International Conference on Computational Science, ICCS 2010

Dimensional recursion for multivariate adaptive integration

Elise de Doncker and Karlis Kaugars
Department of Computer Science, Western Michigan University, Kalamazoo, MI 48008, United States

Abstract
We consider multivariate integrals which can be expressed as iterated integrals over product regions. The iteration
over the dimensions is applied recursively for a numerical evaluation. We evaluate a scheme for setting the tolerated
error in the interface between the integration levels and address the efficiency of the resulting method with respect to
time and space requirements.

Keywords: Dimensional recursion, automatic iterated integration, relative error interface, discriminant sensory
process, biometrics.

1. Introduction
The user of an integration algorithm or software generally wants to approximate an integral to within a prescribed
accuracy, which may be absolute ($\varepsilon_a$) or relative ($\varepsilon_r$) or a combination of both. The goal is then to obtain a numerical
approximation $Qf$ to an integral

$$If = \int_D f(\vec{x})\, d\vec{x}$$

(1)

and an (absolute) error estimate $Ef$, in order to satisfy an accuracy requirement where the actual error $Ef = Qf - If =
\rho If$ does not exceed the estimated error $Ef$ (in absolute value); and the latter does not exceed the prescribed tolerated
error, i.e.,

$$|Qf - If| \leq Ef \leq \max \{ \varepsilon_a, \varepsilon_r |If| \}.$$  

(2)

A basic integral approximation to (1) can be calculated by using an integration rule (quadrature or cubature rule,
for univariate and multivariate integrals, respectively), which is a weighted sum of function values. An error estimate
can be obtained by comparing two or more rule approximations.

We use basic rule approximations designed to be of a polynomial degree of accuracy, i.e., exact for polynomials up
to (and not beyond) the corresponding degree. Thus, depending on the integrand behavior, a basic rule approximation
by itself will not generally provide sufficient accuracy so that we will need a more elaborate strategy to increase the
integrand sampling, such as an adaptive integration which attempts to handle problem areas by region partitioning. The
latter will clearly suffer from a dimensionality curse since partitioning in a higher-dimensional space is not effective.
However, by invoking an integration method recursively across dimensions, it may be possible to retain benefits of
its lower-dimensional application. This leads to recursive implementations of numerical \textit{iterated integration}, for an integral over a product region $\mathcal{D} = \mathcal{D}_1 \times \ldots \times \mathcal{D}_\ell$ which can be written in the form

$$If = \int_{\mathcal{D}_1} d\mathcal{x}^{(1)} \ldots \int_{\mathcal{D}_\ell} d\mathcal{x}^{(\ell)} f(\mathcal{x}^{(1)}, \ldots, \mathcal{x}^{(\ell)}).$$

We have applied numerical iterated integration with adaptive integration strategies (on the levels $j = 1, \ldots, \ell$) for the computation of \textit{loop integrals} in high energy physics, e.g., in [1, 2, 3, 4, 5], and for other mathematical modeling problems in [6], where the focus was on the applications and the interface problem was not addressed. The technique is implemented recursively using the lower-dimensional code across successive groups of dimensions. Iterated integration was implemented with non-adaptive (lattice) methods in [7].

An interface problem [8] arises between the applications on successive levels, particularly with respect to setting the requested accuracy from each level to the next lower level. The problem was addressed for double integration using non-specific one-dimensional codes and with respect to absolute error in [9]. An extension of the latter for integrals of the form (3) was given in [10]. In this paper we evaluate a scheme applied particularly for a two-level recursion.

The two-level dimensional recursion is directly applicable to the mathematical modeling of differences in taste perceptions or discriminant sensory processes. Since Thurstone [11] defined a \textit{discriminal process} in 1927 as a “process by which the organism identifies, distinguishes or reacts to stimuli”, a large body of work has emerged in this area, much of which in collaborations with the food and beverage industries. Discriminal sensory probabilities are naturally expressed as integrals over $\mathbb{R}^n \times C_n$ where $C_n$ is an $n$-dimensional sphere or a portion thereof. We apply an automatic numerical strategy with an adaptive integration on both of the $n$-variate integration levels.

The error tolerance interface is outlined in Section 2. We describe a two-dimensional experimental implementation in Section 3 and give results with respect to accuracy and memory use. Section 5 describes the $N$-dimensional application to a probability integral occurring in biometrical modeling of di

\textbf{2. Error Tolerance Interface}

We will first outline the error tolerance interface for double integration, as it is essentially the same for a more general two-level iterated integration.

Let us denote the integral

$$If = I = I_1 = \int_{a_1}^{b_1} dx \int_{a_2}^{b_2} dy f(x, y) = \int_{a_1}^{b_1} dx \ g(x)$$

where $g(x) = \int_{a_2}^{b_2} dy f(x, y)$. On the inner (lower) level we obtain, for a given $x$, an approximation $Q_2 = Q_2(x) \approx \int_{a_2}^{b_2} dy f(x, y) = g(x)$ with actual absolute error $E_2 = E_2(x)$, so that $Q_2(x) = \int_{a_1}^{b_1} dy f(x, y) - E_2(x)$. On the outer (top) level, the integral approximation is $Q = Q_1 \approx I = I_1$ and we have

$$Q_1 = \int_{a_1}^{b_1} dx \ Q_2(x) - E_1 = \int_{a_1}^{b_1} dx \ [\int_{a_2}^{b_2} dy f(x, y) - E_2(x)] - E_1$$

$$= I - \int_{a_1}^{b_1} dx \ E_2(x) - E_1,$$

so that

$$|Q - I| \leq \int_{a_1}^{b_1} dx \ |E_2(x)| + |E_1| \leq \varepsilon_a^{(2)} |b_2 - a_2| + \varepsilon_a^{(1)}$$

assuming that $|E_2(x)| \leq \varepsilon_a^{(2)}$ (for each $x$) and $|E_1| \leq \varepsilon_a^{(1)}$ where $\varepsilon_a^{(2)}$ and $\varepsilon_a^{(1)}$ are the tolerated errors on the inner and outer level, respectively. Thus in order to achieve overall absolute accuracy $\varepsilon_a$ it suffices to require that

$$\varepsilon_a^{(2)} |b_2 - a_2| + \varepsilon_a^{(1)} \leq \varepsilon_a.$$
If we furthermore produce associated absolute error estimates \( E_i(x) \), \( i = 1, 2 \) with absolute values in between the actual and the tolerated errors, then

\[
| Q - I | \leq | b_2 - a_2 | \max_x | E_i^{(i)}(x) | + E_i^{(1)} \leq E_i^{(2)} | b_2 - a_2 | + E_i^{(1)}.
\]

In practice, this needs to be implemented with guarding factors. Estimated errors are generally conservative but may also be under-estimated especially in difficult cases. It is furthermore recommended to evaluate the inner integral more accurately than the outer integral, as errors on the evaluation points may be perceived as roundoff error by the outer integration procedure.

An extension of (4) for the general integral (3) follows as

\[
| Q - I | \leq E_i^{(1)} + | D_1 | E_i^{(2)} + \ldots + \prod_{j=1}^{l-1} | D_j | E_i^{(l)}
\]

where \( | D_j | \) is the volume of region \( D_j \).

A modification of (4) to account for relative error can be given in the form

\[
| Q - I | \leq \int_{a_1}^{b_1} dx | q_2(x) | \int_{a_2}^{b_2} dy f(x,y) | + q_1 \leq E_i^{(2)} \int_{a_1}^{b_1} dx \int_{a_2}^{b_2} dy f(x,y) | + E_i^{(1)}
\]

where \( q_1 \) and \( q_2(x) \) represent outer and inner actual relative errors. The bounds in (6) are thus expressed in terms of absolute value functions.

3. Experimental Code

In order to test and evaluate iterated integration and error estimation, we have developed a C language implementation of \( N \) times iterated one-dimensional integration. Subsequently we refer to the experimental code as \( IQ2D \). The integration program uses the Gauss-Kronrod 15-point integration rule pair in both directions and fully supports 2D iterated integration. The code implements the adaptive integration meta-algorithm of Figure 3 at each level of the iterated integration.

The algorithm is implemented recursively, with the difference among dimensions appearing in the methods used to evaluate a region. For the final (inner) dimension of integration, region evaluation consists of evaluating the integrand at multiple points and constructing a result and error estimate using the traditional Gauss-Kronrod integration points. At each other level of integration, region evaluation consists of creating one subregion along the next dimension for each of the Gauss-Kronrod evaluation points and then making a recursive call to evaluate that region. For example, using a 15-point rule with a two dimensional function results in 15 recursive calls at the top level with each call having a fixed \( x_1 = x \) coordinate. Each of these calls then constructs a priority queue which evaluates the function at 15 points in the \( x_2 = y \) dimension with the common \( x_1 \) coordinate.

The estimated error parameter is computed locally along each dimension while the evaluation limit is treated as a global constant value. Termination of a recursive call occurs when the local error condition is met or the global evaluation limit is reached. Each of the upper dimensions terminates when either the current recursive call has exceeded the evaluation limit or when the estimated error along the dimension is smaller than the requested accuracy.

4. Two-dimensional Tests and Memory Use

4.1. Numerical results

As a simple two-dimensional example let us consider an integral from the paper on (the adaptive Monte Carlo
The integrand function \( f(x,y) \) is depicted over the unit square in Figure 2 for \( \alpha = 10^{-1} \) and has a ridge of height \( 2(1-x)/\alpha \) along \( y = 1 - x \). The function parameter is \( \alpha = 10^{-p} \) where \( p = 1, 2, \ldots, 10 \), which causes an increasingly difficult anomaly as the ridge becomes higher and steeper.

Table 1 lists the actual error and the number of integrand evaluation points used. The maximum number of integrand evaluations was set to 100 million. The absolute error tolerance for the outer integration was set at \( 1 \times 10^{-8} \) while the error tolerance of the inner integration was initially set at \( 2 \times 10^{-9} \). For the fixed error criterion case in IQ2D, the inner error tolerance was held constant. For the variable criterion the tolerance for computing the inner integrals (i.e., the function evaluations for the outer integration) was scaled with respect to the length of the outer interval \( 2 \times 10^{-9} \). Changing the inner tolerance to reflect the size of the outer region results in significantly fewer integrand evaluations for most cases. On the other hand, a conservative error estimate can have benefits, as in the cases of \( p = 7 \) and \( p = 8 \). The higher tolerance on the inner integral results in a failed integration when a variable inner tolerance is used. The very high and steep peak is missed by the evaluation points, and the routine underestimates the actual error and terminates prematurely.

For comparison, Table 1 gives results of an iterated integration where (a modification of) the Q one-dimensional integration program D [13] is used in both coordinate directions. The 1D integration procedure adheres to the adaptive algorithm of Figure 3 and terminates when the requested accuracy is achieved or the maximum allowed number of integrand evaluations is reached. The function peak was partially missed for \( p = 7 \) resulting in a failed integration. The maximum number of evaluations (9,975) for the outer integration was reached for \( p = 11 \). The function peak was missed at \( p = 12 \) where the procedure stopped prematurely. Thus the D application is somewhat better at dealing with higher peaks but uses more integrand evaluations than IQ2D for smaller values of \( p \).

Table 1 furthermore lists results obtained with the multivariate adaptive integration package D [14]. The routine breaks down at \( \alpha = 10^{-5} \) where it runs up to the maximum allowed number of function evaluations. The

![Table 1: Comparison of function evaluations for Dice function](image-url)
accuracy deteriorates significantly from this point on as $\alpha$ decreases. We use the integration rule of degree 7 provided by $D_cuhre$ which uses a fairly low number of points, as is recommended for problems which require intensive partitioning of the integration region.

4.2. Memory Use

The use of iterated integration can also significantly reduce the memory usage of adaptive integration by subdividing the global priority queue into a series of local priority queues. Table 2 lists the number of regions used while generating the results listed in Table 1. The $D_0$ regions column lists the total number of regions evaluated along the first dimension of integration. These regions remain in memory for the duration of the computation. The $D_1$ regions are kept in a local priority queue, and the value reported in Table 2 is the maximum number of regions active at any time along $D_1$.

In the worst case, the total memory footprint of the algorithm is $O(RD_0 + RD_1)$, i.e. the regions stored at the first level of iteration and the maximum region storage required at the second level along with some constant number of local variables, parameters, etc.

For the simple integral produced by $p = 0$, there is no advantage to iterated integration, and there may be a disadvantage - if the integral could be solved with the function evaluation points generated by crossing two 15-point integration rules, only a single region would be needed as opposed to the four required for iterated integration. When iterated integration fails - e.g. the case of $p = 9$, the first $D_0$ iteration point results in a $D_1$ integration which fails to meet the error criterion, the iterated method results in a local priority queue which is just as large as the global queue needed by a multi-dimensional integration routine.

In non-trivial cases where iterated integration succeeds, there is a large memory utilization advantage to the method. The inner integrands meet the error condition within a few hundred regions - indeed for most values $p < 8$ the inner priority queue size is limited to fewer than 100 regions, while the global priority queue is limited to fewer than 50 regions.

A direct comparison of the memory utilization with a $N$-Dimensional integration routine is difficult due to the differences in the number of function evaluations used per region as well as the region subdivision patterns from different algorithms. The $D_cuhre$ routine uses 65 function evaluations per region (with its parameter set to 4). Assuming for the moment that the method would succeed, we can take the number of function evaluations used during iterated integration and estimate the number of regions needed for a 2D integration routine - these values are listed in the estimated 2D regions column in Table 2 above. The iterated integration routine outperforms the potential 2D integrator’s memory footprint by a substantial amount, ranging from a high of 65% of the memory needed for $p = 1$ to a low of 1% for $p = 8$.

The amount of memory used to represent a region depends on the specific implementation of the priority queue – using dynamic allocation requires two heap pointers while a fixed size implementation can implement the priority queue in an array. The dynamic allocation model has significant benefits in the case of iterated integration, allowing the $D_2$ integration routines to limit the amount of memory used. In either case, the region structure needs to store at least five doubles for a 2D integration (the fixed $D_0$ point, the minimum and maximum of the integration region in $D_1$,

<table>
<thead>
<tr>
<th>$p$</th>
<th>$V$ Rgns $D_0$</th>
<th>$\varepsilon^{(2)}_a$ Rgns $D_1$</th>
<th>$F$ Rgns $D_0$</th>
<th>$\varepsilon^{(2)}_a$ Rgns $D_1$</th>
<th>Est. 2D Rgns.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>15</td>
<td>9</td>
<td>15</td>
<td>335</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>31</td>
<td>19</td>
<td>31</td>
<td>1407</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
<td>43</td>
<td>29</td>
<td>47</td>
<td>2967</td>
</tr>
<tr>
<td>4</td>
<td>39</td>
<td>57</td>
<td>39</td>
<td>57</td>
<td>5240</td>
</tr>
<tr>
<td>5</td>
<td>41</td>
<td>67</td>
<td>41</td>
<td>69</td>
<td>6681</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>79</td>
<td>39</td>
<td>83</td>
<td>7812</td>
</tr>
<tr>
<td>7</td>
<td>33</td>
<td>95</td>
<td>39</td>
<td>95</td>
<td>8497</td>
</tr>
<tr>
<td>8</td>
<td>99</td>
<td>131</td>
<td>43</td>
<td>149</td>
<td>34312</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>6650795</td>
<td>1</td>
<td>6650795</td>
<td>1538461.62</td>
</tr>
</tbody>
</table>

Table 2: Region evaluations by dimension Dice function
the result and the estimated error). In this case (to allow for the maximum 100 million integrand evaluations), a space of more than 19 million doubles has to be assigned for the region storage and manipulation in D.

For its interval storage by means of an ordered list structure, the D routine uses four double arrays and one integer array each of which has the length of the total number of intervals generated. A set of these arrays is passed in the call to invoke the next level in the interface. Thus, only \( \ell \) sets of these are needed for an \( \ell \) level interface. In this case, to allow \( n = 333 \) intervals in each coordinate direction for double integration (\( \ell = 2 \)), the array space required in memory at any one time will not exceed that of \( 4.5n \times \ell = 3000 \) doubles. Using this space, 333 intervals are generated in 332 bisections and 30 points are used per bisection. Thus the total number of points allowed per direction is \( 15 + 332 \times 30 = 9,975 \). If an equal amount of work is done in the two coordinate directions, then the total number of points that can be generated in the two-dimensional region is \( 9,975^2 = 99,500,625 \).

5. Discriminal Sensory Probability

5.1. Definition and Notations

In this section we consider an application of two-level dimensional recursive integration. We studied evaluation techniques for discriminant taste testing in [15, 16] for the duo-trio and triangular models.

In the test trials, two sets of objects are considered, \( S_x \) and \( S_y \), where \( S_x \) is referred to as the standard set. The stimulus objects \( S_x \) and \( S_y \) are selected from \( S_x \) and the object \( S_y \) is taken from \( S_y \).

In the duo-trio method, \( S_x \) is identified to the subject as being from the standard set. The other two objects are then presented in random order, and the subject is asked which of those is more similar to the standard. In the triangular method, the three objects are given to the subject at random and the trial consists of identifying the less similar object.

The stimuli \( S_x, S_y, \) and \( S_z \) correspond to the \( n \)-vectors of sensory values \( \vec{x}_1, \vec{x}_2, \) and \( \vec{y} \), respectively, where the \( x \)-vectors are drawn from an \( n \)-variate normal distribution with mean \( \vec{\mu}_x \) and variance-covariance matrix \( V_x \), and \( \vec{y} \) from an independent \( n \)-variate normal distribution with mean \( \vec{\mu}_y \) and variance-covariance matrix \( V_y \).

Thus for the duo-trio method, the probability \( P \) of a correct response is given by

\[
P = \Pr ( \| \vec{u} \| < \| \vec{v} \| ),
\]

with \( \vec{u} = \vec{x}_1 - \vec{x}_2 \) and \( \vec{v} = \vec{x}_1 - \vec{y} \), which can be expressed by the \( N = 2n \)-dimensional integral

\[
P = \int_{\mathbb{R}^n} \int_{C_n} f(\vec{u}, \vec{v}) \ d\vec{u} \ d\vec{v}
\]

where

\[
f(\vec{u}, \vec{v}) = \frac{\exp\left[-\frac{1}{2}(\vec{z} - \vec{\mu}_x) V^{-1} (\vec{z} - \vec{\mu}_x)^T\right]}{(2\pi)^n |V|^{1/2}},
\]

with \( \vec{z} = \left(\frac{\vec{u}}{\vec{v}}\right), \vec{\mu}_x = \left(\frac{\vec{\mu}_x}{\vec{\mu}_y}\right), \) the combined variance-covariance matrix \( V = \left(\begin{array}{cc} V_x & V_x V_y V_x^{-1} \\ V_y V_x V_y^{-1} & V_y \end{array}\right) \) and \( |V| = \det(V) \). The integration region \( C_n \) for \( \vec{u} \) is the sphere \( \| \vec{u} \| < \| \vec{v} \| \) centered at the origin and of radius \( \| \vec{v} \| \).

The probability of a correct response for the triangular method is is given by a similar integral. In this paper we will further focus on the duo-trio probability.

5.2. Representations

The integral (9) over \( \mathbb{R}^n \times C_n \) can be interpreted as an \textit{iterated integral} where the inner integral over \( C_n \) yields the integrand function of the outer integral over \( \mathbb{R}^n \),

\[
P = \int_{\mathbb{R}^n} \ d\vec{v} \left[ \int_{C_n} d\vec{u} \ f(\vec{u}, \vec{v}) \right].
\]

By using spherical coordinates for the inner integration this is

\[
P = \int_{\mathbb{R}^n} d\vec{v} \left[ \int_0^{\sqrt{v_1^2 + \ldots + v_n^2}} dr \int_0^\pi d\varphi_1 \cdots \int_0^\pi d\varphi_{n-2} \int_0^{2\pi} d\varphi_{n-1} \ f(r, \varphi, \vec{v}) \ J(\varphi) \right],
\]
where the transformed function is $\tilde{f}(r, \varphi, \tilde{v}) = f(\bar{u}(r, \varphi), \tilde{v})$ and $\tilde{J}(\tilde{v})$ is the Jacobian of the transformation.

In view of the near symmetry for small $\| \bar{\mu} \|$ it may be advantageous to also use spherical coordinates for the integration over $\mathbb{R}^n$ (as mentioned in [16]). This gives

$$\mathcal{P} = \int_0^{\infty} dR \int_0^{\pi} d\theta_1 \cdots \int_0^{\pi} d\theta_{n-2} \int_0^{2\pi} d\theta_{n-1} \left[ \int_0^{\sqrt{1 - r^2}} dr \int_0^{\alpha} d\varphi_1 \cdots \int_0^{\alpha} d\varphi_{n-2} \int_0^{2\pi} d\varphi_{n-1} \tilde{f}(r, \varphi, R, \tilde{v}) \tilde{J}(\varphi, \tilde{v}) \right],$$

(12)

with $\tilde{J}(\varphi, \tilde{v})$ as the new Jacobian. Whereas transformations to obtain constant limits were necessary in [15, 16] (to allow integration over a hyper-rectangular region), that is not needed in (11) and (12) since the inner limits are determined by the evaluation points of the outer integration.

### 5.3. N-dimensional Numerical Results

We evaluate the probability $\mathcal{P}$ according to (10) as a two-level dimensional-recursive integral over $\mathbb{R}^n \times C_n$. The integration variables are transformed to spherical coordinates as in (12) for the inner and outer integration. The outer integration is performed over a sphere of radius truncated to 8 in view of the exponential decay of the integrand. For the matrix $V$ in the exponent of the integrand function we use $V = \left(\frac{2}{l} \frac{l}{2}\right)$ where $I$ is the identity matrix.

Table 3 gives results obtained for $\bar{\mu} = \vec{0}$ and $n = 2, 3, 4, 5$ with the iterated integration on the right and with $D$ (for the corresponding $2n$-dimensional integrals) on the left. The exact integral for $\bar{\mu} = \vec{0}$ is 0.5 (cf. [15]). For both methods, the result, number of integrand evaluations and the time (obtained with the GNU function $etime$) are listed.

The computations were run on a system with 2.26 quad core Xeon processor with an 8MB cache (model E5520) and containing 12GB of ram.

The absolute error tolerance strategy is implemented according to (5), i.e.,

$$|Q - I| \leq \varepsilon_{\alpha}^{(1)} + |D| \varepsilon_{\alpha}^{(2)}$$

(13)

with $|D|$ as the volume of the outer integration region.

As was the case with the double integration, the dimensional recursion is performed with far less memory use than the corresponding $2n$-dimensional integration. For $D$, the user specifies a maximum number of function evaluations and must supply the necessary workspace in a (double precision) work array that is passed to the (Fortran) routine via the integration call from the main program. For the tests on the left of Table 3, the maximum number of function evaluations was set to 10 million for dimensions $2n = 4$ and 6 and the work array needs to be of length 923,094 for a 4-dimensional integration, and of length 496,914 for 6 dimensions. The maximum number of function evaluations was set to 100 million for dimensions 8 and 10, requiring a work array of length 2,398,098 and 948,462, respectively.

In comparison, with the maximum number of function evaluations set to 10,000 on the inner and outer levels of the iterated integration, the array lengths needed are 1922, 1298, 942 and 704 for dimensions 4, 6, 8 and 10, respectively. Recall that only about twice that space is needed for the region storage on the two recursion levels.

For this experiment, the iterated strategy takes less function evaluations for the larger values of $n$ and takes less time than the standard $2n$-dimensional method. For the 10-dimensional integral ($n = 5$), the iterated method reaches the maximum number of evaluations (10,000) in the outer integration and the standard integration reaches the overall 100 million maximum (indicated with $^*$).
6. Conclusions and Future Work

Developing a recursive set of routines for iterated integration has allowed us to begin experimenting with iterated integration and error tolerance and estimation code in a structured fashion without rewriting code for each integrand problem. It has led to new insights about the computation of error tolerance and estimates and will form the experimental platform for our future work.

This code base will be the basis for continued work towards a generalized n-dimensional iterated integration codebase. The use of the C language allows us to more naturally implement recursive routines and incorporate our previous work with parallel codes, as well as explore emerging parallel technologies such as Cuda.

Both the IQ2D and Dqage can underestimate the errors for particularly difficult integrals, and correcting this situation will be a primary focus of our continuing work. We will be evaluating error estimation techniques in the underlying rule implementation and exploring alternative techniques of error estimation such as result convergence.

In addition to the current work utilizing absolute error tolerance, the code still needs updates to include relative error tolerances. At present, the IQ2D code only implements the Gauss-Kronrod rule sets, and including iterated integration of several different multi-dimensional rules is a promising research direction.

References