CS 81

CALCULATION OF GAUSS QUADRATURE RULES

BY

GENE H. GOLUB AND JOHN H. WELSCH

TECHNICAL REPORT NO. CS 81
NOVEMBER 3, 1967

COMPUTER SC IENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

CALCULATION OF GAUSS QUADRATURE RULES*

Gene H. Golub and John H. Welsch

Introduction

Most numerical integration techniques consist of approximating the integrand by a polynomial in a region or regions and then integrating the polynomial exactly. Often a complicated integrand can be factored into a non-negative "weight" function and another function better approximated by a polynomial, thus

$$\int_a^b g(t)dt = \int_a^b \omega(t)f(t)dt \approx \sum_{i=1}^{N} w_i f(t_i) .$$

Hopefully, the quadrature rule $\{w_j, t_j\}_{j=1}^{N}$ corresponding to the weight function $\omega(t)$ is available in tabulated form, but more likely it is not. We present here two algorithms for generating the Gaussian quadrature rule defined by the weight function when:

a) the three term recurrence relation is known for the orthogonal polynomials generated by $\omega(t)$, and

b) the moments of the weight function are known or can be calculated.

*The work of the first author was in part supported by the Office of Naval Research and the National Science Foundation; the work of the second author was in part supported by the Atomic Energy Commission.

1. Definitions and Preliminaries

Let $w(x) \geq 0$ be a fixed weight function defined on $[a, b]$. For $\omega(x)$, it is possible to define a sequence of polynomials $p_0(x), p_1(x), \ldots$ which are orthonormal with respect to $\omega(x)$ and in which $p_n(x)$ is of exact degree n so that

$$\int_a^b \omega(x) p_m(x) p_n(x) dx = \begin{cases} 1 & \text{when } m = n \\ 0 & \text{when } m \neq n \end{cases}. \qquad (1.1)$$

The polynomial

$$p_n(x) = k_n \prod_{i=1}^{n} (x - t_i), \qquad k_n > 0,$$

has n real roots

$$a < t_1 < t_2 < \ldots < t_n < b .$$

The roots of the orthogonal polynomials play an important role in -Gaussian quadrature.

Theorem: Let $f(x) \in C^{2N}[a, b]$, then

$$\int_a^b \omega(x) f(x) dx = \sum_{j=1}^{N} w_j f(t_j) + \frac{f^{(2N)}(\xi)}{(2N)!\, k_N^2}, \qquad (a < \xi < b),$$

where

$$w_j = -\frac{k_{N+1}}{k_N}\,\frac{1}{p_{N+1}(t_j)\,p_N'(t_j)}\,,$$

$$\left(p_N'(t_j) = \left.\frac{dp_N(t)}{dt}\right|_{t=t_j}\right), \qquad j = 1,\ 2,\ \ldots,\ N\ .$$

Thus the Gauss quadrature rule is exact for all polynomials of degree $< 2N-1$. Proofs of the above statements and Theorem can be found in Davis and Rabinowitz [4], chapter 2.

Several algorithms have been proposed for calculating

$$\{w_j,\ t_j\}_{j=1}^{N};\quad \text{cf [8], [9].}$$

In this note, we shall give effective numerical algorithms which are based on determining the eigenvalues and the first component of the eigenvectors of a symmetric tri-diagonal matrix.

2. Generating the Gauss Rule

Any set of orthogonal polynomials, $\{p_j(x)\}_{j=1}^{N}$, satisfies a three term recurrence relationship:

$$p_{j+1}(x) = (a_{j+1}x + b_{j+1})\,p_j(x) - c_{j+1}p_{j-1}(x) \qquad (2.1)$$

$$j = 0,\ 1,\ 2,\ \ldots,\ N-1; \qquad p_{-1}(x) \equiv 0, \qquad p_0(x) \equiv 1,$$

with

$$a_j > 0, \qquad c_j > 0\ .$$

The coefficients $\{a_j, b_j, c_j\}$ have been tabulated for a number of weight functions $\omega(x)$, cf[6]. In section 4 we shall give a simple method for generating $\{a_j, b_j, c_j\}$ for any weight function.

Following Wilf [10], we may identify (2.1) with the matrix equation

$$x\begin{bmatrix} p_0(x) \\ p_1(x) \\ \cdot \\ \cdot \\ \cdot \\ p_{N-1}(x) \end{bmatrix} = \begin{bmatrix} -b_1/a_1, & 1/a_1, & 0 & & \\ c_2/a_2, & -b_2/a_2, & 1/a_2 & & \\ 0 & \cdot & \cdot & \cdot & \\ & \cdot & \cdot & \cdot & \\ & & & & 1/a_{N-} \\ & & & c_N/a_N, & -b_N/a_N \end{bmatrix} \begin{bmatrix} p_0(x) \\ p_1(x) \\ \cdot \\ \cdot \\ \cdot \\ p_{N-1}(x) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \\ p_N(x)/a_1 \end{bmatrix}$$

or, equivalently in matrix notation

$$x\underset{\sim}{p}(x) = T\underset{\sim}{p}(x) + \frac{1}{a_N} p_N(x)\underset{\sim}{e}_N$$

where T is the tri-diagonal matrix and $\underset{\sim}{e}_N = (0, 0, \ldots, 0, 1)^T$. Thus $p_N(t_j) = 0$ if and only if

$$t_j\underset{\sim}{p}(t_j) = T\underset{\sim}{p}(t_j)$$

where t_j is an eigenvalue of the tri-diagonal matrix T. In [10], it is shown that T is symmetric if the polynomials are orthonormal. If T is not symmetric, then we may perform a diagonal similarity

transformation which will yield a symmetric tri-diagonal matrix J .

Thus

$$DTD^{-1} = J = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ 0 & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & \beta_{N-1} & \alpha_N \end{bmatrix}$$

where

$$a_1 = -\frac{b_i}{a_i}, \quad \beta_i = \left(\frac{c_{i+1}}{a_i a_{i+1}}\right)^{\frac{1}{2}} \tag{2.2}$$

It is shown by Wilf [10] that as a consequence of the Christoffel-Darboux identity

$$w_j[\underset{\sim}{p}(t_j)]^T[\underset{\sim}{p}(t_j)] = 1 \qquad = 1, 2, \ldots, N \tag{2.3}$$

where $\underset{\sim}{p}(t_j)$ corresponds to the normalized eigenvector associated with the eigenvalue t_j . Suppose that the eigenvectors of T are calculated so that

$$J\underset{\sim}{q}_j = t_j\underset{\sim}{q}_j \qquad j = 1, 2, \ldots, N \tag{2.4}$$

with

$$\underset{\sim}{q}_j^T \underset{\sim}{q}_j = 1 .$$

Then if

$$\underset{\sim}{q}_j^T = (q_{0j}, q_{1j}, \cdots, q_{N-1,j}),$$
$$q_{0j}^2 = w_j (p_0(t_j))^2 \qquad (2.5)$$

by (2.3) . Thus from (1.1), we see

$$w_j = \frac{q_{0j}^2}{p_0^2(t_j)} = \frac{q_{0j}^2}{k_0^2} = q_{0j}^2 \times \int_a^b \omega(x)dx \equiv q_{0j}^2 \times \mu_0 . \qquad (2.6)$$

Consequently, if one can compute the eigenvalues of T and the first component of the orthonormal eigenvectors, one is able to determine the Gauss quadrature rule.

3. The Q-R algorithm

One of the most effective methods of computing the eigenvalues and eigenvectors of a symmetric matrix is the Q-R algorithm of Francis [5]. The Q-R algorithm proceeds as follows:

Begin with the given matrix $J = J_0^{(0)}$, compute the factorization

$$J^{(0)} = Q^{(0)} R^{(0)}$$

where $Q^{(0)^T}Q^{(0)} = I$ and $R^{(0)}$ is an upper triangular matrix, and then multiply the matrices in reverse order so that

$$J^{(1)} = R^{(0)}Q^{(0)} = Q^{(0)^T}J^{(0)}Q^{(0)} .$$

Now one treats the matrix $J^{(1)}$ in the same fashion as the matrix $J^{(0)}$, and a sequence of matrices is obtained by continuing ad infinitum, Thus

$$\begin{aligned} J^{(i)} &= Q^{(i)}R^{(i)}, \\ \text{--- } J^{(i+1)} &= R^{(i)}Q^{(i)} = Q^{(i+1)}R^{(i+1)} \end{aligned} \tag{3.1}$$

so that

$$\begin{aligned} J^{(i+1)} &= Q^{(i)^T}J^{(i)}Q^{(i)} \\ &= Q^{(i)^T}Q^{(i-1)^T} \cdots Q^{(0)^T}JQ^{(0)}Q^{(1)} \cdots Q^{(i)} . \end{aligned} \tag{3.2}$$

Since the eigenvalues of J are distinct and real for orthogonal polynomials, a real translation parameter λ may be chosen so that the eigenvalues of $J^{(i)} - \lambda I$ are distinct in modulus. Under these conditions, it is well known [5] that $J^{(i)} - \lambda I$ converges to the diagonal matrix of eigenvalues of $J - \lambda I$ as $i \to \infty$ and that $P^{(i)} = Q^{(0)} \times Q^{(1)} \times \ldots \times Q^{(i)}$ converges to the orthogonal matrix of eigenvectors of J . The method has the advantage that the matrix $J^{(i)} - \lambda I$ remains tri-diagonal throughout the computation.

Francis has shown that it is not necessary to compute the decomposition (3.1) explicitly but it is possible to do the calculation (3.2) directly. Let

$$\{S^{(i)}\}_{k,1} = \{Q^{(i)}\}_{k,1} \qquad (k = 0, 1, \ldots, N-1),$$

(i.e., the elements of the first column of $S^{(i)}$ are equal to the elements of the first column of $Q^{(i)}$). Then if

i) $K^{(i+1)} = S^{(i)^T} J^{(i)} S^{(i)}$,

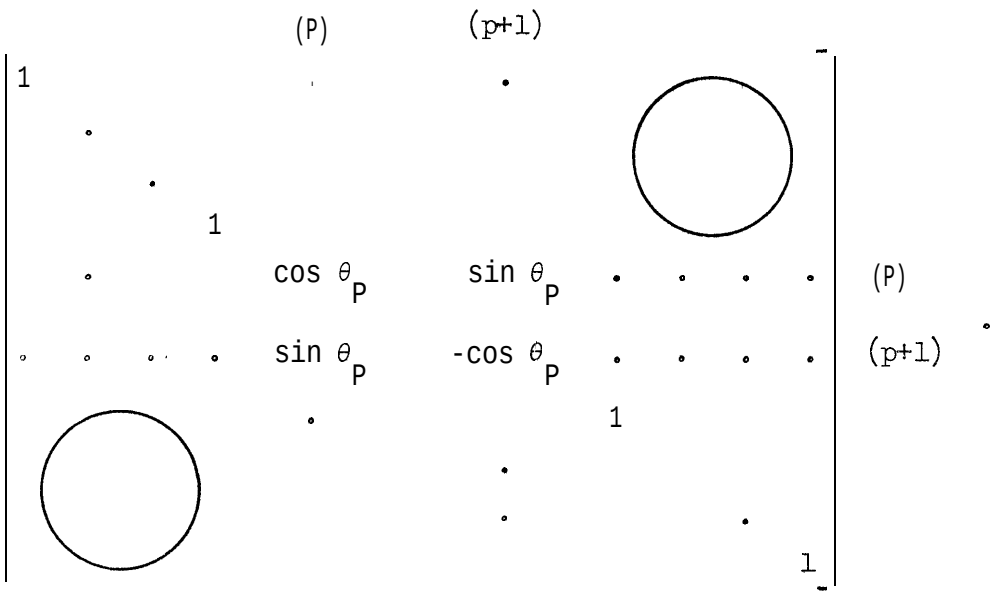
ii) $K^{(i+1)}$ is a tri-diagonal matrix,

iii) $J^{(i)}$ is non-singular,

iv) the sub-diagonal elements of $K^{(i+1)}$ are positive,

it follows that $K^{(i+1)} = J^{(i+1)}$.

For the tri-diagonal matrices, the calculation is quite simple. Dropping the iteration counter i, let



Then $\cos \theta_1$ is chosen so that

$$\{Z_1 J\}_{k,1} = 0 \qquad k = 2, 3, \ldots, \quad .$$

Let

$$J = \begin{bmatrix} a_1 & b_1 & 0 & & & \\ b_1 & a_2 & b_2 & & & \\ 0 & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & & \\ & & \cdot & \cdot & b_{N-1} & \\ & & & & b_{N-1} & a_N \end{bmatrix}$$

The matrix

$$Z_1 J Z_1 = \begin{bmatrix} a_1' & b_1' & d_1 & & & \\ b_1' & a_2' & b_2' & 0 & & \\ d_1 & b_2' & a_3 & b_3 & & \\ & 0 & b_3 & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & b_{Ni-1} \\ & & & & b_{N-1} & a_N \end{bmatrix},$$

where the primes indicate altered elements of J; then

$$K = Z_{N-1} Z_{N-2} \cdots Z_1 J Z_1 \cdots Z_{N-1}$$

and $Z_2, \ldots, Z_{N-1}$ are constructed so that K is tri-diagonal. The product of all the orthogonal rotations yields the matrix of orthogonal eigenvectors. To determine $\{w_j\}_{j=1}^{N}$, however, we need only the first component of the orthonormal eigenvector. Thus, using (2.3)

$$\underset{\sim}{q}^T \equiv [q_{01}, q_{02}, \ldots, q_{0N}] = [1, 0, 0, \ldots 0] \times \prod_{i=0}^{\infty} (Z_1^{(i)} \times Z_2^{(i)} \times \ldots \times Z_{N-1}^{(i)})$$

and it is not necessary to compute the entire matrix of eigenvectors. More explicitly, for j = 1, 2, ..., N-1

$$\sin \theta_j^{(i)} = d_{j-1}^{(i)} / [(d_{j-1}^{(i)})^2 + (\bar{b}_{j-1}^{(i)})^2]^{\frac{1}{2}},$$

$$\cos \theta_j^{(i)} = \bar{b}_{j-1}^{(i)} / [(d_{j-1}^{(i)})^2 + (\bar{b}_{j-1}^{(i)})^2]^{\frac{1}{2}},$$

$$a_j^{(i+1)} = \bar{a}_j^{(i)} \cos^2 \theta_j^{(i)} + 2\tilde{b}_j^{(i)} \cos \theta_j^{(i)} \sin \theta_j^{(i)} + a_{j+1}^{(i)} \sin^2 \theta_j^{(i)},$$

$$\bar{a}_{j+1}^{(i)} = \bar{a}_{j-1}^{(i)} \cos^2 \theta_j^{(i)} + 2\tilde{b}_j^{(i)} \cos \theta_j^{(i)} \sin \theta_j^{(i)} + a_{j+1}^{(i)} \sin^2 \theta_j^{(i)},$$

$$b_{j-1}^{(i+1)} = \bar{b}_{j-1}^{(i)} \cos \theta_j^{(i)} + d_{j-1}^{(i)} \sin \theta_j^{(i)} = [(\bar{b}_{j-1}^{(i)})^2 + (d_{j-1}^{(i)})^2]^{\frac{1}{2}}$$

$$\bar{b}_j^{(i)} = (\bar{a}_j^{(i)} - a_{j+1}^{(i)}) \sin \theta_j^{(i)} \cos \theta_j^{(i)} + \tilde{b}_j^{(i)} (\sin^2 \theta_j^{(i)} - \cos^2 \theta_j^{(i)}),$$

$$\tilde{b}_{j+1}^{(i)} = -b_{j+1}^{(i)} \cos \theta_j^{(i)},$$

$$d_j^{(i)} = b_{j+1}^{(i)} \sin \theta_j^{(i)},$$

$$z_j^{(i+1)} = \bar{z}_j^{(i)} \cos \theta_j + z_{j+1}^{(i)} \sin \theta_j,$$

$$\bar{z}_{j+1}^{(i)} = \bar{z}_j^{(i)} \sin \theta_j^{(i)} - z_{j+1}^{(i)} \cos \theta_j^{(i)},$$

with

$$d_0^{(i)} = b_1^{(i)}, \quad \bar{b}_0^{(i)} = (a_1^{(i)} - \lambda^{(i)}).$$

Initially

$$z_1^{(0)} = 1, \quad z_j^{(0)} = 0 \quad \text{for} \quad j = 2, \ldots, N$$

so that

$$\underset{\sim}{z}^{(i)T} \to \underset{\sim}{q}^T \quad \text{as} \quad i \to \infty .$$

In the actual computation, no additional storage is required for

$$\{\bar{a}_j^{(i)}, \bar{b}_j^{(i)}, \tilde{b}_j^{(i)}, \bar{z}_j^{(i)}\}$$

since they may overwrite

$$\{a_j^{(i)}, b_j^{(i)}, z_j^{(i)}\} .$$

We choose $\lambda^{(i)}$ as an approximation to an eigenvalue; usually, it is related to the eigenvalues of the matrix

$$\begin{vmatrix} a_{N-1}^{(i)} & b_{N-1}^{(i)} \\ b_{N-1}^{(i)} & a_{N}^{(i)} \end{vmatrix} .$$

When $b_{N-1}^{(i)}$ is sufficiently small, $a_N^{(i)}$ is taken as eigenvalue and N is replaced by N-1 .

4. Determining the Three Term Relationship from the Moments

For many weight functions, the three term recursion relationship of the orthogonal polynomials have been determined. In some situations, however, the weight function is not known explicitly but one has a set of 2N+1 moments, viz.

$$\mu_k = \int_a^b \omega(x) x^k dx \qquad k = 0, 1, \ldots . 2N .$$

Let

$$M = \begin{bmatrix} \mu_0, \mu_1, \cdots, \mu_N \\ \mu_1, \cdots, \mu_{N+1} \\ \vdots \\ \mu_N, \cdots, \mu_{2N} \end{bmatrix} ,$$

$$D_j = \det \begin{bmatrix} \mu_0, \mu_2, \cdots, \mu_j \\ \mu_1, \mu_2, \cdots, \mu_{j+1} \\ \cdot \\ \mu_j, \mu_{j+1}, \cdots, \mu_{2j} \end{bmatrix} , \qquad j = 0, 1, r.., N-1,$$

and

$$F_j = \det \begin{bmatrix} \mu_0, \mu_1, \cdots \mu_{j-1}, & \mu_{j+1} \\ \mu_1, \mu_2, \cdots, \mu_j, & \mu_{j+2} \\ \vdots & \vdots \\ \mu_j, \mu_{j+1}, \cdots, \mu_{2j-1}, & \mu_{2j+1} \end{bmatrix}, \qquad j = 1, 2, \ldots . N-1 .$$

It is shown in [1] that

$$xp_j(x) = \beta_{j-1}p_{j-1}(x) + \alpha_j p_j(x) + \beta_j p_{j+1}(x) \qquad (4.1)$$

$$\text{for } j = 1, 2, \ldots,$$

where

$$a_j = \left(\frac{F_{j-1}}{D_{j-1}} - \frac{F_{j-2}}{D_{j-2}}, \qquad F_{-1} = 0, \; F_0 = \mu_1\right) \qquad j = 1, 2, \ldots . N$$

$$\beta_j = \frac{\sqrt{D_{j-2}D_j}}{D_{j-1}}, \quad (D_{-1} = 1, \; D_0 = \mu_0) \qquad j = 1, 2, \ldots . N-1 .$$

Note that the tri-diagonal matrix so generated is symmetric.

In [7], Rutishauser gives explicit formulas in terms of determinants for the Gauss-Crout decomposition of a matrix. We may use these relationships to evalute the coefficients

$$\{a_j\}_{j=1}^{N}, \quad \{\beta_j\}_{j=1}^{N-1} .$$

Let R denote the Cholesky decomposition of M so that

$$M = R^T R$$

and R is an upper triangular matrix whose elements are

$$r_{ii} = \left(m_{ii} - \sum_{i=1}^{i-1} r_{ki}^2\right)^{\frac{1}{2}}$$

and

$$r_{ij} = \left(m_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}\right)/r_{ii}, \qquad i < j, \tag{4.2}$$

for i and j between 1 and n .

Then, from the formulas of Rutishauser

$$\left.\begin{aligned} \alpha_j &= \frac{r_{j,j+1}}{r_{j,j}} - \frac{r_{j-1,j}}{r_{j-1,j-1}} && j = 1, 2, \ldots . N \\ \beta_j &= \frac{r_{j+1,j+1}}{r_{j,j}} && j = 1, 2, \ldots . N-1 \end{aligned}\right\} \tag{4.3}$$

with $r_{0'0} = 1$, $r_{0,1} = 0$.

There are other means for evaluating $\{\alpha_j\}, \{\beta_j\}$ but it is the opinion of the authors that the above method will lead to the most accurate formulas.

5. Description of Computational Procedures

In the following section there are three ALGOL 60 procedures for performing the algorithms presented above. We have tried to keep the identifiers as close to the notation of the equations as possible without

sacrificing storage or efficiency. The weights and abscissas of the quadrature rule are the result of the procedure GAUSSQUADRULE which must be supplied with the recurrence relation by either procedure GENORTHOPOLY or procedure CLASSICORTHOPOLY . The former requires the moments of the weight function and the latter the name of the particular orthogonal polynomial. A short description of each procedure follows.,

CLASSICORTHOPOLY produces μ_0 and the three term recurrence relationship (a_j, b_j, c_j) for six well-known kinds of orthogonal polynomials:

KIND = 1, Legendre Polynomials $P_n(x)$ on [-1.0, +1.0], $\omega(x) = 1.0$.

KIND = 2, Chebyshev Polynomials of the first kind $T_n(x)$ on [-1.0, +1.0], $\omega(x) = (1 - x^2)^{-\frac{1}{2}}$.

KIND = 3, Chebyshev Polynomials of the second kind $U_n(x)$ on c-1.0, +1.0], $w(x) = (1 - x^2)^{+\frac{1}{2}}$.

KIND = 4, Jacobi Polynomials $P_n^{(\alpha,\beta)}(x)$ on [-1.0, +1.0], $\omega(x) = (1-x)^{\alpha}(1+x)^{\beta}$ for $\alpha > -1$ and $\beta > -1$.

KIND = 5, Laguerre Polynomials $L_n^{(\alpha)}(x)$ on [0, +∞], $w(x) = e^{-x}x^{\alpha}$ for $\alpha > -1$.

KIND = 6, Hermite Polynomials $H_n(x)$ on [-∞, +∞], $\omega(x) = e^{-x^2}$.

Notice that this procedure requires a real procedure to evaluate the gamma function $\Gamma(x)$.

GENORTHOPOLY uses the 2N+1 moments of the weight function which are supplied in MU[0] through MU[2⊗N] to compute the α_j's and β_j's of formula (4.1). First, The Cholesky decomposition (formula 4.2) of the moment matrix is placed in the upper right triangular part of the array R, then the formulas (4.3) are used to compute the α_j's and β_j's which are placed in the arrays A and B respectively.

GAUSSQUADRULE has two modes of operation controlled by the Boolean parameter SYMM which indicates whether the tri-diagonal matrix is symmetric or not. When the recursion relation is produced by GENORTHOPOLY, SYMM is true; when produced by CLASSICORTHOPOLY, SYMM is false. If SYMM is false, the matrix is symmetricized using the formulas (2.2). The diagonal elements α_i are stored in A[I] and the off diagonal elements β_i are stored in B[I] .

Beginning at label SETUP, several calculations and initializations are done: the ℓ_1 norm of the tri-diagonal matrix and the relative zero tolerance are computed; the first component of each eigenvector W[I] and the Q-R iteration are initialized. LAMBDA is a variable subtracted off the diagonal elements to accelerate convergence of the Q-R iteration and control to some extent in what order the eigenvalues (abscissas) are found. It begins with a value outside and to the right of the interval containing the abscissas (=NORM) and moves to the left as the abscissas are found; thus the abscissas will be in ascending order in the array T (just to be sure an exchange sort is used at label SØRT).

The maximum (EIGMAX) of the eigenvalues (LAMBDA1 and LAMBDA2) of the lower 2 × 2 submatrix is compared to the maximum (RHO) from the last iteration. If they are close, LAMBDA is replaced by EIGMAX .

This scheme seems to stabalize LAMBDA and speed convergence immediately after deflation.

An eigenvalue has been found when the last off diagonal element falls below EPS (see section 7). Tts value is placed in T[I] and the corresponding weight W[I] is computed from formula (2.5). This convergence test and the test for matrix splitting are done following label INSPECT. Only the Lower block (from K to M) needs to be transformed by the Q-R equation given in formulas (3.3) .

6. The ALGOL 60 Procedures

```
procedure classicorthopoly(kind,  alfa,  beta, n, a, b, c, muzero);
     value kind, n, alfa, beta;
     integer kind, n;  real alfa, beta, muzero;
     real array a, b, c;
begin comment   This procedure supplies the coefficients of the three term
          recurrence relationship for various classical orthogonal polynomials. ;
     integer i;  real abl, pi;
     switch swt := legendre, chebyshev1, chebyshev2, jacobi,
          laguerre, hermite;
     pi := 3.14159265359;
     go to swt[kind];
legendre: muzero := 2.0;
     comment  P(x)  in [-1, 1], w(x) = 1.0  ;
     for i:=1 step 1 until n do
          begin a[i] := (2×i-1)/i; b[i]:= 0;  c[i]:= (i-1)/i  end;
     go to return;
chebyshev1:  muzero := pi ;
     comment T (x) in [-1, 1], w(x) = (1-x↑2)↑(-.5) ;
     for i:=1 step 1 until. n do
          begin a[i] := 2; b[i] := 0; c[i]:= 1 end;
    a[i] := 1;    go to return;
chebyshev2:  muzero  := pi/2.0;
     comment u(x) in [-1, 1], W(x) = (1-x↑2)↑.5;
     for i:=1 step 1 until n do
          begin a[i] := 2; b[i] :=0; c[i] := 1 end;
     go to return;
```

```
jacobi:  muzero:=2↑(alfa+beta+1)×gamma(alfa+1)×gamma(beta+1)
            /gamma(alfa+beta+2) ;
     comment  P (alfa,beta)(x) in [-1, 1], w(x) = (1-x)↑alfa×(1+x)↑beta
          alfa > -1 and beta > -1 ;
     a[i]:= 0.5×(alfa+beta+2); b[i]:= 0.5×(alfa-beta);
     for i:=2 step 1 until n do

     begin  ab1:= 2×i×(i+alfa+beta);
          a[i]:= (2×i+alfa+beta-1)×(2×i+alfa+beta)/ab1;
          ab1:= (2×i+alfa+beta-2)×ab1;
          b[i]:= (2×i+alfa+beta-1)×(alfa↑2-beta↑2)/ab1;
          c[i]:= 2×(i-1+alfa)×(i-1+beta)×(2×i+alfa+beta)/ab1;
     end;
     go to return;
laguerre: muzero:= gamma(alfa+1.0) ;
     comment L(alfa)(x) in [0, infinity), w(x) = exp(-x)×x↑alfa,
          alfa > -1;
     for i:=1 step 1 until n do
     begin a[i]:= -1/i;        := (2×i-1+alfa)/i;
          c[i]  := (i-1+alfa)/i;
     end;
     go to return ;
hermite: muzero := sqrt(pi);
     comment  H(x)  in (-infinity,+infinity),  w(x) = exp(-x↑2) ;
     for i:=1 step 1 until n do
          begin a[i]:= 2;  b[i]  := 0;  c[i]:= 2×(i-1) end;
return:  end  classicorthopoly ;
```

```
procedure  genorthopoly(n, mu, a, b);
     value n;  integer n;
     real array mu, a, b;
begin comment    Given the 2n+1 moments of the weight function,
          generate the recursion coefficients of the orthogonal
          polynomials.  ;
     real array   r[0:n+1,0:n+1]; real sum ;
     integer i, j, k;
comment     Place the Cholesky decomposition of the moment matrix in r[];
     for i:=1 step 1 until n+1 do
     for j:=i step 1 until n+1 do
    . begin sum:= mu[i+j-2] ;
          for k:=i-1 step -1 until 1 do
             sum:= sum-r[k,i]xr[k,j];
          r[i,j]:= (if i=j en sqrt(sum) else sum/r[i,i]);
     end;
comment   Compute the recursion coefficients'from the decomposition r[];
     r[0,0] := 1.0; r[0,1] := 0;
     for i:=0 step 1 until n+1 do
     a[n] := r[n,n+1]/r[n,n]-r[n-1,n]/r[n-1,n-1] ;
     for j:=n-1 step -1 until 1 do
     begin b[j] := r[ j+1,j+1]/r[j,j];
          a[j] := r[j,j+1]/r[j,j]-r[ j-1, j]/r[ j-1,j-1];
     end ;
end' genorthopoly ;
```

```
procedure gaussquadrule(n, a, b, c, muzero, symm, t, w);
    value n, muzero, symm;
    integer n; real muzero; boolean symm;
    real array a, b, c, t, w;
begin comment  Given the coefficients a, b, c of the three term recurrence
        relation : p[k+1] = (a[k+1]x+b[k+1])p[k]-c[k+1]p[k-1], this procedure
        computes the abscissas t and the weights w of the gaussian type
        quadrature rule associated with the orthogonal polynomial by QR
        type iteration with origin shifting;
    integer i, j, k, m, ml;
    real norm, eps, ct, st, ct2, st2, sc, aa, ai, aj, a2, eigmax, lambda,
        lambda1, lambda2, rho, r, det, bi, bj, b2, wj, cj;
    boolean ex;
    real procedure max(x,y); value x, y;  real x, y;
        max := if x≥y then x else y;
    if symm then go to setup;
comment  Symmetrize the matrix, if required.;
        for i:=1 step 1 until n-1 do
        begin ai := a[i];  a[i] := -b[i]/ai;
            b[i] := sqrt(c[i+1]/(ai×a[i+1]));
        end;
        a[n] := -b[n]/a[n];
comment  Find the maximum row sum norm and initialize w[I;
setup :  b[0] := 0; norm := 0;
    for i:=1 step 1 until n-1  do
    begin norm := max(norm, abs(a[i])+abs(b[i-1]) +abs(b[i-1]));
        w[i] := 0;
```

```
      end;
      norm := max(norm, abs(a[n])+abs(b[n-1]));
      w[1] := 1.0; w[n] := 0; m := n;
      eps:=norm×1.0 ×8↑(-13); comment  Relative zero tolerance:
      lambda := lambda1 := lambda2 := rho := norm;
comment   Look for convergence of lower diagonal element;
inspect:  if m=0 then go to sort else i := k := m1 := m-1;
      if abs(b[m1])≤eps then
           begin t[m] := a[m]; w[m] := muzero×w[m]↑ 2;
                rho := (if lambda1<lambda2 then lambda1 else lambda2);
                m := m1; go to inspect;
           end;
comment   Small of diagonal element means matrix can be split;
      for i:=i-1 while abs(b[i])>eps do k := i;
comment   Find eigenvalues of lower 2x2 and select accelerating shift;
      b2 := b[m1]↑2;   det := sqrt((a[mi]-a[m])↑2+4.0×b2) ;
       aa := a[m1]+a[m];
      lambda2 := 0.5×(if aa>0 then aa +det else aa-det);
      lambda1 := (a[m1]×a[m]-b2)/lambda2;
      eigmax := max(lambda1,lambda2);
      if abs(eigmax-rho)≤0.125×abs(eigmax) then lambda := rho := eigmax
           else rho := eigmax;
comment   Transform block from k to m;
      cj := b[k];  b[k-1] := a[k]-lambda;
      for j:=k step 1 until m1 do
```

```
      begin r := sqrt(cj↑2+b[j-1]↑2) ;
            st := cj/r;  st2 := st↑2;
            ct := b[j-1]/r; ct2 := ct↑2;
            sc := st×ct; aj := a[j];
            bj := b[j]; wj := w[j];
            a[j] := aj×ct2+2.0×bj×sc+a[j+1]×st2;
            b[j] := (aj-a[j+1])×sc+bj×(st2-ct2);
            a[j+1] := aj×st2-2.0×bj×sc+a[j+1]×ct2;
            cj := b[j+1]×st; b[j+1] := -b[j+1]×ct; b[j-1]:=r
            w[j] := wj×ct+w[j+1]×st; w[j+1] := wj×st-w[j+1]×ct;
      end;
      b[k-1] := 0 go to inspect;
comment   Arrange abscissas in ascending order;
sort:  for m:=n step -1 until 2 do
       begin ex := false;
          for i:=2 step 1 until m do
             if t[i-1]>t[i] then
                begin r:= t[i-1]; t[i-1] := t[ i];
                   t[i]:= r;  r:=w[i-1];
                   w[i-1] := w[i]; w[i] :=r;
                   ex:= true ;
                end ;
             if ¬ ex then go to return ;
          end;
return : end gaussquadrule ;
```

7. Test Program and Results

The procedures in section 6 have been extensively tested in Burroughs B5500 Algol and IBM OS/360 Algol. There are two machine dependent items which must be mentioned. First, the constant used to define the "relative zero tolerance" EPS in procedure GUASSQUADRULE is dependent on the length of the fraction part of the floating-point number representation ($=8^{-13}$ for the 13 octal digit fraction on the B5500, and $= 16^{-14}$ for a 14 hexadecimal digit long-precision fraction on the IBM 360). Second, the moment matrix M defined in section 4 usually becomes increasingly ill conditioned with increasing N . Thus the round-off errors generated during Cholesky decomposition in GENORTHOPOLY cause an ill conditioned M to appear no longer positive definite and the procedure fails on taking the square root of a negative number.

The procedure GAUSSQUADRULE proves to be quite stable and when the recursion coefficients are known or supplied by the procedure CLASSICORTHOPOLY it loses only several digits off of full-word accuracy even for N = 50 . Procedure GENORTHOPOLY usually failed to produce the recursion coefficients from the moments when N was about 20 for the IBM 360.

The test program given below is designed to compare the two methods of generating the quadrature rules--from the moments or the recursion coefficients. N can be increased until GENORTHOPOLY fails. Numerical results may be checked against tables for Gauss-Legendre quadrature in [9] and Gauss-Laguerre quadrature in [2]. In the Table, we compare the abscissas and weights of the Gauss-Laguerre quadrature rule with $\alpha = -0.75$ and N = 10 computed by (1) the analytic recurrence

relationship and the Q-R algorithm; (2) the moment matrix and the Q-R algorithm; (3) Concus et. al. [2]. The calculations for (1) and (2) were performed on the IBM 360.

```
begin
comment   Driver program for gaussrule;
     real array a, b, c, mu, t, w[0:10];
     real muzero;  integer i, n;
       n := 10;
comment   Legendre polynomials. ;
          outstring (1, 6 legendre quadrature.9 );
          classicorthopoly(1, 0, 0, n, a, b, c, muzero);
          gaussquadrule(n, a, b, c, muzero, false, t, w);
    . outstring(1, 6 abscissas: ' ); outarray (1,t);
     outstring(1, 6 weights:' ); outarray(1,w);
     for i:=0 step 1 until 2×n do mu[i] := 0;
     for i:=0 step 2 until 2×n do mu[i] := 2.0/(i+1);
          genorthopoly(n, mu, a, b);
          muzero :=  mu[0];
          gaussquadrule(n, a, b, c, muzero, true, t, w);
     outstring (1,6abscissas:' );  outarray (1, t);
     outstring(1, 6 weights:' ); outarray(1, w);
comment    Laguerre polynomials. ;
          outstring(1, 6laguerre quadrature.  alpha =-0.5' );
          classicorthopoly(5, -0.5, 0, n, a, b, c, muzero);
          gaussquadrule(n, a, b, c, muzero, false, t, w);
     outstring (1, 6 abscissas :' );  outarray (1, t);
     outstring (1, 6 weights: ' ); outarray (1, w);
     mu[0] := muzero := 1.7724538509 ; comment gamma (0.5);
     for i:=1 step 1 until 2×n do
          mu[i] := (i-0.5)×mu[i-1];
          genorthopoly(n, mu, a, b);
          gaussquadrule(n, a, b, c, muzero, true, t, w);
```

```
    outstring(1, ' abscissas:' ); outarray (1, t);
    outstring(1, ' weights:' ); outarray (1, w);
end;
```

TABLE

A Comparison of the Abscissas and Weights
of the Gauss-Laguerre Quadrature Rule
with $\alpha = -0.75$ and $N = 10$

	Analytic Recurrence Relationship + QR	Moment Matrix + QR	Concus et al [2].
	ABSCISSAS		
1	$2.766655867080153_{10}-2$	$2.766655862878470_{10}-2$	$2.76665586707972_{10}-2$
2	$4.547844226059642_{10}-1$	$4.547844219368714_{10}-1$	$4.5478442260594_{10}-1$
3	1.382425761158619_{10}	1.382425759256314_{10}	1.382425761158599_{10}
4	2.833980012092734_{10}	2.833980008561162_{10}	2.833980012092697_{10}
5	4.850971448764968_{10}	4.850971443442301_{10}	4.850971448764914_{10}
6	7.500010942642896_{10}	7.500010935563904_{10}	7.500010942642825_{10}
7	$1.088840802383446_{10}+1$	$1.088840801516104_{10}+1$	$1.088840802383404_{10}+1$
8	$1.519947804423765_{10}+1$	$1.519947803419274_{10}+1$	$1.519947804423760_{10}+1$
9	$2.078921462107018_{10}+1$	$2.078921460989977_{10}+1$	$2.078921462107010_{10}+1$
10	$2.857306016492223_{10}+1$	$2.857306015294401_{10}+1$	$2.857306016492210_{10}+1$
	WEIGHTS		
1	2.566765557790853_{10}	2.566765556932285_{10}	2.566765557790772_{10}
2	$7.733479703443168_{10}-1$	$7.733479706154000_{10}-1$	$7.733479703443410_{10}-1$
3	$2.331328349732182_{10}-1$	$2.331328353678223_{10}-1$	$2.331328349732190_{10}-1$
4	$4.643674708956677_{10}-2$	$4.643674724992909_{10}-2$	$4.643674708956700_{10}-2$
5	$5.549123502036256_{10}-3$	$5.549123531829512_{10}-3$	$5.549123502036250_{10}-3$
6	$3.656466626776441_{10}-4$	$3.656466653186007_{10}-4$	$3.656466626776380_{10}-4$
7	$1.186879857102525_{10}-5$	$1.186879867642139_{10}-5$	$1.186879857102450_{10}-5$
8	$1.584410942056844_{10}-7$	$1.584410958350144_{10}-7$	$1.584410942056780_{10}-7$
9	$6.193266726796867_{10}-10$	$6.193266797518338_{10}-10$	$6.193266726796840_{10}-10$
10	$3.037759926517691_{10}-13$	$3.037759963698451_{10}-13$	$3.037759926517500_{10}-13$

(Underlined figures are those which disagree with Concus et al [2].)

8. References

[1] N. Ahiezer and M. Krein, Some Questions in the Theory of Moments, Translations of Mathematical Monograph, Vol. 2, American Mathematical Society, Providence, 1962, Article VI.

[2] P. Concus, D. Cassatt, G. Jaehnig and E. Melky, "Tables for the evaluation of

$$\int_0^\infty x^\beta e^{-x} f(x)dx$$

by Gauss-Laguerre Quadrature," Math, Comp., 17(1963), pp. 245-256.

[3] P. Davis and I. Polonsky, "Numerical interpolation, differentiation, and integration," Handbook of Mathematical Functions, (edited by M. Abramowitz and I. Stegun), National Bureau of Standards, Applied Mathematics Series 55, U. S. Government Printing Office, Washington, D. C., pp. 875-924.

[4] P. Davis and P. Rabinowitz, Numerical Integration, Blaisdell Publishing Company, Waltham, Mass., 1967.

[5] J. Francis, "The Q-R transformation. A unitary analogue to the L-R transformation," Comput. J., 4 (1961, 1962), pp. 265-271, 332-345.

[6] U. Hochstrasser, "Orthogonal polynomials, "Handbook of Mathematical Functions, (edited by M. Abramowitz and I. Stegun), National Bureau of Standards, Applied Mathematics Series 55, U. S. Government Printing Office, Washington, D. C., pp.771-802.

[7] H. Rutishauser, "Solution of eigenvalue problems with the L-R transformation," Further Contributions to the Solution of Simultaneous Linear Equations and the Determination of Eigenvalues, National Bureau of Standards, Applied Mathematics Series 49, U. S. Government Printing Office, Washington, D. C., pp. 47-81.

[8] H. Rutishauser, "On a modification of the Q-D algorithm with Graeffe-type convergence," ZAMP, 13 (1963), pp. 493-496.

[9] A. Stroud and D. Secrest, Gaussian Quadrature Formulas, Prentice-Hall, Englewood Cliffs, N. J., 1966.

[10] H. Wilf, Mathematics for the Physical Sciences, John Wiley and Sons, Inc., New York, 1962.