An algorithm for numerical integration based on quasi-interpolating splines [☆]

C. Dagnino and V. Demichelis

Università di Torino, Dipartimento di Matematica, Via Carlo Alberto 10, 10123 Torino, Italy

E. Santi

Università di L'Aquila, Dipartimento di Energetica, 67100 L'Aquila, Italy

In this paper product quadratures based on quasi-interpolating splines are proposed for the numerical evaluation of integrals with an L_1-kernel and of Cauchy Principal Value integrals.

AMS subject classification: 65D30, 65D32.

1. Introduction

Splines have been used for numerical integration ever since they entered the numerical analysis scene [13]. However, only recently they have been applied to the numerical evaluation of integrals such as

$$I(Kf) = \int_{-1}^{1} K(x)f(x)\mathrm{d}x, \tag{1}$$

and of Cauchy principal value integrals such as

$$J(uf;\lambda) = -\!\!\!\!\!\!\int_{-1}^{1} u(x)\frac{f(x)}{x-\lambda}\mathrm{d}x, \quad -1<\lambda<1, \tag{2}$$

where $K \in L_1[-1,1]$, f is bounded in $[-1,1]$ for case (1) and u and f are such that $J(uf;\lambda)$ exists for case (2).

Some authors [1–6,16] have proposed and studied product rules for (1) and (2) based on interpolating or approximating splines. However, their results are not completely satisfactory, since they have some restrictions on the spacing of spline knots [2–6], or on the accuracy of the quadrature [16], or on the convergence properties [1].

[☆] Work sponsored by "Ministero dell'Università e Ricerca Scientifica" of Italy.

Then, since one would like to extend the above results to splines with arbitrary knot spacing as well as to splines with an order of accuracy comparable to the best spline approximation, in order to do it, in this paper we propose product rules for (1) and (2) based on approximation of f by quasi-interpolating (q-i) splines $Q_n f$, [14,17].

2. On q-i spline quadrature rules

Let

$$\Delta = \{-1 \equiv x_0 < x_1 < \ldots < x_k < x_{k+1} \equiv +1\} \tag{3}$$

be a partition of the interval $\mathfrak{I} \equiv [-1, 1]$ and let

$$\bar{\Delta} = \max_{0 \leqslant i \leqslant k} (x_{i+1} - x_i), \quad \underline{\Delta} = \min_{0 \leqslant i \leqslant k} (x_{i+1} - x_i).$$

Then there exists an associated 4-quasi uniform partition

$$\Delta^* = \{-1 \equiv x_0^* < x_1^* < \ldots < x_l^* < x_{l+1}^* \equiv +1\} \tag{4}$$

with $\Delta^* \subseteq \Delta$, such that

$$\frac{\bar{\Delta}}{2} \leqslant \underline{\Delta}^* \leqslant \bar{\Delta}^* \leqslant 2\bar{\Delta}. \tag{5}$$

In fact, letting $x_0^* = -1$, we can define $x_1^*, x_2^*, \ldots, x_l^*$ recursively by

$$x_j^* = \min\left\{ x_i : x_{j-1}^* + \frac{\bar{\Delta}}{2} \leqslant x_i \leqslant x_{j-1}^* + \frac{3}{2}\bar{\Delta} \text{ and } x_i \leqslant 1 - \frac{\bar{\Delta}}{2} \right\}. \tag{6}$$

This process does not stop until $1 - 2\bar{\Delta} \leqslant x_l^*$. Now we let $x_{l+1}^* = 1$. The property (5) follows by construction [7].

We return now to the operator Q_n. Let m be an integer and $n = m + l$; then corresponding to the partition Δ^* we define the extended one

$$\begin{aligned} y_1 &= y_2 = \ldots = y_m = x_0^*, \\ y_{m+1} &= x_1^*, \ldots, y_n = x_l^*, \\ y_{n+1} &= y_{n+2} = \ldots = y_{n+m} = x_{l+1}^*. \end{aligned} \tag{7}$$

The set of normalized B-splines $B_1^{(m)}, B_2^{(m)}, \ldots, B_n^{(m)}$ of order m, associated with this extended partition, forms a basis for $S_m(\Delta^*)$, where $S_m(\Delta^*)$ is the class of polynomial splines of the order m with knots y_i [7].

For each $i = 1, 2, \ldots, n$, let

$$\tau_{ij} = y_i + (y_{i+m} - y_i)\frac{(j-1)}{(m-1)}, \quad j = 1, 2, \ldots, m, \tag{8}$$

and

$$\alpha_{ij} = \sum_{\nu=1}^{j} \frac{\xi^{(\nu)} \psi_{ij}^{(\nu-1)}(0)}{(\nu-1)!}, \quad j = 1, 2, \ldots, m, \tag{9}$$

where

$$\xi_i^{(\nu)} = \frac{(-1)^{\nu-1}(\nu-1)!}{(m-1)!} \prod_{r=1}^{m-1} (t - y_{i+r}), \tag{10}$$

$$\psi_{ij}(t) = \prod_{r=1}^{j-1} (t - \tau_{ir}), \quad \psi_{i1}(t) = 1. \tag{11}$$

For any $f \in B(\mathfrak{I})$ [1] we consider the following quasi-interpolating operator

$$Q_n f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{m} \alpha_{ij} [\tau_{i1}, \tau_{i2}, \ldots, \tau_{ij}] f \right) B_i^{(m)}(x), \tag{12}$$

where $[\tau_{i1}, \ldots, \tau_{ij}] f$ is the $(j-1)$th divided difference.

We remark that $Q_n f$ is a linear operator mapping $B(\mathfrak{I})$ into $S_m(\Delta^*) \subseteq S_m(\Delta)$ and depending only on values of f in a small neighborhood of x.

Moreover, Q_n reproduces polynomials of order m (i.e. degree $<m$).

If we use the quasi-interpolating (12) to approximate f in (1) and in (2), we obtain respectively the following quadratures:

$$I_n(Kf) = I(KQ_n f) \simeq I(Kf), \tag{13}$$

$$J_n(uf; \lambda) = J(uQ_n f; \lambda) \simeq J(uf; \lambda), \tag{14}$$

for which we can provide a satisfactory quadrature theory.

In fact, this scheme places no restriction on the order of the splines and very few restrictions on the spacing of the knots. Moreover, it guarantees a precision degree equal to $m-1$.

Regarding convergence, we can prove that the sequence of product rules $\{I_n(Kf)\}$ converges to $I(Kf)$ for all $f \in C(\mathfrak{I})$. These convergence results are for an arbitrary partition Δ subject only to the condition

$$\bar{\Delta} \to 0 \text{ as } k \text{ or, equivalently, } n \to \infty. \tag{15}$$

In our discussion we need the following lemma directly deduced from [14].

LEMMA 2.1

Let $1 \leqslant s \leqslant m$ and $\|g\| = \max_{x \in \mathfrak{I}} |g(x)|$. Then for all $f \in C^{s-1}(\mathfrak{I})$ and for all $j = 0, 1, \ldots, s-1$,

[1] $B(\mathfrak{I}) = \{f : f$ is a real valued function on $\mathfrak{I}$ and $|f(x)| < \infty$ for all $x \in \mathfrak{I}\}$.

$$\|D^{(j)}(f - Q_n f)\| \leqslant C(\bar{\Delta})^{s-j-1} \omega_{m-s+1}(f^{(s-1)}, \bar{\Delta}), \tag{16}$$

where C is a constant dependent only on m, ω_{m-s+1} is the modulus of continuity of order $m - s + 1$ and $D^{(j)}$ is the jth derivative operator.

The following theorem provides a convergence result and a bound for the quadrature error.

THEOREM 2.1
Let $1 \leqslant s \leqslant m$. Then, for all $f \in C^{s-1}(\mathfrak{J})$ and for all $K \in L_1(\mathfrak{J})$,

$$I(Kf) - I_n(Kf) = O((\bar{\Delta})^{s-1} \omega_1(f^{(s-1)}, \bar{\Delta})), \tag{17}$$

where ω_1 is the traditional modulus of continuity of f.

Proof
Since

$$|I(Kf) - I_n(Kf)| \leqslant \|f - Q_n f\| I(|K|), \tag{18}$$

from lemma 2.1, where we put $j = 0$, the thesis follows. □

Now we consider the quadrature error $J(uf; \lambda) - J_n(uf; \lambda)$, for which in the following theorem we derive a bound.

THEOREM 2.2
Let $2 \leqslant s \leqslant m$. Then, for all $f \in C^{s-1}(\mathfrak{J})$,

$$J(uf; \lambda) - J_n(uf; \lambda) = O((\bar{\Delta})^{s-2} \omega_1(f^{(s-1)}, \bar{\Delta})). \tag{19}$$

Proof
With the help of the Mean Value Theorem and lemma 2.1, where we assume $j = 0, 1$, we can easily obtain

$$|J(uf; \lambda) - J_n(uf; \lambda)| \leqslant C\omega_{m-s+1}(f^{(s-1)}, \bar{\Delta}) \cdot U, \tag{20}$$

where

$$U = \left\{ (\bar{\Delta})^{s-2} \int_{-1}^{1} |u(x)| \mathrm{d}x + (\bar{\Delta})^{s-1} \left| \fint_{-1}^{1} \frac{u(x)}{x - \lambda} \mathrm{d}x \right| \right\}.$$

From (20) the thesis follows. □

3. On the computational procedure QSIPQR

We remark that, since we can write

$$Q_nf(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij}B_i^{(m)}(x)f(\tau_{ij}), \tag{21}$$

where

$$v_{ij} = \sum_{r=j}^{m} \frac{\alpha_{ir}}{\prod_{\substack{s=1\\ s\neq j}}^{r}(\tau_{ij}-\tau_{is})}, \tag{22}$$

then

$$I_n(Kf) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mu_{ij}f(\tau_{ij}), \tag{23}$$

where $\mu_{ij} = v_{ij}I(KB_i^{(m)})$.

The real values $w_i = I(KB_i^{(m)}), i = 1, 2, \ldots, n$, are the weights of the product integration rule

$$R_n(Kf) = \sum_{i=1}^{n} w_if(z_i) \simeq I(Kf) \tag{24}$$

with prefixed nodes $z_i \in [y_i, y_{i+m}]$ based on approximating splines with knots (7) [16]. We recall that the precision degree of (24) is $\leqslant 1$; it is $= 1$ only if $\{z_i\}_{i=1}^{n}$ are the Schoenberg points (see [11]).

The construction of nodes $\{\tau_{ij}\}$ and weights $\{\mu_{ij}\}$ of the rule (23) is made by the algorithm QSIPQR, whose computing task is broken down into a series of modules. Given an initial points partition Δ, in module DELTA of QSIPQR, we generate the associated 4-quasi uniform partition Δ^* and then the extended partition (7). Successively, by (8) we obtain the nodes $\{\tau_{ij}\}$.

The computation of the weights $\{\mu_{ij}\}$ is obtained by the modules VIJ and INTKB. For any fixed i, by (22), the module VIJ computes the values $v_{ij}, j = 1, 2, \ldots, m$, with α_{ir} defined by (9) [17].

Recalling the definition of the classical symmetric functions $\text{symm}_j(t_1, t_2, \ldots, t_p)$, an elementary calculation shows that

$$\text{symm}_j(t_1, t_2, \ldots, t_p) = \sum_{1\leqslant i_1<i_2<\ldots<i_j\leqslant p} t_{i_1}t_{i_2}\ldots t_{i_j}.$$

Therefore we can use the symmetric functions to compute α_{ir}. In fact, we can write

$$\alpha_{ir} = \sum_{j=0}^{r-1}(-1)^j\xi_i^{(r-j)}\text{symm}_j(\tau_{i1}, \ldots, \tau_{i,r-1})$$

and, from (10),

$$\xi_i^{(\nu)} = \frac{(\nu-1)!(m-\nu)!}{(m-1)!}\operatorname{symm}_{\nu-1}(y_{i+1},\dots,y_{i+m-1}). \tag{25}$$

Then the final computational formula for α_{ir} is:

$$\alpha_{ir} = \sum_{\nu=1}^{r}(-1)^{r-\nu}\frac{(\nu-1)!(m-\nu)!}{(m-1)!}C_{i,\nu-1}D_{i,r-\nu}, \tag{26}$$

where

$$C_{i,\nu-1} = \operatorname{symm}_{\nu-1}(y_{i+1},\dots,y_{i+m-1})$$

and

$$D_{i,r-\nu} = \operatorname{symm}_{r-\nu}(\tau_{i1},\dots,\tau_{i,r-1}).$$

For any fixed i, the module INTKB evaluates $I(KB_i^{(m)})$. Following [11], on the set of knots $\{y_i\}_{i=1}^n$ we can define

$$B_i^{(1)}(x) = \begin{cases} 1, & y_i \leqslant x < y_{i+1}, \\ 0, & \text{otherwise,} \end{cases} \tag{27}$$

and the B-splines of order m are generated by the stable recursive method

$$\begin{aligned} B_i^{(m)}(x) &= \frac{x-y_i}{y_{i+m-1}-y_i}B_i^{(m-1)}(x) \\ &\quad + \frac{y_{i+m}-x}{y_{i+m}-y_{i+1}}B_{i+1}^{(m-1)}(x), \quad i=1,2,\dots,n. \end{aligned} \tag{28}$$

Through (28) the module INTKB implements a recurrence formula to obtain the values $\{w_i\}_{i=1}^n$. Letting the integrals be

$$I_p(KB_i^{(q)}) = \int_{-1}^{1} x^p K(x) B_i^{(q)}(x)\mathrm{d}x, \quad p=0,1,\dots,m-q; q=1,2,\dots,m, \tag{29}$$

we insert (28) into (29) to find the recurrence formula:

$$\begin{aligned} I_p(KB_i^{(m)}) &= \frac{I_{p+1}(KB_i^{(m-1)}) - y_i I_p(KB_i^{(m-1)})}{y_{i+m-1}-y_i} \\ &\quad + \frac{y_{i+m}I_p(KB_{i+1}^{(m-1)}) - I_{p+1}(KB_{i+1}^{(m-1)})}{y_{i+m}-y_{i+1}}. \end{aligned} \tag{30}$$

This formula, starting with the sequence of integrals

$$I_p(KB_i^{(1)}) = \begin{cases} \int_{y_i}^{y_{i+1}} K(x)x^p dx, & \text{when } i=m,\dots,n, \\ 0, & \text{otherwise,} \end{cases}$$
$$p = 0,1,2,\dots,m-1, \tag{31}$$

is used to evaluate the terms $I(KB_i^{(m)}) = I_0(KB_i^{(m)})$. Each of the integrals in (31) is evaluated by the module CALINT, here proposed for the case of kernel

$$K(x) = \ln|x-\lambda|, \quad \lambda \in (-1,1).$$

For another choice of kernel K the module CALINT has to be replaced by another appropriate one, specific for the function K considered.

Now we can report the closed form expressions of (31) for the kernel here considered. Let:

$$A_i = \frac{y_i^{p+1} - \lambda^{p+1}}{p+1} \ln|\lambda - y_i|, \tag{32}$$

$$S_i = \frac{1}{p+1} \sum_{k=0}^{p} \lambda^k \frac{y_{i+1}^{p-k+1} - y_i^{p-k+1}}{p-k+1}, \tag{33}$$

then, for $i = m, \ldots, n$,

$$I_p(\ln|\lambda - x|B_i^{(1)}) = \begin{cases} A_{i+1} - A_i - S_i, & \text{if } \lambda \notin [y_i, y_{i+1}] \\ & \text{or } \lambda \in (y_i, y_{i+1}), \\ A_{i+1} - S_i, & \text{if } \lambda = y_i, \\ -A_i - S_i, & \text{if } \lambda = y_{i+1}. \end{cases} \tag{34}$$

A FORTRAN 77 code, whose listing and diskette are given in [9], implements the above method to generate the nodes $\{\tau_{ij}\}$ and the weights $\{\mu_{ij}\}$ of (23).

4. On the computational procedure QSICPV

We can write

$$J_n(uf;\lambda) = \sum_{i=1}^{n} \sum_{j=1}^{m} \nu_{ij}(\lambda) f(\tau_{ij}), \tag{35}$$

where

$$\nu_{ij}(\lambda) = v_{ij} J(uB_i^{(m)};\lambda). \tag{36}$$

We remark that the real values $\{\bar{w}_i(\lambda) = J(uB_i^{(m)};\lambda), i = 1,2,\ldots,n\}$ are the weights of the integration rule for the CPV integrals (2),

$$\bar{R}_n(uf;\lambda) = \sum_{i=1}^{n} \bar{w}_i(\lambda) f(z_i) \simeq J(uf;\lambda), \tag{37}$$

with prefixed nodes $z_i \in [y_i, y_{i+m}]$ based on approximating splines with knots (7) [16]. The precision degree of (37) is $\leqslant 1$; it is $= 1$ only if z_i are the Schoenberg points (see [11]).

Given an initial points partition Δ, in order to generate the knots $\{y_i\}$ of the extended partition, we use the module DELTA as in QSIPQR. If one of the knots coincides with λ, then the procedure stops with a message, otherwise it evaluates nodes $\{\tau_{ij}\}$ and the real value $\{v_{ij}\}$ by the module VIJ.

If $\lambda \neq y_i, \forall i = 1, 2, \dots, n$, then for every i, the module INTKB evaluates the weights $\bar{w}_i(\lambda)$.

Defining the integrals

$$J_p(uB_i^{(q)}; \lambda) = \int_{-1}^{1} u(x) x^p \frac{B_i^{(q)}(x)}{x-\lambda} \mathrm{d}x, \quad p = 0, 1, \dots, m-q; q = 1, 2, \dots, m, \tag{38}$$

by means of (28), we obtain

$$J_p(uB_i^{(m)}; \lambda) = \frac{J_{p+1}(uB_i^{(m-1)}; \lambda) - y_i J_p(uB_i^{(m-1)}; \lambda)}{y_{i+m-1} - y_i} + \frac{y_{i+m} J_p(uB_i^{(m-1)}; \lambda) - J_{p+1}(uB_{i+1}^{(m-1)}; \lambda)}{y_{i+m} - y_{i+1}}. \tag{39}$$

The above recurrence formula, starting with the sequence of integrals

$$J_p(uB_i^{(1)}; \lambda) = \int_{-1}^{1} u(x) \frac{x^p B_i^{(1)}(x)}{x-\lambda} \mathrm{d}x, \quad p = 0, 1, \dots, m-1, \tag{40}$$

is used to evaluate the terms:

$$J(uB_i^{(m)}; \lambda) = J_0(uB_i^{(m)}; \lambda). \tag{41}$$

In order to calculate the elements in (40), of the recurrence basis, we can write

$$J_p(uB_i^{(1)}; \lambda) = \\ = \begin{cases} \int_{y_i}^{y_{i+1}} u(x) \frac{x^p}{x-\lambda} \mathrm{d}x = & \sum_{k=0}^{p-1} \lambda^k \int_{y_i}^{y_{i+1}} u(x) x^{p-k-1} \mathrm{d}x + \\ & + \lambda^p J_0(uB_i^{(1)}; \lambda), \quad i = m, \dots, n, \\ 0, & \text{otherwise}. \end{cases} \tag{42}$$

Each of the integrals in (42) is evaluated by the module CPVINT, using a closed expression, if it exists.

When a closed expression for (42) does not exist then, similarly to [12], a numerical method must be used.

A FORTRAN 77 code, whose listing and diskette are given in [9], implements the above method to generate the nodes $\{\tau_{ij}\}$ and the weights $\{\nu_{ij}\}$ of (35).

5. Application and final remarks

Several numerical applications that test the performance of the rules (13) and (14) have been made, and the results obtained are available [9].

In order to construct the B-splines we must choose the initial partition Δ. For our numerical examples we have chosen the following two Δ-partitions, both satisfying (15):

$$U\text{: uniform } \Delta = \left\{x_i = -1 + \frac{2i}{k+1}, i = 0, 1, \ldots, k+1\right\},$$

$$P\text{: perfect } \Delta = \left\{x_i = \cos\left(\frac{k-i+1}{k+1}\pi\right), i = 0, 1, \ldots, k+1\right\}.$$

We remark that if $\Delta = U$ then $\Delta^* = \Delta$, whereas if $\Delta = P$ we have used the constructive theorem 2.1 to generate Δ^* from Δ.

An interesting open question, concerning the above formulas and here not considered, is their convergence for larger classes of functions f. Recently, we have investigated this problem and the results obtained are reported in [7,8].

References

[1] A. Alaylioglu, D.S. Lubinsky and D. Eyre, Product integration of logarithmic singular integrands based on cubic splines, J. Comp. Appl. Math. 11 (1984) 353–366.

[2] C. Dagnino and A. Palamara Orsi, Product integration of piecewise continuous integrands based on cubic spline interpolation at equally spaced nodes, Numer. Math. 52 (1988) 459–466.

[3] C. Dagnino, Product integration of singular integrands based on cubic spline interpolation at equally spaced nodes, Numer. Math. 57 (1990) 97–104.

[4] C. Dagnino and E. Santi, On the evaluation of one-dimensional Cauchy principal value integrals by rules based on cubic spline interpolation. Computing 43 (1990) 267–276.

[5] C. Dagnino and E. Santi, Spline product quadrature rules for Cauchy singular integrals, J. Comp. Appl. Math. 33 (1990) 133–140.

[6] C. Dagnino and E. Santi, On the convergence of spline product rules for Cauchy principal value integrals, J. Comp. Appl. Math. 36 (1991) 181–187.

[7] C. Dagnino, V. Demichelis and E. Santi, Numerical integration based on quasi-interpolating splines, Computing 50 (1993) 149–163.

[8] C. Dagnino and P. Rabinowitz, Product integration of singular integrands using quasi-interpolating splines, submitted for publication (1992).

[9] C. Dagnino, V. Demichelis and E. Santi, A FORTRAN code for computing nodes and weights of quadratures based on quasi-interpolating splines, Int. Report (1992).

[10] P.J. Davis and P. Rabinowitz, *Numerical Integration*, 2nd ed. (Academic Press, New York, 1984).

[11] C. de Boor, *A Practical Guide to Splines*, Applied Mathematical Sciences, vol. 27 (Springer, New York, 1978).

[12] A. Gerasoulis, Piecewise-polynomial quadratures for Cauchy singular integrals, SIAM J. Numer. Anal. 23 (1986) 891–902.

[13] T.N.E. Greville, Spline functions, interpolation and numerical quadrature, in: *Mathematical Methods for Digital Computers*, Vol. 2, eds. A. Ralston and H.S. Wolf (Wiley, New York, 1967) pp. 156–168.

[14] T. Lyche and L.L. Schumaker, Local spline approximation methods, J. Approx. Theory 15 (1975) 294–325.

[15] P. Rabinowitz, The convergence of non-interpolatory product integration rules, in: *Numerical Integration* (Reidel, 1987).
[16] P. Rabinowitz, Numerical integration based on approximating splines, J. Comp. Appl. Math. 33 (1990) 73–83.
[17] L.L. Schumaker, *Spline Functions* (Wiley, 1981).