatif chez les ordinateurs à la précision et la portée de virgule flottante limitées. Il est possible de circonvenir ces problèmes en employant la mise à l'échelle et en calculant en termes de rapports.

Cet article contient un sommaire de théorie générale de polynômes orthogonaux, y compris une dérivation d'approximation fonctionnelle et de quadrature de Gauss du point de vue de la théorie de matrice. Nous présentons ensuite une dérivation de l'algorithme de Wheeler ainsi que les deux variations mentionnées ci-dessus. Dans l'appendice A nous documentons un ensemble de programmes FORTRAN qui met à exécution cette théorie, incluant des programmes pour évaluer les coefficients de récurrence, les valeurs et dérivés polynomiaux, les points et poids pour la quadrature de Gauss, intégrant des fonctions valorisées par échelle et par vecteur, et s'approchant par approximation des fonctions en série de ces polynômes orthogonaux. Pour illustrer ceci, nous tabulons les premiers 128 coefficients de récurrence, ainsi que les points et les poids, pour la quadrature de Gauss $\int_0^1 \ln(1/x)\, F(x)\, dx$, d'ordres 2,3,4,5,6,8,10,16,32,64,et 128. L'appendice B contient le code de source MAPLE© pour faire des calculs avec ces polynômes, et indique comment le code peut être modifié pour accommoder d'autres fonctions de poids et d'intervalles. Pour servir les fins d'illustration, nous présentons les valeurs précises des premiers dix coefficients de récurrence et de polynômes orthogonaux.

Contents

List of Figures

List of Tables

1 Introduction

Solutions to a wide variety of wave propagation problems in such areas as acoustics and electromagnetics can be obtained in terms of integral representations with Green's function kernels [3]. Depending on whether boundary conditions are known or not, these integral representations may either be evaluated numerically, or used to derive integral equation(s), whose solution allows the desired physical fields to be evaluated by further integration. The Green's functions are obtained from solutions of differential equations, such as Helmholtz' Equation, with singular (point) forcing functions. Thus the Green's functions themselves have an inherently singular nature, as the observation point approaches the source point. This singular nature requires that care be used in working with integral representations for fields solutions if the observation point lies close to a boundary. This is particularly so in solving associated integral equations, for the observation point then lies within the domain of integration.

Green's functions for the two-dimensional Helmholtz equation generally have a logarithmic singularity. For instance, the Green's function for a line source [3] is given in terms of the order-zero Hankel function. Successively higher derivatives of this Green's function acquire increasingly severe algebraic singularities [4], while retaining a logarithmically singular component. Similarly, the Green's function for the axially symmetric Helmholtz equation [5, 6, 7], while not available in closed form, can be shown to be logarithmically singular, either from expressions involving elliptic integrals as considered in the preceeding references, or from expressions involving hypergeometric functions [8]. This latter approach shows that successively higher derivatives of the axially symmetric Green's function also acquire increasingly severe algebraic singularities, while retaining a logarithmically singular component.

In working with integral equations, the Green's function singularities must be handled properly. Frequently, the algebraic components can be treated analytically [8]. Provided that only limited accuracy is required, the logarithmic singularities can be handled by straightforward numerical quadrature methods, ignoring the slowly-varying logarithmic term. The errors in higher-order quadrature schemes generally involve higher-order derivatives of the integrand, and it is here that the logarithm, with its increasingly-singular higher-order derivatives, can cause problems. Thus for greater accuracy, more specialized schemes, such as Gaussian quadrature [2] may be required.

For the near-singular case where the observation point approaches a boundary, but does not reach it, sets of Gaussian quadrature rules, using the singular terms as weight functions and parametrized by the separation distance from the boundary, can be constructed [9]. For points lying on the boundary, the integrals over boundary elements (where only a logarithmic singularity causes problems) can generally be converted [8], by a change of variable, to integrals of the form

$$\int_0^1 \ln(1/x)\, F(x)\, dx.$$

Here, the positive weight function $\ln(1/x)$ can be used to generate Gaussian quadrature rules that allow the above integral to be evaluated to high accuracy. This will be considered in more detail in this report.

Crow [10] has recently considered a related quadrature scheme which is designed to be exact for integrands of the form

$$\pi(x) + \varpi(x) \ln(x)$$

where π and ϖ are polynomials. He obtains a set of linear equations for the appropriate orthogonal polynomials by using shifted Legendre polynomials with a non-linear dependence on the (unknown) points. Iterative and Newton methods are then used to solve the resulting equations, with high-precision arithmetic. Results are given for up to 7-point rules. Crow notes that since the above integrand is scale-invariant, his rules

are also exact for quadrature on the interval $(0, h)$. It follows that the 4-point rule is suitable in boundary integral equation methods for Hankel-function kernels using quadratic basis functions.

The main thrust of this document is to describe an improved method for computing polynomials $\{\Lambda_n(x)\}$ orthogonal with respect to the positive weight function $\ln(1/x)$ on the interval $x \in [0, 1]$. An implementation of Wheeler's [1] algorithm was recently published in Numerical Recipes [2]. However, the modified moments used in his approach are prone to underflow on computers with limited precision and floating-point range. The problems can be circumvented by employing scaling, and by working in terms of ratios, procedures which will be considered in Sections 3.2 and 3.3.

Section 2 of this report presents a summary of relevant general orthogonal polynomial theory, including derivations of functional approximation in orthogonal polynomial series, and of Gaussian quadrature, from a matrix theory viewpoint. The matrix approach readily leads to such results as the positivity of Gaussian quadrature weights, the exactness of n-point Gaussian quadrature for polynomials of degree $\leq 2n-1$ and a pair of discrete orthogonality relationships involving values of the n polynomials of order $< n$, when evaluated at the zeros of the polynomial of order n.

Section 3 considers the fundamental problem of evaluation of the recursion coefficients for orthogonal polynomials. A derivation of Wheeler's Algorithm [1], as given in Numerical Recipes [2], is presented. It is then shown how the modified moments which he used can lead to underflow problems. Two additional methods are then given, which work with various ratios; these approaches lead to better-behaved methods.

Appendix A documents a set of FORTRAN routines implementing various aspects of the theory considered in the body of this report, as applied to the polynomials $\{\Lambda_n(x)\}$. The package includes routines for evaluation of the recursion coefficients, polynomial values and derivatives, points and weights for Gaussian quadrature, integration of scalar- and vector-valued functions, and approximation of functions in series of orthogonal polynomials. By way of illustration, the first 128 recursion coefficients are tabulated, together with points and weights for Gaussian quadrature with a logarithmic weight function, for rules of orders $2, 3, 4, 5, 6, 8, 10, 16, 32, 64$ and 128.

Appendix B considers the application of symbolic algebra software to the orthogonal polynomials $\{\Lambda_n(x)\}$. It contains MAPLE© source code for working with these polyomials, and it is indicated how the code can be modified to accommodate other weight functions and intervals. For illustrative purposes, exact values of the first ten recursion coefficients and orthogonal polynomials are given. The first ten orthogonal polynomials, normalized to unity at $x = 1$, are also plotted.

2 Summary of Orthogonal Polynomial Theory

2.1 Orthogonal Polynomial Recursions

Let $W(x) > 0$ be a positive weight function defined on the interval $a < x < b$. Then

$$\langle f \,|g\rangle \equiv \int_a^b W(x)\, f(x)\, g(x)\, dx \tag{1}$$

defines an inner product [11]. A family of polynomials $\{p_n(x),\ (n = 0, 1, \ldots, \infty)\}$ orthogonal with respect to this weight function satisfies

$$\langle p_n \,|p_k\rangle = N_k^2 \delta_{n,k}, \tag{2}$$

where $p_k(x)$ is a polynomial of degree k, N_k^2 is a normalization factor, and where

$$\delta_{n,k} = \begin{cases} 1 & (n = k) \\ 0 & (n \neq k) \end{cases} \tag{3}$$

is the Kronecker delta-function. Such a family can be constructed recursively using the scheme [2]

$$p_{-1}(x) \equiv 0; \qquad p_0(x) \equiv 1; \qquad p_{n+1}(x) = (x - a_n)\, p_n(x) - b_n p_{n-1}(x), \quad (n = 0, 1, \ldots, \infty), \tag{4}$$

where the coefficients are given by

$$a_n \equiv \langle x p_n \,|p_n\rangle / N_n^2 \quad (n = 0, 1, \ldots, \infty) \tag{5}$$

and

$$b_n \equiv N_n^2 / N_{n-1}^2 \quad (n = 1, 2, \ldots, \infty). \tag{6}$$

In this scheme, the value of b_0 is arbitrary. Once the coefficients $\{a_n\}$ and $\{b_n\}$ are known, Eq.(4) represents an efficient method for computing values of $p_n(x)$. Note that $p_n(x)$ is of degree n, and is monic (i.e., the coefficient of x^n is unity). If desired, the (simultaneous) recursion scheme

$$p'_{-1}(x) = 0; \qquad p'_0(x) = 0; \qquad p'_{n+1}(x) = p_n(x) + (x - a_n)\, p'_n(x) - b_n p'_{n-1}(x), \quad (n = 0, 1, \ldots, \infty) \tag{7}$$

gives the derivatives.

Alternative normalizations are frequently useful. Given the coefficients $\{b_n\}$, the norms N_n^2 can be computed from

$$N_0^2 = \int_a^b W(x)\, dx; \qquad N_n^2 = N_0^2 \prod_{j=1}^{n} b_j, \quad (n \geq 1). \tag{8}$$

Thus an orthonormal family, satisfying

$$\langle \hat{p}_n \,|\hat{p}_k\rangle = \delta_{n,k}, \quad (n, k = 0, 1, \ldots, \infty) \tag{9}$$

can be defined by

$$\hat{p}_n(x) \equiv \pm p_n(x) / N_n, \tag{10}$$

where $N_n > 0$ by definition. The choice of sign here is to be the same for all polynomials in the family. Upon substitution of definition (10) into the monic recurrence relation (4), it follows that the orthonormal polynomials satisfy the recursion scheme

$$\hat{p}_{-1}(x) \equiv 0; \qquad \hat{p}_0(x) \equiv 1/N_0; \qquad \hat{p}_{n+1}(x) = \{(x - a_n)\, \hat{p}_n(x) - \beta_n\, \hat{p}_{n-1}(x)\} / \beta_{n+1}, \quad (n = 0, 1, \ldots), \tag{11}$$

where

$$\beta_n = \sqrt{b_n} \equiv N_n / N_{n-1}, \quad (n = 1, 2, \ldots). \tag{12}$$

The associated recursion scheme

$$\hat{p}'_{-1}(x) \equiv 0; \quad \hat{p}'_0(x) \equiv 0; \quad \hat{p}'_{n+1}(x) = \left\{\hat{p}_n(x) + (x - a_n)\, \hat{p}'_n(x) - \beta_n\, \hat{p}'_{n-1}(x)\right\} / \beta_{n+1}, \quad (n = 0, 1, \ldots, \infty) \tag{13}$$

gives the derivatives of this orthonormal family.

It is noted in the next subsection that the orthogonal polynomials cannot vanish at the either of the endpoints $\{a, b\}$. Consequently, the scaled polynomials

$$q_{d,n}(x) \equiv p_n(x) / p_n(d), \qquad (d \in \{a, b\}) \tag{14}$$

are well-defined, orthogonal, and satisfy $q_{d,n}(d) = 1$. These scaled polynomials can be computed efficiently as follows. Let ratios R_n be defined by

$$R_n \equiv p_n(d)/p_{n-1}(d), \quad (n = 1, 2, \ldots). \tag{15}$$

It then follows from Eq.(4) that the scaled polynomials $q_{d,n}$, together with the ratios R_n, can be computed from the joint recursion scheme

$$\begin{gathered} R_1 = d - a_0; \quad q_{d,0}(x) = 1; \quad q_{d,1}(x) = (x - a_0)/R_1; \\ \left.\begin{array}{rcl} R_{n+1} & = & (d - a_n) - b_n/R_n, \\ q_{d,n+1}(x) & = & \{(x - a_n)\, q_{d,n}(x) - b_n q_{d,n-1}(x)/R_n\}/R_{n+1}, \end{array}\right\} \quad (n = 1, 2, \ldots, \infty). \end{gathered} \tag{16}$$

The additional recursion

$$\begin{gathered} q'_{d,0}(x) = 0; \qquad q'_{d,1}(x) = 1/R_1; \\ q'_{d,n+1}(x) = \left\{q_{d,n}(x) + (x - a_n)\, q'_{d,n}(x) - b_n q'_{d,n-1}(x)/R_n\right\}/R_{n+1}, \quad (n = 1, 2, \ldots, \infty) \end{gathered} \tag{17}$$

yields the derivatives.

2.2 Orthogonal Polynomial Zeros

The polynomial $p_n(x)$ has n roots [2], x_k^n, $(k = 1, \ldots, n)$, which satisfy

$$a < x_1^n < x_2^n < \cdots < x_n^n < b. \tag{18}$$

Moreover, the zeros of consecutive polynomials interlace [2], i.e.,

$$a < x_1^{n+1} < x_1^n < x_2^{n+1} < x_2^n < \cdots < x_n^n < x_{n+1}^{n+1} < b. \tag{19}$$

This feature provides a method for computing all the zeros of a consecutive set of orthogonal polynomials, in that the zeros of a given polynomial, in conjunction with the interval endpoints, serve as brackets for the zeros of the next higher polynomial in the sequence.

As indicated in Numerical Recipes [2], the first $n - 1$ instances of Eq.(11) can be arranged in matrix form as

$$x \begin{pmatrix} \hat{p}_0(x) \\ \hat{p}_1(x) \\ \vdots \\ \hat{p}_{n-1}(x) \end{pmatrix} = \begin{pmatrix} a_0 & \beta_1 & 0 & \ldots & \ldots & 0 \\ \beta_1 & a_1 & \beta_2 & 0 & & \vdots \\ 0 & \beta_2 & a_2 & \beta_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \beta_{n-2} & a_{n-2} & \beta_{n-1} \\ 0 & \ldots & \ldots & 0 & \beta_{n-1} & a_{n-1} \end{pmatrix} \cdot \begin{pmatrix} \hat{p}_0(x) \\ \hat{p}_1(x) \\ \vdots \\ \hat{p}_{n-1}(x) \end{pmatrix} + \beta_n \hat{p}_n(x) \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}. \tag{20}$$

Using a symbolic matrix notation, this becomes

$$x\hat{\mathbf{p}}^n(x) = \mathsf{J}^n \hat{\mathbf{p}}^n(x) + \beta_n \hat{p}_n(x)\, \hat{\mathbf{e}}_n, \tag{21}$$

where $\hat{\mathbf{p}}^n(x)$ is the column n-vector

$$\hat{\mathbf{p}}^n(x) = (\hat{p}_0(x), \hat{p}_1(x), \cdots \hat{p}_{n-1}(x))^T, \tag{22}$$

$\hat{\mathbf{e}}_n$ is the unit column n-vector

$$\hat{\mathbf{e}}_n = (0, 0, \ldots, 0, 1)^T, \tag{23}$$

and J^n is the symmetric, tridiagonal, $n \times n$ matrix

$$\mathsf{J}^n = \begin{pmatrix} a_0 & \beta_1 & 0 & \cdots & \cdots & 0 \\ \beta_1 & a_1 & \beta_2 & 0 & & \vdots \\ 0 & \beta_2 & a_2 & \beta_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \beta_{n-2} & a_{n-2} & \beta_{n-1} \\ 0 & \cdots & \cdots & 0 & \beta_{n-1} & a_{n-1} \end{pmatrix}. \tag{24}$$

Thus, if the abbreviation

$$\mathbf{r}_k^n \equiv \hat{\mathbf{p}}^n (x_k^n), \quad (k = 1, \ldots, n) \tag{25}$$

is introduced, it is clear (since $\hat{p}_n (x_k^n) = 0$) that

$$\mathsf{J}^n \mathbf{r}_k^n = x_k^n \mathbf{r}_k^n \tag{26}$$

(i.e., x_k^n is an eigenvalue of J^n, with associated eigenvector $\mathbf{r}_k^n$).

Because (by Eq.(18)) the eigenvalues are distinct, the eigenvectors are orthogonal [11]. Thus

$$\mathbf{r}_k^n \cdot \mathbf{r}_j^n \equiv (w_k^n)^{-1} \delta_{j,k}, \quad (j, k = 1, \ldots, n). \tag{27}$$

This relationship serves to define the terms w_k^n, which therefore have the form

$$w_k^n = \frac{1}{\left\{ [\hat{p}_0 (x_k^n)]^2 + [\hat{p}_1 (x_k^n)]^2 + \cdots + [\hat{p}_{n-1} (x_k^n)]^2 \right\}}, \quad (k = 1, \ldots, n). \tag{28}$$

Clearly, the quantities w_k^n are all positive.

Now let matrix R^n have the eigenvectors $\mathbf{r}_k^n$ as columns, so that

$$\mathsf{R}^n \equiv \left(\mathbf{r}_1^n \mid \mathbf{r}_2^n \mid \cdots \mid \mathbf{r}_{n-1}^n \right) = \begin{pmatrix} \hat{p}_0 (x_1^n) & \hat{p}_0 (x_2^n) & \cdots & \hat{p}_0 (x_n^n) \\ \hat{p}_1 (x_1^n) & \hat{p}_1 (x_2^n) & \cdots & \hat{p}_1 (x_n^n) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{p}_{n-1} (x_1^n) & \hat{p}_{n-1} (x_2^n) & \cdots & \hat{p}_{n-1} (x_n^n) \end{pmatrix}, \tag{29}$$

where the individual elements are

$$\mathsf{R}_{jk}^n = \hat{p}_j (x_k^n), \quad (j = 0, \ldots, n-1, \quad k = 1, \ldots, n). \tag{30}$$

Further, let W^n be the diagonal matrix

$$\mathsf{W}^n \equiv \begin{pmatrix} w_1^n & 0 & \cdots & 0 \\ 0 & w_2^n & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & w_n^n \end{pmatrix}, \tag{31}$$

where the w_k^n are defined in Eq.(28). Then it is clear that Eq.(27) is equivalent to the matrix relationship

$$(\mathsf{R}^n)^T \mathsf{R}^n \mathsf{W}^n = \mathsf{I}^n, \tag{32}$$

where I^n is the $n \times n$ identity matrix. In component form, both Eqs.(32) and (27) become

$$\sum_{m=0}^{n-1} \hat{p}_m\left(x_i^n\right) \hat{p}_m\left(x_j^n\right) = \delta_{i,j}/w_i^n, \quad (i,j = 1,2,\ldots,n). \tag{33}$$

Moreover, since a matrix left-inverse is also a right-inverse, Eq.(32) may be expressed alternatively as

$$\mathsf{R}^n \mathsf{W}^n \left(\mathsf{R}^n\right)^T = \mathsf{I}^n, \tag{34}$$

which is equivalent to

$$\sum_{m=1}^{n} w_m^n \hat{p}_i\left(x_m^n\right) \hat{p}_j\left(x_m^n\right) = \delta_{i,j}, \quad (i,j = 0,1,\ldots,n-1) \tag{35}$$

in component form.

In full matrix form, the eigenvalue/vector relationships of Eq.(26) may be combined as

$$\mathsf{J}^n \mathsf{R}^n = \mathsf{R}^n \mathsf{X}^n, \tag{36}$$

where X^n is the diagonal matrix

$$\mathsf{X}^n \equiv \begin{pmatrix} x_1^n & 0 & \cdots & 0 \\ 0 & x_2^n & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & x_n^n \end{pmatrix} \tag{37}$$

of eigenvalues. It follows from Eq.(32) that

$$\left(\mathsf{R}^n\right)^T \mathsf{J}^n \mathsf{R}^n = \left(\mathsf{W}^n\right)^{-1} \mathsf{X}^n \tag{38}$$

(i.e., that R^n diagonalizes J^n via an orthogonal transformation).

It follows from the results of this section, that the complete set of zeros of $\hat{p}_n(x)$ can be found using either (i) a polynomial root-finding scheme (such a Newton's method), possibly using successive root sets as brackets (see Eq.(19)) for the next set in the sequence, or (ii) a technique for finding the eigenvalues of a real symmetric matrix [2].

2.3 Functional Approximation

If function $f(x)$ is analytic on $[a,b]$ and has bounded squared norm $\langle f \,|f\rangle$, then it can be expanded in the (convergent) Fourier Series [11]

$$f(x) = \sum_{k=0}^{\infty} \alpha_k \hat{p}_k(x), \tag{39}$$

since the functions $\hat{p}_k(x)$ constitute an orthonormal set. By orthonormality, the coefficients are

$$\alpha_m = \langle f \,|\hat{p}_m\rangle, \quad (m = 0,1,\ldots,\infty) \tag{40}$$

and, as a corollary to Bessel's inequality [11],

$$\lim_{j\to\infty} \alpha_j = 0. \tag{41}$$

The n-term partial sum

$$f^n(x) = \sum_{k=0}^{n-1} \alpha_k \hat{p}_k(x) \tag{42}$$

can be viewed as an approximation to $f(x)$, with error

$$f_\epsilon^n(x) \equiv f(x) - f^n(x) = \sum_{k=n}^{\infty} \alpha_k \hat{p}_k(x). \tag{43}$$

Evaluation of the truncated sum (42), at the roots x_j^n of $\hat{p}_n(x)$, gives

$$f^n(x_j^n) = \sum_{k=0}^{n-1} \alpha_k \hat{p}_k(x_j^n), \quad (j = 1, \ldots, n). \tag{44}$$

It follows that

$$\sum_{j=1}^{n} w_j^n \hat{p}_m(x_j^n) f^n(x_j^n) = \sum_{k=0}^{n-1} \alpha_k \left(\sum_{j=1}^{n} w_j^n \hat{p}_m(x_j^n) \hat{p}_k(x_j^n) \right),$$

which in turn leads to

$$\alpha_m = \sum_{j=1}^{n} w_j^n \hat{p}_m(x_j^n)\ f^n(x_j^n) \tag{45}$$

on account of the identity (35). Thus the expansion coefficients can in principal be computed from the truncated approximation (42) in terms of discrete, finite sums, as opposed to the general integral definition (40).

In matrix form, Eq.(44) can be written as

$$\mathbf{f}^n = (\mathsf{R}^n)^T \boldsymbol{\alpha}^n, \tag{46}$$

where $\mathbf{f}^n$ and $\boldsymbol{\alpha}^n$ are the column n-vectors

$$\mathbf{f}^n \equiv (f_1^n, f_2^n, \cdots f_n^n)^T \tag{47}$$

and

$$\boldsymbol{\alpha}^n \equiv (\alpha_0, \alpha_1, \cdots \alpha_{n-1})^T \tag{48}$$

respectively. The matrix version of Eq.(45) is then

$$\boldsymbol{\alpha}^n = \mathsf{R}^n \mathsf{W}^n \mathbf{f}^n. \tag{49}$$

The inverse relationship between Eqs.(49) and (46) is consistent with the matrix results (34) and (32).

In practice, the preceeding approximation scheme, based on truncation, may not be directly useful. This is because the truncated expansion is not known, unless the coefficients α_m have been computed from the full function using the integral forms of Eq.(40).

An alternative method is to base an approximation scheme on the related coefficients

$$\tilde{\alpha}_m^n \equiv \sum_{j=1}^{n} w_j^n \hat{p}_m(x_j^n) f(x_j^n), \qquad (m = 0, 1, \ldots, n-1), \tag{50}$$

which have a definition analogous to Eq.(45), but based on the original (as opposed to the truncated) function. An approximating function is then defined by

$$\tilde{f}^n(x) \equiv \sum_{k=0}^{n-1} \tilde{\alpha}_k^n \hat{p}_k(x). \tag{51}$$

This modified approximation is actually an interpolant, agreeing with the original function at the points x_j^n. This can be verified by evaluating Eq.(51) at x_r^n, replacing the coefficients by Eq. (50), and reversing the summation order :

$$\begin{aligned}\tilde{f}^n\left(x_r^n\right) &= \sum_{k=0}^{n-1}\left\{\sum_{j=1}^{n} w_j^n\,\hat{p}_k\left(x_j^n\right) f\left(x_j^n\right)\right\}\hat{p}_k\left(x_r^n\right), \\ &= \sum_{j=1}^{n} w_j^n\left\{\sum_{k=0}^{n-1}\hat{p}_k\left(x_j^n\right)\hat{p}_k\left(x_r^n\right)\right\} f\left(x_j^n\right).\end{aligned}$$

It then follows from Eq.(33) that

$$\tilde{f}^n\left(x_r^n\right) = f\left(x_j^n\right), \tag{52}$$

as claimed.

From Eq.(43) it is evident that the error in the approximate coefficients (50), compared to the true coefficients (45), is

$$\epsilon_m^n \equiv \tilde{\alpha}_m^n - \alpha_m = \sum_{j=1}^{n} w_j^n\,\hat{p}_m\left(x_j^n\right) f_e^n\left(x_j^n\right). \tag{53}$$

This may be viewed as representing the error incurred by approximating the inner product integral (40) (which is equal to Eq.(45)) by the discrete sum (50). Now consider this error term in more detail. From Eq.(53), together with Eq.(43), it follows that

$$\tilde{\alpha}_m^n - \alpha_m = \sum_{j=1}^{n} w_j^n\,\mathcal{P}_m^n\left(x_j^n\right) + \sum_{j=1}^{n} w_j^n\,\hat{p}_0\left(x_j^n\right)\left\{\sum_{k=2n-m}^{\infty}\alpha_k\hat{p}_k\left(x_j^n\right)\right\}, \tag{54}$$

where

$$\mathcal{P}_m^n\left(x\right) \equiv \sum_{k=n}^{2n-m-1}\alpha_k\hat{p}_k\left(x\right)\hat{p}_m\left(x\right). \tag{55}$$

This latter term is a polynomial of degree $2n-1$ in x. Consequently, there exists a (non-unique) set of expansion coefficients $\{\Gamma_{j\ell}^{nm}\}$ such that

$$\mathcal{P}_m^n\left(x\right) = \sum_{\substack{j,\,\ell=0 \\ j+\ell<2n}}^{n}\Gamma_{j\ell}^{nm}\hat{p}_j\left(x\right)\hat{p}_\ell\left(x\right), \tag{56}$$

which involves only orthogonal polynomials of degree n or less. From the preceding pair of equations it follows that

$$\sum_{\substack{j,\,\ell=0 \\ j+\ell<2n}}^{n}\Gamma_{j\ell}^{nm}\left\langle\hat{p}_j\,|\hat{p}_\ell\right\rangle = \sum_{k=n}^{2n-m-1}\alpha_k\left\langle\hat{p}_k\,|\hat{p}_m\right\rangle.$$

Invoking the orthonormality of the polynomials shows that the right-hand side is zero (since $m<k$), and consequently that

$$\sum_{\ell=0}^{n-1}\Gamma_{\ell\ell} = 0 \tag{57}$$

since $\left\langle\hat{p}_j\,|\hat{p}_\ell\right\rangle = \delta_{j,\ell}$. Replacement of $\mathcal{P}_m^n\left(x_j^n\right)$ with the expression (56) and subsequent reversal of the order of summation, allows the first term in Eq.(54) to be written as

$$\sum_{j=1}^{n} w_j^n\,\mathcal{P}_m^n\left(x_j^n\right) = \sum_{\substack{s,\,\ell=0 \\ s+\ell<2n}}^{n}\Gamma_{s\ell}^{nm}\left\{\sum_{j=1}^{n} w_j^n\hat{p}_s\left(x_j^n\right)\hat{p}_\ell\left(x_j^n\right)\right\} = \sum_{\ell=0}^{n-1}\Gamma_{\ell\ell}^{nm} = 0.$$

The quantity in braces in the middle term is $\delta_{s,\ell}$ (by Eq.(35)), and the last equality follows from Eq.(57). The outcome is that the error term (54) can be written as

$$\tilde{\alpha}_m^n - \alpha_m = \sum_{k=2n-m}^{\infty} \alpha_k \left\{ \sum_{j=1}^{n} w_j^n \, \hat{p}_0\left(x_j^n\right) \hat{p}_k\left(x_j^n\right) \right\} \tag{58}$$

and depends only on the ("true") expansion coefficients α_k of order $k \geq 2n - m$ (see Eq.(41)).

The lowest-order difference $\tilde{\alpha}_0^n - \alpha_0$ is of particular interest and has the expression

$$\tilde{\alpha}_0^n - \alpha_0 = \sum_{k=2n}^{\infty} \alpha_k \left(\sum_{j=1}^{n} w_j^n \, \hat{p}_0\left(x_j^n\right) \hat{p}_k\left(x_j^n\right) \right). \tag{59}$$

Thus the error represented by the left-hand side depends only on the expansion coefficients α_k, $k \geq 2n$. In particular, $\tilde{\alpha}_0^n = \alpha_0$ whenever f is a polynomial of degree less than $2n$.

2.3.1 Clenshaw's Method

Clenshaw's method is an efficient procedure for the evaluation of functional series where the expansion functions satisfy a three-term recurrence relation. In particular, the method is applicable to the summation of series of orthogonal polynomials of the form given in Eqs.(42) or (51). The general case is considered in Section 5.5 of Numerical Recipes [2].

For the case of orthogonal polynomials, the method may be summarized as follows. The recurrence relations for the orthogonal polynomials under all of the recurrences considered have the general form

$$\pi_{n+1}(x) = \check{a}_n(x)\pi_n(x) - \check{b}_n(x)\pi_{n-1}(x), \quad (n \geq 1), \tag{60}$$

where

$$\pi_{-1}(x) = 0, \quad \text{and} \quad \pi_0(x) = \text{const.} \tag{61}$$

The term $\check{b}_0(x)$ is arbitary. Thus it follows from the preceding pair of equations that

$$\pi_1(x) = \check{a}_0(x)\pi_0(x). \tag{62}$$

Two cases of particular interest are (i) the monic orthonormal polynomials (4), in which case

$$\check{a}_n(x) \equiv (x - a_n) \quad \text{and} \quad \check{b}_n(x) \equiv b_n \tag{63}$$

and (ii) the orthonormal polynomials (11), in which case

$$\check{a}_n(x) \equiv (x - a_n)/\sqrt{b_{n+1}} \quad \text{and} \quad \check{b}_n(x) \equiv \sqrt{b_n/b_{n+1}}\,. \tag{64}$$

Clenshaw's scheme is designed to evaluate the functional sum

$$F(x) \equiv \sum_{k=0}^{N-1} \varphi_k \pi_k(x), \tag{65}$$

where the coefficients $\{\varphi_k\}$ are assumed known. The method is based on the reverse recursion scheme

$$\begin{array}{rcll} y_k(x) & \equiv & 0, & (k \geq N), \\ y_k(x) & \equiv & \varphi_{N-1}, & (k = N-1) \quad \text{and} \\ y_k(x) & = & \check{a}_k(x)y_{k+1}(x) - \check{b}_k(x)y_{k+2}(x) + \varphi_k, & (k = N-2, N-3, \ldots, 0). \end{array} \tag{66}$$

The series value is then given by

$$F(x) = y_0(x)\,\pi_0(x). \tag{67}$$

This can be seen by 'solving' the system (66) for the coefficients (φ_k), and by substituting into Eq.(65) to get the initial result

$$F(x) = \sum_{k=0}^{N-1} \left\{ y_k(x) - \check{a}_k(x) y_{k+1}(x) + \check{b}_k(x) y_{k+2}(x) \right\} \pi_k(x).$$

Subsequent shifting of the dummy index of summation, to bring the auxiliary parameters $\{y_k(x)\}$ to a common index, gives

$$F(x) = y_0(x)\pi_0(x) + \sum_{k=1}^{N-1} \left\{ \pi_k(x) - \check{a}_{k-1}(x)\pi_{k-1}(x) + \check{b}_{k-1}(x)\pi_{k-2}(x) \right\} y_k(x).$$

The initial values for the recurrence schemes for $y_k(x)$ and $\pi_k(x)$ are also used in this derivation. However, every term in the sum vanishes due to Eq.(60), giving the result (67).

Thus, values of the sum (65) can quickly be computed without explicit evaluation of the orthogonal polynomials themselves, provided that their recurrence coefficients are available. In particular, Eq.(51) can provide a useful functional approximation scheme, especially in cases where $f\,(x)$ is difficult to compute in comparison to the method outlined in this section.

2.4 Gaussian Quadrature

From the expansion (39) it follows that

$$I(f) \equiv \int_a^b W\,(x)\; f\,(x)\; dx = (1/\hat{p}_0) \sum_{n=0}^{\infty} \alpha_n \left\langle \hat{p}_0 \,|\hat{p}_n \right\rangle = \alpha_0/\hat{p}_0. \tag{68}$$

Let an operator $\mathcal{S}^n$ be defined by

$$\mathcal{S}^n\,(f) \equiv \sum_{m=1}^{n} w_m^n\, f\,(x_j^n) = \tilde{\alpha}_0^n/\hat{p}_0. \tag{69}$$

From Eq.(59), it follows that

$$\mathcal{S}^n\,(f) - I(f) = \left(\tilde{\alpha}_0^n - \alpha_0\right)/\hat{p}_0 = \sum_{k=2n}^{\infty} \alpha_k \left(\sum_{j=1}^{n} w_j^n \hat{p}_k\,(x_j^n) \right). \tag{70}$$

The operator $\mathcal{S}^n$ represents the n–point Gaussian quadrature approximation to the integral $I(f)$, with sample points x_j^n, $(j = 1, \ldots, n)$ and positive integration weights w_j^n, $(j = 1, \ldots, n)$. Thus, the sample points are given by the n roots of $\hat{p}_n\,(x)$, and the weights are then given by Eq.(28). The error term is given by the right-most term in Eq.(70), and depends only on the expansion coefficients α_k, $k \geq 2n$. In particular, the scheme is exact for all polynomials of degree $\leq 2n-1$.

It is now apparent that use of the coefficients (50) corresponds to evaluation of the integrals (40) (which represent the true coefficients) by n-point Gaussian quadrature. The error is given by Eq.(58) and depends on successively lower-order coefficients as m increases. In particular, (as noted earlier) the error in $\tilde{\alpha}_0^n$ depends only on $\alpha_k, k \geq 2n$. In contrast, the error in $\tilde{\alpha}_n^n$ depends on $\alpha_k, k \geq n$.

It may be verified that the expression (28) for the weights is equivalent to Eq.(4.5.27) of Numerical Recipes [2]. Alternative forms are also available. For example, Eq.(4.5.9) of Numerical Recipes [2] gives

$$w_k^{(n)} = \frac{N_{n-1}^2}{p_{n-1}(x_k^n)\, p_n'(x_k^n)}. \tag{71}$$

3 Computation of the Recursion Coefficients

Knowledge of the coefficients a_n and b_n of Eq.(4) is of paramount importance for working with either the orthogonal polynomials themselves, or with functional series such as (42) or (51) derived from them. For many classical orthogonal polynomials, analytical expressions are available [4] for these coefficients. In the general case, however, numerical methods must be used.

In theory, the recursion coefficients can be computed from the joint recurrence provided by Eqs. (4), (5) and (6). Symbolic algebra software, where precision concerns are immaterial, can be employed in this task. This will be considered briefly in Appendix B. However, as discussed by Gautschi [12], for example, the process is likely to be so unstable as to be virtually useless, when implemented in a standard computer program using finite precision and dynamic range. Consequently, alternative approaches must be used. These are based on modified moments using a second set of orthogonal polynomials. In particular, Wheeler's [1] algorithm, as discussed in Section 4.5 of Numerical Recipes [2], provides a practical alternative approach. This method will be outlined in subsection 3.1 below.

An important component of Wheeler's method is knowledge of the modified moments of the second set of polynomials with respect to the weight function under consideration. It will be pointed out below that in the case of the logarithmic weight function (to be considered later), the modified moments decrease so rapidly with order that underflow problems can occur, especially on computers with a limited floating-point dynamic range. The rapid variation can also lead to loss of precision.

Due to the deficiencies of Wheeler's method, two variations were developed, based on normalization and ratios. These will be presented below in sections 3.2 and 3.3. The first alternative method (section 3.2) works in terms of ratios of the modified moments. For the logarithmic weight function, at least, these ratios are slowly varying and are easier to compute that the moments themselves. The second alternative method (Section 3.3) works in terms of ratios of the expansions coefficients. The original motivation for this idea was the observation that in Wheeler's algorithm (Section 3.1; see also Figure 1 and Table i), only three rows of the table are involved at any step, and the method is therefore homogeneous in the expansion coefficients. Consequently, a scaling can be performed at any stage of the algorithm. Method C (outlined below) puts this scaling on a more formal footing.

It was found that the modified schemes are free from underflow problems, and permit the calculation of the recursion coefficients to very high order, in the case of the logarithmic weight function.

3.1 Method A (Wheeler's Method)

Suppose that it is desired to determine the orthogonal polynomials $\{p_n(x)\}$ orthogonal with respect to a weight function $W(x)$. Further, suppose that $\{\pi_n(x)\}$ is a second (non-orthogonal with respect to $W(x)$) monic set governed by a known recurrence relation

$$\pi_{n+1}(x) = (x - \tilde{a}_n)\,\pi_n(x) - \tilde{b}_n \pi_{n-1}(x), \quad (n = 0, 1, \ldots, \infty), \tag{72}$$

with starting values $\pi_{-1}(x) = 0$ and $\pi_0(x) = 1$. The task at hand is then the evaluation of the coefficients a_n and b_n in the recursion relation (4) for the polynomials p_n.

Suppose that the inner products (or modified moments)

$$\nu_\ell \equiv \langle 1 \,|\pi_\ell\rangle\,, \quad (\ell = -1, 0, \ldots, \infty) \tag{73}$$

are known, and let additional inner products $\sigma_{k,\ell}$ be defined by

$$\sigma_{k,\ell} \equiv \langle p_k \,|\pi_\ell\rangle\,, \quad (k, \ell \geq -1). \tag{74}$$

It may be noted that

$$\sigma_{-1,\ell} = \langle 0 \,|\pi_\ell\rangle = 0, \quad \text{and} \quad \sigma_{0,\ell} = \langle 1 \,|\pi_\ell\rangle = \nu_\ell. \tag{75}$$

The polynomial $\pi_\ell(x)$ is of degree ℓ, and so can be expanded in terms of the polynomials $p_j(x)\,, (0 \leq j \leq \ell)$ as

$$\pi_\ell(x) \equiv \sum_{j=0}^{\ell} \gamma_{\ell,j} p_j(x)\,. \tag{76}$$

Since both sets of polynomials are monic, it follows that

$$\gamma_{\ell,\ell} = 1. \tag{77}$$

Taking the inner product of the expansion (76) with $p_k(x)$, and subsequently invoking orthogonality, gives

$$\sigma_{k,\ell} = \begin{cases} \gamma_{\ell,k} N_k^2, & (0 \leq k \leq \ell) \\ 0 & (k > \ell) \end{cases}\,. \tag{78}$$

Equations (77) and (78) therefore show that

$$\sigma_{k,k} = N_k^2, \quad (k = 0, 1, \ldots, \infty), \tag{79}$$

which will be used below to evaluate the coefficients b_n.

From the basic p-recurrence relation (4), it follows that

$$p_k = x p_{k-1} - a_{k-1} p_{k-1} - b_{k-1} p_{k-2}$$

and consequently that

$$\langle p_k \,|\pi_\ell\rangle = \langle x p_{k-1} \,|\pi_\ell\rangle - a_{k-1} \langle p_{k-1} \,|\pi_\ell\rangle - b_{k-1} \langle p_{k-2} \,|\pi_\ell\rangle\,.$$

By Eq.(74), this is equivalent to

$$\sigma_{k,\ell} = \langle x p_{k-1} \,|\pi_\ell\rangle - a_{k-1}\, \sigma_{k-1,\ell} - b_{k-1}\, \sigma_{k-2,\ell}. \tag{80}$$

Similarly, from the π-recurrence relation (72) it follows that

$$\sigma_{k-1,\ell+1} = \langle x p_{k-1} \,|\pi_\ell\rangle - \tilde{a}_\ell\, \sigma_{k-1,\ell} - \tilde{b}_\ell\, \sigma_{k-1,\ell-1}. \tag{81}$$

Elimination of the term $\langle x p_{k-1} \,|\pi_\ell\rangle$ between Eqs.(80) and (81) leads to the recurrence relation

$$\sigma_{k,\ell} = \sigma_{k-1,\ell+1} - (a_{k-1} - \tilde{a}_\ell)\, \sigma_{k-1,\ell} - b_{k-1}\, \sigma_{k-2,\ell} + \tilde{b}_\ell\, \sigma_{k-1,\ell-1} \tag{82}$$

for the terms $\sigma_{k,\ell}$, with starting values given by Eq.(75).

In principal, this equation allows computation of $\{\sigma_{k,\ell}, \quad (\ell = k, k+1, \ldots)\}$ for $k = 1, 2, \ldots$. It may be noted, however, that the (as yet) unknown coefficients a_n and b_n appear in this relationship. These unknown coefficients may be expressed in terms of the elements $\sigma_{k,\ell}$ as follows. From Eqs.(6) and (79), it follows that

$$b_k = \sigma_{k,k}/\sigma_{k-1,k-1}, \quad (k = 1, 2, \ldots). \tag{83}$$

Furthermore, setting $k = m+1$ and $\ell = m$ in Eq.(82) shows that

$$\sigma_{m+1,m} = \sigma_{m,m+1} - (a_m - \tilde{a}_m)\,\sigma_{m,m} - b_m\,\sigma_{m-1,m} + \tilde{b}_m\,\sigma_{m,m-1}. \tag{84}$$

Now, Eq.(78) shows the terms $\sigma_{m+1,m}$ and $\sigma_{m,m-1}$ to be zero. Thus, upon replacement of b_m according to Eq.(83), and replacement of m by k, it follows that

$$a_k = \tilde{a}_k - \sigma_{k-1,k}/\sigma_{k-1,k-1} + \sigma_{k,k+1}/\sigma_{k,k}, \quad (k = 1, 2, \ldots). \tag{85}$$

The case $k = 0$ is slightly different. Note that $p_0(x) = \pi_0(x) = 1$, $p_1(x) = x - a_0$ and $\pi_1(x) = x - \tilde{a}_0$. It then follows that

$$p_1 = \pi_1 + (\tilde{a}_0 - a_0)\,\pi_0,$$

so that

$$\langle p_0 \,|p_1\rangle = 0 = \sigma_{0,1} + (\tilde{a}_0 - a_0)\,\sigma_{0,0}.$$

In terms of the starting values (75), this becomes

$$a_0 = \tilde{a}_0 + \nu_1/\nu_0. \tag{86}$$

Now it may be noted that in Eq.(82), $\sigma_{k,\ell}$ depends only on a_{k-1} and b_{k-1}. Similarly, in Eqs.(83) and (85), it is seen that computation of a_k and b_k requires only computation of $\sigma_{j,\ell}, (j \le k)$. Consequently, these equations (Wheeler's Algorithm) provide a complete recurrence scheme for generation of the coefficients.

This scheme may be summarized (as in Numerical Recipes [2], with a minor typographical error corrected) as follows. To generate $a_k, (k = 0, 1, \ldots, n-1)$ and $b_k, (k = 1, 2, \ldots, n-1)$, proceed as outlined in Table i. This algorithm is illustrated schematically in Figure 1. It is evident that the scheme constructs a infinite,

	-1	0	+1	+2	• • •	*l*-1	*l*	*l*+1
-1	0	0	0	0	• • •			
0	0	ν_0	ν_1	ν_2	• • •			
+1	0	0	•	•	• • •			
• • •	0	0	0	•				
k-2								
k-1								
k								

Figure 1: Schematic illustration of the recursion for Method A

upper triangular array of values. Moreover, if the scheme is truncated, only an inverted pyramid of values, terminating on $\sigma_{n,n}$ is needed. It is evident that the (truncated) recursion thus requires the starting values $\sigma_{-1,1\ldots 2n-2}$ and $\sigma_{0,0\ldots 2n-1}$, as indicated.

Table i: Method A (Wheeler's Algorithm) for computing the recurrence coefficients for one set of orthogonal polynomials, given the modified moments of a second set.

- Initialize :
  - ‡ $\sigma_{-1,\ell} = 0, \quad (\ell = 1, 2, \ldots, 2n-2),$
  - ‡ $\sigma_{0,\ell} = \nu_\ell, \quad (\ell = 0, 1, \ldots, 2n-1),$
  - ‡ $b_0 = 0$ and,
  - ‡ $a_0 = \tilde{a}_0 + \nu_1/\nu_0.$
- Iterate for $k = 1, 2, \ldots, n-1$:
  - ‡ $\sigma_{k,\ell} = \sigma_{k-1,\ell+1} - (a_{k-1} - \tilde{a}_\ell)\,\sigma_{k-1,\ell} - b_{k-1}\,\sigma_{k-2,\ell} + \tilde{b}_\ell\,\sigma_{k-1,\ell-1}, \quad (\ell = k, k+1, \ldots, 2n-k-1),$
  - ‡ $a_k = \tilde{a}_k - \sigma_{k-1,k}/\sigma_{k-1,k-1} + \sigma_{k,k+1}/\sigma_{k,k}$ and
  - ‡ $b_k = \sigma_{k,k}/\sigma_{k-1,k-1}.$

3.2 Method B

As noted in the introduction to this section, direct implementation of Wheeler's Algorithm can lead to underflow, in the computation of higher-order coefficients, for some classes of orthogonal polynomials. For example, for the weight function $W(x) \equiv \ln(1/x)$ combined with shifted Legendre polynomials on $0 \le x \le 1$, the modified moments [2] are $\nu_j = (-1)^j (j!)^2/j/(j+1)/(2j)!$. It may be shown, using properties of the Γ-function [4], that these moments have the asymptotic behavior

$$|\nu_j| \sim \frac{\sqrt{\pi}}{j^{3/2} 4^j}, \quad (j \to \infty) \tag{87}$$

and so have a combined algebraic and exponential decay with increasing index. This is illustrated in Fig. 2, where the heavy line plots $\log_{10}\{-\log_{10}(|\nu_{10^m}|)\}$ as a function of m, for $0 \le m \le 4$ (corresponding to indices ranging from 1 to 10^4). Table ii shows the underflow limits and indices for this case. The entries

Table ii: Underflow limits and indices, for ν_j with weight function $W(x) \equiv \ln(1/x)$ and shifted Legendre polynomials. The underflow limits are for Language Systems FORTRAN on a Macintosh.

Precision	Label	Underflow	j_u
Real*4	'sp'	$1.2 * 10^{-38}$	59
Real*8	'dp'	$2.3 * 10^{-308}$	505
Real*12	'ep'	$1.7 * 10^{-4932}$	8182

in Columns 1 and 3 are based on Language Systems FORTRAN© [13], on a Macintosh© computer. The first column of the table indicates the precision (single, double or extended), while the third indicates the associated underflow limit. These levels are indicated by the horizontal lines on Fig. 2. The labels on these

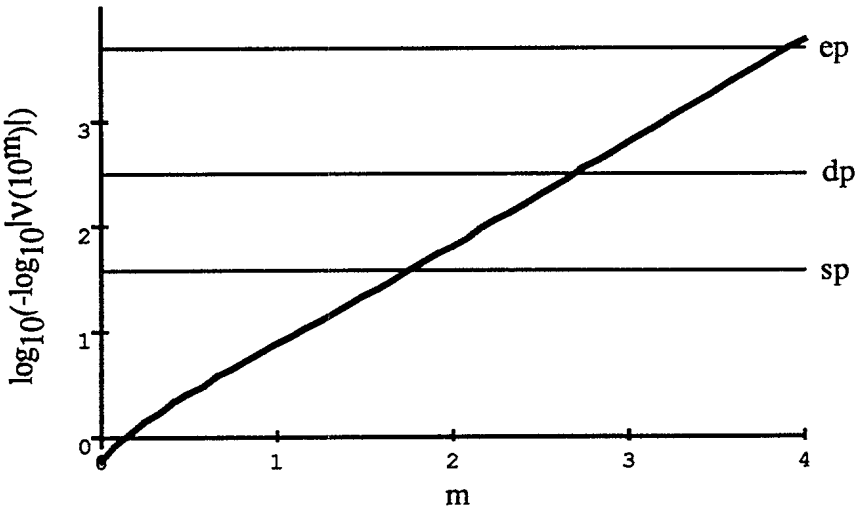


Figure 2: Behavior of modified moments for shifted Legendre polynomials, with a logarithmic weight function on $(0, 1)$.

lines are indicated in the second column of Table ii. The final column of Table ii gives the index j_u for which the associated value of $|\nu_j|$ is below the underflow limit. These limits correspond to the intersections of the sloping and horizontal lines on Figure 2 (by way of $j = 10^m$). Thus in single precision, at most ≈ 59 recurrence coefficients can be computed.

Although the situation improves somewhat at higher precisions and with increased dynamic range, it may be preferable to work in terms of normalized variables. Specifically, let the ratios ρ_ℓ be defined by

$$\rho_\ell \equiv \nu_{\ell+1}/\nu_\ell, \quad (\ell = 0, 1, \ldots) \tag{88}$$

and the scaled inner products $\tau_{k,\ell}$ by

$$\tau_{k,\ell} \equiv \sigma_{k,\ell}/\nu_\ell. \tag{89}$$

With reference to Figure 1 (with appropriately modified starting values), it is evident that this method is equivalent to scaling the columns of the triangular table.

It may be verified that the associated recursion scheme is as shown in Table iii. The pattern of the recursion is again as depicted in Fig. 1. However, the values ν_ℓ in the second row are now all replaced by unity, and the table entries are now $\tau_{k,\ell}$.

3.3 Method C

A further normalization can be attained by working with the ratios

$$\eta_{k,\ell} \equiv \tau_{k,\ell}/\tau_{k,k}, \quad (\ell \geq k). \tag{90}$$

With reference to Figure 1 (with appropriately modified starting values), it is evident that this method is equivalent to normalizing the rows of the triangular table so that the diagonal entries are all unity.

Table iii: Method B for computing the recurrence coefficients for one set of orthogonal polynomials, given the modified moments of a second set. This approach uses ratios of the modified moments to scale the columns of $\sigma_{k,\ell}$.

- Initialize :
  - ‡ $\tau_{-1,\ell} = 0, \quad (\ell = 1, 2, \ldots, 2n-2),$
  - ‡ $\tau_{0,\ell} = 1, \quad (\ell = 0, 2, \ldots, 2n-1),$
  - ‡ $b_0 = 0$ and
  - ‡ $a_0 = \tilde{a}_0 + \rho_0.$
- Iterate for $k = 1, 2, \ldots, n-1$:
  - ‡ $\tau_{k,\ell} = \rho_\ell \tau_{k-1,\ell+1} - (a_{k-1} - \tilde{a}_\ell)\,\tau_{k-1,\ell} - b_{k-1}\,\tau_{k-2,\ell} + \tilde{b}_\ell\,\tau_{k-1,\ell-1}/\rho_{\ell-1},$ $(\ell = k, k+1, \ldots, 2n-k-1),$
  - ‡ $a_k = \tilde{a}_k - \rho_{k-1}\tau_{k-1,k}/\tau_{k-1,k-1} + \rho_k \tau_{k,k+1}/\tau_{k,k}$ and
  - ‡ $b_k = \rho_{k-1}\tau_{k,k}/\tau_{k-1,k-1}.$

Division of the recurrence relation for $\tau_{k,\ell}$ by $\tau_{k,k}$ leads to

$$\eta_{k,\ell} = \left\{ \rho_\ell \eta_{k-1,\ell+1} - (a_{k-1} - \tilde{a}_\ell)\,\eta_{k-1,\ell} - \left(\frac{b_{k-1}\tau_{k-2,k-2}}{\tau_{k-1,k-1}}\right)\eta_{k-2,\ell} + \left(\frac{\tilde{b}_\ell \eta_{k-1,\ell-1}}{\rho_{\ell-1}}\right)\right\}\left(\frac{\tau_{k-1,k-1}}{\tau_{k,k}}\right).$$

The relationship

$$b_k = \rho_{k-1}\tau_{k,k}/\tau_{k-1,k-1}$$

then leads to the recurrence relation

$$\eta_{k,\ell} = \left\{ \rho_\ell \eta_{k-1,\ell+1} - (a_{k-1} - \tilde{a}_\ell)\,\eta_{k-1,\ell} - \rho_{k-2}\eta_{k-2,\ell} + \tilde{b}_\ell \eta_{k-1,\ell-1}/\rho_{\ell-1} \right\}(\rho_{k-1}/b_k) \tag{91}$$

for the ratios $\eta_{k,\ell}$. The computation of the coefficients $\tilde{a}_k$ is achieved simply from

$$a_k = \tilde{a}_k - \rho_{k-1}\eta_{k-1,k} + \rho_k \eta_{k,k+1}, \tag{92}$$

which follows directly from the Method B form for a_k, together with Eq.(90). Computation of $\tilde{b}_k$ is slightly less obvious. From the Eq.(90), it is clear that $\eta_{k,k} \equiv 1$. Consequently, the expression

$$b_k = \tilde{b}_k + \{\rho_k\,\eta_{k-1,k+1} - (a_{k-1} - \tilde{a}_k)\,\eta_{k-1,k} - \rho_{k-2}\,\eta_{k-2,k}\}\,(\rho_{k-1}) \tag{93}$$

follows from Eq.(91).

These results then give rise to the recurrence scheme shown in Table iv. The pattern of the recursion is again as shown in Fig. 1. The values ν_ℓ in the second row are again all replaced by unity, and in addition, the diagonal entries are also unity. The table entries are, of course, now $\eta_{k,\ell}$.

For comparison purposes, Methods A and C were compared in single-precision, using double-precision results from Method C as the reference, Table v shows the results of the single-precision test run on a Macintosh

Table iv: Method C for computing the recurrence coefficients for one set of orthogonal polynomials, given the modified moments of a second set. This approach uses ratios of the expansion coefficients to scale the rows of $\sigma_{k,\ell}$.

- Initialize :
  - ‡ $\eta_{-1,\ell} = 0, \quad (\ell = 1, 2, \ldots, 2n-2),$
  - ‡ $\eta_{0,\ell} = 1, \quad (\ell = 0, 1, \ldots, 2n-1),$
  - ‡ $b_0 = 0$ and
  - ‡ $a_0 = \tilde{a}_0 + \rho_0.$
- Iterate for $k = 1, 2, \ldots, n-1$:
  - ‡ $b_k = \tilde{b}_k + \rho_{k-1}\left\{\rho_k\,\eta_{k-1,k+1} - (a_{k-1} - \tilde{a}_k)\,\eta_{k-1,k} - \rho_{k-2}\,\eta_{k-2,k}\right\},$
  - ‡ $\eta_{k,k} = 1,$
  - ‡ $\eta_{k,\ell} = (\rho_{k-1}/b_k)\left\{\rho_\ell \eta_{k-1,\ell+1} - (a_{k-1} - \tilde{a}_\ell)\,\eta_{k-1,\ell} - \rho_{k-2}\,\eta_{k-2,\ell} + \tilde{b}_\ell\,\eta_{k-1,\ell-1}/\rho_{\ell-1}\right\}$
    $(\ell = k+1, k+2, \ldots, 2n-k-1),$
  - ‡ $a_k = \tilde{a}_k - \rho_{k-1}\,\eta_{k-1,k} + \rho_k\,\eta_{k,k+1}.$

Table v: Performance of Method A in single precision. The first column is the recursion coefficient index. Column 2 contains the absolute difference in values a_k computed by Method A in single precision and by Method C in double precision. Column 3 contains similar differences for b_k. Method A failed completely at $k = 38$ due to underflow. Computations were done on a Macintosh using Language Systems FORTRAN.

k	δa_k	δb_k
25	1.21×10^{-08}	2.14×10^{-11}
26	2.45×10^{-08}	5.05×10^{-10}
27	3.45×10^{-08}	8.52×10^{-09}
28	2.44×10^{-08}	9.75×10^{-09}
29	3.03×10^{-08}	2.91×10^{-09}
30	3.35×10^{-08}	9.59×10^{-09}
31	5.50×10^{-08}	1.07×10^{-08}
32	1.84×10^{-06}	5.14×10^{-08}
33	1.47×10^{-05}	1.54×10^{-06}
34	3.65×10^{-04}	3.31×10^{-05}
35	3.84×10^{-03}	8.49×10^{-04}
36	4.27×10^{-02}	1.99×10^{-04}
37	2.86×10^{-01}	1.49×10^{-02}

using Language Systems FORTRAN. The single-precision implementation of Method A failed completely at $k = 38$ due to underflow. Also, values close to this limit are degraded in accuracy. In contrast, the maximum error in the first 1024 coefficients was found to be 4.17×10^{-06} for a_k and 1.04×10^{-06} for b_k, when comparing results from single- and double-precision implementations of Method C.

4 Summary

This report presents an improved method for computing polynomials $\{\Lambda_n(x)\}$ orthogonal with respect to the positive weight function $\ln(1/x)$ on the interval $x \in [0, 1]$. The new method, which works in terms of ratios, is less prone to underflow and loss of accuracy on computers with limited precision and floating-point range, when compared to Wheeler's [1] algorithm as recently presented in Numerical Recipes [2].

By way of background, the report also contains a summary of general orthogonal polynomial theory, including a derivation of functional approximation and Gaussian quadrature, from a matrix theory viewpoint.

A suite of FORTRAN routines, which implement some of the theoretical concepts, is described in an Appendix. Tabulated results produced by these routines include the first 128 recursion coefficients, together with points and weights of orders 2, 3, 4, 5, 6, 8, 10, 16, 32, 64 and 128 for Gaussian quadrature of $\int_0^1 \ln(1/x)\, F(x)\, dx$.

In addition, a second Appendix presents MAPLE© source code for exact manipulations with these polyomials using symbolic algebra. It also indicates how the code can be modified to accommodate other weight functions and intervals, which give rise to other families of orthogonal polynomials. The symbolic algebra procedures are used to produce the exact (rational) tabulated values of the first ten recursion coefficients, and of the first ten orthogonal polynomials.

References

[1] John C. Wheeler. Modified moments and Gaussian quadratures. *Rocky Mountain Journal of Mathematics*, 4(2):287–296, 1974.

[2] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in FORTRAN: The Art of Scientific Computing*. Cambridge University Press, Cambridge, second edition, 1992.

[3] D. Colton and R. Kress. *Integral Equation Methods in Scattering Theory*. Wiley, New York, 1983.

[4] Milton Abramowitz and Irene Stegun, editors. *Handbook of Mathematical Functions*. Dover Publications, Inc., New York, 1965.

[5] H. Igarishi and T. Honma. On axially and helically symmetric fundamental solutions to modified Helmholtz-type equations. *Applied Mathematical Modelling*, **16**:314–319, June 1992.

[6] M. Tezer-Sezgin and S. Dost. On the fundamental solutions of the axisymmetric Helmholtz-type equations. *Applied Mathematical Modelling*, **17**:47–51, January 1993.

[7] Stephen D. Gedney and Raj Mittra. The use of the FFT for the efficient solution of the problem of electromagnetic scattering by a body of revolution. *IEEE Transactions on Antennas and Propagation*, **38**(3):313–322, March 1990.

[8] T. W. Dawson. Underwater acoustic radiation from a flooded cylinder with impulsively driven flexible endplates. Technical Memorandum 94-66, Defence Research Establishment Pacific, F.M.O., Victoria, B.C., Canada, V0S 1B0, May 1994.

[9] Earlin Lutz. Exact Gaussian quadrature methods for near-singular integrals in the boundary element method. *Eng. Analysis with Boundary Elements*, **9**(3):233–245, 1992.

[10] J.A. Crow. Quadrature of integrands with a logarithmic singularity. *Mathematics of Computation*, **60**(201):297–301, January 1993.

[11] Allan M. Krall. *Linear Methods of Applied Analysis.* Advanced Book Program. Addison-Wesley Publishing Company, Reading, Massachusetts, 1973.

[12] Walter Gautschi. Questions of numerical condition related to polynomials. In *Recent Advances in Numerical Analysis*, pages 45–72. Academic Press, New York, 1978.

[13] Language Systems Corporation, 441 Carlisle Drive, Herndon, VA 22070, USA. *Language Systems FORTRAN 3.0 Reference Manual*, 1991.

[14] Waterloo Maple Software, 160 Columbia Street West, Waterloo, Ontario, Canada, N2L 3L3.

[15] Michaek P. Shatz and George H. Polychronopoulos. An algorithm for the evaluation of radar propagation in the spherical earth diffraction region. *The Lincoln Laboratory Journal*, 1(2):145–152, August 1988.

A Subroutine Documentation and Examples

Several subroutines were written to implement some of the preceding theoretical results for polymials $\{\Lambda_k(x)\}$ orthogonal with respect to $W(x) = \ln(1/x), x \in [0,1]$.

Computation of the recursion coefficients is based on Method C. As indicated in Numerical Recipes [2], appropriate reference polynomials are the monic shifted Legendre polynomials

$$\pi_n(x) \equiv \left\{(n!)^2/(2n)!\right\} P_n(2x-1), \tag{A1}$$

with recurrence coefficients (see Eq.(72)

$$\tilde{a}_n = 1/2 \quad \text{and} \quad \tilde{b}_n = 1/\left\{4\left(4 - 1/j^2\right)\right\} \tag{A2}$$

and modified moments

$$\nu_0 = 1, \qquad \nu_n = (-1)^n n!^2/\left\{n(n+1)(2n)!\right\}, \ (n \geq 1). \tag{A3}$$

The ratios

$$\rho_0 = -\frac{1}{4} \quad \text{and} \quad \rho_\ell = -\frac{j(j+1)}{(4j+2)(j+2)}, \quad (\ell \geq 1) \tag{A4}$$

are also required in Method C.

These routines are described below with respect to calling sequences and purpose. Although source listings are not provided, they may be obtained by contacting the author. The routines have been written with efficiency in mind. The "easy" routines maintain internal tables of previously computed values, so that multiple calls do not lead to recalculation of tabulated results. Furthermore, the implementation of Method C stores only three rows of the inner product table $\eta_{k,\ell}$, making the memory requirements for this array $O(N)$, instead of $O(N^2)$.

A.1 "Easy" Orthogonal Polynomial Routines

The routines in this subsection are concerned with (i) evaluation of recursion coefficients, (ii) Gaussian quadrature points and weights, and (iii) polynomial values and derivatives for the polynomials $\{\Lambda_k(x)\}$, and well as for the monic Legendre polynomials, denoted by $\{P_k(x)\}$, which are orthogonal with respect to $W(x) = 1, x \in [-1, 1]$. These routines are intended to be easy to use, and work space is allocated internally to keep the calling sequences brief. The maximum presently supported polynomial order is $N \leq 256$. For the routines of this section, the following FORTRAN declarations apply to the argument lists :

```
Integer*4   N
Real*8      A(0:N),   B(0:N),   R(0:N)
Real*8      X(1:N),   W(1:N)
```

- **Subroutine :** `Twd_LwOp_Coeffs_RDP` :

  Calling Sequence :

  ```
  Call        Twd_LwOp_Coeffs_RDP( N, A, B, R )
  ```

  Notes :
  This routine returns the recursion coefficients $a_k, b_k, (k = 0, \ldots, \mathtt{N})$, for the monic polynomials $\{\Lambda_k(x)\}$, in arrays `A` and `B` respectively. On exit, array `R` contains contains the values $R_k, (k = 0, \ldots, \mathtt{N})$ (see Eq.(15), with $d = 1$).

- **Subroutine :** `Twd_LwOp_PtsWts_RDP` :

  Calling Sequence :

  ```
  Call        Twd_LwOp_PtsWts_RDP( N, X, W )
  ```

  Notes :
  This routine returns the points $x_k^{\mathtt{N}}$, $(k = 1, \ldots, \mathtt{N})$ and weights $w_k^{\mathtt{N}}$, $(k = 1, \ldots, \mathtt{N})$, in arrays `X` and `W` respectively, for `N`-point Gaussian quadrature with logarithmic weight function $W(x) = \ln(1/x)$, $x \in [0, 1]$.

- **Subroutine :** `Twd_LwOp_PxdPdx_RDP` :

  Calling Sequence :

  ```
  Call        Twd_LwOp_PxdPdx_RDP( N, Norm, x, P, dP_dx )
  ```

  Notes :
  This routine returns values `P` $\leftrightarrow \Lambda_{\mathtt{N}}(\mathtt{x})$ and derivatives `dP_dx` $\leftrightarrow \Lambda'_{\mathtt{N}}(\mathtt{x})$ with normalization controlled by `Norm`. Monic and orthonormal polynomials are selected with `Norm` $= 0$ and `Norm` $= 1$ respectively. Similarly, setting `Norm` $= 2$ and `Norm` $= 3$ selects polynomials normalized to unity at `x` $= 0$ and `x` $= 1$ respectively, as discussed in Eq.(14).

- **Subroutine :** `Twd_UwOp_Coeffs_RDP` :

  Calling Sequence :

```
Call        Twd_UwOp_Coeffs_RDP( N, A, B, R )
```

Notes :
This routine is exactly analogous to `Twd_LwOp_Coeffs_RDP`, except that the returned coefficients and ratios pertain to the monic Legendre polynomials $\{P_k(x)\}$.

- **Subroutine :** `Twd_UwOp_PtsWts_RDP` :

  Calling Sequence :

  ```
  Call        Twd_UwOp_PtsWts_RDP( N, X, W )
  ```

  Notes :
  This routine is analogous to `Twd_LwOp_PtsWts_RDP`, except that the output points and weights are for `N`-point Gaussian quadrature with unit weight function $W(x) = 1,\ x \in [-1, +1]$.

- **Subroutine :** `Twd_UwOp_PxdPdx_RDP` :

  Calling Sequence :

  ```
  Call        Twd_UwOp_PxdPdx_RDP( N, Norm, xx, P, dP_dx )
  ```

  Notes :
  This routine is parallel to `Twd_LwOp_PxdPdx_RDP`, but returns Legendre polynomial values and derivatives under the various normalizations.

A.2 Scalar Gaussian Quadrature Routines

The routines in this section are concerned with Gaussian quadrature of real, double-precision, single-argument scalar functions. The following FORTRAN declarations apply to this subsection :

```
Real*8      Twd_LwGq_IntegA_RDP
Real*8      Twd_LwGq_IntegB_RDP
Real*8      Twd_LwGq_IntegC_RDP
Real*8      Twd_UwGq_IntegA_RDP
Real*8      Int
Integer*4   N
Real*8      a, b, c
Real*8      Integrand
External    Integrand
```

The variable `Integrand` $\leftrightarrow F(x)$ represents the name of a FORTRAN FUNCTION subroutine, whose single double-precision real argument represents the value of x, and whose double-precision real value returns the value of $F(x)$. The built-in upper limit on `N` is currently $\mathtt{N} \leq 256$, but can easily be changed.

There are four routines under this heading :

- **Function :** `Twd_UwGq_IntegA_RDP` :

  Calling Sequence :

```
Int     =   Twd_UwGq_IntegA_RDP( Integrand, a, b, N )
```

Notes :
This routine returns an approximation `Int` to the integral

$$I(a,b) \equiv \int_{x=a}^{b} F(x)\,dx$$

using `N`-point Gaussian quadrature based on the Legendre polynomials. The integral is transformed using the identity

$$I(a,b) = \frac{b-a}{2} \int_{u=-1}^{+1} F\left(\frac{b+a}{2} + u\frac{b-a}{2}\right) du,$$

which is then evaluated by Gaussian quadrature.

- **Function :** `Twd_LwGq_IntegA_RDP` :

Calling Sequence :

```
Int     =   Twd_LwGq_IntegA_RDP( Integrand, a, c, N )
```

Notes :
This routine returns an approximation `Int` to the integral

$$J(a,c) \equiv \int_{x=0}^{a} F(x)\,\ln\left(|cx|\right)\,dx$$

using `N`-point Gaussian quadrature based on the Legendre polynomials and on the polynomial $\Lambda_N(x)$. The integral can be re-written as

$$J(a,c) = \frac{a}{2}\ln|ac| \int_{x=-1}^{+1} F\left(a(1+v)/2\right)\,dv - a\int_{x=0}^{1} \ln(1/x)\,F(au)\,du.$$

The first integral and second integrals are evaluated using `N`-Gaussian quadrature based on the polynomials $P_N(x)$ and $\Lambda_N(x)$ respectively. These results are valid for both $a > 0$ and $a < 0$.

- **Function :** `Twd_LwGq_IntegB_RDP` :

Calling Sequence :

```
Int     =   Twd_LwGq_IntegB_RDP( Integrand, a, b, c, N )
```

Notes :
This routine returns an approximation `Int` to the integral

$$I(a,b,c) \equiv \int_{x=a}^{b} F(x)\,\ln\left(|cx|\right)\,dx$$

using `N`-point Gaussian quadrature based on the Legendre polynomials and on the polynomials $\Lambda_N(x)$. The integral can be rewritten as

$$I(a,b,c) = I(0,b,c) - I(0,a,c).$$

The resulting pair of integrals are evaluated by way of two calls to routine `Twd_LwGq_IntegA_RDP`.

- **Function :** `Twd_LwGq_IntegC_RDP` :

  Calling Sequence :

  ```
  Int     =   Twd_LwGq_IntegC_RDP( Integrand, a, b, c, N )
  ```

  Notes :
  This routine returns an approximation `Int` to the integral

  $$I(a,b,c) \equiv \int_{x=a}^{b} F(x)\,\ln(|cx|)\,dx$$

  using `N`-point Gaussian quadrature based on the Legendre polynomials and on the polynomials $\Lambda_N(x)$. The integral is evaluated *either* as

  $$I(a,b,c) = I(0,b,c) - I(0,a,c), \qquad (ab \leq 0),$$

  (the two component integrals being evaluated by two calls to routine `Twd_LwGq_IntegA_RDP`), *or* by standard `N`-Gaussian quadrature of the function $F(x)\,\ln(|cx|)$, based on $P_N(x)$, when $ab > 0$. The latter may be inaccurate if one of a or b is close to 0, since the logarithmic singularity will then influence the accuracy of the standard Gaussian quadrature.

A.3 Vector Gaussian Quadrature Routines

The routines in this section are concerned with Gaussian quadrature of real, double-precision, single-argument vector-valued functions, and so compute vectors of integrals. In particular, these can be used to integrate complex-valued functions of a single real variable using a vector length of 2.

The following FORTRAN declarations apply to this subsection :

```
Integer*4   N
Real*8      a, b, c
Real*8      Integrand
External    Integrand
Integer*4   VecLen
Real*8      Integral(VecLen)
```

Here, `Integrand` represents the name of a FORTRAN SUBROUTINE, with the calling convention
`Call Integrand(x,F,VecLen)`
where `x` is a double-precision real argument representing the value of x, and where `F` is an output array, of length `VecLen`, whose entries contain the components of the `VecLen`-component vector function $\vec{F}(x)$. The built-in upper limit on `N` is currently $N \leq 256$.

There are four routines under this heading, with sample calling sequences :

```
Call    Twd_LwGq_IntegA_VDP( Integrand, Integral, VecLen, a, c, N )
Call    Twd_LwGq_IntegB_VDP( Integrand, Integral, VecLen, a, b, c, N )
Call    Twd_LwGq_IntegC_VDP( Integrand, Integral, VecLen, a, b, c, N )
Call    Twd_UwGq_IntegA_VDP( Integrand, Integral, VecLen, a, b, N )
```

Each routine is analogous to the associated scalar routine in Section A.2, except that the approximate integrals are returned in the associated `VecLen`-component argument array `Integral`.

A.4 Functional Approximation Routines

The routines in this subsection are concerned with approximation of a real, double-precision, single-argument function $F(x)$ in a series of orthonormal polynomials $\hat{\Lambda}_k(x), (k = 0, \ldots, N-1)$, as considered in Section 2.3. There are two routines, one to evaluate the expansion coefficients, and the second to evaluate the resultant series approximation.

Specifically, routine `Twd_LwOp_FcnApr_Coeffs_RDP` evaluates the first `N` coefficients $\tilde{\alpha}_k^N$ of Eq.(50). The calling sequence for evaluation of the coefficients is

```
Call        Twd_LwOp_FcnApr_Coeffs_RDP( N, F, C, X, W, A, B, R, P )
```

The argument list conventions here are

```
Integer*4   N
Real*8      F
Real*8      C(0:N-1)
Real*8      X(1:N), W(1:N)
Real*8      A(0:N), B(0:N), R(0:N)
Real*8      P(1:N,0:2)
```

Arrays `X,W,A,B,R` and `P` are work arrays. The first `N` coefficients $\tilde{\alpha}_k^N$, $(k = 0, \ldots, N-1)$ are returned in array `C`. `F` is the name of a FORTRAN FUNCTION that returns the real, double-precision value of $F(x)$, given a single real, double-precision argument x (i.e., with the conventions) :

```
Function    F(X)
Real*8      F, X
```

Similarly, the companion FUNCTION routine `Twd_LwOp_FcnApr_Value_RDP` returns values of the associated expansion $\tilde{f}^n(x)$ as defined in Eq.(51). The calling sequence is

```
Apr     =   Twd_LwOp_FcnApr_Value_RDP( x, N, C, A, B, R )
```

where `Apr` will be assigned the value of $\tilde{f}^n(x)$. For this procedure, the argument list conventions are :

```
Real*8      Twd_LwOp_FcnApr_Value_RDP, Apr
Real*8      x
Integer*4   N
Real*8      C(0:N-1)
Real*8      A(0:N), B(0:N), R(0:N)
```

Arrays A,B and R are work arrays, while x is the desired evaluation point, assumed to satisfy $0 \leq x \leq 1$. It is further assumed that the coefficients C were computed by a previous call to routine Twd_LwOp_FcnApr_Coeffs_RDP.

A.5 Utility Routines

This section documents some lower-level routines that may also be of interest to the user. Some of these form the working core for the "easy" routines documented above.

- **Subroutine : Twd_LwOp_Coeffs_RDPa :**

  Calling Sequence :

```
Call        Twd_LwOp_Coeffs_RDPa( N, A, B, Alp, Bet, Rho, Eta )
```

  Argument List Conventions :

```
Integer*4   N
Real*8      A(0:N-1), B(0:N-1)
Real*8      Alp( 0:2*N-2 ), Bet(0:2*N-2), Rho(0:2*N-2), Eta(0:2*N-1,0:2)
```

  Notes :
  This routine returns the recursion coefficients a_k and b_k, $(k = 0, 1, \ldots, \mathrm{N}-1)$ in arrays A and B respectively, for the monic polynomials $\{\Lambda_k(x)\}$. The routine implements Method C, summarized in Table iv of Section 3.3. Arrays Alp,Bet,Rho and Eta are work arrays used in implementing the method. On exit, the first three contain values of $\tilde{a}_k$, $\tilde{b}_k$ and ρ_k respectively. Array Eta is used to compute $\eta_{k,\ell}$. This array has three columns, and only stores the latest three rows of the table, as indicated in Figure 1, to conserve memory. There is no internal upper limit on N, so it is up to the calling program to allocate sufficient memory.

- **Subroutine : Twd_UwOp_Coeffs_RDPa :**

  Calling Sequence :

```
Call        Twd_UwOp_Coeffs_RDPa( N, A, B )
```

  Argument List Conventions :

```
Integer*4   N
Real*8      A(0:N-1), B(0:N-1)
```

  Notes :
  This routines returns the recursion coefficients a_k and b_k, $(k = 0, 1, \ldots, \mathrm{N}-1)$ in arrays A and B respectively, for the monic Legendre polynomials $\{P_k(x)\}$.

- **Subroutine : Twd_GqGen_PtsWts_RDPa :**

  Calling Sequence :

```
Call        Twd_GqGen_PtsWts_RDPa( N, A, B, NormSq0, X, W, D, E, Z )
```

Argument List Conventions :

```
Integer*4   N
Real*8      A(0:N-1), B(0:N-1)
Real*8      X(N),     W(N)
Real*8      D(0:N-1), E(0:N-1), Z(0:N-1)
Real*8      NormSq0
```

Notes :
This routine returns the points and weights for N-point Gaussian quadrature based on monic orthogonal polynomials defined by a user-supplied set of recursion coefficients a_k and b_k, $(k = 0, 1, \ldots, \mathtt{N}-1)$ in arrays A and B respectively. The N points and weights are returned in arrays X and W respectively. Arrays D,E and Z are work arrays. Parameter

$$\mathtt{NormSq0} \equiv \langle p_0 \, | p_0 \rangle \tag{A5}$$

(see also Eq.(8)) is to be supplied by the calling program. There is no internal upper limit on N, so it is up to the calling program to allocate sufficient memory.

- **Subroutine :** `Twd_OpGen_PxdPdx_RDP` :

Calling Sequence :

```
Call        Twd_OpGen_PxdPdx_RDP( N, Norm, x, P, dP_dx, xLo, xHi, NormSq0, A, B )
```

Argument List Conventions :

```
Integer*4   N
Integer*4   Norm
Real*8      x, P, dP_dx, xLo, xHi, NormSq0
Real*8      A(0:N), B(0:N)
```

Notes :
This routine returns the values and derivatives for orthogonal polynomials defined by a user-supplied set of recursion coefficients a_k and b_k, $(k = 0, 1, \ldots, \mathtt{N})$, in arrays A and B respectively, on the interval $\mathtt{xLo} \leq x \leq \mathtt{xHi}$. These endpoint values are to be supplied by the calling program, as is the value of NormSq0, defined in Eq.(A5). User-supplied integer values of Norm, from 0 to 3 inclusive, set the normalization to monic, orthonormal, unity at $x = 0$ or unity at $x = 1$ respectively. On output, P contains the appropriately normalized polynomial value, and dP_dx its derivative.

- **Function :** `TWD_OpGen_Series_Sum_RDP` :

Calling Sequence :

```
Sum     =   TWD_OpGen_Series_Sum_RDP( N, Norm, x, xLo, xHi, NormSq0, C, A, B )
```

Argument List Conventions :

```
Real*8      TWD_OpGen_Series_Sum_RDP, Sum
Integer*4   N, Norm
Real*8      x, xLo, xHi, NormSq0
Real*8      C(0:N-1)
Real*8      A(0:N), B(0:N)
```

Notes :
This routine uses Clenshaw's algorithm (see Section 2.3.1) to compute the value of an expansion

$$S(x) \equiv \sum_{k=0}^{N-1} C_k \begin{cases} p_k(x) & (\texttt{Norm} = 0) \\ \hat{p}_k(x) & (\texttt{Norm} = 1), \end{cases}$$

comprising a series of orthogonal polynomials defined by a user-supplied set of recursion coefficients a_k and b_k, $(k = 0, 1, \ldots, \texttt{N})$ in arrays `A` and `B` respectively, on the interval $\texttt{xLo} \leq x \leq \texttt{xHi}$. These endpoint values are to be supplied by the calling program, as is the value of `NormSq0`, defined in Eq.(A5), and the expansion coefficients C_k in array `C`.

- **Subroutine :** `Twd_GqGen_PW_All_RDPa` :

Calling Sequence :

```
Call        Twd_GqGen_PW_All_RDPa( N, A, B, xLo, xHi, NrmSq0, Xpts, Wgts )
```

Argument List Conventions :

```
Integer*4   N
Real*8      A(0:N), B(0:N)
Real*8      xLo, xHi, NrmSq0
Real*8      Xpts(1:N*(N+1)/2)
Real*8      Wgts(1:N*(N+1)/2)
```

Notes :
Given a user-supplied set of recursion coefficients a_k and b_k, $(k = 0, 1, \ldots, \texttt{N})$ in arrays `A` and `B` respectively, which are assumed to define a monic set of orthogonal polynomials on the interval $\texttt{xLo} \leq x \leq \texttt{xHi}$, and a value of `NormSq0` (defined in Eq.(A5)), this routine returns all points and weights for j-point Gaussian quadrature, $j = 1, 2, \ldots, \texttt{N}$. Succesively higher-order sets of points and weights are stored sequentially in their respective linear arrays, with the correspondence

$$\begin{array}{rcl} x_k^j & \leftrightarrow & \texttt{Xpts}(k + j(j-1)/2) \\ w_k^j & \leftrightarrow & \texttt{Wgts}(k + j(j-1)/2) \end{array} \quad (k = 1, \ldots, j, \quad j = 1, \ldots, N).$$

A.6 Examples

A.6.1 Recursion coefficients

Table vi contains the first 128 recursion coefficients a_k and b_k for the monic orthogonal polynomials $\{\Lambda_n(x)\}$. These were computed using routine `Twd_LwOp_Coeffs_RDP`. Note that the values of b_k have been multiplied by 10 prior to tabulation, as indicated in the top line of the table.

A.6.2 Gaussian quadrature points and weights

Tables vii through ix contain points and weights for n-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \leq x \leq 1$, for orders $n = 2, 3, 4$. See Section 2.4 for the notation. These tables are seen to agree with previously-published (see Table 25.7 of Abramowitz and Stegun [4], which date back to 1952) values, to

Table vi: The first 128 recursion coefficients for monic polynomials orthogonal with respect to $\ln(1/x)$, $0 \leq x \leq 1$, computed using routine `Twd_LwOp_Coeffs_RDP`.

k	a_k	$10b_k$	k	a_k	$10b_k$
0	0.25000 00000 00000	0.00000 00000 00000	64	0.49997 11997 14645	0.62495 97291 88132
1	0.46428 57142 85714	0.48611 11111 11111	65	0.49997 20631 76787	0.62496 09669 55887
2	0.48548 24464 56171	0.58684 80725 62358	66	0.49997 28883 41804	0.62496 21486 34268
3	0.49210 30818 71361	0.60728 58391 89179	67	0.49997 36774 42873	0.62496 32775 57706
4	0.49502 84987 58354	0.61482 02019 69369	68	0.49997 44325 52667	0.62496 43568 16660
5	0.49657 95116 43558	0.61840 80953 18848	69	0.49997 51555 97004	0.62496 53892 78812
6	0.49750 13013 04958	0.62039 06295 44560	70	0.49997 58483 67165	0.62496 63776 07866
7	0.49809 40182 04252	0.62159 91915 82894	71	0.49997 65125 31018	0.62496 73242 80885
8	0.49849 78019 78253	0.62238 93767 16667	72	0.49997 71496 43087	0.62496 82316 03672
9	0.49878 53226 55693	0.62293 38867 99075	73	0.49997 77611 53675	0.62496 91017 24788
10	0.49899 73531 67154	0.62332 47750 66563	74	0.49997 83484 17136	0.62496 99366 48248
11	0.49915 82216 78013	0.62361 47348 38110	75	0.49997 89126 99401	0.62497 07382 45019
12	0.49928 31802 15736	0.62383 56835 95357	76	0.49997 94551 84814	0.62497 15082 63482
13	0.49938 21876 70866	0.62400 78643 42726	77	0.49997 99769 82365	0.62497 22483 38936
14	0.49946 19720 96619	0.62414 46153 53519	78	0.49998 04791 31373	0.62497 29600 02258
15	0.49952 72124 26730	0.62425 50130 29884	79	0.49998 09626 06674	0.62497 36446 87802
16	0.49958 12447 30037	0.62434 54062 35662	80	0.49998 14283 23363	0.62497 43037 40602
17	0.49962 64998 05822	0.62442 03431 42552	81	0.49998 18771 41140	0.62497 49384 22965
18	0.49966 47829 28014	0.62448 31505 98195	82	0.49998 23098 68281	0.62497 55499 20498
19	0.49969 74576 41224	0.62453 63072 43563	83	0.49998 27272 65292	0.62497 61393 47628
20	0.49972 55694 71812	0.62458 16901 83745	84	0.49998 31300 48255	0.62497 67077 52670
21	0.49974 99310 06759	0.62462 07418 96409	85	0.49998 35188 91907	0.62497 72561 22473
22	0.49977 11815 42551	0.62465 45855 36031	86	0.49998 38944 32474	0.62497 77853 86693
23	0.49978 98296 37453	0.62468 41060 86524	87	0.49998 42572 70280	0.62497 82964 21722
24	0.49980 62839 48614	0.62471 00084 46911	88	0.49998 46079 72150	0.62497 87900 54310
25	0.49982 08759 01574	0.62473 28596 46797	89	0.49998 49470 73630	0.62497 92670 64896
26	0.49983 38765 74983	0.62475 31199 58313	90	0.49998 52750 81035	0.62497 97281 90683
27	0.49984 55094 28689	0.62477 11661 19530	91	0.49998 55924 73347	0.62498 01741 28487
28	0.49985 59600 05191	0.62478 73088 74134	92	0.49998 58997 03961	0.62498 06055 37355
29	0.49986 53834 00178	0.62480 18063 62586	93	0.49998 61972 02315	0.62498 10230 40994
30	0.49987 39100 69945	0.62481 48744 48922	94	0.49998 64853 75390	0.62498 14272 30025
31	0.49988 16503 85902	0.62482 66947 58821	95	0.49998 67646 09104	0.62498 18186 64060
32	0.49988 86982 35927	0.62483 74209 90571	96	0.49998 70352 69611	0.62498 21978 73640
33	0.49989 51338 93979	0.62484 71839 10190	97	0.49998 72977 04499	0.62498 25653 62021
34	0.49990 10263 23137	0.62485 60953 35040	98	0.49998 75522 43912	0.62498 29216 06847
35	0.49990 64350 36423	0.62486 42513 33313	99	0.49998 77992 01590	0.62498 32670 61692
36	0.49991 14116 09901	0.62487 17348 10881	100	0.49998 80388 75836	0.62498 36021 57502
37	0.49991 60009 20382	0.62487 86176 15941	101	0.49998 82715 50420	0.62498 39273 03938
38	0.49992 02421 63593	0.62488 49622 61467	102	0.49998 84974 95420	0.62498 42428 90620
39	0.49992 41696 96215	0.62489 08233 42740	103	0.49998 87169 68009	0.62498 45492 88296
40	0.49992 78137 35783	0.62489 62487 10109	104	0.49998 89302 13186	0.62498 48468 49926
41	0.49993 12009 45233	0.62490 12804 44085	105	0.49998 91374 64463	0.62498 51359 11694
42	0.49993 43549 23324	0.62490 59556 69942	106	0.49998 93389 44504	0.62498 54167 93958
43	0.49993 72966 17866	0.62491 03072 41303	107	0.49998 95348 65726	0.62498 56898 02134
44	0.49994 00446 75355	0.62491 43643 16242	108	0.49998 97254 30858	0.62498 59552 27526
45	0.49994 26157 37942	0.62491 81528 44783	109	0.49998 99108 33467	0.62498 62133 48096
46	0.49994 50246 96626	0.62492 16959 83027	110	0.49999 00912 58454	0.62498 64644 29198
47	0.49994 72849 07892	0.62492 50144 46261	111	0.49999 02668 82508	0.62498 67087 24251
48	0.49994 94083 79702	0.62492 81268 11097	112	0.49999 04378 74548	0.62498 69464 75378
49	0.49995 14059 31703	0.62493 10497 74891	113	0.49999 06043 96125	0.62498 71779 14010
50	0.49995 32873 33649	0.62493 37983 79194	114	0.49999 07666 01805	0.62498 74032 61440
51	0.49995 50614 25374	0.62493 63862 02820	115	0.49999 09246 39530	0.62498 76227 29355
52	0.49995 67362 21055	0.62493 88255 29172	116	0.49999 10786 50953	0.62498 78365 20334
53	0.49995 83190 00092	0.62494 11274 91662	117	0.49999 12287 71761	0.62498 80448 28308
54	0.49995 98163 86518	0.62494 33022 00447	118	0.49999 13751 31966	0.62498 82478 39005
55	0.49996 12344 18573	0.62494 53588 53172	119	0.49999 15178 56198	0.62498 84457 30357
56	0.49996 25786 09814	0.62494 73058 31983	120	0.49999 16570 63960	0.62498 86386 72896
57	0.49996 38540 02905	0.62494 91507 88730	121	0.49999 17928 69888	0.62498 88268 30113
58	0.49996 50652 17082	0.62495 09007 19957	122	0.49999 19253 83979	0.62498 90103 58810
59	0.49996 62164 90131	0.62495 25620 33071	123	0.49999 20547 11822	0.62498 91894 09420
60	0.49996 73117 15584	0.62495 41406 04849	124	0.49999 21809 54801	0.62498 93641 26321
61	0.49996 83544 75753	0.62495 56418 33275	125	0.49999 23042 10301	0.62498 95346 48118
62	0.49996 93480 71120	0.62495 70706 83574	126	0.49999 24245 71891	0.62498 97011 07928
63	0.49997 02955 46539	0.62495 84317 29159	127	0.49999 25421 29505	0.62498 98636 33627

Table vii: Points and weights for 2-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \leq x \leq 1$. See Section 2.4 for the notation.

k	x_k^2	$10w_k^2$
1	0.11200 88061 66976	7.18539 31903 03845
2	0.60227 69081 18738	2.81460 68096 96157

Table viii: Points and weights for 3-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \leq x \leq 1$. See Section 2.4 for the notation.

k	x_k^3	$10w_k^3$
1	0.06389 07930 87325	5.13404 55223 23633
2	0.36899 70637 15619	3.91980 04120 14874
3	0.76688 03039 38941	0.94615 40656 61490

Table ix: Points and weights for 4-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \leq x \leq 1$. See Section 2.4 for the notation.

k	x_k^4	$10w_k^4$
1	0.04144 84801 99383	3.83464 06814 51352
2	0.24527 49143 20602	3.86875 31777 47628
3	0.55616 54535 60276	1.90435 12695 01423
4	0.84898 23945 32985	0.39225 48712 99598

the corresponding nuumber of digits. Note that the weights have been multiplied by 10 prior to tabulation, as indicated in the first line of the tables.

Similarly, Tables x through xiii contain points and weights for rules of orders $5, 6, 8$ and 10 respectively, the weights again having been scaled up by a factor of 10.

Table x: Points and weights for 5-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \leq x \leq 1$. See Section 2.4 for the notation.

k	x_k^5	$10w_k^5$
1	0.02913 44721 51972	2.97893 47178 28943
2	0.17397 72133 20898	3.49776 22651 32241
3	0.41170 25202 84902	2.34488 29004 40524
4	0.67731 41745 82820	0.98930 45951 66332
5	0.89477 13610 31008	0.18911 55214 31958

Table xi: Points and weights for 6-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \leq x \leq 1$. See Section 2.4 for the notation.

k	x_k^6	$10w_k^6$
1	0.02163 40058 44117	2.38763 66257 85474
2	0.12958 33911 54951	3.08286 57327 39469
3	0.31402 04499 14765	2.45317 42656 32103
4	0.53865 72173 51802	1.42008 75656 64767
5	0.75691 53373 77403	0.55454 62232 48862
6	0.92266 88513 72120	0.10168 95869 29323

Table xii: Points and weights for 8-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \leq x \leq 1$. See Section 2.4 for the notation.

k	x_k^8	$10w_k^8$
1	0.01332 02441 60892	1.64416 60472 80030
2	0.07975 04290 13895	2.37525 61002 33063
3	0.19787 10293 26188	2.26841 98443 19190
4	0.35415 39943 51909	1.75754 07900 60700
5	0.52945 85752 34917	1.12924 03024 67590
6	0.70181 45299 39100	0.57872 21071 77823
7	0.84937 93204 41106	0.20979 07374 21330
8	0.95332 64500 56360	0.03686 40710 40276

Finally, Tables xiv through xvii contain points and weights for rules of orders $16, 32, 64$ and 128 respectively. As indicated in the first line of the appropriate Table, the weights have been scaled up by a factor of 10 for order 16, and by 100 for the remaining orders.

Table xiii: Points and weights for 10-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \le x \le 1$. See Section 2.4 for the notation.

k	x_k^{10}	$10w_k^{10}$
1	0.00904 26309 62200	1.20955 13195 45709
2	0.05397 12662 22501	1.86363 54256 40712
3	0.13531 18246 39251	1.95660 87327 77602
4	0.24705 24162 87160	1.73577 14218 29073
5	0.38021 25396 09332	1.35695 67299 54840
6	0.52379 23179 71843	0.93646 75853 81102
7	0.66577 52055 16425	0.55787 72735 14160
8	0.79419 04160 11966	0.27159 81089 92333
9	0.89816 10912 19003	0.09515 18260 28485
10	0.96884 79887 18633	0.01638 15763 35983

Table xiv: Points and weights for 16-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \le x \le 1$. See Section 2.4 for the notation.

k	x_k^{16}	$10w_k^{16}$	k	x_k^{16}	$10w_k^{16}$
1	0.00389 78344 87116	0.60791 71004 35919	9	0.51508 24733 81463	0.61850 49458 19652
2	0.02302 89456 16873	1.02915 67751 75816	10	0.60755 61204 47728	0.45435 24650 77267
3	0.05828 03983 06240	1.22355 66204 60092	11	0.69637 56532 28214	0.31098 97475 15819
4	0.10867 83650 91054	1.27569 24693 70161	12	0.77843 25658 73265	0.19459 76592 73609
5	0.17260 94549 09844	1.23013 57460 00708	13	0.85085 02697 15391	0.10776 25496 32055
6	0.24793 70544 70578	1.11847 24485 54854	14	0.91108 68572 22272	0.04972 54289 00877
7	0.33209 45491 29917	0.96596 38515 21247	15	0.95702 55717 03542	0.01678 20111 00512
8	0.42218 39105 81949	0.79356 66435 14730	16	0.98704 78002 47984	0.00282 35376 46684

Table xv: Points and weights for 32-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \le x \le 1$. See Section 2.4 for the notation.

k	x_k^{32}	$100w_k^{32}$	k	x_k^{32}	$100w_k^{32}$
1	0.00107 56110 83792	2.03890 23531 54664	17	0.50762 95526 50971	3.24197 03228 94050
2	0.00625 08100 87023	3.74416 76436 66338	18	0.55534 43282 86804	2.79615 25965 01122
3	0.01582 84408 49814	4.87955 11203 57846	19	0.60256 43674 27937	2.37210 61881 06421
4	0.02976 35030 15651	5.65239 84075 09978	20	0.64885 63380 84223	1.97609 25961 18262
5	0.04795 20750 21797	6.15676 01719 90689	21	0.69379 51760 69503	1.61305 82375 39249
6	0.07024 38350 80692	6.44894 33565 74191	22	0.73696 80350 96385	1.28664 89279 67584
7	0.09644 67338 72306	6.56834 30288 33478	23	0.77797 81219 04546	0.99923 63018 15180
8	0.12633 01772 38289	6.54540 37084 36427	24	0.81644 83812 19166	0.75195 73343 58694
9	0.15962 78853 77058	6.40525 41613 95632	25	0.85202 49965 97508	0.54476 75287 39952
10	0.19604 07756 73163	6.16955 09885 58196	26	0.88438 06747 87704	0.37650 78475 83961
11	0.23523 99914 32046	5.85747 36333 93480	27	0.91321 76830 00779	0.24498 50691 86080
12	0.27687 01174 81971	5.48627 84107 51176	28	0.93827 06102 20815	0.14706 49124 22519
13	0.32055 25883 33085	5.07160 82789 42976	29	0.95930 88243 61079	0.07877 69901 56616
14	0.36588 92770 17391	4.62766 16905 61017	30	0.97613 85919 08961	0.03543 04104 73817
15	0.41246 62429 65958	4.16727 86211 18943	31	0.98860 47831 99629	0.01173 86473 47101
16	0.45985 76113 66114	3.70197 83226 27815	32	0.99659 16301 80025	0.00195 21909 16545

Table xvi: Points and weights for 64-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \le x \le 1$. See Section 2.4 for the notation.

k	x_k^{64}	$100w_k^{64}$	k	x_k^{64}	$100w_k^{64}$
1	0.00028 63969 01797	0.64109 93950 58534	33	0.50383 66814 38029	1.66038 99787 77912
2	0.00164 10367 20737	1.23529 37507 72743	34	0.52804 49651 43190	1.54442 93222 21178
3	0.00414 13159 35145	1.68676 94334 73099	35	0.55218 89367 55956	1.43079 18015 09238
4	0.00778 87699 28314	2.04876 03988 43113	36	0.57621 18421 96677	1.31995 79356 90781
5	0.01257 86985 82090	2.34477 71611 69991	37	0.60005 72019 40674	1.21236 42015 97806
6	0.01850 24498 28946	2.58818 94667 53056	38	0.62366 89447 17307	1.10840 30958 43763
7	0.02554 80481 50165	2.78771 33648 36502	39	0.64699 15402 22086	1.00842 32674 68626
8	0.03370 04779 72389	2.94955 08075 41042	40	0.66997 01305 31072	0.91272 97329 40723
9	0.04294 18498 28581	3.07839 80706 70188	41	0.69255 06599 09467	0.82158 41829 98415
10	0.05325 15179 98297	3.17795 28046 81535	42	0.71468 00027 09675	0.73520 53888 91627
11	0.06460 61763 91145	3.25137 55694 10373	43	0.73630 60890 58329	0.65376 97139 03945
12	0.07697 99448 21567	3.30118 22947 78981	44	0.75737 80280 36728	0.57741 17345 62452
13	0.09034 44517 78752	3.32966 90232 40796	45	0.77784 62280 64820	0.50622 49746 31866
14	0.10466 89169 78808	3.33884 58380 45768	46	0.79766 25142 05268	0.44026 27538 20274
15	0.11992 02355 50195	3.33052 50175 84220	47	0.81678 02421 11235	0.37953 91520 69723
16	0.13606 30649 16931	3.30636 22788 59602	48	0.83515 44083 49308	0.32403 00893 53052
17	0.15305 99149 87106	3.26788 65664 67337	49	0.85274 17568 37350	0.27367 45200 28632
18	0.17087 12419 74785	3.21652 18726 55692	50	0.86950 08811 46136	0.22837 57400 13968
19	0.18945 55460 04614	3.15360 33868 19404	51	0.88539 23224 23150	0.18800 28043 11634
20	0.20876 94725 29381	3.08038 95322 87360	52	0.90037 86627 07046	0.15239 20517 53933
21	0.22876 79175 08325	2.99807 09722 70645	53	0.91442 46134 11690	0.12134 87332 19360
22	0.24940 41362 35040	2.90777 73526 34261	54	0.92749 70987 69398	0.09464 87390 09617
23	0.27062 98556 60144	2.81058 23368 26164	55	0.93956 53340 33434	0.07204 04205 57492
24	0.29239 53900 20765	2.70750 73417 66805	56	0.95060 08982 49191	0.05324 65011 60545
25	0.31464 97595 62983	2.59952 42804 36339	57	0.96057 78014 09509	0.03796 60700 01343
26	0.33734 08121 22809	2.48755 75431 53796	58	0.96947 25458 06336	0.02587 66533 19744
27	0.36041 53473 14590	2.37248 53959 25771	59	0.97726 41813 60236	0.01663 63562 46252
28	0.38381 92430 62054	2.25514 09347 00769	60	0.98393 43545 89609	0.00988 60684 87447
29	0.40749 75841 95938	2.13631 27047 81504	61	0.98948 73504 26102	0.00525 17267 80792
30	0.43139 47928 32822	2.01674 50722 50659	62	0.99385 01240 67191	0.00234 66268 38610
31	0.45545 47602 42077	1.89713 84170 77525	63	0.99707 23081 67015	0.00077 37776 27402
32	0.47962 09799 01604	1.77814 92042 46003	64	0.99912 60630 05244	0.00012 82943 42311

Table xvii: Points and weights for 128-point Gaussian quadrature with weight function $\ln(1/x)$ for $0 \le x \le 1$. See Section 2.4 for the notation.

k	x_k^{128}	$100w_k^{128}$	k	x_k^{128}	$100w_k^{128}$
1	0.00007 46263 70041	0.19297 33610 27425	65	0.50192 37684 65858	0.84031 50906 51361
2	0.00042 28620 78738	0.38382 04411 52068	66	0.51411 28101 35989	0.81075 99926 84412
3	0.00106 33609 65987	0.53884 25817 74340	67	0.52629 36555 23050	0.78146 53671 45377
4	0.00199 70952 85879	0.67160 02350 41916	68	0.53845 90577 18779	0.75246 42834 84591
5	0.00322 41717 13099	0.78808 76990 75065	69	0.55060 17786 47016	0.72378 83877 21264
6	0.00474 42908 27687	0.89169 43555 18543	70	0.56271 45933 86107	0.69546 79026 21267
7	0.00655 68625 17207	0.98460 51652 65136	71	0.57479 02944 83234	0.66753 16282 42947
8	0.00866 10533 28387	1.06834 85990 86488	72	0.58682 16962 58109	0.64000 69429 22012
9	0.01105 58106 62086	1.14405 58925 24418	73	0.59880 16390 93481	0.61291 98047 56578
10	0.01373 98769 30034	1.21259 98346 99321	74	0.61072 29937 09915	0.58629 47536 51950
11	0.01671 17987 02124	1.27467 58338 00755	75	0.62257 86654 22289	0.56015 49139 73776
12	0.01996 99331 12521	1.33085 24496 71093	76	0.63436 15983 75504	0.53452 19978 58711
13	0.02351 24526 67771	1.38160 45257 31557	77	0.64606 47797 56875	0.50941 63092 22538
14	0.02733 73490 82108	1.42733 58076 80786	78	0.65768 12439 82699	0.48485 67485 04803
15	0.03144 24365 04050	1.46839 49038 71897	79	0.66920 40768 56527	0.46086 08181 81946
16	0.03582 53543 57259	1.50508 68617 63847	80	0.68062 64196 96644	0.43744 46290 80100
17	0.04048 35699 37879	1.53768 17622 84168	81	0.69194 14734 30329	0.41462 29075 12687
18	0.04541 43808 62200	1.56642 12285 36099	82	0.70314 25026 52428	0.39240 90032 67163
19	0.05061 49174 28291	1.59152 34404 48183	83	0.71422 28396 45861	0.37081 48984 58508
20	0.05608 21449 35832	1.61318 70565 75242	84	0.72517 58883 61639	0.34985 12172 69811
21	0.06181 28659 95440	1.63159 43217 12110	85	0.73599 51283 56039	0.32952 72365 90520
22	0.06780 37228 49992	1.64691 35579 70304	86	0.74667 41186 82588	0.30985 08975 67521
23	0.07405 11997 24321	1.65930 11821 71879	87	0.75720 65017 36542	0.29082 88180 75879
24	0.08055 16252 15302	1.66890 33545 30274	88	0.76758 60070 49566	0.27246 63061 17412
25	0.08730 11747 31164	1.67585 73369 41028	89	0.77780 64550 32358	0.25476 73741 51915
26	0.09429 58729 86560	1.68029 26201 18965	90	0.78786 17606 62986	0.23773 47543 63030
27	0.10153 15965 58144	1.68233 18649 41669	91	0.79774 59371 18745	0.22136 99148 60683
28	0.10900 40765 04111	1.68209 16931 31345	92	0.80745 30993 49364	0.20567 30768 17906
29	0.11670 89010 50089	1.67968 33547 75995	93	0.81697 74675 89436	0.19064 32325 39806
30	0.12464 15183 43015	1.67521 32944 15160	94	0.82631 33708 07984	0.17627 81644 59896
31	0.13279 72392 73938	1.66878 36330 17563	95	0.83545 52500 93102	0.16257 44650 57395
32	0.14117 12403 70240	1.66049 25797 74488	96	0.84439 76619 69644	0.14952 75576 88015
33	0.14975 85667 57305	1.65043 47849 89451	97	0.85313 52816 48013	0.13713 17183 18610
34	0.15855 41351 89338	1.63870 16432 73515	98	0.86166 29062 02071	0.12538 00981 55166
35	0.16755 27371 48797	1.62538 15546 00984	99	0.86997 54576 74321	0.11426 47471 51864
36	0.17674 90420 13576	1.61056 01494 79418	100	0.87806 79861 06473	0.10377 66383 87970
37	0.18613 76002 90981	1.59432 04834 28739	101	0.88593 56724 93602	0.09390 56932 97734
38	0.19571 28469 17280	1.57674 32051 16010	102	0.89357 38316 60146	0.08464 08077 37368
39	0.20546 91046 21520	1.55790 67017 97779	103	0.90097 79150 56010	0.07596 98788 72160
40	0.21540 05873 52148	1.53788 72251 57163	104	0.90814 35134 71110	0.06787 98328 65537
41	0.22550 14037 64859	1.51675 90001 53819	105	0.91506 63596 66766	0.06035 66533 50734
42	0.23576 55607 70006	1.49459 43191 22929	106	0.92174 23309 22325	0.05338 54106 64953
43	0.24618 69671 37816	1.47146 36230 33934	107	0.92816 74514 95538	0.04695 02918 24747
44	0.25675 94371 59555	1.44743 55715 55687	108	0.93433 78949 95188	0.04103 46312 20362
45	0.26747 66943 62752	1.42257 71033 47012	109	0.94024 99866 64573	0.03562 09420 06234
46	0.27833 23752 78481	1.39695 34878 05284	110	0.94590 02055 74460	0.03069 09481 63463
47	0.28932 00332 58678	1.37062 83693 39307	111	0.95128 51867 24190	0.02622 56172 09761
48	0.30043 31423 41399	1.34366 38051 06133	112	0.95640 17230 49682	0.02220 51935 31385
49	0.31166 51011 61865	1.31612 02970 22983	113	0.96124 67673 37083	0.01860 92323 10655
50	0.32300 92369 07125	1.28805 68187 77837	114	0.96581 74340 40909	0.01541 66340 22503
51	0.33445 88093 12087	1.25953 08384 67344	115	0.97011 10010 05500	0.01260 56794 72175
52	0.34600 70146 94661	1.23059 83374 23762	116	0.97412 49110 88651	0.01015 40653 46127
53	0.35764 69900 27711	1.20131 38257 22613	117	0.97785 67736 86234	0.00803 89402 47295
54	0.36937 18170 45462	1.17173 03548 15615	118	0.98130 43661 56522	0.00623 69411 85475
55	0.38117 45263 82020	1.14189 95276 73881	119	0.98446 56351 42602	0.00472 42304 93011
56	0.39304 81017 39584	1.11187 15067 97452	120	0.98733 86977 90621	0.00347 65331 35540
57	0.40498 54840 83956	1.08169 50203 98862	121	0.98992 18428 59909	0.00246 91743 87128
58	0.41697 95758 64880	1.05141 73670 43149	122	0.99221 35317 16963	0.00167 71178 38695
59	0.42902 32452 58768	1.02108 44189 95537	123	0.99421 23991 94125	0.00107 50037 08408
60	0.44110 93304 31322	0.99074 06245 03614	124	0.99591 72542 60025	0.00063 71874 22478
61	0.45323 06438 17556	0.96042 90092 16661	125	0.99732 70803 30115	0.00033 77784 35258
62	0.46537 99764 16696	0.93019 11769 28847	126	0.99844 10345 31889	0.00015 06792 60567
63	0.47755 01020 99450	0.90006 73098 10460	127	0.99925 84422 26526	0.00004 96247 07774
64	0.48973 37819 25098	0.87009 61682 79796	128	0.99977 87533 06208	0.00000 82215 52537

As a check on the accuracy of these points and weights, the moments

$$I_p \equiv \int_0^1 \ln(1/x)\, dx = (p+1)^{-2} \tag{A6}$$

can be estimated by Gaussian quadrature (see Section 2.4) as

$$S_p^k \equiv \sum_{j=1}^{n} w_j^k \left(x_j^k\right)^p. \tag{A7}$$

Table xviii contains numerically computed value of the errors

$$\epsilon_p^k \equiv \log_{10}\left\{\max\left(10^{-99}, \left|I_p - S_p^k\right|\right)\right\} \tag{A8}$$

for moments 0 through 32, for the rules listed above. The moment index p runs down the the first column of the table, and the integration rule orders are listed across the top row. It is evident from the Table that the errors are of the order 10^{-15} or better, for these double-precision computations, provided that $2k > p$. Since the moment integrand term x^p is a particular polynomial of degree p, the tabulated results are consistent with the expectation of k-point Gaussian quadrature being exact [2] (see also Section 2.4) for polynomials of degree $2k-1$ or less.

A.6.3 Functional approximation

This final part considers some sample results for functional approximation. The two functions

$$f(x) = e^x \quad \text{and} \quad g(x) = \sin(8\pi x)$$

were chosen for the examples. For a first test, the expansion coefficients $\tilde{\alpha}_m^n, (m = 0, 1, \ldots, n)$ of Eq.(50) were evaluated for these two functions, for several values of n, using routine `Twd_LwOp_FcnApr_Coeffs_RDP`. The resulting pairs of series approximations (51) were then evaluated at 32 uniformly distributed points on $[0, 1]$ using routine `Twd_LwOp_FcnApr_Value_RDP`, and the absolute difference between the original functions and the associated series approximations.

The results are shown in Table xix. The approximation order n is indicated in the first columns. The second column gives the maximum error $\max_x \left|\tilde{f}^n(x) - f(x)\right|$ for function f, while the third column gives the analogous quantity for function g. Function g, with its greater variation over $[0, 1]$, requires more coefficients to achieve a comparable accuracy to that of the smoother function f.

For illustrative purposes, Table xx lists the expansion coefficients of order 32 for these two functions. The first column contains the index, and columns two and three contain the expansion coefficients for functions f and g respectively. As noted in conjunction with Table xix, the greater variation of function g over $[0, 1]$ induces a slower decay rate in the expansion coefficients, compared to those of the smoother function f.

B Exact Computations Using Algebraic Manipulation Software

As is demonstrated in Appendix A, the methods considered in this paper can be implemented in programming languages such as FORTRAN. However, concerns about stability, round-off error and accuracy must be considered. The recursive methods alternatively can be implemented *exactly* in symbolic algebra software

Table xviii: Errors in approximation of the moments $I_p \equiv \int_0^1 \ln(1/x)\,dx = (p+1)^{-2}$ by way of the quadrature sums $S_p^k \equiv \sum_{j=1}^n w_j^k (x_j^k)^p$. The table entries are $\log_{10}|I_p - S_p^k|$, with values of p running down the first column, and of k across the top row. Values -99.0 indicate a numerical difference of 0.

	2	3	4	5	6	8	10	16	32	64	128
0	−15.7	−15.5	−99.0	−16.0	−15.7	−99.0	−16.0	−16.0	−16.0	−15.7	−14.8
1	−16.3	−16.0	−16.6	−16.6	−15.9	−16.3	−15.9	−16.3	−16.3	−15.7	−15.3
2	−16.3	−16.2	−17.2	−16.5	−16.3	−16.5	−15.9	−16.3	−16.1	−15.9	−15.5
3	−16.4	−16.3	−17.2	−16.6	−16.4	−16.4	−16.2	−16.7	−16.6	−16.0	−15.7
4	−2.5	−16.5	−17.2	−16.6	−16.5	−16.5	−16.3	−16.9	−16.8	−16.1	−15.9
5	−2.3	−16.6	−17.1	−16.8	−16.7	−16.7	−16.5	−16.9	−16.7	−16.2	−16.1
6	−2.2	−3.8	−17.1	−16.8	−16.7	−16.9	−16.7	−16.9	−16.8	−16.2	−16.3
7	−2.1	−3.3	−17.1	−17.0	−16.8	−17.5	−16.9	−17.1	−17.2	−16.3	−16.4
8	−2.1	−3.0	−5.0	−17.0	−16.9	−18.2	−17.1	−17.1	−17.1	−16.4	−16.8
9	−2.2	−2.9	−4.4	−17.2	−16.9	−17.5	−17.2	−17.2	−17.4	−16.4	−17.3
10	−2.2	−2.8	−4.0	−6.2	−17.1	−17.3	−17.5	−17.2	−17.4	−16.5	−18.6
11	−2.2	−2.7	−3.8	−5.5	−17.1	−17.3	−17.7	−17.3	−17.7	−16.6	−17.4
12	−2.3	−2.7	−3.6	−5.0	−7.4	−17.2	−18.0	−17.3	−17.8	−16.6	−17.0
13	−2.3	−2.7	−3.5	−4.7	−6.6	−17.2	−18.4	−17.4	−17.9	−16.6	−17.0
14	−2.4	−2.7	−3.4	−4.5	−6.1	−17.2	−18.1	−17.4	−18.1	−16.6	−17.0
15	−2.4	−2.7	−3.3	−4.3	−5.7	−17.2	−17.8	−17.5	−17.8	−16.6	−16.9
16	−2.5	−2.7	−3.2	−4.1	−5.4	−9.8	−17.7	−17.5	−17.7	−16.7	−16.9
17	−2.5	−2.7	−3.2	−4.0	−5.2	−8.9	−17.7	−17.5	−17.4	−16.7	−16.9
18	−2.6	−2.7	−3.2	−3.9	−5.0	−8.3	−17.7	−17.5	−17.5	−16.7	−16.8
19	−2.6	−2.7	−3.1	−3.8	−4.8	−7.8	−17.6	−17.5	−17.4	−16.7	−16.9
20	−2.6	−2.7	−3.1	−3.7	−4.6	−7.4	−12.2	−17.5	−17.3	−16.7	−16.9
21	−2.7	−2.8	−3.1	−3.7	−4.5	−7.0	−11.2	−17.6	−17.3	−16.7	−16.9
22	−2.7	−2.8	−3.1	−3.6	−4.4	−6.8	−10.5	−17.6	−17.2	−16.7	−16.9
23	−2.8	−2.8	−3.1	−3.6	−4.3	−6.5	−9.9	−17.6	−17.2	−16.8	−16.9
24	−2.8	−2.8	−3.1	−3.6	−4.2	−6.3	−9.4	−17.6	−17.2	−16.8	−16.9
25	−2.8	−2.9	−3.1	−3.5	−4.2	−6.1	−9.0	−17.6	−17.2	−16.8	−16.9
26	−2.9	−2.9	−3.1	−3.5	−4.1	−5.9	−8.7	−17.6	−17.2	−16.8	−16.9
27	−2.9	−2.9	−3.1	−3.5	−4.1	−5.8	−8.4	−17.6	−17.2	−16.8	−16.9
28	−2.9	−2.9	−3.1	−3.5	−4.0	−5.7	−8.1	−17.7	−17.2	−16.8	−16.9
29	−3.0	−3.0	−3.1	−3.4	−4.0	−5.5	−7.8	−17.7	−17.2	−16.8	−16.9
30	−3.0	−3.0	−3.1	−3.4	−3.9	−5.4	−7.6	−17.7	−17.2	−16.8	−16.9
31	−3.0	−3.0	−3.1	−3.4	−3.9	−5.3	−7.4	−17.7	−17.2	−16.8	−16.9
32	−3.0	−3.0	−3.1	−3.4	−3.9	−5.2	−7.2	−17.7	−17.2	−16.8	−16.9

Table xix: Errors in various-order approximation of the functions $f(x) = e^x$ and $g(x) = \sin(8\pi x)$ in series of the orthonormal polynomials $\left\{\hat{\Lambda}_k(x)\right\}$. The first column gives the number of expansion coefficients used. The next two columns give the computed maximum absolute errors between the original function and its series approximation. The second column gives error for $f(x)$ and the third for $g(x)$.

4	3.50×10^{-03}	$2.13 \times 10^{+00}$
8	1.95×10^{-08}	$4.83 \times 10^{+00}$
16	1.56×10^{-13}	5.77×10^{-01}
32	2.68×10^{-13}	1.14×10^{-09}
64	4.62×10^{-13}	1.78×10^{-12}
128	2.33×10^{-11}	6.33×10^{-12}

such as MAPLE© [14]. Such methods will generally be slower for serious applications, but the results can be used for checking results from other implementations.

By way of illustration, this Appendix will present source code for the MAPLE© implemenation of the recursive methods for the coefficients and related quantities, for the monic polynomials $\{\Lambda_k(x)\}$, orthogonal with respect to $\ln(1/x)$, $x \in (0,1)$. The appropriate quantities for evaluation of the recursion coefficients are defined in Eqs.(A1–A3) and (A4).

It should be noted that this code can readily be modified to other weight functions, intervals, and their associated sets of orthogonal polynomials. For instance, in procedure `Moment` below, MAPLE© computes its own values of the moments $I_p \equiv \int_a^b W(x)\, x^p\, dx$. If these moments are known analytically for other weight functions, the procedure can readily be modified to suit.

As an example, complex Airy functions can [15] be computed by Gaussian quadrature of the integral

$$\mathrm{Ai}(z) = \frac{z^{-1/4} e^{-\zeta}}{2\sqrt{\pi}} \int_0^\infty \frac{W(x)}{1 + x/\zeta} dx,$$

where $\zeta \equiv \frac{2}{3} z^{3/2}$, and the weight function is

$$W(x) \equiv \frac{x^{-1/2} e^{-x}}{\pi\sqrt{2\pi}} K_{1/3}(x).$$

The associated moments can be evaluated analytically as

$$I_p \equiv \int_0^\infty W(x)\, x^p\, dx = \frac{\Gamma\left(3p + \frac{1}{2}\right)}{54^p p! \Gamma\left(p + \frac{1}{2}\right)}.$$

With the single modification of procedure `Moment` to assign this value, the procedures of Sections B.1 and B.2 below will compute recursion coefficients and monic polynomials orthogonal with respect to the above weight function.

Similarly, the MAPLE© procedures for evaluating the recursion coefficients by Methods A–C need only be amended to incorporate the appropriate changes to the modified moments and their ratios.

Table xx: Expansion coefficients for the functions $f(x) = e^x$ and $g(x) = sin(8\pi x)$ in 32-term series of the orthonormal polynomials $\hat{\Lambda}_k(x)$. The first column gives the expansion coefficient index. The second and third columns give the computed coefficients for $f(x)$ and $g(x)$ respectively.

k	$\tilde{\alpha}_k^{32}(f)$	$\tilde{\alpha}_k^{32}(g)$
0	$1.317902151454404 \times 10^{+00}$	$1.513147973332309 \times 10^{-01}$
1	$3.215909470153954 \times 10^{-01}$	$-1.605806198896547 \times 10^{-01}$
2	$4.046470581943436 \times 10^{-02}$	$1.497296618564923 \times 10^{-01}$
3	$3.391081982857586 \times 10^{-03}$	$-1.243736473313042 \times 10^{-01}$
4	$2.128182523800280 \times 10^{-04}$	$2.433436053019316 \times 10^{-02}$
5	$1.067359304879240 \times 10^{-05}$	$3.112713228522277 \times 10^{-02}$
6	$4.457792159317762 \times 10^{-07}$	$-2.280085598774621 \times 10^{-01}$
7	$1.595018434429016 \times 10^{-08}$	$2.215491641139984 \times 10^{-01}$
8	$4.991874500805387 \times 10^{-10}$	$-2.252383733472458 \times 10^{-01}$
9	$1.388266410375285 \times 10^{-11}$	$1.007947829682903 \times 10^{-01}$
10	$3.480453903698794 \times 10^{-13}$	$3.535712315760193 \times 10^{-01}$
11	$8.722415371491829 \times 10^{-15}$	$-2.329783722688169 \times 10^{-01}$
12	$9.080934348394947 \times 10^{-16}$	$-1.837600443381935 \times 10^{-01}$
13	$-3.225782466324627 \times 10^{-15}$	$1.289128315481664 \times 10^{-01}$
14	$6.223875276647247 \times 10^{-16}$	$5.508052362041811 \times 10^{-02}$
15	$-2.279639464461522 \times 10^{-15}$	$-3.962708862989713 \times 10^{-02}$
16	$2.273676987787562 \times 10^{-15}$	$-1.118071794197425 \times 10^{-02}$
17	$-3.274586175404815 \times 10^{-15}$	$8.150306803105618 \times 10^{-03}$
18	$1.774817145260379 \times 10^{-15}$	$1.666557021297416 \times 10^{-03}$
19	$-1.186251854937755 \times 10^{-15}$	$-1.224371914790243 \times 10^{-03}$
20	$1.038713326844396 \times 10^{-15}$	$-1.918012045700227 \times 10^{-04}$
21	$2.757468114183092 \times 10^{-15}$	$1.416186243327427 \times 10^{-04}$
22	$1.387914517811243 \times 10^{-15}$	$1.764650227160004 \times 10^{-05}$
23	$2.135868750675787 \times 10^{-15}$	$-1.307377422077332 \times 10^{-05}$
24	$-3.884313736881640 \times 10^{-15}$	$-1.331777280332852 \times 10^{-06}$
25	$5.471273797517592 \times 10^{-16}$	$9.890323445500162 \times 10^{-07}$
26	$4.906961389815464 \times 10^{-16}$	$8.411188344017722 \times 10^{-08}$
27	$-1.070911696147492 \times 10^{-15}$	$-6.257297696477150 \times 10^{-08}$
28	$-4.106210640436494 \times 10^{-15}$	$-4.517613524952792 \times 10^{-09}$
29	$-2.081332322608582 \times 10^{-15}$	$3.365298516475788 \times 10^{-09}$
30	$1.111514802809279 \times 10^{-15}$	$2.087931470469386 \times 10^{-10}$
31	$3.598460128336693 \times 10^{-15}$	$-1.621891906964176 \times 10^{-10}$

This Appendix will conclude with some MAPLE©-generated example results. Exact values of the first ten recursion coefficients, and for the first ten orthogonal polynomials are given, along with a plot of the first ten (scaled) orthogonal polynomials.

B.1 Polynomials

The polynomials $\Lambda_n(x)$ are denoted by $Lp(n,x)$ in the MAPLE© code. The following procedure for evaluating these polynomials is a direct implementation of Eq.(4), assuming that the coefficients a_n and b_n are known. These will be considered in later subsections.

Procedure(s) for evaluation of polynomials :

```
Lp              :=   proc( n, x )
  option             remember;
  if                 type(n,integer)   then
    if    n<0  then  0;
    elif  n=0  then  1;
    elif  n=1  then (x-a(n-1))*Lp(n-1,x) :
    else            (x-a(n-1))*Lp(n-1,x) - b(n-1)*Lp(n-2,x) :
    fi;
    collect         (",x);
  else              'Lp'(n,x);
  fi;
end;
```

B.2 Classical Method

The procedures in this subsection implement the classical method of moments for Eqs.(5) and (6). Procedure `W` defines the weight function, while `InProd` implements the inner product (1). The inner product is computed by expanding the polynomial product into a single polynomial, and computing the integral term-by-term using the classical moments $\mu_k \equiv -\int_0^1 \ln(x)\, x^k\, dx$. These latter are evaluated in procedure `Moment`. The "remember" option saves previously computed values for an increase in efficiency. The squared norms N_k^2 are computed in procedure `NormSq` using the inner product according to Eq.(2).

Procedure(s) for evaluation of inner products :

```
W               :=   proc(x)
  -log(x);
end;

Moment          :=   proc(k)
  local              x;
  option             remember;
  x               :=  'x';
  if                 type(k,integer)   then
     if    k<0 then  0
     else            int( W(x)*x^k, x=0..1 );
     fi;
  else              'Moment(k)';
```

```
  fi;
end;

InProd          :=   proc(p1,p2,x)
  local              prd, deg, i;
  option             remember;
  prd           :=   collect(expand(p1*p2),x);
  deg           :=   degree(prd,x);
  sum ( 'coeff(prd,x,'i')*Moment('i')', 'i' = 0..deg );
end;

NormSq          :=   proc(k)
  local              x;
  option             remember;
  x             :=  'x';
  if                 type(k,integer)   then
    if    k<0  then  0;
    else             InProd(Lp(k,x),Lp(k,x),x):
    fi;
  else              'NormSq'(k);
  fi;
end;
```

Procedures a and b are direct implementations of Eqs.(5) and (6) respectively.

Procedure(s) for evaluation of a_k, Classical Method :

```
a               :=   proc( j )
  local              x;
  option             remember;
  if                 type(j,integer)   then
    if    j<0  then  0;
    else             InProd(x*Lp(j,x),Lp(j,x),x)/NormSq(j)
    fi;
  else              'a'(j);
  fi;
end;
```

Procedure(s) for evaluation of b_k, Classical Method :

```
b               :=   proc( j )
  local              x;
  option             remember;
  if                 type(j,integer)   then
    if   j<=0  then  0;
    else             NormSq(j) / NormSq(j-1)
    fi;
  else              'b'(j);
  fi;
end;
```

B.3 Method A

The procedures in this subsection are concerned with the direct implementation of Wheeler's algorithm. The next two procedures (aT and bT) compute the recursion coefficients $\tilde{a}_j$ and $\tilde{b}_j$ for the monic shifted Legendre polynomials, as given by Eq.(A2).

Procedure(s) for evaluation of $\tilde{a}_j$:

```
aT          :=   proc(j)
   option        remember;
   if            type(j,integer)  then
      if    j>=0 then   1/2;
      else            0;
      fi;
   else          'aT'(j);
   fi;
end;
```

Procedure(s) for evaluation of $\tilde{b}_j$:

```
bT          :=   proc(j)
   option        remember;
   if            type(j,integer)  then
      if    j <1 then   0
      else       1/(16-4/j^2);
      fi;
   else         'bT'(j);
   fi;
end;
```

Next, procedure Nu evaluates the modified moments (A3) :

Procedure(s) for evaluation of ν_k :

```
Nu          :=   proc(j)
   option        remember;
   if            type(j,integer)  then
      if    j <0 then   0;
      elif  j =0 then   1;
      elif  j>=1 then
                  (-1)^j*(j!)^2/j/(j+1)/(2*j)!;
      fi;
   else         'Nu'(j);
   fi;
end;
```

Finally, all of the pieces are in place for implementation of Wheeler's algorithm as shown in Table i. Procedure Sig computes $\sigma_{k,\ell}$:

Procedure(s) for evaluation of $\sigma_{k,\ell}$:

```
Sig         :=   proc(k,l)
   option        remember;
   if            type(k,integer)  and  type(l,integer)  then
```

```
      if    k<=-1 then   0;
      elif  k = 0 then   Nu(l);
      else        Sig(k-1,l+1)
                 -Sig(k-1,l  )*(aA(k-1)-aT(l))
                 -Sig(k-2,l  )* bA(k-1)
                 +Sig(k-1,l-1)*         bT(l)
      fi;
   else          'Sig'(k,l);
   fi;
end;
```

Similarly, procedures **aA** and **bA** compute the polynomial recurrence coefficients a_k and b_k respectively, using Wheeler's Method :

Procedure(s) for evaluation of a_k, Method A :

```
aA           :=   proc(k)
   option          remember;
   if              type(k,integer)   then
      if    k <0   then   0
      elif  k =0   then   aT(0) + Nu(1)/Nu(0);
      else
         aT(k)  - Sig(k-1,k)/Sig(k-1,k-1)
                + Sig(k,k+1)/Sig(k  ,k  );
      fi;
   else           'aA'(k);
   fi;
end;
```

Procedure(s) for evaluation of b_k, Method A :

```
bA           :=   proc(k)
   option          remember;
   if              type(k,integer)   then
      if     k<1   then   0
      else         Sig(k,k)/Sig(k-1,k-1);
      fi;
   else           'bA'(k);
   fi;
end;
```

B.4 Method B

The procedures in this subsection constitute the MAPLE© implementation as presented in Table iii. Procedure **Rho** computes the ratios ρ_j of Eq.(A4) :

Procedure(s) for evaluation of ρ_k :

```
Rho          :=   proc(j)
   option          remember;
   if              type(j,integer)  then
      if     j<0   then   0
```

```
      elif   j=0  then   -1/4;
      else      -(j/(j+1/2))*((j+1)/(j+2))/4;
      fi;
   else          'Rho'(j);
   fi;
end;
```

Procedure Tau computes $\tau_{k,\ell}$:

Procedure(s) for evaluation of $\tau_{k,\ell}$:

```
Tau          :=   proc(k,l)
   option          remember;
   if              type(k,integer)  and
                   type(l,integer)         then
      if    k<=-1 then   0;
      elif  k = 0 then   1;
      else        Tau(k-1,l+1)                 * Rho(l)
                 -Tau(k-1,l  )*(aB(k-1)-aT(l))
                 -Tau(k-2,l  )* bB(k-1)
                 +Tau(k-1,l-1)*         bT(l) / Rho(l-1)
      fi;
   else          'Tau'(k,l);
   fi;
end;
```

Procedures aB and bB compute the corresponding Method B values of the polynomial recurrence coefficients a_k and b_k :

Procedure(s) for evaluation of a_k, Method B :

```
aB           :=   proc(k)
   option          remember;
   if              type(k,integer)   then
      if     k<0  then   0
      elif   k=0  then   aT(0) + Rho(0);
      else
         aT(k)  - Rho(k-1)*Tau(k-1,k)/Tau(k-1,k-1)
                + Rho(k  )*Tau(k,k+1)/Tau(k  ,k  );
      fi;
   else          'aB'(k);
   fi;
end;
```

Procedure(s) for evaluation of b_k, Method B :

```
bB           :=   proc(k)
   option          remember;
   if              type(k,integer)   then
      if    k <1  then   0
      else        Rho(k-1)*Tau(k,k)/Tau(k-1,k-1);
      fi;
   else          'bB'(k);
   fi;
```

```
end;
```

B.5 Method C

Finally, the procedures in this section are concerned with the fully scaled Method C, as presented in Table iv. The scaled inner products $\eta_{k,\ell}$ are computed by procedure Eta.

Procedure(s) for evaluation of $\eta_{k,\ell}$:

```
Eta          :=   proc(k,l)
   option          remember;
   if              type(k,integer)  and  type(l,integer)  then
      if    k<=-1 then   0;
      elif  k = 0 then   1;
      elif  k > 0 then
         if       k = l  then   1;
         elif     k = 1  then
            ( Rho(k-1) / bC(k) ) *
              (   Rho(l)           *Eta(k-1,l+1)
               - (aC(k-1) - aT(l) )*Eta(k-1,l  )
               + (bT(l) / Rho(l-1))*Eta(k-1,l-1) )
         elif     k > 1  then
            ( Rho(k-1) / bC(k) ) *
              (   Rho(l)           *Eta(k-1,l+1)
               -  Rho(k-2)         *Eta(k-2,l  )
               - (aC(k-1) - aT(l) )*Eta(k-1,l  )
               + (bT(l) / Rho(l-1))*Eta(k-1,l-1) )
         else     0;
         fi;
      fi;
   else          'Eta(k,l)';
   fi;
end;
```

The associated Method C values of the polynomial recurrence coefficients a_k and b_k are evaluated by procedures aC and bC :

Procedure(s) for evaluation of a_k, Method C :

```
aC           :=   proc(k)
   option          remember;
   if              type(k,integer)   then
      if     k<0  then   0
      elif   k=0  then   aT(0)  + Rho(0);
      else
         aT(k)  - Rho(k-1)*Eta(k-1,k)
                + Rho(k  )*Eta(k,k+1);
      fi;
   else          'aB'(k);
   fi;
end;
```

Procedure(s) for evaluation of b_k, Method C :

```
bC          :=  proc(k)
   option       remember;
   if           type(k,integer)   then
      if    k<1  then    0
      elif  k=1  then
         bT(k) + Rho(k-1)*(      Rho(k)     * Eta(k-1,k+1)
                          -(aC(k-1)-aT(k)) * Eta(k-1,k  ) )
      else
         bT(k) + Rho(k-1)*(      Rho(k)     * Eta(k-1,k+1)
                          -(aC(k-1)-aT(k)) * Eta(k-1,k  )
                          -     Rho(k-2)   * Eta(k-2,k  ) )
      fi;
   else         'bC'(k);
   fi;
end;
```

B.6 Examples

As expected, all four MAPLE© methods for evaluation of the recurrence coefficients give the same values. The first ten values of the polynomial recursion coefficients a_k, $(k = 0, 1, \ldots, 9)$ are

$$
\begin{aligned}
a_0 &= \frac{1}{4} \\
a_1 &= \frac{13}{28} \\
a_2 &= \frac{8795}{18116} \\
a_3 &= \frac{124351943}{252694908} \\
a_4 &= \frac{43450203422161}{87773135347044} \\
a_5 &= \frac{2350608674255710485401 3941}{4733599794473525918062 6044} \\
a_6 &= \frac{3429343438514001166060586581447320148 27}{6893134609937196845985539362237437588 52} \\
a_7 &= \frac{1370473272796736925399268247030806611845310896727470 15}{2751434915315026414423175037598397322631460760007690 52} \\
a_8 &= \frac{1363248503264472819079210465311574552506204405592494129007942989538787 3}{2734713167950829423649864246552342749961179977850619600799126924145574 8} \\
a_9 &= \frac{3774287584068443845903069228058467370268556462526222452496246871937955946324640826512966 9}{7566957993015766121724650617591240095734835782406954843113336815937366131863410397059359 6}
\end{aligned}
\tag{B1}
$$

in exact rational form. Similarly exact values of b_k, $(k = 0, 1, \ldots, 9)$ are

$$
\begin{aligned}
b_0 &= 0 \\
b_1 &= \frac{7}{144} \\
b_2 &= \frac{647}{11025} \\
b_3 &= \frac{71180289}{1172105200} \\
b_4 &= \frac{332349955856}{5405644687527} \\
b_5 &= \frac{3967248102309963159437 5}{6415259005082182745619 36} \\
b_6 &= \frac{434286744216380246109027147868474 47}{700021443804201423473849094692097649} \\
b_7 &= \frac{3280755863573189905233164756820779953521806520655 1}{5277928137613544054752582913959579252952707064196 00} \\
b_8 &= \frac{2207327481598004999136185564108285606747256313683185646219556633 6}{3546537849412642903327484260313256564326239536786679817663982650 25} \\
b_9 &= \frac{8911407975015412056192295789506670405305519774657223193264409794266913509642683100 23}{1430554375650743151150166338850559299843458434749257373478825583532068755065394076481 6 .}
\end{aligned}
\tag{B2}
$$

The first ten (monic) orthogonal polynomials, as generated by MAPLE© are

$$
\begin{aligned}
\Lambda_0(x) &= 1 \\
\Lambda_1(x) &= x \quad -1/4 \\
\Lambda_2(x) &=. x^2 \quad -\tfrac{5}{7}x + \tfrac{17}{252} \\
\Lambda_3(x) &= x^3 \quad -\tfrac{3105}{2588}x^2 + \tfrac{5751}{16175}x - \tfrac{4679}{258800} \\
\Lambda_4(x) &= x^4 \quad -\tfrac{165196}{97641}x^3 + \tfrac{67227}{75943}x^2 - \tfrac{79564}{531601}x + \tfrac{2296639}{478440900} \\
\Lambda_5(x) &= x^5 \quad -\tfrac{17692971425}{8090435556}x^4 + \tfrac{2449515716800}{1474481880081}x^3 \\
&\quad -\tfrac{112304929775}{218441760012}x^2 + \tfrac{203478628075}{3567882080196}x \\
&\quad -\tfrac{1461977847751}{1155993793983504} \\
\Lambda_6(x) &= x^6 \quad -\tfrac{15700658824389411}{5850859031888599}x^5 + \tfrac{7604560816456422375}{2831815771434081916}x^4 \\
&\quad -\tfrac{873930519668513600}{707953942858520479}x^3 + \tfrac{182360435978518375}{707953942858520479}x^2 \\
&\quad -\tfrac{5756887477447529}{2831815771434081916}x + \tfrac{1654296840628723409}{4995323020809720499824} \\
\Lambda_7(x) &= x^7 \quad -\tfrac{3747642360380611053475 45}{1178140600766962504339 48}x^6 \\
&\quad +\tfrac{1970348797534910591224 6983}{4977644063590416580834303}x^5 \\
&\quad -\tfrac{1939101144210753476588 83625}{7964230501744666529334 8848}x^4 \\
&\quad +\tfrac{4629614918529636824062 37375}{6022949316944404062809 50663}x^3 \\
&\quad -\tfrac{2808724186513574552586 35313}{2409179726777761625123 802652}x^2 \\
&\quad +\tfrac{3743992762908617669916 6095}{5420654385249963656528 555967}x \\
&\quad -\tfrac{2993820619101904789561 5767}{3469218806559976740178 27581888}
\end{aligned}
\tag{B3}
$$

$$
\begin{aligned}
\Lambda_8(x) &= x^8 \quad -\tfrac{8592127439414827792392700638640}{2335404534493957255219087217249}x^7 \\
&\quad +\tfrac{2879902591882908608456143471299476}{5254660202611403824242946238 81025}x^6 \\
&\quad -\tfrac{2475301915444601804380583685 57296}{5838511336234893138047718043 1225}x^5 \\
&\quad +\tfrac{7161446304080014882127587211 54645}{3946833663294787761320257397 15081}x^4 \\
&\quad -\tfrac{1501555692963855159769862124 531760}{3552150296953089851882316574 35729}x^3 \\
&\quad +\tfrac{2338414406348495433553215416820980}{4775668732586693191197511450552 4801}x^2 \\
&\quad -\tfrac{2701732605213401529312417844193584}{1193917183146673297799377862638 120025}x \\
&\quad +\tfrac{8664359260807654792244720107099399}{3868291673395221484869984274947508881 00}
\end{aligned}
$$

$$
\begin{aligned}
\Lambda_9(x) &= x^9 \quad -\tfrac{4891855785102952408187410379095220250 8097}{1170980499334945245200632340900301954 1252}x^8 \\
&\quad +\tfrac{6135802694564908064033814211583445704899968}{8460334107694979396574568663004681618554 57}x^7 \\
&\quad -\tfrac{8186859116721137274090364333413315434420 19}{1208619158242139913796366951857811659793 51}x^6 \\
&\quad +\tfrac{1112397799628431789617873199477157178494298 3}{3021547895605349784490917379644529149483775}x^5 \\
&\quad -\tfrac{5683865526086732186767845078821934087341 55}{4834476632968559655185467807431246639174 04}x^4 \\
&\quad +\tfrac{4327138606971270080504217710113896787024320}{2042566377429216454315860148639701705051 0319}x^3 \\
&\quad -\tfrac{3965819327923284534862632807313065577374 75}{2042566377429216454315860148639701705051 0319}x^2 \\
&\quad +\tfrac{4986555479273341472512568079597721216175 7855}{6920214886730185347222134183591309376712 8960772}x \\
&\quad -\tfrac{4009950193434285017744948709762382735370 4151}{6920214886730185347222134183591309376712 8960772 00}.
\end{aligned}
$$

The first ten scaled polynomials $q_{1,k}(x)$, $(k = 0, 1, \ldots, 9)$ (see Eq.(14)) are plotted in Figure 3. The appropriate subscript k is indicated by the subscripts adjacent to the lines.

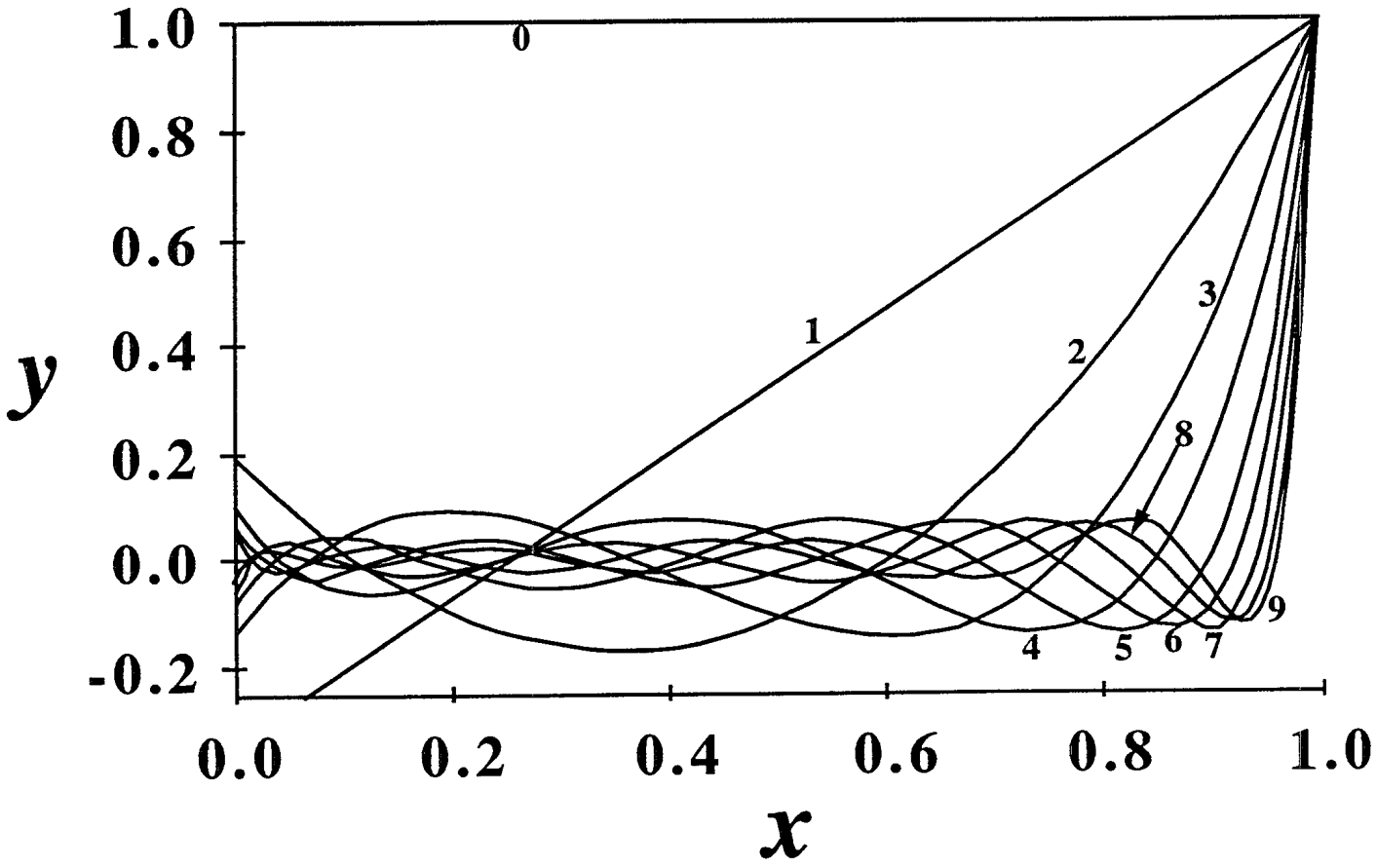


Figure 3: Plots of the polynomials $q_{1,k}(x)$, $(k = 0, 1, \ldots, 9)$, orthogonal with respect to $W(x) = \ln(1/x)$ on $[0, 1]$. See also Eq.(14).

Title:	GQLW - A Package of Subroutines for Polynomials Orthogonal with Respect to ln(1/x), Including Theory and Applications
Author(s):	T.W. Dawson
Report Series:	DREP Technical Memorandum 94-108
Date:	June 1994
Classification:	Unclassified

Distribution List:

1 - DSIS

9 - DREP

- Library (1)
- Dr. T. Dawson (6)
- Dr. G. Brooke (1)
- Dr. D. Thomson (1)

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1 ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research Establishment Pacific
FMO Victoria, B.C. V0S 1B0

2. SECURITY CLASSIFICATION (overall security classification of the document. including special warning terms if applicable)

Unclas

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

CQLW - A Package of Subroutines for Polynomials Orthogonal with Respect to ln(1/x), including Theory and Applications (U)

4 AUTHORS (Last name, first name, middle initial)

T.W. Dawson

5 DATE OF PUBLICATION (month and year of publication of document)

June 1994

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

vi + 47

6b. NO. OF REFS (total cited in document)

15

7 DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

Defence Research Establishment Pacific
FMO Victoria, B.C. V0S 1B0

9a PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

94-108

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

- (X) Unlimited distribution
- ( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
- ( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
- ( ) Distribution limited to government departments and agencies; further distribution only as approved
- ( ) Distribution limited to defence departments; further distribution only as approved
- ( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

DCD03 2/06/87

SECURITY CLASSIFICATION OF FORM

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both offical languages unless the text is bilingual).

The main thrust of this document is an improved method for computing polynomials $\{A_n(x)\}$ orthogonal with respect to the positive weight function $\ln(1/x)$ on the interval $x \in [0, 1]$. An implementation of Wheeler's algorithm was recently published in Numerical Recipes. However, the modified moments used are prone to underflow on computers with limited precision and floating-point range. The problems can be circumvented by employing scaling, and by working in terms of ratios.

The paper contains a summary of general orthogonal polynomial theory, including a derivation of functional approximation and Gaussian quadrature from a matrix theory viewpoint. Subsequently, a derivation of Wheeler's algorithm, together with the two variations mentioned above, is presented. Appendix A documents a set of FORTRAN routines implementing the theory, including routines for evaluating the recursion coefficients, polynomial values and derivatives, points and weights for Gaussian quadrature, integrating scalar- and vector-valued functions, and approximating functions in series of these orthogonal polynomials. By way of illustration, the first 128 recursion coefficients are tabulated, together with points and weights, for Gaussian quadrature of $\int_0^1 \ln(1/x)\, F(x)\, dx$, of orders $2, 3, 4, 5, 6, 8, 10, 16, 32, 64$ and 128. Appendix B contains MAPLE© source code for working with these polyomials, and indicates how the code can be modified to accommodate other weight functions and intervals. For illustrative purposes, exact values of the first ten recursion coefficients and orthogonal polynomials are given.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Orthogonal Polynomials
Functional Approximation
Gaussian Quadrature
Matrix Methods
Symbolic Algebra Software

SECURITY CLASSIFICATION OF FORM

5/15

#143688

NO. OF COPIES NOMBRE DE COPIES	COPY NO. COPIE N°	INFORMATION SCIENTIST'S INITIALS INITIALES DE L'AGENT D'INFORMATION SCIENTIFIQUE
1	1	JL

AQUISITION ROUTE / FOURNI PAR ▶ DREP

DATE ▶ 05 Jul 94

DSIS ACCESSION NO. / NUMÉRO DSIS ▶ 95-00064

DND 1158 (6-87)

National Defence / Défense nationale

PLEASE RETURN THIS DOCUMENT TO THE FOLLOWING ADDRESS:

DIRECTOR
SCIENTIFIC INFORMATION SERVICES
NATIONAL DEFENCE
HEADQUARTERS
OTTAWA, ONT. - CANADA K1A 0K2

PRIÈRE DE RETOURNER CE DOCUMENT À L'ADRESSE SUIVANTE:

DIRECTEUR
SERVICES D'INFORMATION SCIENTIFIQUES
QUARTIER GÉNÉRAL
DE LA DÉFENSE NATIONALE
OTTAWA, ONT. - CANADA K1A 0K2