

A SET OF SYMMETRIC QUADRATURE RULES ON TRIANGLES AND TETRAHEDRA*

Linbo Zhang, Tao Cui and Hui Liu

*LSEC, ICMSEC, Academy of Mathematics and Systems Science,
Chinese Academy of Sciences, Beijing 100190, China*

Email: zlb@lsec.cc.ac.cn, tcui@lsec.cc.ac.cn, liuhui@lsec.cc.ac.cn

Abstract

We present a program for computing symmetric quadrature rules on triangles and tetrahedra. A set of rules are obtained by using this program. Quadrature rules up to order 21 on triangles and up to order 14 on tetrahedra have been obtained which are useful for use in finite element computations. All rules presented here have positive weights with points lying within the integration domain.

Mathematics subject classification: 65D32, 65D30, 65M60.

Key words: Finite element, Numerical integration, Quadrature, Cubature, Triangle, Tetrahedron.

1. Introduction

In finite element computations, numerical integration is widely used for computing integrals of functions or bilinear forms. For triangular meshes numerical integrations on line segments, triangles, and tetrahedra are needed. In contrast to quadrilaterals or hexahedra on which quadrature formulas can be naturally derived from tensor products of one-dimensional Gauss quadrature rules, high-order non-tensor product quadrature rules on triangles and tetrahedra are difficult to construct. In fact, many of the non-tensor product rules published in finite element textbooks contain either negative weights or points outside of the integration domain, which are undesirable for numerical computations. As a result, it is a common practice in many finite element packages to use quadrature rules associated with tensor products of one of the Gauss-Jacobi rules; these rules are unsymmetric and generally require (as many as twice in three-dimensions) more function evaluations.

There have been many studies searching for quadrature rules on triangles and tetrahedra, both numerically and analytically. The problem of finding quadrature rules generally leads to a problem of finding the zeros or minima of high-order multi-variate polynomials, which is known to be extremely difficult. Many methods have been developed for computing quadrature rules; we refer the reader to [1, 2, 3, 4, 5, 6, 7, 8, 9] and the references therein.

In this paper, we present a program for computing symmetric quadrature rules on triangles and tetrahedra and a set of quadrature rules obtained using this program. The underlying algorithm turns the problem of computing quadrature rules into nonlinear least square solution of systems of polynomial equations, and makes use of MINPACK [10] which is a publicly available well known minimization package. All rules presented here are fully symmetric and have positive weights with quadrature points lying within the integration domain. We believe that at least some of the rules presented in this paper are new. We prefer symmetric rules

* Received February 17, 2008 / accepted April 18, 2008 /

on triangles and tetrahedra because they are naturally related to the geometric symmetry of the integration domain and can be represented in a compact form using symmetry orbits, and more importantly, the symmetry of the quadrature points may be exploited, together with the symmetry of finite element basis functions, to reduce the computational cost in the calculation of mass and stiffness matrices.

After this short introduction, the rest of the paper is organized as follows. In Section 2 we give a brief description on quadrature rules and define some notations used. In Section 3 we present our program and the underlying numerical algorithm for computing quadrature rules. In Section 4 we report the quadrature rules found using this program. In the final section we give some concluding remarks.

2. Notations

Let T be a d -dimensional simplex, here $d = 2$ (triangle) or 3 (tetrahedron). A quadrature rule \mathcal{R} on T is defined as a set of point and weight pairs: $\mathcal{R} = \{(p_i, w_i) \mid i = 1, \dots, n\}$, such that for any function $f(x)$ defined on a domain containing T and the points p_i , its integral on T can be approximated by:

$$\int_T f(x) dx \approx |T| \sum_{i=1}^n f(p_i) w_i, \quad (2.1)$$

where $n \in \mathbb{N}$ is the number of points, p_i are the quadrature points, w_i are the associated weights, $|T|$ denotes the area ($d = 2$) or volume ($d = 3$) of T .

A quadrature rule is said to be of (*algebraic*) order p if (2.1) is exact for all polynomials of degree not exceeding p . It is clear that if a quadrature rule is of order 0, then the sum of the weights must be equal to 1.

When dealing with a simplex it is often convenient to use barycentric coordinates. Let v_i , $i = 1, \dots, d+1$, be the vertices of T . Then the barycentric coordinates $(\xi_1, \dots, \xi_{d+1})$ of a point p with respect to T is determined by:

$$p = \sum_{i=1}^{d+1} \xi_i v_i \quad \text{and} \quad \sum_{i=1}^{d+1} \xi_i = 1.$$

Barycentric coordinates are invariant under affine transformations and $p \in T$ if and only if all its barycentric coordinates lie in the interval $(0, 1)$.

A quadrature rule \mathcal{R} is said to be *symmetric* if it is invariant under permutations of the barycentric coordinates. More precisely, let $(\xi_1, \dots, \xi_{d+1})$ be a quadrature point of \mathcal{R} associated with weight w , then for any permutation i_1, \dots, i_{d+1} of the indices $1, \dots, d+1$, the point $(\xi_{i_1}, \dots, \xi_{i_{d+1}})$ is also a quadrature point of \mathcal{R} associated with the same weight. For a symmetric quadrature rule, the set of quadrature points can be naturally divided into symmetry orbits, with each orbit containing all the points generated by permuting the barycentric coordinates of a single point. The symmetry orbits can be classified into a number of permutation stars, which are summarized in Tables 2.1 and 2.2, in which the notations for the permutation stars are from [11] which have the advantage of easily distinguishing stars on triangles and on tetrahedra.

3. The Numerical Algorithm

Denote by $\mathbb{P}^{(p)}$ the set of polynomials of degree less than or equal to p . The problem of finding an n -point quadrature rule of order p consists of finding the quadrature points p_i and

Table 2.1: Permutation stars on a triangle.

permutation star	barycentric coordinates	# of points
$S_3(\frac{1}{3})$	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	1
$S_{21}(a)$	$(a, a, 1 - 2a)$	3
$S_{111}(a, b)$	$(a, b, 1 - a - b)$	6

Table 2.2: Permutation stars on a tetrahedron.

permutation star	barycentric coordinates	# of points
$S_4(\frac{1}{4})$	$(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$	1
$S_{31}(a)$	$(a, a, a, 1 - 3a)$	4
$S_{22}(a)$	$(a, a, \frac{1}{2} - a, \frac{1}{2} - a)$	6
$S_{211}(a, b)$	$(a, a, b, 1 - 2a - b)$	12
$S_{1111}(a, b, c)$	$(a, b, c, 1 - a - b - c)$	24

weights w_i , $i = 1, \dots, n$, such that

$$|T| \sum_{i=1}^n f(p_i) w_i = \int_T f(x) dx, \quad \forall f(x) \in \mathbb{P}^{(p)}, \quad (3.1)$$

where T is a triangle or a tetrahedron. Eq. (3.1) can be regarded as a system of algebraic equations with p_i and w_i as unknowns, which can be solved using various numerical or analytical methods.

For symmetric quadrature rules, the set of n points is divided into symmetry orbits. Let us first consider the case of triangles. Suppose we can decompose n as $n = n_1 + 3n_2 + 6n_3$, where $n_1 = 0$ or 1 , $n_2 \geq 0$, $n_3 \geq 0$. Then the set of n points can be divided into n_1 S_1 orbit, n_2 S_{21} orbits, and n_3 S_{111} orbits. S_1 corresponds to a fixed point which is the barycenter of the triangle, points in S_{21} are determined by a single unknown abscissae, and points in S_{111} are determined by two unknown abscissas. For each symmetry orbit the weights are the same. Thus the total number of unknowns in (3.1) is reduced to $n_2 + 2n_3$ unknown abscissas and $n_1 + n_2 + n_3$ unknown weights. The symmetry can also be exploited to reduce the number of trial polynomials (number of equations) in (3.1) (see, e.g., [9]). In our case, we use monomials of the form $x_1^{k_1} x_2^{k_2}$ with $k_1 \leq k_2$ and $k_1 + k_2 \leq p$. Without loss of generality, we choose T to be the standard triangle $\{(x_1, x_2) \mid 0 \leq x_1 \leq 1, 0 \leq x_1 + x_2 \leq 1\}$. Using the formula

$$\int_T x_1^{k_1} x_2^{k_2} = \frac{k_1! k_2!}{(2 + k_1 + k_2)!},$$

we get a system of algebraic equations whose unknowns are $n_2 + 2n_3$ independent abscissas and $n_1 + n_2 + n_3$ weights. This system is turned into a nonlinear least square problem which is solved by calling the MINPACK routine `lmderr1` or `lmdif1` [10]. Both routines can find local minima of nonlinear least square problems but the solution found depends heavily on the initial guess. The difference between `lmderr1` and `lmdif1` is that the former requires the analytical Jacobian at each iteration, while the latter computes Jacobians using finite differences.

In the solution process, the $n_1 + n_2 + n_3$ weights can be regarded either as independent unknowns (in this case, we are solving a nonlinear least square problem with $n_1 + 2n_2 + 3n_3$ unknowns), or as dependent on the abscissas (in this case, we are solving a nonlinear least square problem with $n_2 + 2n_3$ unknowns in which a linear least square problem with $n_1 + n_2 + n_3$

unknowns is solved for computing the weights at each function evaluation of the nonlinear least square problem).

The case of tetrahedra is similar. The points are divided into n_1 S_4 orbit, n_2 S_{31} orbits, n_3 S_{22} orbits, n_4 S_{211} orbits, and n_5 S_{1111} orbits, with $n_1 + 4n_2 + 6n_3 + 12n_4 + 24n_5 = n$, where $n_1 = 0$ or 1 , $n_k \geq 0$, $k = 2, 3, 4, 5$. There are $n_2 + n_3 + 2n_4 + 3n_5$ unknown abscissas and $n_1 + n_2 + n_3 + n_4 + n_5$ unknown weights, and for the standard tetrahedron $T = \{(x_1, x_2, x_3) \mid 0 \leq x_1 \leq 1, 0 \leq x_1 + x_2 \leq 1, 0 \leq x_1 + x_2 + x_3 \leq 1\}$, we have:

$$\int_T x_1^{k_1} x_2^{k_2} x_3^{k_3} = \frac{k_1! k_2! k_3!}{(3 + k_1 + k_2 + k_3)!}.$$

The actual algorithm used for computing quadrature rules is described in Algorithm 3.1, in which a “good” solution means a solution with zero residual, positive weights, and points lying inside the domain \bar{T} . To speed up the search for a “good” solution, the algorithm is parallelized by computing with different initial guesses on different processes, and all processes exit once a “good” solution is found by any process.

Algorithm 3.1. *Algorithm for computing an n -point order p quadrature rule.*

for each decomposition $n = n_1 + 3n_2 + 6n_3$ (2d) or $n = n_1 + 4n_2 + 6n_3 + 12n_4 + 24n_5$ (3d) **do**

repeat

 (1) randomly choose an initial guess.

 (2) find a least square solution to (3.1) subject to the given decomposition.

 (3) if the solution is “good” then stop.

until maximum number of initial guesses tried.

end for

The probability to find a “good” solution with Algorithm 3.1 depends heavily on the way the initial guesses are chosen. An approach similar to [2] is used in our code which controls distribution of the initial guesses according to the extremal measure. Numerical comparison confirmed that this approach does reduce the average number of initial guesses tried before a “good” solution is found.

We have implemented Algorithm 3.1 in a C program. The source code can be freely downloaded at <http://lsec.cc.ac.cn/phg/download/quadrule.tar.bz2>. For convenience of users, a copy of necessary MINPACK files is included in the distribution. The program can use either `lmdcr1` or `lmdif1`, and with the weights as either dependent or independent variables, its behaviour can be easily changed by modifying some macro definitions at the top of the source file. It is possible to modify the program to compute other kinds of quadrature rules, such as rules with specific weight functions or on other types of integration domains.

4. Quadrature Rules

In this section we present the quadrature rules obtained by using the program described in Section 3. In the actual computations the weights were treated as dependent variables. The routine `lmdif1` was used because it seems to much more likely find a “good” solution than the routine `lmdcr1`, though the latter is much faster. The computations were performed using

S_{31}	.0861403311024363536537208740298857	.0058134382678884505495373338821455
S_{31}	.2087626425004322968265357083976176	.0196255433858357215975623333961715
S_{31}	.0141049738029209600635879152102928	.0003875737905908214364538721248394
S_{211}	.1021653241807768123476692526982584 .5739463675943338202814002893460107	.0116429719721770369855213401005552
S_{211}	.4075700516600107157213295651301783 .09222787013902013000000000000000	.0052890429882817131317736883052856
S_{211}	.0156640007402803585557586709578084 .7012810959589440327139967673208426	.0018310854163600559376697823488069
S_{211}	.2254963562525029053780724154201103 .4769063974420887115860583354107011	.0082496473772146452067449669173660
S_{1111}	.39059842812814580000000000000000 .2013590544123922168123077327235092 .0161122880710300298578026931548371	.0030099245347082451376888748208987
S_{1111}	.1061350679989021455556139029848079 .0327358186817269284944004077912660 .0035979076537271666907971523385925	.0008047165617367534636261808760312
S_{1111}	.5636383731697743896896816630648502 .2302920722300657454502526874135652 .1907199341743551862712487790637898	.0029850412588493071187655692883922
S_{1111}	.3676255095325860844092206775991167 .2078851380230044950717102125250735 .33121048851934490000000000000000	.0056896002418760766963361477811973
S_{1111}	.7192323689817295295023401840796991 .1763279118019329762157993033636973 .0207602362571310090754973440611644	.0041590865878545715670013980182613
S_{1111}	.5278249952152987298409240075817276 .4372890892203418165526238760841918 .0092201651856641949463177554949220	.0007282389204572724356136429745654
S_{1111}	.5483674544948190728994910505607746 .3447815506171641228703671870920331 .0867217283322215394629438740085828	.0054326500769958248216242340651926

5. Conclusion

We have presented a program for computing symmetric quadrature rules on triangles and tetrahedra. It is expected that these rules are useful for users developing finite element codes. The computations were done on the teracluster LSSC-II of the State Key Laboratory of Scientific and Engineering Computing, Chinese Academy of Sciences. For quadrature rules of modest orders (< 10), the program works quite well and can find solutions quickly, but it tends to be slower when the order becomes higher because the number of possible decompositions into symmetry orbits to try and the size of the system of polynomial equations to solve become large. For orders above 20 it is almost impossible to distinguish local and global minima of the nonlinear least square problems due to roundoff errors. During numerical computations many rules which have fewer points than those reported here, but have either negative weights or points outside of the integration domain, have been found and discarded. Through numerical computations we found that the way the initial guesses are chosen is essential for the efficiency and robustness of the algorithm, which is considered to be a possible future study subject.

Acknowledgments. The research is partially supported by the 973 Program under the grant 2005CB321702 and by China NSF under the grants 10531080 and 60873177.

References

- [1] H. Li, J. Sun and Y. Xu, Discrete fourier analysis, cubature and interpolation on a hexagon and a triangle, *SIAM J. Numer. Anal.*, to appear.
- [2] M.A. Taylor, B.A. Wingate and L.P. Bos, A cardinal function algorithm for computing multivariate quadrature points, *SIAM J. Numer. Anal.*, **45**:1 (2007), 193–205.
- [3] R. Cools, An encyclopaedia of cubature formulas, *J. Complex.*, **19**:3 (2003), 445–453.
- [4] S. Heo and Y. Xu, Constructing fully symmetric cubature formulae for the sphere, *Math. Comput.*, **70**:233 (2001), 269–279.
- [5] P. Hammer and A. Stroud, Numerical integration over simplexes, *Mathematical Tables and Aids to Computation*, **10** (1956), 137–139.
- [6] A. Stroud, Approximate Calculation of Multiple Integrals, Prentice-Hall, Englewood Cliffs, N.J., 1971.
- [7] A. Stroud and D. Secrest, Gaussian Quadrature Formulas, Prentice-Hall, Englewood Cliffs, N.J., 1966.
- [8] P. Solin, K. Segeth and I. Dolezel, Higher-Order Finite Element Methods, Chapman and Hall/CRC Press, 2003.
- [9] S. Wandzura and H. Xiao, Symmetric quadrature rules on a triangle, *Comput. Math. Appl.*, **45** (2003), 1829–1840.
- [10] J. More, B. Garbow and K. Hillstom, MINPACK, <http://www.netlib.org:80/minpack/>.
- [11] C.A. Felippa, A compendium of FEM integration rules for finite element work, *Eng. Computation*, **21** (2004), 867–890.
- [12] P.A. Nelson, GNU bc, <http://www.gnu.org/software/bc/>.
- [13] L.B. Zhang, Parallel Hierarchical Grid, http://lsec.cc.ac.cn/phg/index_en.htm/.