INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 6, 317–324 (1986)

ON THE EFFICIENT EVALUATION OF CERTAIN INTEGRALS IN THE GALERKIN F. E. METHOD

A. STANIFORTH AND C. BEAUDOIN

Recherche en prévision numérique, Atmospheric Environment Service, 2121 Trans-Canada Highway, Dorval, P. Q., Canada H9P 1J3

SUMMARY

The use of linear finite elements in fluid dynamic problems requires the evaluation of integrals of polynomial expressions, which arise from product terms in the equations of motion. An algorithm based on Simpson quadrature is presented and its efficiency compared with that of the more usual one, based on Gaussian quadrature. For both algorithms, the integrations are exact provided that the polynomial integrand is at most cubic. It is found that the Simpson algorithm is twice as efficient as the corresponding Gaussian one, for the evaluation of integrals in one, two and three space dimensions. This doubling of efficiency is a consequence of the vanishing of the basis functions at certain points, a property that can be exploited in the Simpson algorithm, but not in the Gaussian one. It is thought that the use of the Simpson algorithm will prove to be beneficial in many finite element fluid dynamic codes, because the evaluation of product terms generally represents a significant fraction of the total computational cost.

INTRODUCTION

The use of linear finite elements in one, two and three space dimensions has become quite popular for fluid dynamic problems in simple geometries,[1,2] and leads to the need to evaluate integrals of the form

$$I_k = \int u(\mathbf{x})v(\mathbf{x})\theta_k(\mathbf{x})\mathrm{d}\mathbf{x} \tag{1}$$

where $\theta_k(\mathbf{x})$ is the basis function associated with the kth node $\mathbf{x}_k$, $u(\mathbf{x})$ and $v(\mathbf{x})$ are finite-element expansions of the form

$$f(\mathbf{x}) = \sum_{m=1}^{M} f_m \theta_m(\mathbf{x}), \tag{2}$$

and $\mathbf{x}$ is the one, two- or three-dimensional space vector. As noted by Gresho *et al.*,[1] such integrals implicitly contain many terms, particularly in two and three dimensions, and in many applications their evaluation contributes significantly to the total computational cost; it is consequently important to evaluate them efficiently. In fact, it was this costly evaluation that motivated Gresho *et al.*[1] to examine an '... *ad hoc* (non-Galerkin) modification called centroid advection velocity'. They also caution the reader that this modification '... may, however, introduce some aliasing error in contrast to the honest Galerkin finite element method, which does not'. It is not our intention to discuss here the merits of this alternative approach, but rather to describe how the calculations of the 'honest Galerkin finite element method' may be made more efficient in this context, and consequently computationally more attractive.

0271–2091/86/060317–08$05.00

Received 19 August 1985
Revised 25 November 1985

Because the integrand is a polynomial it is possible to analytically integrate the individual terms and express the integral as a weighted sum of products of the form $u_i v_j$, where u_i and v_j are the values of u and v at the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, neighbours of the node $\mathbf{x}_k$. This is rarely done in practice, however, since it is computationally expensive.

The usual method of evaluation (see, for instance, Reference 1, p. 562 and Reference 3, p. 32) is to use Gaussian quadrature over subdomains, where the quadrature formula is chosen to be the one of lowest order that exactly integrates the polynomial integrand. In the present paper we show that the computational cost of evaluating such integrals may be halved by using Simpson quadrature instead of the more usual Gaussian quadrature. Although Simpson quadrature has been successfully used in the past to evaluate FE integrals,[4,5] its efficiency advantage for bilinear and trilinear elements does not appear to be generally well known, thus motivating this short contribution. We first compare the computational efficiencies of the two methods in one dimension, and then show how to extend this result to higher dimensions.

EVALUATION IN ONE DIMENSION BY GAUSSIAN QUADRATURE

In one dimension (1) and (2) reduce to

$$I_k = \int_{x_1}^{x_M} u(x)v(x)e_k(x)\mathrm{d}x \tag{3}$$

and

$$f(x) = \sum_{m=1}^{M} f_m e_m(x), \tag{4}$$

where $e_k(x)$ is the Chapeau basis function (Reference 3, p. 27) centred on node x_k of a mesh of points $x_1 < x_2 < x_3 < \cdots < x_M$, and $f_m = f(x_m)$. These basis functions are illustrated in Figure 1; they form an interpolatory basis and $f(x)$ is piecewise linear. Since $e_k(x)$ is zero over most of the domain, (3) simplifies to

$$I_1 = R_1, \tag{5}$$

$$I_k = R_k + S_{k-1}, \quad k = 2, 3, \ldots, (M-1), \tag{6}$$

$$I_M = S_{M-1}, \tag{7}$$

where

$$R_k = \int_{x_k}^{x_{k+1}} u(x)v(x)e_k(x)\mathrm{d}x, \quad k = 1, 2, \ldots, (M-1) \tag{8}$$

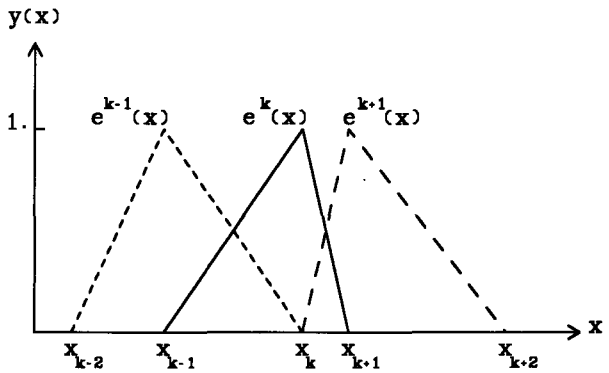

Figure 1. Three adjacent Chapeau basis functions: $e^{k-1}(x)$—short dashed line; $e^k(x)$—full line; $e^{k+1}(x)$—long dashed line

and

$$S_k = \int_{x_k}^{x_{k+1}} u(x)v(x)e_{k+1}(x)\mathrm{d}x, \quad k = 1, 2, \ldots, (M-1). \tag{9}$$

For a given interval $x_k \leqslant x \leqslant x_{k+1}$, R_k is the contribution over this interval to I_k, whereas S_k is the contribution to I_{k+1}.

The integrands of (8) and (9) are cubics by construction, and can therefore be evaluated exactly by the two-point Gaussian quadrature rule (Reference 6, section 2.7):

$$\int_{x_k}^{x_{k+1}} f(x)\mathrm{d}x = (x_{k+1} - x_k)[f(\xi_k^-) + f(\xi_k^+)]/2, \tag{10}$$

where

$$\xi_k^{\pm} = (x_k + x_{k+1})/2 \pm (x_{k+1} - x_k)/(2\sqrt{3}). \tag{11}$$

Thus (8) and (9) may be rewritten as

$$R_k = \frac{(x_{k+1} - x_k)}{2}[u(\xi_k^-)v(\xi_k^-)e_k(\xi_k^-) + u(\xi_k^+)v(\xi_k^+)e_k(\xi_k^+)] \tag{12}$$

and

$$S_k = \frac{(x_{k+1} - x_k)}{2}[u(\xi_k^-)v(\xi_k^-)e_{k+1}(\xi_k^-) + u(\xi_k^+)v(\xi_k^+)e_{k+1}(\xi_k^+)]. \tag{13}$$

For $x \in [x_k, x_{k+1}]$ we have by definition that

$$e_k(x) = \frac{(x_{k+1} - x)}{(x_{k+1} - x_k)}, \tag{14}$$

$$e_{k+1}(x) = \frac{(x - x_k)}{(x_{k+1} - x_k)}, \tag{15}$$

$$u(x) = u_k e_k(x) + u_{k+1} e_{k+1}(x), \tag{16}$$

and similarly for $v(x)$. In particular

$$e_k(\xi_k^{\pm}) = \tfrac{1}{2}(1 \mp 1/\sqrt{3}), \tag{17}$$

$$e_{k+1}(\xi_k^{\pm}) = \tfrac{1}{2}(1 \pm 1/\sqrt{3}). \tag{18}$$

$$u(\xi_k^{\pm}) = u_k e_k(\xi_k^{\pm}) + u_{k+1} e_{k+1}(\xi_k^{\pm}), \tag{19}$$

and a similar result obtains for $v(x)$.

Assuming that all mesh-dependent (i.e. data-independent) quantities are pre-computed once and for all, we can put together the above results as:

Algorithm G1 (Gaussian quadrature in one dimension)

(i) For $k = 1, 2, \ldots, (M-1)$, compute

$$A_k^{\pm} = u_k + [e_{k+1}(\xi_k^{\pm})/e_k(\xi_k^{\pm})]u_{k+1}, \tag{20}$$

$$B_k^{\pm} = v_k + [e_{k+1}(\xi_k^{\pm})/e_k(\xi_k^{\pm})]v_{k+1}, \tag{21}$$

$$C_k^{\pm} = A_k^{\pm} B_k^{\pm}. \tag{22}$$

(ii) For $k = 1, 2, \ldots, (M-1)$, compute

$$R_k = [(x_{k+1} - x_k)e_k^3(\xi_k^-)/2]C_k^- + [(x_{k+1} - x_k)e_k^3(\xi_k^+)/2]C_k^+ \tag{23}$$

and

$$S_k = [(x_{k+1} - x_k)e_k^2(\xi_k^-)e_{k+1}(\xi_k^-)/2]C_k^- + [(x_{k+1} - x_k)e_k^2(\xi_k^+)e_{k+1}(\xi_k^+)/2]C_k^+ . \tag{24}$$

(iii) For $k = 1, 2, \ldots, M$ compute I_k using (5)–(7).

In Algorithm G1, all the terms in square brackets are data-independent and may be precomputed. These terms have been grouped together in this way in order to precompute as many mesh-dependent quantities as possible, and thus minimize the total number of arithmetic operations in the context of a problem requiring many integral evaluations on a fixed mesh. Denoting a multiplication by m and an addition by a, the number of arithmetic operations per node for steps (i)–(iii) are $2(3m + 2a)$, $(4m + 2a)$ and $(1a)$, respectively. The total number of arithmetic operations per node for Algorithm G1 is thus $(10m + 7a)$.

EVALUATION IN ONE DIMENSION BY SIMPSON QUADRATURE

Instead of using the two-point Gaussian quadrature rule (10) to evaluate (8) and (9), we use the Simpson rule

$$\int_{x_k}^{x_{k+1}} f(x)\mathrm{d}x = (x_{k+1} - x_k)[f(x_k) + 4f(x_{k+1/2}) + f(x_{k+1})]/6, \tag{25}$$

where

$$x_{k+1/2} = (x_k + x_{k+1})/2. \tag{26}$$

Simpson's rule is also exact for cubics and application to (8) and (9) yields [using (14)–(16)]

$$R_k = (x_{k+1} - x_k)(u_k v_k + 2u_{k+1/2}v_{k+1/2})/6 \tag{27}$$

and

$$S_k = (x_{k+1} - x_k)(2u_{k+1/2}v_{k+1/2} + u_{k+1}v_{k+1})/6, \tag{28}$$

where

$$u_{k+1/2} = \tfrac{1}{2}(u_k + u_{k+1}) \tag{29}$$

and

$$v_{k+1/2} = \tfrac{1}{2}(v_k + v_{k+1}). \tag{30}$$

Note that there is no contribution to R_k at $x = x_{k+1}$ by virtue of the fact that $e_k(x_{k+1}) \equiv 0$, by definition. Similarly, there is no contribution to S_k at $x = x_k$, since $e_{k+1}(x_k) \equiv 0$.

Using (5)–(7) and (27)–(30) leads to:

Algorithm S1 (Simpson quadrature in one dimension)

(i) For $k = 1, 2, \ldots, (M-1)$, compute

$$D_{k+1/2} = u_k + u_{k+1} \tag{31}$$

and

$$E_{k+1/2} = v_k + v_{k+1}. \tag{32}$$

(ii) For $k = 1, 2, \ldots, (M-1)$, compute

$$F_{k+1/2} = [(x_{k+1} - x_k)/12] D_{k+1/2} E_{k+1/2}. \tag{33}$$

(iii) Compute for $k = 1, 2, \ldots, M$

$$G_k = u_k v_k. \tag{34}$$

(iv) Compute

$$I_1 = [(x_2 - x_1)/6] G_1 + F_{3/2}, \tag{35}$$

$$I_k = [(x_{k+1} - x_{k-1})/6] G_k + F_{k-1/2} + F_{k+1/2}, \tag{36}$$

for $k = 2, 3, \ldots, (M-1)$, and

$$I_M = [(x_M - x_{M-1})/6] G_M + F_{M-1/2}. \tag{37}$$

Terms have again been grouped together to minimize the total number of arithmetic operations, and those in square brackets are data-independent and may be precomputed. The number of arithmetic operations per node for steps (i)–(iv) of Algorithm S1 are $(2a)$, $(2m)$, $(1m)$ and $(1m + 2a)$, respectively, giving a total of $(4m + 4a)$. Comparing this total (8 arithmetic operations per node) with that of the corresponding Gaussian algorithm ($10m + 7a = 17$ arithmetic operations per node), we find that *the Simpson algorithm is approximately twice as fast as the Gaussian algorithm.*

EVALUATION IN 2-D AND 3-D

Evaluations in two and three dimensions of integrals of the form of (1) are performed by a dimension-reduction technique whereby, for example, a 3-D evaluation is reduced to a set of 2-D evaluations, each of which in turn is further reduced to a set of 1-D evaluations. To illustrate the technique, we examine the evaluation using Simpson quadrature of the 2-D integral

$$J_{m,n} = \int_{x_1}^{x_M} \int_{y_1}^{y_N} u(x, y) v(x, y) e_m(x) e_n(y) \mathrm{d}x \mathrm{d}y, \tag{38}$$

where

$$u(x, y) = \sum_{p=1}^{M} \sum_{r=1}^{N} u_{p,r} e_p(x) e_r(y), \tag{39}$$

with a similar expansion for $v(x, y)$.

For $2 \leqslant n \leqslant N-1$ the limits of integration over y in (38) reduce to the interval $y_{n-1} \leqslant y \leqslant y_{n+1}$. By applying Simpson quadrature to each of the two intervals $y_{n-1} \leqslant y \leqslant y_n$ and $y_n \leqslant y \leqslant y_{n+1}$ and noting that $e_n(y_{n\pm 1}) = 0$, $e_n(y_n) = 1$, $e_n(y_{n\pm 1/2}) = \frac{1}{2}$, where $y_{n\pm 1/2} = (y_n + y_{n\pm 1})/2$, we may rewrite (38) as

$$\begin{aligned} J_{m,n} = &\left\{ \left(\frac{y_n - y_{n-1}}{6} \right) \left[\int_{x_1}^{x_M} u(x, y_n) v(x, y_n) e_m(x) \mathrm{d}x + 2 \int_{x_1}^{x_M} u(x, y_{n-1/2}) v(x, y_{n-1/2}) e_m(x) \mathrm{d}x \right] \right\} \\ &+ \left\{ \left(\frac{y_{n+1} - y_n}{6} \right) \left[\int_{x_1}^{x_M} u(x, y_n) v(x, y_n) e_m(x) \mathrm{d}x + 2 \int_{x_1}^{x_M} u(x, y_{n+1/2}) v(x, y_{n+1/2}) e_m(x) \mathrm{d}x \right] \right\} \end{aligned} \tag{40}$$

For $n = 1$, $J_{m,1}$ is given by the second curly-bracketed term of (40) and for $n = N$, $J_{m,N}$ is given by the first curly-bracketed term.

From (39), and (14) and (15) we see that

$$u(x, y_n) = \sum_{p=1}^{M} u_{p,n} e_p(x) \tag{41}$$

and

$$u(x, y_{n\pm 1/2}) = \sum_{p=1}^{M} \frac{(u_{p,n} + u_{p,n\pm 1})}{2} e_p(x)$$

$$= \sum_{p=1}^{M} u_{p,n\pm 1/2} e_p(x) \quad \text{(say)}, \tag{42}$$

with similar results for $v(x, y_n)$ and $v(x, y_{n\pm 1/2})$.

The integrals appearing in (40) therefore correspond to one-dimensional integrals along the lines $y = y_*$, where $y_* = y_n$ or $y_{n\pm 1/2}$, and these can be evaluated using the one-dimensional algorithm of the previous section. Putting these results together, and grouping terms to minimize the number of arithmetic operations, we have:

Algorithm S2 (Simpson quadrature in two dimensions)

(i) For $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, (N-1)$, compute

$$D_{m,n+1/2} = u_{m,n} + u_{m,n+1}, \tag{43}$$

and

$$E_{m,n+1/2} = v_{m,n} + v_{m,n+1}. \tag{44}$$

(ii) For $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, (N-1)$, compute (using Algorithm S1)

$$F_{m,n+1/2} = [(y_{n+1} - y_n)/12] \int_{x_1}^{x_M} D_{n+1/2}(x) E_{n+1/2}(x) e_m(x) \mathrm{d}x, \tag{45}$$

where

$$D_{n+1/2}(x) = \sum_{m=1}^{M} D_{m,n+1/2} e_m(x) \tag{46}$$

and

$$E_{n+1/2}(x) = \sum_{m=1}^{M} E_{m,n+1/2} e_m(x). \tag{47}$$

(iii) For $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$, compute (using Algorithm S1)

$$G_{m,n} = \int_{x_1}^{x_M} u(x, y_n) v(x, y_n) e_m(x) \mathrm{d}x. \tag{48}$$

(iv) For $m = 1, 2, \ldots, M$ compute

$$J_{m,1} = [(y_2 - y_1)/6] G_{m,1} + F_{m,3/2} \tag{49}$$

$$J_{m,n} = [(y_{n+1} - y_{n-1})/6] G_{m,n} + F_{m,n-1/2} + F_{m,n+1/2}, \tag{50}$$

for $n = 2, 3, \ldots, (N-1)$, and

$$J_{m,N} = [(y_N - y_{N-1})/6] G_{m,N} + F_{m,N-1/2}. \tag{51}$$

In Algorithm S2, the square-bracketed terms are again data-independent, and are assumed to have been precomputed. Comparing the form of Algorithm S2 with that of Algorithm S1, we see that they are analogous to one another, step for step. The number of arithmetic operations

per node for steps (i)–(iv) of Algorithm S2 are $(2a)$, $(5m + 4a)$, $(4m + 4a)$ and $(1m + 2a)$, respectively, giving a total of $(10m + 12a) = 22$ arithmetic operations. This is approximately half the cost of the corresponding two-dimensional Gaussian algorithm which requires $(28m + 21a) = 49$ arithmetic operations.

It is straightforward to extend the above method for the evaluation of three-dimensional integrals and we therefore omit the details. In Table I, we summarize the numbers of arithmetic operations required by three different methods, namely computation by

(a) weighted sums of the products of local values (as described in the Introduction)
(b) Gaussian quadrature
(c) Simpson quadrature.

We see that Simpson quadrature is twice as efficient as Gaussian quadrature for integrals in one, two and three space dimensions.

Table I. Operation counts per node of three different methods for evaluating integrals of the form (1), as a function of the number of space dimensions; m denotes a multiplication, a an addition and 'ops' either a multiplication or an addition.

	Number of space dimensions		
	1	2	3
(a) local weighted sums	$7m + 6a = 13$ ops	$49m + 48a = 97$ ops	$343m + 342a = 685$ ops
(b) Gaussian quadrature	$10m + 7a = 17$ ops	$28m + 21a = 49$ ops	$64m + 49a = 113$ ops
(c) Simpson quadrature	$4m + 4a = 8$ ops	$10m + 12a = 22$ ops	$22m + 28a = 50$ ops

Further economies are possible in the context of a fluid dynamic model code. For example, for an advection velocity u that appears in different terms in different equations, it is possible to avoid recomputing quantities such as $D_{m,n+1/2}$ of (43). The use of Simpson quadrature instead of Gaussian quadrature is equally beneficial for the computation of other integrals arising from product terms, such as $\int u(x)\partial v/\partial x_i \theta^k(x)\mathrm{d}x$. The only restriction is that the integrand be such that Simpson quadrature is exact.

Finally, the described method for evaluating integrals is highly vectorizable (a very important aspect of the programming of supercomputers). For example, a 3-D evaluation on a $51 \times 51 \times 51$ mesh using Simpson quadrature takes 0·11 s on a Cray 1-S computer, which corresponds to approximately 60 Mflops (millions of floating-point operations per second).

ACKNOWLEDGEMENT

We would like to thank Dr. Clive Temperton and an anonymous reviewer for reviewing the manuscript and for their helpful comments. The authors are also indebted to Ms. Maryse Ferland and Ms. Jeanne Mongeau for their expert typing of the manuscript.

REFERENCES

1. P. Gresho, S. T. Chan, R. L. Lee and C. D. Upson, 'A modified finite element method for solving the time-dependent, incompressible Navier–Stokes equations. Part 1: theory', *Int. j. numer. methods fluids*, **4**, 557–598 (1984).
2. M. J. P. Cullen and K. W. Morton, 'Analysis of evolutionary error in finite element and other methods', *J. Comput. Phys*, **34**, 245–267 (1980).

3. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice Hall, Englewood Cliffs, N.J., 1973.
4. A. N. Staniforth and H. L. Mitchell, 'A semi-implicit finite-element barotropic model', *Mon. Wea. Rev.*, **105,** 154–169 (1977).
5. W. G. Gray and M. Van Genuchten, 'Economical alternatives to Gaussian quadrature over isoparametric quadrilaterals', *Int. j. numer. methods eng.*, **12,** 1478–1484 (1978).
6. P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, Academic Press, N.Y., 1975.