Journal of Computational Mathematics, Vol.27, No.1, 2009, 89–96.

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: $\mathcal{R} = \{(p_i, w_i) \mid i = 1, \cdots, n\}$, 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:

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

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, $i = 1, \cdots, d+1$, be the vertices of T. Then the barycentric coordinates $(\xi_1, \cdots, \xi_{d+1})$ 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 $(\xi_1, \cdots, \xi_{d+1})$ be a quadrature point of $\mathcal{R}$ associated with weight w, then for any permutation $i_1, \cdots, i_{d+1}$ of the indices $1, \cdots, d+1$, the point $(\xi_{i_1}, \cdots, \xi_{i_{d+1}})$ 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, $i = 1, \cdots, n$, such that

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

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 `lmder1` 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 `lmder1` 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 `lmder1` 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 `lmder1`, though the latter is much faster. The computations were performed using

double precision, and the results were post-processed by applying a few Newton iterations to them using a program written in GNU bc [12] to bring their precision to 34 digits.

The numbers of points for the quadrature rules found are listed in Tables 4.1 and 4.2, they are either smaller than, equal to, or comparable to the numbers of points of the quadrature rules available in the literature.

Table 4.1: Numbers of points for the quadrature rules on triangles.

Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
# points	1	3	6	6	7	12	15	16	19	25	28	33	37	46	52	55	61	72	73	88	91

Table 4.2: Numbers of points for the quadrature rules on tetrahedra.

Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# points	1	4	8	14	14	24	36	46	61	81	109	140	171	236

In Sections 4.1 and 4.2 some sample rules are listed. An electronic form of the complete list of rules, including orders 1–21 on triangles and orders 1–14 on tetrahedra, which is suitable for use in C or C++ codes, can be found in the file `src/quad.c` in the source distribution of the parallel adaptive finite element toolbox PHG [13] which is distributed under the GNU Lesser General Public License. The macros for defining the rules can be found in the file `include/quad.h`. In fact, the quadrature rules listed here were automatically extracted from `src/quad.c` using a Bash script.

4.1. Quadrature rules on triangles

16-point order 8 rule on triangle		
Orbit	Abscissas	Weight
S_3	.33333333333333333333333333333333	.14431560767778716825109111104890646
S_{21}	.17056930775176020662229350149146 45	.10321737053471825028179155029212 90
S_{21}	.05054722831703097545842355059659 89	.03245849762319808031092592834178 06
S_{21}	.45929258829272315602881551449416 93	.09509163426728462479389610438858 43
S_{111}	.26311282963463811342178578628464 36	.02723031417443499426484469007390 89
	.00839477740995760533721383453929 44	
46-point order 14 rule on triangle		
Orbit	Abscissas	Weight
S_3	.33333333333333333333333333333333	.05859628522602859412789380634775 60
S_{21}	.00997976080645843241529352958205 24	.00173515122972526756806186388080 94
S_{21}	.47997789352118838981055286508838 99	.02616378255861452177782885918197 83
S_{21}	.15381195917696690000000000000000 00	.00391972924240182909652082757014 54
S_{21}	.07402347711698781000000000000000 00	.01224735975694086609728698992625 05
S_{21}	.13035468250333000000000000000000 00	.02819962850325796010736630715156 57
S_{21}	.23061722602665313429960537009838 31	.05088708718595948529603482754545 40
S_{21}	.42233208341914782411440871379139 39	.05045343990160359919102089713411 89
S_{111}	.78623738593466100332962211403309 00	.01706364421223345129002539938494 72
	.19061636003190090424614328286530 34	
S_{111}	.63055214366060744162240907556881 29	.00968346642550660040752096309341 94
	.36232313774354714461832673435977 29	
S_{111}	.62657732985630631423351231375342 65	.03638575592848500562201132776427 17
	.29077120588366741502481681748167 32	

S_{111}	.914209984929625412239967099385046 9 .071165710877750762547592450292433 6	.006964663373518412425399722504241 3
88-point order 20 rule on triangle		
Orbit	Abscissas	Weight
S_3	.333333333333333333333333333333333 3	.012537607994496565735856367723948
S_{21}	.215874305932991973190254543840182 8	.027471869876424213748453549607359 8
S_{21}	.075376766529747278097285430945916 3	.009765272277051423041364691429423 7
S_{21}	.010300828137221792113686216009696 9	.001398419535391823523923363159786 7
S_{21}	.493602211298700165511920832145053 6	.009292102625185182630428203403033 0
S_{21}	.461550938106925296741048710291518 0	.016577876032366925326023625035184 0
S_{111}	.328621406424236993303497460950913 3 .429340570258210375213958800466398 4	.020667762348665076961421970012972 9
S_{111}	.260480361786568756419593017081153 5 .101577534280969446168755006196179 7	.020822235521154507306878556199329 7
S_{111}	.137074235846455300000000000000000 0 .710065973001130159987904074564407 9	.009568638419849060688875845045832 0
S_{111}	.146726945872299784304160988487453 0 .498545477678414849389622696707611 9	.024452770968972463885643920702408 9
S_{111}	.026998977742553290000000000000000 0 .049186722672582001619703712577587 2	.003155730630630534003826400320729 6
S_{111}	.061871785933617026841712470012233 9 .779660146540569395360350619076810 8	.012136796365321296937013309080757 4
S_{111}	.047724367427621996208352680104293 4 .370491539149547636920149620256738 8	.014966480143886449036524911851570 7
S_{111}	.120600515186364379967233787040079 4 .863346948754752648497987996092521 7	.006327593321777739569324032750439 8
S_{111}	.002697147796709787671648914501282 7 .056194938187745502987892301986588 7	.001342560312063695884979851298143 3
S_{111}	.003015633277942362657276259823471 0 .208675006748421350957594463061357 7	.002776076916347554067729356155801 5
S_{111}	.029905375788457018806928773864338 6 .721151240912034091028104150205094 1	.010739844474184941555173447447951 7
S_{111}	.006756654222460988539945817519227 8 .640055441940541889904053668272164 7	.005367805738187453205247410021269 7

4.2. Quadrature rules on tetrahedra

46-point order 8 rule on tetrahedron		
Orbit	Abscissas	Weight
S_{31}	.039675423070389901265071329539389 5	.006397147779902321321451420335173 0
S_{31}	.314487800698096313784160562697148 3	.040190448020966172488161158479817 8
S_{31}	.101986693062703300000000000000000 0	.024307975504770321174869108771922 6
S_{31}	.184203696949191512275946417348909 2	.054858892413697440466924123990391 4
S_{22}	.063436287754539892405141238701898 3	.035719612234099182464950968996617 6
S_{211}	.021690162067728004802662482624930 2 .719931922039465935889434953352734 8	.007183190697852539409451105219803 8
S_{211}	.204480080636795714241335574872745 3 .580577190128809224175398171390620 4	.016372181945319117540938139756119 1
236-point order 14 rule on tetrahedron		
Orbit	Abscissas	Weight
S_{31}	.327253362523848563909309669268528 9	.004065113665270767043620883683563 6
S_{31}	.044761304466685080883794209647884 2	.002214538533445578143759956950007 1

S_{31}	.08614033110243635365372087402988 57	.00581343826788845054953733388214 55
S_{31}	.20876264250043229682653570839761 76	.01962554338583572159756233339617 15
S_{31}	.01410497380292096006358791521029 28	.00038757379059082142364538721248 394
S_{211}	.10216532418077681234766925269825 84 .57394636759433382028140028934601 07	.01164297197217703698552134010055 52
S_{211}	.40757005166001071572132956513017 83 .09222787013902013000000000000000 00	.00528904298828171313177368830528 56
S_{211}	.01566400074028035855575867095780 84 .70128109595894403271399676732084 26	.00183108541636005593766978234880 69
S_{211}	.22549635625250290537807241542011 03 .47690639744208871158605833541070 11	.00824964737721464520674496691736 60
S_{1111}	.39059842812814580000000000000000 00 .20135905441239221681230773272350 92 .01611228807103002985780269315483 71	.00300992453470824513768887482089 87
S_{1111}	.10613506799890214555561390298480 79 .03273581868172692849440040779126 60 .00359790765372716669079715233859 25	.00080471656173675346362618087603 12
S_{1111}	.56363837316977438968968166306485 02 .23029207223006574545025268741356 52 .19071993417435518627124877906378 98	.00298504125884930711876556928839 22
S_{1111}	.36762550953258608440922067759911 67 .20788513802300449507171021252507 35 .33121048851934490000000000000000 00	.00568960024187607669633614778119 73
S_{1111}	.71923236898172952950234018407969 91 .17632791180193297621579930336369 73 .02076023625713100907549734406116 44	.00415908658785457156700139801826 13
S_{1111}	.52782499521529872984094240075817 276 .43728908922034181655262387608419 18 .00922016518566419494631775549492 20	.00072823892045727243561364297456 54
S_{1111}	.54836745449481907289949105056077 46 .34478155061716412287036718709203 31 .08672172833222153946294387400858 28	.00543265007699582482162423406519 26

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. Hillstrom, 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/`.