Interval Computations
No 4, 1993

Verified Calculation of the Nodes and Weights for Gaussian Quadrature Formulas

Ulrike Storck

Using Gaussian Quadrature for the verified calculation of an integral assumes that enclosures of the nodes and weights are given. The traditional method of computing the nodes and weights is ill-conditioned and cannot be performed with interval arithmetic. Since the nodes and weights can be determined by use of orthonormal polynomials, an algorithm is presented which first calculates approximations and then computes enclosures of the orthonormal polynomials.

Верифицированное вычисление узлов и весов для квадратурных формул Гаусса

У. Шторк

Применение гауссовых квадратур для верифицированного вычисления интеграла предполагает известными оболочки узлов и весов. Традиционный метод вычисления узлов и весов плохо обусловлен и не может быть реализован при помощи интервальной арифметики. Поскольку узлы и веса можно определить, используя ортонормальные полиномы, представлен алгоритм, который вначале рассчитывает приближения, а затем вычисляет оболочки для ортонормальных полиномов.

1 Introduction

Our aim is to determine a verified enclosure of the integral

$$J := \int_a^b w(x)f(x)dx$$

with a weight function $w(x)$ which fulfills the assumptions for Gaussian quadrature formulas and with a smooth function $f(x)$. We use Gaussian Quadrature for which the following holds (see [8]):

Theorem 1. *Let $f(x) \in C^{2n}[a,b]$. Then there exist x_i and w_i for $i = 1,\ldots,n$ and k_n with*

$$\int_a^b w(x)f(x)dx = \sum_{i=1}^{n} w_i f(x_i) + \frac{f^{(2n)}(\xi)}{(2n)!k_n^2}$$

with $\xi \in [a,b]$. The nodes x_i, the weights w_i, and the error coefficient k_n are determined by the weight function $w(x)$ and the integration bounds a and b.

This theorem leads to an enclosure algorithm which requires interval arithmetic (see [3]), automatic differentiation (see [6]) and enclosures of the nodes x_i, the weights w_i, and the error coefficient k_n:

$$J \in \diamondsuit J = \diamondsuit\left(\sum_{i=1}^{n} \diamondsuit w_i \diamondsuit f(\diamondsuit x_i) + \frac{\diamondsuit f^{(2n)}([a,b])}{(2n)!\diamondsuit\diamondsuit k_n^2}\right).$$

Here '$\diamondsuit$' symbolizes enclosures and interval arithmetic. Since we use the optimal scalar product (see [3]), a single rounding symbol $\diamondsuit$ is used in front of the summation symbol.

In order to realize this enclosure algorithm, we have to find tight enclosures of the nodes, of the weights, and of the error coefficient. This is the subject of this paper.

2 The traditional method

The traditional method for the determination of the values we need is described in [8] and is sketched only here. By defining the scalar product

$$(f,g) := \int_a^b w(x) f(x) g(x)\, dx$$

on the real vector space $C[a,b]$ of continuous functions on $[a,b]$, we obtain

Theorem 2. *There exist uniquely determined, orthogonal polynomials $\tilde{p}_j$ of degree j with*

$$(\tilde{p}_i, \tilde{p}_k) = 0 \quad \textit{for } i \neq k.$$

They satisfy the recursive relation

$$\begin{aligned} &\tilde{p}_0(x) := 1,\ \tilde{p}_{-1}(x) = 0, \\ &\tilde{p}_{i+1}(x) := (x - \tilde{\alpha}_{i+1})\tilde{p}_i(x) - \tilde{\beta}^2_{i+1}\tilde{p}_{i-1}(x) \quad \textit{for } i \geq 0 \end{aligned} \tag{1}$$

with

$$\begin{aligned} &\tilde{\alpha}_{i+1} := (x\tilde{p}_i, \tilde{p}_i)/(\tilde{p}_i, \tilde{p}_i) \quad \textit{for } i \geq 0 \\ &\tilde{\beta}^2_{i+1} := \begin{cases} 0 & \textit{for } i = 0 \\ (\tilde{p}_i, \tilde{p}_i)/(\tilde{p}_{i-1}, \tilde{p}_{i-1}) & \textit{for } i \geq 1. \end{cases} \end{aligned} \tag{2}$$

The polynomials $\tilde{p}_i$ are the orthogonal polynomials corresponding to the weight function $w(x)$. If we arrange the coefficients $\tilde{\alpha}_i$ and $\tilde{\beta}_i$ in a tridiagonal matrix $\widetilde{A}$ in the following way

$$\widetilde{A} = \begin{pmatrix} \tilde{\alpha}_1 & \tilde{\beta}_2 & & \\ \tilde{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \tilde{\beta}_n \\ & & \tilde{\beta}_n & \tilde{\alpha}_n \end{pmatrix}$$

the eigenvalues of $\widetilde{A}$ are equal to the nodes x_i.

We obtain the weights from the eigenvectors $v^{(i)}$, $i = 1, \ldots, n$ via

$$w_i = (v_1^{(i)})^2$$

with $v^{(i)} := (v_1^{(i)}, \ldots, v_n^{(i)})^T$ which is normed by $(v^{(i)})^T v^{(i)} = (p_0, p_0) = \int_a^b w(x)\,dx$.

Finally, the error coefficient is given by

$$1/k_n^2 = \big(\tilde{p}_n(x), \tilde{p}_n(x)\big).$$

The determination of $\tilde{\alpha}_i$ and $\tilde{\beta}_i$ by (1) and (2) leads to an ill-conditioned problem (see [1]). The performance of (1) and (2) with interval arithmetic (even with multiple precision) results in rapid growth in overestimation of the enclosures and unacceptable results.

3 The new algorithm

Our new algorithm is based on the determination of the orthonormal polynomials corresponding to the weight function. By using the coefficients of these orthonormal polynomials $p_i(x)$, we can compute the coefficients α_i, β_i, and k_n.

If we employ interval arithmetic, tight enclosures of the coefficients of p_i must be obtained. We first assume that tight enclosures of the moments $\mu_i = \int_a^b x^i w(x)\,dx$ for $i = 0, \ldots, 2n$ are given. They enable us to calculate enclosures of the scalar products (p_j, p_k) with $0 \le j, k \le n$, since the polynomials are linear combinations of the monomials. Then, by a modification of the orthogonalization algorithm of Gram-Schmidt, tight enclosures of the orthonormal polynomials are obtained. The Gram-Schmidt algorithm is presented here:

for $(n+1)$ linearly independent polynomials q_i, $i = 0, \ldots, n$, the algorithm:

$$\begin{aligned} \textit{for} \quad & i = 0, \ \ldots, \ n \\ & p_i = q_i - \sum_{j=0}^{i-1} (q_i, p_j) p_j \qquad (3) \\ & p_i = p_i / \sqrt{(p_i, p_i)} \qquad (4) \end{aligned}$$

leads to $(n+1)$ orthonormal polynomials p_i.

If we use floating-point operations in this algorithm, we will not get orthonormal polynomials in general. On the other hand, if we use interval arithmetic the results will soon become unacceptably large overestimations. In order to avoid this, we introduce two iteration procedures. In the first iteration procedure, Step (3) followed by the substitution $q_i := p_i$ is iterated several times and performed with multiple precision floating-point operations. Therefore, we obtain good approximations of the orthogonal polynomials. Then, in the second iteration procedure, each iteration step consists of (3), (4), and the substitution $\diamondsuit q_i := \diamondsuit p_i$ or $\diamondsuit q_i := \diamondsuit p_i \cap \diamondsuit q_i$, and is executed with multiple interval precision.

After having determined the enclosures of the orthonormal polynomials, we calculate enclosures of α_i and β_i by (see [1])

$$\begin{aligned} p_{-1}(x) &= 0, \text{ and} \\ \beta_{i+2}p_{i+1}(x) &= (x-\alpha_{i+1})p_i(x) - \beta_{i+1}p_{i-1}(x). \end{aligned}$$

The $\diamondsuit\alpha_i$ and $\diamondsuit\beta_i$ we obtained have to be arranged in a matrix

$$A = \begin{pmatrix} \diamondsuit\alpha_1 & \diamondsuit\beta_2 & & \\ \diamondsuit\beta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \diamondsuit\beta_n \\ & & \diamondsuit\beta_n & \diamondsuit\alpha_n \end{pmatrix}$$

whose eigenvalues and eigenvectors yield the nodes and weights as in the traditional method. The error coefficient k_n is equal to the coefficient of x^n in $p_n(x)$ (see [9]).

Thus, we get an algorithm which is presented here in a simplified form. We only assume tight enclosures of the moments $\diamondsuit\mu_k$ that are required for the calculation of the scalar products $\diamondsuit(\diamondsuit p_j, \diamondsuit p_i)$.

Enclosure algorithm for $\diamondsuit x_i$, $\diamondsuit w_i$, $\diamondsuit k_n$:

1) input($\diamondsuit\mu_i$, $i = 0, \ldots, 2n$)

2) $p_0 = 1/\sqrt{(1,1)} = (\mu_0)^{-1/2}$
   for i =1, $\ldots$, n do
   begin
   $p_i = 0$, $p_{i,i} = 1$

repeat (floating-point iteration)
$p_{i\_old} = p_i$
$p_i = p_i - \sum_{j=0}^{i-1}(p_j, p_i)p_j$
until $(|p_i - p_{i\_old}| < eps)$
$\diamondsuit p_i = \diamondsuit\Big(p_i - \sum_{j=0}^{i-1} \diamondsuit(\diamondsuit p_j, p_i)\diamondsuit p_j\Big) \diamondsuit\!\!\diamondsuit \sqrt{(p_i, p_i)}$
repeat (interval iteration)
$\diamondsuit p_{i\_old} = \diamondsuit p_i$
$\diamondsuit p_i = \diamondsuit\Big(\diamondsuit p_i - \sum_{j=0}^{i-1} \diamondsuit(\diamondsuit p_j, \diamondsuit p_i)\diamondsuit p_j\Big) \cap \diamondsuit p_i$
$\diamondsuit p_i = \diamondsuit p_i \diamondsuit\!\!\diamondsuit \sqrt{(\diamondsuit p_i, \diamondsuit p_i)}$
until $(diam(\diamondsuit p_i) < eps)$ or $(\diamondsuit p_{i\_old} = \diamondsuit p_i)$
determine $\diamondsuit\alpha_i$, $\diamondsuit\beta_i$ by use of the orthonormal polynomials:
$\diamondsuit\beta_{i+1} = \diamondsuit p_{i-1,i-1} \diamondsuit\!\!\diamondsuit p_{i,i}$,
$\diamondsuit\alpha_i = (\diamondsuit p_{i-1,i-2} \diamondsuit\!\!\diamondsuit \beta_{i+1} \diamondsuit p_{i,i-1}) \diamondsuit\!\!\diamondsuit p_{i-1,i-1}$
end

3) determine enclosures of the eigenvalues and eigenvectors of the corresponding tridiagonal matrix A

4) determine $\diamondsuit x_i$, $\diamondsuit w_i$, $i = 1, \ldots, n$ and $\diamondsuit k_n = \diamondsuit p_{n,n}$

5) output($\diamondsuit x_i$, $\diamondsuit w_i$, $\diamondsuit k_n$)

Some implementation hints:

- The polynomials p_i with degree i are represented as one-dimensional arrays: $p_i(x) = p_{i,0} + p_{i,1} \cdot x + p_{i,2} \cdot x^2 + \cdots + p_{i,i} \cdot x^i$.

- The linearly independent polynomials given at the beginning of the algorithm are the monomials $p_i = x^i$, except for p_0 which is calculated directly. This guarantees that the space spanned by these polynomials does not change during floating-point iteration. (This would happen otherwise.)

- The optimal scalar product (see [3]) and multiple precision (interval) arithmetic (see [5]) are needed. Therefore the algorithm was implemented in Pascal-XSC (see [2]).

- We use an algorithm to determine enclosures of the eigenvalues and eigenvectors of a tridiagonal matrix. This algorithm consists of the two procedures developed in [4, 7]. The first one, implemented in single precision, delivers good approximations of the eigenvalues and eigenvectors. These approximations serve as inputs for the second procedure (see [7]), which was implemented with multiple interval precision and leads to results with high accuracy.

The precision of the computed enclosures of the weights and nodes depends on the precision of the algorithm, on the degree n, and on the Gaussian Quadrature formula. In order to obtain enclosures for the degree $n = 2, \ldots, m$ which are consistent with an error bound given by the user, we use an adaptive algorithm for the choice of the precision:

1) $n := 2$

2) While $n \leq m$ do

   (a) Determine $\diamondsuit x_i^{(n)}$, $\diamondsuit w_i^{(n)}$, $i = 1, \ldots, n$ by our enclosure algorithm. Determine

   $log\_max\_error[n]$:= log_{10} (maximal relative error of $\{\diamondsuit x_i^{(n)}, \diamondsuit w_i^{(n)} | i = 1, \ldots, n\}$)

   (b) With $log\_max\_error[j]$, $j = 2, \ldots, n$ calculate
   app := approximation of $log\_max\_error[m]$
   by performing the Neville algorithm (see [8]),

   (c) If $approx > log_{10}$(error bound)
   then mantissa length := mantissa length
   +(app$-log_{10}$(error bound))
   determine $\diamondsuit x_i^{(n)}$, $\diamondsuit w_i^{(n)}$, $i = 1, \ldots, n$,
   $log\_max\_old := log\_max\_err[n]$
   determine $log\_max\_error[n]$ analogously to (0a)
   $log\_max\_err[j]$:= $log\_max\_err[j]$ +
   $(log\_max\_old - log\_max\_err[n])$
   for $j = 2, \ldots, n-1$.

Some implementation hints:

- The precision in our algorithm is enlarged by increasing the mantissa length by at least of the difference (approx $-$ log_{10} (error bound)), assuming the mantissa length is expressed in base 10.
- The increasement of $log\_max\_err[j]$ in (0c) is necessary for further calculations of the approximations of $log\_max\_err[m]$ by the Neville algorithm.
- At the beginning of our algorithm, we choose for the mantissa length approximately the value $(m + constant)$ with $constant \approx 100$.

4 Some numerical results

We give numerical examples for two different weight functions. The following tables contain

- the degree, $\underline{i}$,
- the number of floating-point iterations, $\underline{n_{fl}}$,
- the number of interval iterations, $\underline{n_{it}}$,
- the maximal diameter of the enclosures of the scalar products $\diamondsuit(\diamondsuit p_i, \diamondsuit p_j)$, $j = 0, \ldots, i-1$; denoted by $\underline{\text{diam sc.}}$, and
- some weights and nodes, $\underline{w_j}$, $\underline{x_j}$ with $1 \leq j \leq i$.

Example 1. For Gauss-Legendre quadrature ($w(x) = 1$, $a = -1$, and $b = 1$), enclosures of the nodes and weights for $n = 2, \ldots, 20$ and with a maximal relative error of $1e-16$ were determined. For $n = 2$, 3, and 4, the algorithm was executed with approximately 128 decimal digits, and for $n = 5, \ldots, 20$ with approximately 350 decimal digits. We get the following information about the orthonormal polynomials:

i	4	5	10	15	20
n_{fl}	2	1	2	2	2
n_{it}	1	1	1	1	1
diam. sc.	8.2E-136	2.6E-311	9.8E-272	6.4E-207	2.4E-112

and the weights and nodes:

i	15	20
w_1	$3.07532419961172\binom{7}{6}6E-002$	$1.76140071391521\binom{2}{0}E-002$
w_2	$7.03660474881081\binom{4}{2}E-002$	$4.06014298003869\binom{6}{3}E-002$
w_3	$1.07159220467 17\binom{20}{19}E-001$	$6.26720483341090\binom{8}{5}E-002$
x_1	$-9.87992518020485\binom{3}{5}E-001$	$-9.9312859918509\binom{61}{47}E-001$
x_2	$-9.3727339240070\binom{58}{60}E-001$	$-9.63971927277913\binom{6}{9}E-001$
x_3	$-8.48206583410427\binom{2}{4}E-001$	$-9.1223442825132\binom{61}{58}E-001$

Example 2. For Gauss-Laguerre quadrature $\big(w(x) = e^{-x},\, a = 0,\, b = \infty\big)$, enclosures of the nodes and weights up to degree 10 were calculated with a maximal relative error of $1e-14$. At the beginning of the algorithm, the calculations were performed using a mantissa length of approximately 100 digits. For $n = 4$, the mantissa length was increased to 172, and for $n = 7$ to 224. We get the following information about the orthonormal polynomials:

i	3	6	8	10
n_{fl}	2	2	2	2
n_{it}	1	1	1	1
diam. sc.	2.6E-126	5.2E-168	8.6E-162	1.0E-111

and the weights and nodes:

i	8	10
w_1	$3.69188589341637\binom{6}{5}E-001$	$3.08441115765020\binom{2}{0}E-001$
w_{i-1}	$8.48574671627253\binom{3}{1}E-007$	$1.83956482397963\binom{1}{0}E-009$
w_i	$1.04800117487151\binom{1}{0}E-009$	$9.9118272196090\binom{11}{08}E-013$
x_1	$1.7027963230510\binom{11}{9}E-001$	$1.37793470540492\binom{5}{4}E-001$
x_{i-1}	$1.57406786412780\binom{1}{0}E+001$	$2.19965858119807\binom{7}{6}E+001$
x_i	$2.28631317368892\binom{7}{6}E+001$	$2.9920697012273\binom{90}{89}E+001$

However, numerical experiences have shown that a mantissa length with more than 350 digits does not yield more significant precision in the results.

5 Higher dimensions

The construction of orthonormal polynomials in higher dimensions is the same as in one dimension (assuming each polynomial is represented as a one-dimensional array, that is, every monomial has a corresponding array element for its coefficient).

References

[1] Gautschi, W. *On the construction of Gaussian quadrature rules from modified moments.* Math. Comp. **24** (1970), pp. 245–260.

[2] Klatte, R., Kulisch, U., Neaga, M., Ratz, D., and Ullrich, Ch. *Pascal-XSC.* Springer, Berlin, 1992.

[3] Kulisch, U. and Miranker, W. (eds) *Computer arithmetic in theory and practice.* Academic Press, New York, 1981.

[4] Lohner, R. *Enclosing all eigenvalues of symmetric matrices.* In: Ullrich, Ch. and Wolff von Gudenberg, J. (eds) "Accurate Numerical Algorithms", Research Reports ESPRIT, Springer, Berlin, 1989.

[5] Lohner, R. *Interval arithmetic in staggered correction format.* In: Adams, E. and Kulisch, U. (eds) "Scientific Computing with Automatic Result Verification", Academic Press, New York, 1993.

[6] Rall, L. B. *Automatic differentiation: techniques and applications.* Lecture Notes in Computer Science **120**, Springer, Berlin, 1981.

[7] Rump, S. M. *Solving algebraic problems with high accuracy.* In: Kulisch, U. and Miranker, W. (eds) "A New Approach to Scientific Computation", Academic Press, New York, 1983.

[8] Stoer, J. *Einführung in die Numerische Mathematik I.* Springer, Berlin, 1983.

[9] Stroud, A. H. *Approximative calculation of multiple integrals.* Prentice-Hall, Englewood Cliffs, N. J., 1971.

Institute for Applied Mathematics
University of Karlsruhe
Kaiserstr. 12
D-76128 Karlsruhe
Germany