Efficient computation of cubature rules with application to new asymmetric rules on the triangle

Efficient computation of cubature rules with application to new asymmetric rules on the triangle

Stefanos-Aldo Papanicolopulos[a,*]

[a] *Institute for Infrastructure & Environment, School of Engineering, The University of Edinburgh, Edinburgh, EH9 3JL, UK*

Abstract

This paper presents a new, efficient method for computing cubature rules, based on least-squares minimisation and the use of orthogonal bases. The method, which can be applied for any integration domain, is tested here for the case of asymmetric cubature rules on the triangle showing how the computation of the necessary basis, and its derivatives, can be optimised. The numerical results presented include three new cubature rules with fewer points than known rules of the same degree.

Keywords: Cubature, triangle, asymmetric rules, orthogonal bases, least-squares minimisation
2000 MSC: Primary 65D32, Secondary 65D30

1. Introduction

The numerical computation of a multiple integral over a given domain ("cubature") is of interest to many different disciplines. A classical reference on cubature is the book of Stroud [1], with further information to be found, among others, in [2] and in chapter 6 of [3]. An extensive list of (then) known cubature formulas (or "rules") can be found in [4, 5] and online at the Encyclopaedia of Cubature Formulas [6]. Many other results have been published in the literature since; in the case of cubature on the triangle, for example, these include [7–13].

Cools [2] has discussed the "science behind the art" of obtaining cubature rules. In most practical cases, cubature rules are obtained by the iterative, numerical solution of a set of non-linear equations, taking into account possible symmetries. Considering again the example of cubature on the triangle, early results obtained in this way include among others [14–16], while different methods to obtain cubature rules were successfully used in [7, 8, 12].

This paper presents a new, efficient method for computing cubature rules using least-squares minimisation to iteratively solve a set of polynomial equations. The method uses concepts introduced for cubature rules in [7, 9, 13] together with the least-squares method of Golub and Pereyra [17] to increase the efficiency and robustness of the iterative solutions. While the proposed method has general applicability to any domain and imposed symmetry, it is here further applied to the specific case of asymmetric cubature rules on the triangle. This shows how significant additional efficiency can be obtained by optimising the computation of the orthogonal basis used and its derivatives, and also provides new cubature rules of theoretical interest.

The structure of the paper is as follows: Section 2 presents a brief overview of how the non-linear equations for cubature rules are obtained. Section 3 gives a summary of the least squares method of Golub and Pereyra [17], which is further simplified in Section 4 when obtaining cubature rules using orthogonal bases. Section 5 examines the specific case of asymmetric cubature rules on the triangle, presenting an efficient way of computing the required matrices, resulting in a number of new rules that are presented in Section 6. Section 7 contains a discussion of the new results in this paper, while Section 8 presents the conclusions.

[*]Corresponding author. Tel.: +44 (0)131 650 7214; Fax: +44 (0)131 650 6554.
Email address: `S.Papanicolopulos@ed.ac.uk` (Stefanos-Aldo Papanicolopulos)

2. Theoretical background

A cubature rule approximates the integral of a function f on a domain Ω (normalised by the domain's measure) as the weighted sum of the function's value evaluated at a set of n_p points $\boldsymbol{x}_i$,

$$\int_\Omega f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \Big/ \int_\Omega \mathrm{d}\boldsymbol{x} \approx \sum_j^{n_p} w_j f(\boldsymbol{x}_j) \tag{1}$$

The cubature rule is of polynomial degree ϕ if equation (1) is exact for all polynomials (in the coordinates $\boldsymbol{x}$) of degree up to ϕ but not exact for at least one polynomial of degree $\phi + 1$.

Since equation (1) is linear in the function f, to obtain a rule of degree (at least) ϕ we only need to ensure that it is exact for a basis of the polynomials of degree ϕ,

$$\mathcal{P}^\phi = \{\psi_i(\boldsymbol{x}) \mid i = 1 \dots n_b\} \tag{2}$$

where we don't show the dependence of ψ_i and n_b on ϕ. Equation (1) therefore becomes

$$\sum_j^{n_p} w_j \psi_i(\boldsymbol{x}_j) = \int_\Omega \psi_i(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \Big/ \int_\Omega \mathrm{d}\boldsymbol{x} \qquad i = 1 \dots n_b \tag{3}$$

which is a non-linear system of equations which must be solved for the coordinates $\boldsymbol{x}_j$ and the weights w_j.

Introducing the $n_b \times n_p$ matrix $\boldsymbol{A}$ and the n_b-element vector $\boldsymbol{b}$ defined as

$$a_{ij} = \psi_i(\boldsymbol{x}_j) \tag{4}$$

$$b_i = \int_\Omega \psi_i(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \Big/ \int_\Omega \mathrm{d}\boldsymbol{x} \tag{5}$$

the non-linear system (3) can be written as

$$\boldsymbol{r} = \boldsymbol{A}(\boldsymbol{x}_j)\boldsymbol{w} - \boldsymbol{b} = \boldsymbol{0} \tag{6}$$

where we have explicitly shown the dependence of $\boldsymbol{A}$ on $\boldsymbol{x}_j$.

Equation (6) can be solved for the unknowns $\boldsymbol{x}_j$ and $\boldsymbol{w}$ by minimising the square of the norm of the vector $\boldsymbol{r}$ with respect to these unknowns and checking that the minimum value is approximately zero. While a suitable least-squares minimisation algorithm can be directly applied to the non-linear system (6), a significant reduction in the number of unknowns can be obtained by exploiting the fact that the system is actually linear in the weights $\boldsymbol{w}$. In the context of obtaining cubature rules, this has been for example done by Zhang et al. [9] and Witherden and Vincent [13] for minimisation problems, and previously by Taylor et al. [18].

3. The minimisation problem

Golub and Pereyra [17] have presented a general method that can be used to solve the system (6) for the unknowns $\boldsymbol{x}_j$ and $\boldsymbol{w}$. This section provides a simplified summary of this method, assuming that the matrix $\boldsymbol{A}$ always has full column rank and that the system has a solution.

As equation (6) is linear in the weights $\boldsymbol{w}$, we eliminate them by computing them as a linear least squares solution of the system. Indicating with $\boldsymbol{A}^+$ the Moore-Penrose pseudoinverse of $\boldsymbol{A}$, we get

$$\boldsymbol{w} = \boldsymbol{A}^+\boldsymbol{b} \tag{7}$$

Substituting (7) into (6) we obtain the non-linear system

$$\boldsymbol{r} = (\boldsymbol{A}\boldsymbol{A}^+ - \boldsymbol{I})\boldsymbol{b} = \boldsymbol{0} \tag{8}$$

which can be solved by minimising the norm of $\boldsymbol{r}$.

Setting

$$M = I - AA^+ \tag{9}$$

we see that $M = M^T$ and $M^2 = M$, and therefore

$$r = -Mb \tag{10}$$

and

$$\|r\|^2 = r^T r = b^T M b \tag{11}$$

where $\|\cdot\|$ is the Euclidean norm.

To obtain the pseudoinverse A^+ we compute the thin QR decomposition of A so that

$$A = QR \tag{12}$$

where Q is $n_b \times n_p$ with $Q^T Q = I$ and R is an upper triangular $n_p \times n_p$ matrix. The pseudoinverse is then given by

$$A^+ = R^{-1} Q^T \tag{13}$$

so that M is computed as

$$M = I - QQ^T \tag{14}$$

If A (and therefore M) depend on a vector of variables v then to minimise r we need, for some algorithms, the Jacobian J given by

$$[J]_{is} = \frac{\partial r_i}{\partial v_s} \tag{15}$$

The Jacobian can be approximately computed using finite differences, but it can also be expressed explicitly in an exact form as

$$[J]_{is} = \sum_{j,k,l} \left(m_{ij} \frac{\partial a_{jk}}{\partial v_s} a^+_{kl} b_l + a^+_{ki} \frac{\partial a_{jk}}{\partial v_s} m_{jl} b_l \right) \tag{16}$$

Starting from equations (15), (10) and (9), equation (16) is directly obtained by expressing equation (5.4) in [17] using the notation of the present paper and using index notation to avoid ambiguity due to the presence of the three-index term $\partial a_{jk}/\partial v_s$. Note that, as r is defined here with the opposite sign from that used in [17], equation (16) for the Jacobian has the opposite sign from equation (5.4) in [17].

4. Using an orthogonal polynomial basis

The procedure in section 3 can be directly used to solve the non-linear system (6) to obtain cubature rules, by choosing any appropriate polynomial basis. While the simplest basis consists of monomials, its numerical behaviour is very poor. We therefore choose the polynomial basis $\mathcal{P}^\phi$ to be orthogonal in the domain Ω [7, 10, 13]. While more expensive to compute than the monomial basis, the orthogonal basis has several advantages as will be seen in the following.

By definition, for an orthogonal basis we have

$$\int_\Omega \psi_i(x)\psi_j(x)\mathrm{d}x = c_i \delta_{ij} \tag{17}$$

Assuming ψ_1 to be the constant element of the basis we get

$$\int_\Omega \psi_1 \psi_1 \mathrm{d}x = c_1 \quad \Rightarrow \quad \int_\Omega \mathrm{d}x = c_1 \psi_1^{-2} \tag{18}$$

and therefore

$$b_i = \int_\Omega \psi_i(x)\mathrm{d}x \Big/ \int_\Omega \mathrm{d}x = \psi_1 \delta_{i1} \tag{19}$$

We therefore need to solve the system

$$\bar{\boldsymbol{r}} = -\boldsymbol{M}\boldsymbol{e}_1 = 0 \tag{20}$$

where $\boldsymbol{e}_1 = \{1, 0, 0, \dots\}^T$, and then compute the weights

$$\boldsymbol{w} = \psi_1 \boldsymbol{A}^+ \boldsymbol{e}_1 \tag{21}$$

Note that in index notation $\bar{r}_i = -m_{i1}$ and $\|\bar{\boldsymbol{r}}\|^2 = m_{11} = -\bar{r}_1$.

The Jacobian is given by

$$[J]_{is} = \sum_{k,j} \left(m_{ij} \frac{\partial a_{jk}}{\partial v_s} a^+_{k1} + a^+_{ki} \frac{\partial a_{jk}}{\partial v_s} m_{j1} \right) \tag{22}$$

In a μ-dimensional space, the vector $\boldsymbol{v}$ of the unknowns can be written as

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(\mu)} \end{bmatrix} \tag{23}$$

where $\boldsymbol{x}^{(\alpha)}$ is the vector of all the coordinate components in the α dimension. Therefore,

$$\boldsymbol{J} = \begin{bmatrix} \dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}^{(1)}} & \dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}^{(2)}} & \cdots & \dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}^{(\mu)}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}^{(1)} & \boldsymbol{J}^{(2)} & \cdots & \boldsymbol{J}^{(\mu)} \end{bmatrix} \tag{24}$$

Since

$$\frac{\partial a_{jk}}{\partial x^{(\alpha)}_s} = \frac{\partial \psi_j(\boldsymbol{x}_k)}{\partial x^{(\alpha)}_s} = \left. \frac{\partial \psi_j(\boldsymbol{x})}{\partial x^{(\alpha)}} \right|_{\boldsymbol{x}=\boldsymbol{x}_s} \delta_{ks} \tag{25}$$

we can define the matrices $\boldsymbol{G}^{(\alpha)}$ as

$$g^{(\alpha)}_{js} = \left. \frac{\partial \psi_j(\boldsymbol{x})}{\partial x^{(\alpha)}} \right|_{\boldsymbol{x}=\boldsymbol{x}_s} \tag{26}$$

so the individual $\boldsymbol{J}^{(\alpha)}$ matrices are then given by

$$[J^{(\alpha)}]_{is} = \sum_j \left(m_{ij} g^{(\alpha)}_{js} a^+_{s1} + a^+_{si} g^{(\alpha)}_{js} m_{j1} \right) \tag{27}$$

In the present paper summation over an index is explicitly indicated by the summation symbol, with no summation being implied for repeated indices. Therefore in equation (27) no summation is implied over s.

It is worth noting that the coordinates $\boldsymbol{x}$ are not necessarily Cartesian, a fact that will be further exploited in the following section.

5. Calculation of basis polynomials and their derivatives

This section presents how the polynomials of an orthogonal basis, and their derivatives, can be efficiently calculated. We consider a triangular domain, for which a standard orthonormal polynomials basis has been proposed in the literature [19–21]. For degree ϕ we can write this basis in the form

$$\psi_{ij} = \sqrt{2(2i+1)(i+j+1)} P_i(d/s) P^{(2i+1,0)}_j (1-2s) s^i \tag{28}$$

where $i = 0, \dots, \phi$, $j = 0, \dots, \phi - i$. The functions P_i and P_j are Jacobi polynomials and the quantities d and s can be written in terms of the barycentric (or areal) coordinates as

$$s = L_2 + L_1, \qquad d = L_2 - L_1 \tag{29}$$

In equation (28) we express in a natural way the basis polynomials ψ using two indices. It is however easy to revert to the single-index notation used elsewhere in this paper, for example by setting

$$\psi_k = \psi_{ij} \qquad \text{with } k = i(\phi + (3 - i)/2) + j + 1 \tag{30}$$

For the needs of the algorithm presented in Section 4, we need to compute the values of all the basis polynomials, and possibly their derivatives, for all (trial) cubature points. While it is possible to do so directly by using equation (28) and any library for computing Jacobi polynomials and their derivatives, it is very inefficient to do so. This happens because Jacobi polynomials are computed through recurrence relations, so that individual computation of each Jacobi polynomial would lead to multiple computations of the same intermediate results. We therefore show in the following how the basis polynomials can be computed using recurrence relations.

Setting

$$Q_i(d, s) = P_i(d/s)s^i \tag{31}$$

and

$$R_{i,j}(s) = \sqrt{2(2i + 1)(i + j + 1)}P_j^{(2i+1,0)}(1 - 2s) \tag{32}$$

we can write

$$\psi_{ij} = Q_i(d, s)R_{i,j}(s) \tag{33}$$

From the recurrence relation for the Jacobi (actually Legendre) polynomial P_i

$$P_0(x) = 1, \quad P_1(x) = x, \quad iP_i(x) = (2i - 1)xP_{i-1}(x) - (i - 1)P_{i-2}(x) \tag{34}$$

we easily obtain

$$Q_0 = 1, \quad Q_1 = d, \quad iQ_i = (2i - 1)dQ_{i-1} - (i - 1)s^2 Q_{i-2} \tag{35}$$

The partial derivatives of Q_i with respect to d and s can be computed either by differentiating the relations (35) or by using the formula

$$(1 - x^2)P_i'(x) = -ixP_i(x) + iP_{i-1}(x) \tag{36}$$

Combining the two approaches results in the following simple formulas

$$\frac{\partial Q_0}{\partial d} = 0, \qquad \frac{\partial Q_i}{\partial d} = iQ_{i-1} + d\frac{\partial Q_{i-1}}{\partial d} \tag{37}$$

$$\frac{\partial Q_0}{\partial s} = 0, \qquad \frac{\partial Q_i}{\partial s} = -s\frac{\partial Q_{i-1}}{\partial d} \tag{38}$$

The recurrence relation for $R_{i,j}(s)$ can be easily computed from the well-known recurrence relation for Jacobi polynomials, obtaining

$$R_{i,0} = \sqrt{2(2i + 1)(i + 1)} \tag{39a}$$

$$R_{i,1} = (c_{i,1}^{(1)} + c_{i,1}^{(2)}s)R_{i,0} \tag{39b}$$

$$R_{i,j} = (c_{i,j}^{(1)} + c_{i,j}^{(2)}s)R_{i,j-1} + c_{i,j}^{(3)}R_{i,j-2} \tag{39c}$$

where

$$c_{i,j}^{(1)} = \sqrt{(i + j + 1)(i + j)}\frac{4(2i^2 + 2ij + j^2 + i)}{j(2i + j + 1)(2i + 2j - 1)} \tag{40a}$$

$$c_{i,j}^{(2)} = -\sqrt{(i + j + 1)(i + j)}\frac{2(2i + 2j + 1)}{j(2i + j + 1)} \tag{40b}$$

$$c_{i,j}^{(3)} = -\sqrt{\frac{i + j + 1}{i + j - 1}}\frac{(2i + j)(j - 1)(2i + 2j + 1)}{j(2i + j + 1)(2i + 2j - 1)} \tag{40c}$$

The quantities in equations (40) can be pre-calculated as they only depend on the indices i and j.

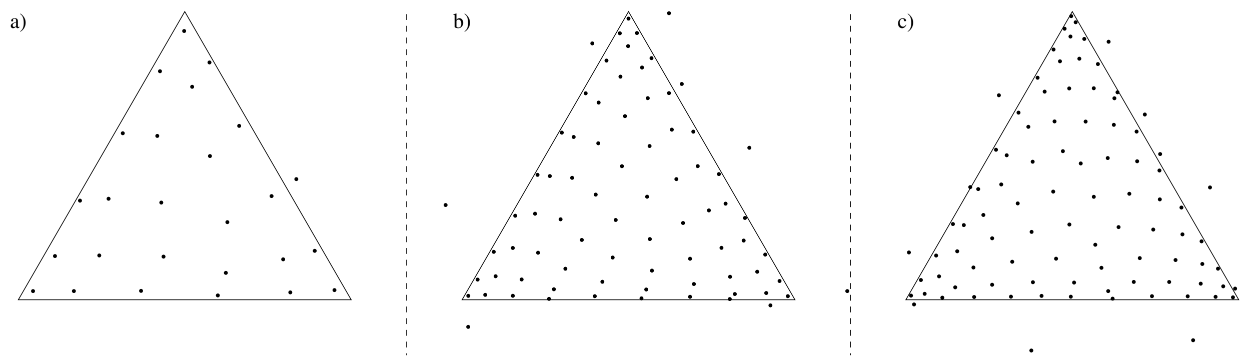

Figure 1: Computed new asymmetric cubature rules on the triangle: a) degree 11, 26 points, PO quality; b) degree 20, 77 points, PO quality; c) degree 22, 92 points, PO quality

The basis polynomials are therefore computed by equation (33), and their derivatives with respect to d are easily calculated as

$$\frac{\partial \psi_{ij}}{\partial d} = \frac{\partial Q_i}{\partial d} R_{i,j} \tag{41}$$

The derivatives with respect to s are obtained by multiplying equations (39) on both sides with Q_i and using equation (33), to obtain recurrence relations for the $\psi_{i,j}$, and then differentiating with respect to s to obtain the recurrence relations

$$\frac{\partial \psi_{i,0}}{\partial s} = \frac{\partial Q_i}{\partial s} \sqrt{2(2i+1)(i+1)} \tag{42a}$$

$$\frac{\partial \psi_{i,1}}{\partial s} = (c_{i,1}^{(1)} + c_{i,1}^{(2)} s) \frac{\partial \psi_{i,0}}{\partial s} + c_{i,1}^{(2)} \psi_{i,0} \tag{42b}$$

$$\frac{\partial \psi_{i,j}}{\partial s} = (c_{i,j}^{(1)} + c_{i,j}^{(2)} s) \frac{\partial \psi_{i,j-1}}{\partial s} + c_{i,j}^{(2)} \psi_{i,j-1} + c_{i,j}^{(3)} \frac{\partial \psi_{i,j-2}}{\partial s} \tag{42c}$$

The equations presented in this section, though clearly not the only possible way to compute the basis polynomials and their derivatives, have been obtained by attempting to minimise the number of computations required.

6. Numerical results

The algorithm presented in this paper, for the case of asymmetric cubature on the triangle, has been implemented in Fortran 95. The initial guess for the coordinates of the cubature points was obtained by considering a uniform random distribution of points inside the triangle.

Using this code, three new cubature rules were found, improving in terms of number of points on previously available results of the same degree. These new rules are shown graphically in Figure 1 and tabulated in Appendix A. Arbitrary precision arithmetic has been used to refine the Fortran results, which are calculated in double-precision but suffer from loss of accuracy due to all the intermediate calculations, to full double-precision accuracy (i.e. the maximum accuracy provided by double-precision numbers).

The first rule is an asymmetric cubature rule of degree 11 with 26 points and of PO quality (all weights positive but one point outside the triangle). Though the presence of a cubature point outside the domain of integration decreases its practical importance, this rule represents an important and long-missing result as all previously available rules had more points [22]. This (degree 11) is the lowest degree for which a rule improving on the number of points used (independently of quality) has been presented for the triangle since the degree seven rule by Gatermann [23]. The other two rules, for degrees 20 and 22 with 77 and 92 points respectively, are also of PO quality but with more points outside the triangle. Once more, they improve in terms of number of points on all previously known cubature rules.

In all three new rules computed here, the number of points is of the form $n_p = 3k + 2$, while it is easily seen that rules possessing rotational or full symmetry cannot have such a number of points. It is therefore necessary to consider asymmetric rules (or rules which only exhibit symmetry along one median) to obtain rules with this number of points. Rotationally symmetric rules of PI quality (that is with positive weights and all points inside the triangle), with one more point for each degree than the rules presented here, have been computed by Xiao and Gimbutas [10].

From the tabulated results in Appendix A, it is easily seen that points outside the triangle are associated with smaller weights, with the weight decreasing with increasing distance. For each combination of degree and number of points presented here, a number of different rules were actually computed. The rules presented here are those with the maximum smallest weight, and therefore with outside points closest to the triangle.

7. Discussion

The method presented in this paper shares common characteristics with those presented by Zhang et al. [9] and Witherden and Vincent [13] (hereinafter referred to respectively as ZCL and WV), in that all methods consider the weights as dependent variables and then solve the resulting problem using non-linear least squares. To achieve efficiency without compromising its robustness, the proposed method differs from ZCL and WV in the following points:[1]

1. Use of an orthogonal basis. This was already done in WV, and previously by Taylor et al. [7, 18], but not in ZCL. The major difference in this paper is that we minimise the number of computations (and associated computation time) by recognising that all basis polynomials need to be computed at the same time and for all cubature points. Section 5 shows how this can be exploited in the case of the triangle. Optimisations based on array storage order and cache size are not considered here, as they are language- and hardware-dependent respectively.
2. Another benefit of using an orthogonal basis is that the vector $\boldsymbol{e}_1$ appears instead of $\boldsymbol{b}$, so that instead of computing the matrix-vector products, only the appropriate columns (or rows) of the relevant matrices are used (compare for example equations (16) and (27)).
3. Analytical computation of the Jacobian. ZCL state that numerical differentiation seems much more likely to find a good solution compared to analytical differentiation, which is however much faster. In informal testing while obtaining the numerical results of Section 6, however, using the analytical Jacobian was much faster without increasing the number of attempts; this may be because of the better numerical behaviour obtained using an orthogonal basis. Note that efficient computation of the Jacobian requires an efficient computation of the derivatives of the basis polynomials, as for example done in Section 5.
4. Avoiding explicit computation of the weights. By closely following the method of Golub and Pereyra [17], we avoid the explicit computation of the weights, which is carried out in both ZCL and WV, and significantly decrease the number of computations required (which decreases not only computation time but also numerical error). Note that if numerical differentiation is used to compute the Jacobian, then only the $\boldsymbol{Q}$ matrix is needed and $\boldsymbol{R}$ is never used and thus never inverted. To compute the Jacobian analytically, $\boldsymbol{R}$ is implicitly inverted to obtain the pseudoinverse $\boldsymbol{A}^+$, the first column of which is actually the vector of the weights (scaled by ψ_1). Even in this case, however, unnecessary intermediate results don't need to be stored.
5. Use of QR factorisation to eliminate the weights. The use of a QR decomposition, as proposed by Golub and Pereyra [17], was shown in our testing to be robust enough numerically, while it is faster than singular value decomposition (SVD) as used in WV. Like SVD, QR decomposition allows for the computation of $\boldsymbol{M}$ in a simple way, without the need to compute the pseudoinverse $\boldsymbol{A}^+$, unlike in the case of the LU decomposition with column pivoting used in ZCL.

The proposed method can be applied to different integration domains, and can also take into account different symmetries of the required cubature rules (either because symmetry is a required characteristic for a given application, or simply to allow a solution to be found by decreasing the number of equations and of unknowns). As mentioned above, however, computational efficiency depends on an appropriate method for computation of the orthogonal basis polynomials. Additionally, imposing symmetry leads to more complex formulas for evaluating the Jacobian $\boldsymbol{J}$, as the

[1]While the ZCL and WV papers only give an overview of the algorithms used, they provide links to source code implementing the respective methods. This comparison therefore is based not only on the published papers, but also on the author's understanding of the source code for both methods, in the version available at the time of the preparation of the present paper.

different types of possible “orbits” must be dealt with in a different way; this complexity however does not translate into significant additional computational cost.

Applying the present method in the case of asymmetric rules on the triangle allows therefore for a simple implementation, as no symmetry is imposed, while at the same time demonstrating how the computation of the orthogonal basis can be efficiently implemented. Additionally, the results obtained and presented in Section 6 are new and of theoretical interest, and thus relevant on their own. Indeed, very few previous works have considered asymmetric cubature rules on the triangle [7, 8], even though such rules are a superset of all symmetric rules and will therefore include the rule with the minimal number of points for a given quality and degree.

The theoretical comparison of the proposed algorithm to the ZCL and WV algorithms provides strong indications that the first one should outperform the other two for any kind of integration domain and imposed symmetry. To further demonstrate the efficiency of the proposed algorithm, the implementation provided for the WV algorithm was modified to allow computation of asymmetric rules on the triangle. Searching for a 26 point rule of degree 11, the average performance for the proposed algorithm was 0.265 seconds per trial rule (i.e. for each initial random guess) while for the WV algorithm it was 2.308 seconds per trial rule. Even more importantly, the proposed algorithm obtained 6 valid rules for 80 tested trial rules, while the WV algorithm obtained 2 valid rules for 307 tested trial rules. While clearly not rigorous, this comparison provides further support for the claim of efficiency of the proposed algorithm.

The performance of the algorithm and its implementation is such that for the 92-point rule of degree 22 a solution was obtained within a few hours, using a single CPU core (as noted above, multiple solutions were obtained to keep the one with outside points closest to the triangle). There is however clearly room for improvement, if the method is to be applied to obtain rules of higher degree, or for three- (and perhaps even higher-) dimensional regions. The main areas where improvements to the implementation could be made are as follows:

1. Implementing large-scale parallelisation either by completely independent parallel runs with different initial guesses (as done in ZCL and WV) or by employing an appropriate global optimisation solver.
2. Using a constrained minimisation algorithm to enable searching only for cubature rules with all points within the integration domain, instead of checking the quality of each rule that is obtained by unconstrained minimisation. This may also involve using a different minimisation algorithm than the trust-region least-squares algorithm used in the present paper.
3. Using better-than-random initial values for the minimisation algorithms, as done in ZCL and also for example by Wandzura and Xiao [24] and by Xiao and Gimbutas [10]. Note however that WV prefer using a random set of initial values due to the simplicity of the method and the almost null computational cost of obtaining the initial values.

It should be noted however that the above ideas for improvement consider only implementation aspects, with the overall algorithm remaining the same as presented in this paper.

8. Conclusions

We have presented a method for the computation of cubature rules using an orthogonal basis and a least-squares minimisation algorithm. While similar methods have been previously proposed in the literature, the present method achieves greater efficiency, by starting with the theoretical treatment of the problem by Golub and Pereyra [17] and optimising it considering the properties arising from the use of an orthogonal basis. An efficient way to compute the orthogonal basis is also presented, for the case of the triangle, showing that such a basis can be used without a major performance penalty.

The numerical results presented for the case of asymmetric cubature on the triangle serve to demonstrate the effectiveness and efficiency of the proposed method. They are however also very interesting in their own right, as they are new cubature rules of theoretical interest, improving on previously available results even for relatively low degrees.

The proposed method is not dependent on a particular domain or imposed symmetry. Various ideas are provided on how the performance of the method can be further improved for three-dimensional domains or for a higher number of points.

Acknowledgements

This research effort is funded from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement n^{o} 618096.

References

[1] A. H. Stroud, Approximate Calculation of Multiple Integrals, Prentice-Hall series in automatic computation, Prentice-Hall, Englewood Cliffs, NJ, USA, 1971.

[2] R. Cools, Constructing cubature formulas: the science behind the art, Acta Numer. 6 (1997) 1–54.

[3] A. R. Krommer, C. W. Ueberhuber, Computational Integration, Society for Industrial and Applied Mathematics, Philadelphia, USA, 1998.

[4] R. Cools, P. Rabinowitz, Monomial cubature rules since "Stroud" — a compilation, J. Comput. Appl. Math. 48 (1993) 309–326.

[5] R. Cools, Monomial cubature rules since "Stroud": a compilation — part 2, J. Comput. Appl. Math. 112 (1999) 21–27.

[6] R. Cools, An encyclopaedia of cubature formulas, J. Complex. 19 (2003) 445–453. Online database at http://nines.cs.kuleuven.be/ecf/.

[7] M. A. Taylor, B. A. Wingate, L. P. Bos, Several new quadrature formulas for polynomial integration in the triangle (2007) arXiv:math/0501496v2 [math.NA].

[8] M. A. Taylor, Asymmetric cubature formulas for polynomial integration in the triangle and square, J. Comput. Appl. Math. 218 (2008) 184–191. Special Issue: Finite Element Methods in Engineering and Science (FEMTEC 2006).

[9] L. Zhang, T. Cui, H. Liu, A set of symmetric quadrature rules on triangles and tetrahedra, J. Comput. Math. 27 (2009) 89–96.

[10] H. Xiao, Z. Gimbutas, A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions, Comput. Math. Appl. 59 (2010) 663–676.

[11] S.-A. Papanicolopulos, New fully symmetric and rotationally symmetric cubature rules on the triangle using minimal orthonormal bases, J. Comput. Appl. Math. 294 (2016) 39–48.

[12] S.-A. Papanicolopulos, Computation of moderate-degree fully-symmetric cubature rules on the triangle using symmetric polynomials and algebraic solving, Comput. Math. Appl. 69 (2015) 650–666.

[13] F. Witherden, P. Vincent, On the identification of symmetric quadrature rules for finite element methods, Comput. Math. Appl. 69 (2015) 1232–1241.

[14] G. R. Cowper, Gaussian quadrature formulas for triangles, Int. J. Numer. Methods Eng. 7 (1973) 405–408.

[15] J. Lyness, D. Jespersen, Moderate degree symmetric quadrature rules for the triangle, IMA J. Appl. Math. 15 (1975) 19–32.

[16] D. A. Dunavant, High degree efficient symmetrical Gaussian quadrature rules for the triangle, Int. J. Numer. Methods Eng. 21 (1985) 1129–1148.

[17] G. H. Golub, V. Pereyra, The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, SIAM J. Numer. Anal. 10 (1973) 413–432.

[18] M. A. Taylor, B. A. Wingate, L. P. Bos, A cardinal function algorithm for computing multivariate quadrature points, SIAM Journal on Numerical Analysis 45 (2007) 193–205.

[19] J. Proriol, Sur une famille de polynomes á deux variables orthogonaux dans un triangle, C.R. Hebd. Seances Acad. Sci. 245 (1957) 2459–2461.

[20] T. H. Koornwinder, Two-variable analogues of the classical orthogonal polynomials, in: R. A. Askey (Ed.), Theory and application of special functions, Academic Press, New York, 1975, pp. 435–495.

[21] M. Dubiner, Spectral methods on triangles and other domains, J. Sci. Comput. 6 (1991) 345–390.

[22] R. Cools, H. J. Schmid, On the (non)-existence of some cubature formulas: gaps between a theory and its applications, J. Complex. 19 (2003) 403–405. Oberwolfach Special Issue.

[23] K. Gatermann, The construction of symmetric cubature formulas for the square and the triangle, Computing 40 (1988) 229–240.

[24] S. Wandzura, H. Xiao, Symmetric quadrature rules on a triangle, Comput. Math. Appl. 45 (2003) 1829–1840.

Appendix A. New cubature rules

Tables A.1, A.2 and A.3 present the cubature rules computed in Section 6. For each rule, the table lists the the weight of each cubature point and its areal coordinates, in increasing order of weight. An asterisk (*) has been placed next to each row corresponding to a point outside the triangle.

Table A.1: Asymmetric PO cubature rule of degree 11 with 26 points

w	L_1	L_2	L_3
1.7703429351573682e-03	0.6258738065923632	-0.0444983835604058	0.4186245769680427 *
1.1407181542799245e-02	0.0290632953572617	0.9408484762789769	0.0300882283637613
1.4450148832022789e-02	0.9330583391486744	0.0334630956930928	0.0334785651582328
1.4571713721870356e-02	0.0318590326985022	0.0355146752201139	0.9326262920813839
1.5082717050370174e-02	0.1624826332186683	0.0139846877219226	0.8235326790594090
1.7381504665411469e-02	0.0132184617612812	0.6427888268154324	0.3439927114232864

Table A.1: Asymmetric PO cubature rule of degree 11 with 26 points (continued)

w	L_1	L_2	L_3
1.9788048913795569e-02	0.5918294793334099	0.3931416942741072	0.0150288263924829
2.3034628317738500e-02	0.8055758967896875	0.0246651880397084	0.1697589151706041
2.4074295611862854e-02	0.8043214954660826	0.1701217153423755	0.0255567891915419
2.5782293138424881e-02	0.1517977741136217	0.8180708557984559	0.0301313700879224
2.7854536597135951e-02	0.0290692872577444	0.1779293919407720	0.7930013208014837
2.8027408557939566e-02	0.0342125596440218	0.8141179370835360	0.1516695032724423
3.0133750535085932e-02	0.0248467017693192	0.3973211985920437	0.5778320996386371
3.5325613244976901e-02	0.3532703142982785	0.6159167605516321	0.0308129251500894
3.9626273680208260e-02	0.3617812618781889	0.0348462422413524	0.6033724958804587
4.7049077672300709e-02	0.7255128905173974	0.1342004546749753	0.1402866548076273
4.7626634579448043e-02	0.1526877583775490	0.1082745708776674	0.7390376707447836
4.9123786306245618e-02	0.5767609187090251	0.3299095455530797	0.0933295357378951
5.5892517086728270e-02	0.0957812464558656	0.5537082795511952	0.3505104739929392
5.6926272805302974e-02	0.1662869182873340	0.6799902364777285	0.1537228452349375
5.7326951870023362e-02	0.5809311067210347	0.0593349142682010	0.3597339790107644
6.3471175918900887e-02	0.1331546134128777	0.2982865414345181	0.5685588451526042
6.3806180979645840e-02	0.3613509750869149	0.4889662400184307	0.1496827848946544
7.4255945293930314e-02	0.3263421583773203	0.1749834270680896	0.4986744145545902
7.7589779768121364e-02	0.4933483827105916	0.2373606333716019	0.2692909839178066
7.8621220374552806e-02	0.2608387984712898	0.4018889231575169	0.3372722783711933

Table A.2: Asymmetric PO cubature rule of degree 20 with 77 points

w	L_1	L_2	L_3
3.3272031899930308e-09	-0.2139722826185562	0.8852704943288995	0.3287017882896566 *
1.1522730667251121e-08	1.1421952868429700	-0.1722740343873985	0.0300787475444285 *
6.1193449764404254e-08	0.5992142624452174	-0.1265986873996475	0.5273844249544302 *
6.7168983640976152e-08	0.1243798718893910	-0.1187856182759406	0.9944057463865496 *
8.5515199758833790e-07	0.0653183349102199	1.0288244853951577	-0.0941428203053776 *
1.5175231945732550e-05	-0.0536667090855444	0.1640223252964244	0.8896443837891200 *
7.6510157049546390e-05	0.2856083204150442	-0.0344098409098498	0.7488015204948056 *
2.1765189868301176e-04	0.9360808436934921	0.0834052574449411	-0.0194861011384333 *
1.4315614431658254e-03	0.8030525284397181	0.1945616339233715	0.0023858376369104
1.9314426547382595e-03	0.0120188880734230	0.0122773002304659	0.9757038116961111
2.0834859747296434e-03	0.0117885985793459	0.9746833066691727	0.0135280947514814
2.4504339436383215e-03	0.9724194714380300	0.0151595907332457	0.0124209378287244
3.1752343328823898e-03	0.2595828898422352	0.7374061786479605	0.0030109315098044
4.1332493970509188e-03	0.0113901722279700	0.0638319730537900	0.9247778547182400
4.2393898837920337e-03	0.0624536748324726	0.0119686440555381	0.9255776811119894
4.2462464843566120e-03	0.5364878869178866	0.4586406435763420	0.0048714695057714
4.5784993771821117e-03	0.0115287726880496	0.9184513045542639	0.0700199227576865
5.0680982671320988e-03	0.9212296278015700	0.0137267427812455	0.0650436294171845
5.7354175259044468e-03	0.0609542503069404	0.9220601718596792	0.0169855778333804
6.1150285138198147e-03	0.1498077783451357	0.0117795744693113	0.8384126471855530
6.1404661979067891e-03	0.7078898074639406	0.0083409100880332	0.2837692824480262
6.1545543320798148e-03	0.8323428152565519	0.0109109589105934	0.1567462258328547
6.8948715222448776e-03	0.0119250456031764	0.8218085619732910	0.1662663924235326
7.0247457608165264e-03	0.1455472291954016	0.8407014888010812	0.0137512820035173
7.1746550318843929e-03	0.0095334490986295	0.4104869400812500	0.5799796108201205
7.2507179234764397e-03	0.6796683414344640	0.3100866480213636	0.0102450105441724
7.2852898422032166e-03	0.0098923273022164	0.5566731853629672	0.4334344873348164

Table A.2: Asymmetric PO cubature rule of degree 20 with 77 points (continued)

w	L_1	L_2	L_3
8.0472822752512336e-03	0.5534560716862930	0.0104547658690505	0.4360891624446565
8.2951743144646277e-03	0.8094842242438749	0.1706822036252221	0.0198335721309030
8.3634959197703732e-03	0.9020135076631229	0.0746138877131692	0.0233726046237078
8.6328876403430689e-03	0.0129694574942775	0.2705576393189556	0.7164729031867668
8.6338660310716417e-03	0.0586275629455061	0.0612222576866121	0.8801501793678818
8.8186775137681390e-03	0.3924249096626205	0.5952063888311978	0.0123687015061817
9.4468854954044334e-03	0.0137405978198458	0.6946591202501084	0.2916002819300458
9.5691674028340313e-03	0.0187516304312782	0.1512275875775564	0.8300207819911654
1.0147724906551308e-02	0.4029563699535816	0.0137359323322952	0.5833076977141232
1.0601836707815451e-02	0.1376506380168706	0.0564581638297467	0.8058911981533827
1.1199272640821794e-02	0.0598726536103753	0.8585595585971510	0.0815677877924737
1.2008691726336927e-02	0.6248823528936678	0.0416470200265497	0.3334706270797824
1.2065306563947097e-02	0.2629381743397160	0.0199626935996523	0.7170991320606317
1.3121419922390531e-02	0.8380490837191127	0.0647623246114735	0.0971885916694138
1.3465012408395130e-02	0.2575339958346665	0.7060473448618915	0.0364186593034420
1.3785919289998268e-02	0.5354016478572056	0.4290126813155198	0.0355856708272746
1.4132111135062390e-02	0.7432315022640500	0.0513341047302288	0.2054343930057212
1.4405132814676555e-02	0.0488883387859558	0.5218697460311620	0.4292419151828821
1.5264441906139244e-02	0.0624726931520615	0.7578158119621619	0.1797114948857766
1.5301286542334050e-02	0.1435972310505592	0.7859738136269185	0.0704289553225222
1.6474551417327365e-02	0.6691011121275246	0.2763262735810618	0.0545726142914136
1.6532264494773961e-02	0.7651859663690987	0.1551000926249013	0.0797139410059999
1.6624974592286139e-02	0.0885954672172786	0.1371782771633756	0.7742262556193458
1.6829113483477516e-02	0.0527740002133012	0.3823667578709374	0.5648592419157614
1.7932849136018492e-02	0.0680360399128474	0.2476912983745129	0.6842726617126397
1.8780448844569661e-02	0.0698062102092748	0.6315959576620008	0.2985978321287244
1.8820167387222608e-02	0.2079871121564631	0.0924058022638531	0.6996070855796838
1.8879556804683955e-02	0.3901829313032221	0.5474116133643799	0.0624054553323979
1.8889095827054366e-02	0.4760763868427125	0.0602551121213251	0.4636685010359623
2.0549961352460224e-02	0.3348863482569718	0.0747904975261003	0.5903231542169280
2.0664562937209037e-02	0.5864230153923678	0.1034891045584583	0.3100878800491739
2.1196592533383173e-02	0.1465901036680476	0.6902217267840939	0.1631881695478585
2.1242623436456073e-02	0.6891984248492541	0.1306777966591577	0.1801237784915882
2.1417656759145985e-02	0.2564310038854510	0.6359512644008602	0.1076177317136887
2.1922234938203348e-02	0.5198079657360537	0.3791169744786023	0.1010750597853440
2.3120581286951603e-02	0.1189889352865325	0.4579430288192932	0.4230680358941744
2.4157383298376990e-02	0.6136703242392350	0.2440243430685658	0.1423053326921992
2.4284601989132075e-02	0.1562497737922975	0.5645689373126001	0.2791812888951025
2.4476493188372424e-02	0.2558715036994906	0.5365717557358474	0.2075567405646620
2.4640553062607307e-02	0.1707341034155206	0.1920780732761519	0.6371878233083275
2.4869718628262086e-02	0.3790868969534791	0.4744250782345639	0.1464880248119570
2.4953403930638818e-02	0.1375806059274538	0.3194314574881331	0.5429879365844131
2.6494936022368213e-02	0.2189111413867021	0.4161899655526268	0.3648988930606711
2.6990664473401601e-02	0.4344120958112691	0.1469767439359339	0.4186111602527970
2.7035897305238948e-02	0.2965501572072155	0.1700101823487815	0.5334396604440030
2.7329813084088887e-02	0.3230707204784158	0.4000967067151431	0.2768325728064411
2.7716535223871964e-02	0.5305962772598443	0.2030397952102567	0.2663639275298991
2.8325725738883217e-02	0.2484602615536088	0.2880666512775105	0.4634730871688807
2.8495989683869940e-02	0.4587355289252984	0.3306635431439684	0.2106009279307332
3.1621731921240055e-02	0.3769856631093792	0.2653966961081434	0.3576176407824775

Table A.3: Asymmetric PO cubature rule of degree 22 with 92 points

w	L_1	L_2	L_3
4.5395639901349092e-09	0.4649821382452427	0.7111505756504594	-0.1761327138957021 *
2.2271274087684915e-08	0.9332413498021373	0.2072453224809921	-0.1404866722831294 *
1.3651386509171615e-07	0.7188146070148182	-0.1085923748471092	0.3897777678322910 *
1.2620323408591148e-06	-0.0753425185865285	0.3657388261010510	0.7096036924854774 *
1.2773886859207504e-06	-0.0729720466214084	0.9087809877046706	0.1641910589167378 *
3.4515501760991334e-06	0.1613900535324765	-0.0567779747516549	0.8953879212191783 *
1.7271454205030007e-05	0.3964730375095246	-0.0393347505474985	0.6428617130379740 *
1.0102564890393311e-04	0.0325670755950674	0.9836159918553805	-0.0161830674504479 *
1.0289895715049877e-04	0.5110834897688819	-0.0167145669135699	0.5056310771446880 *
2.5849787427155538e-04	0.9698153095809794	-0.0084738641497777	0.0386585545687983 *
6.9262488612911405e-04	0.0045668480222933	0.0112732881898780	0.9841598637878287
1.4141283516694330e-03	0.9785347731846658	0.0131017828795071	0.0083634439358271
1.4930106714526691e-03	-0.0020936608309148	0.6111729711663046	0.3909206896646103 *
1.5129779019346722e-03	0.0082446591796961	0.9773748146351398	0.0143805261851641
1.7068811763706268e-03	0.0286161958663017	0.0083941811270468	0.9629896230066514
1.9349920621679056e-03	0.0065762640156403	0.0528100601072915	0.9406136758770682
2.5088294593341612e-03	0.2758666449617112	0.0042128769998266	0.7199204780384622
3.0884974212190485e-03	0.1058689430939781	0.8868858914256998	0.0072451654803221
3.1061258925912809e-03	0.6192264806124585	0.3769418326829147	0.0038316867046268
3.5061520285316948e-03	0.8837762141045503	0.0083705323204879	0.1078532535749618
3.5565092691088905e-03	0.9254105836968477	0.0644641060621398	0.0101253102410124
3.9496670800454524e-03	0.0108650060510515	0.9206641434920366	0.0684708504569120
4.0376177795052796e-03	0.0094414108757236	0.1221323424228500	0.8684262467014264
4.0937996280007192e-03	0.0834379668080314	0.0114591516812569	0.9051028815107116
4.7340311582250246e-03	0.0461305836421291	0.9332880935918074	0.0205813227660635
4.9593015383271003e-03	0.9301326043260487	0.0232412078745276	0.0466261877994238
5.0841050964592767e-03	0.4009202375315670	0.5856759399555980	0.0134038225128350
5.1537586866185813e-03	0.2002986318444794	0.7901613525413712	0.0095400156141494
5.4175393501613104e-03	0.0377632914144659	0.0488000038207257	0.9134367047648084
5.6684521721347064e-03	0.7869933035900340	0.0102546720868125	0.2027520243231536
5.8341342113779666e-03	0.8407378638103820	0.1473222888308409	0.0119398473587771
5.8959182214630989e-03	0.0105197174565544	0.2190350425308016	0.7704452400126441
5.9028999631625856e-03	0.0099790114602826	0.7272897883804312	0.2627312001592862
5.9841005968195210e-03	0.3102831400290192	0.6789943596252085	0.0107225003457724
6.0766492558934536e-03	0.4901895600744398	0.4991745270288127	0.0106359128967475
6.8080298737058088e-03	0.1679663305360452	0.0141053909355621	0.8179282785283927
6.9184793067436195e-03	0.0141127681947915	0.8324474631332752	0.1534397686719333
7.1032560517228288e-03	0.0104234395787747	0.4689980777554518	0.5205784826657734
7.3346357681360919e-03	0.7331463362172260	0.2544424502184292	0.0124112135643448
7.4375132771643090e-03	0.2768726553091220	0.0238950051964320	0.6992323394944460
7.6836516813958281e-03	0.6678935309195156	0.0120851625782621	0.3200213065022223
8.5298898938657950e-03	0.0445015015162890	0.6964091231294402	0.2590893753542709
8.6660639484361069e-03	0.0137349285317674	0.3369393972521638	0.6493256742160688
8.8904913328585393e-03	0.5375478799081402	0.0138730562867832	0.4485790638050765
9.4351999693634109e-03	0.5928350534101339	0.3771591146500973	0.0300058319397688
9.4433029493588047e-03	0.1266301093747175	0.8302226584537487	0.0431472321715338
9.5558787695999292e-03	0.4013663824060378	0.0150259944876660	0.5836076231062962
9.7679257690299608e-03	0.8579542756723133	0.0862277341830809	0.0558179901446058
9.7936899377279218e-03	0.0491710116722590	0.1233215400533555	0.8275074482743855
9.8450167213557546e-03	0.0590361244964827	0.8597698838794501	0.0811939916240672
9.8947652193418236e-03	0.8285527046102029	0.0472711454279718	0.1241761499618253
1.0445653804999097e-02	0.1028881554758129	0.0605783796272693	0.8365334648969179

Table A.3: Asymmetric PO cubature rule of degree 22 with 92 points (continued)

w	L_1	L_2	L_3
1.1527390931442913e-02	0.0252668780123646	0.5906792594832326	0.3840538625044028
1.2767292253125432e-02	0.0522224101343283	0.4464041575932961	0.5013734322723756
1.3478903113459106e-02	0.0557922911259375	0.2219583512296826	0.7222493576443799
1.3588026786923612e-02	0.2330570599001325	0.7148687591961092	0.0520741809037583
1.3672849749700126e-02	0.7602387409043682	0.1780544689852921	0.0617067901103396
1.3838157839677994e-02	0.6537821348500996	0.2843498955459140	0.0618679696039864
1.3924834500668074e-02	0.0701192668918892	0.7616457720344102	0.1682349610737005
1.3977008809975239e-02	0.0856752082434593	0.6201239512863054	0.2942008404702353
1.4123771073724647e-02	0.7174560889009518	0.0555931048067862	0.2269508062922620
1.4798663317517431e-02	0.1975481930963446	0.0685029623840040	0.7339488445196514
1.5291214328912046e-02	0.0864145042160261	0.5134382192706653	0.4001472765133086
1.5553999941421675e-02	0.4755097609964785	0.4648363271004296	0.0596539119030919
1.5802681212130293e-02	0.0678714687817086	0.3319965021961245	0.6001320290221669
1.6234608647348465e-02	0.1469792077847016	0.7410787299023880	0.1119420623129104
1.6257930986965675e-02	0.3512216651844325	0.5879535852949610	0.0608247495206065
1.6687406294486875e-02	0.7412614927620904	0.1208940885567262	0.1378444186811833
1.6711160412937197e-02	0.1240403276591735	0.1425836013649232	0.7333760709759034
1.7224777629975396e-02	0.2138803941211439	0.2704901128161058	0.5156294930627503
1.7249444726274737e-02	0.5887704077391711	0.0628867829706776	0.3483428092901514
1.7277250957003662e-02	0.3212700989541091	0.0716976174401818	0.6070322836057091
1.7866272588789774e-02	0.4545590368099309	0.0648529251240835	0.4805880380659856
1.8010185571439035e-02	0.2888452819409681	0.2381183968667653	0.4730363211922665
1.8768448853723183e-02	0.5321635809001525	0.3570177382033145	0.1108186808965330
1.9379500993648655e-02	0.4736410292722732	0.3354671158671107	0.1908918548606161
1.9862660627271811e-02	0.1372156408075117	0.2439059721040748	0.6188783870884135
2.1294664629939871e-02	0.6213916587330183	0.1350374610368093	0.2435708802301724
2.1526907650302151e-02	0.1526036559137645	0.6340851747840728	0.2133111693021628
2.1633103075462352e-02	0.2535424994047533	0.6145682205207210	0.1318892800745257
2.2237085598750902e-02	0.6304593865191285	0.2250328820610726	0.1445077314197989
2.2636190719411340e-02	0.2308485141633608	0.1507781814727181	0.6183733043639211
2.2921010191850760e-02	0.3606076667650022	0.1476571250366021	0.4917352081983957
2.3361825113449281e-02	0.1688258437775907	0.4968017672246279	0.3343723889977814
2.3654431484015719e-02	0.5129504854212238	0.2349384554629883	0.2521110591157879
2.4182433274350601e-02	0.4874798993088388	0.1447536536811390	0.3677664470100222
2.4335794985536732e-02	0.1414118586611728	0.3795554544211256	0.4790326869177016
2.5376393663440224e-02	0.3799202246016695	0.4775364341798633	0.1425433412184672
2.5667876509629658e-02	0.2599312640326476	0.5045529126340352	0.2355158233333172
2.5948182876164511e-02	0.3606907921924177	0.3781963472779737	0.2611128605296086
2.7637005014311689e-02	0.3859853018308854	0.2567611752699346	0.3572535228991800
2.8360656700894891e-02	0.2549677295279997	0.3697330460402094	0.3752992244317908