
Numerical Algorithms **34:** 325–338, 2003.

A comparison of strategies for the automatic computation of two-dimensional integrals over infinite domains

Michael Hill and Ian Robinson
La Trobe University, Victoria 3086, Australia
E-mail: {hillmj;i.robinson@cs.latrobe.edu.au}

Received 29 November 2001; accepted 30 March 2003

We present sets of parameterized integral test families over each of the domains $[0, \infty)^2$ and $(-\infty, \infty)^2$, exhibiting various rates of integrand decay, and use these families to compare the performances of the two general-purpose two-dimensional cubature algorithms *r2d2lri* and *Cubpack++* for integration over such nonfinite domains. The data collected for this comparison is helpful in identifying the respective advantages and disadvantages of the strategies adopted by the two algorithms when dealing with nonfinite domains.

Keywords: multidimensional integration, automatic cubature, infinite domains, lattice rules, integral test families, Cubpack++, r2d2lri

AMS subject classification: 65D30, 68T35

1. Introduction

We compare the effectiveness of the two general-purpose two-dimensional cubature algorithms *Cubpack++* [2] and *r2d2lri* [6] when evaluating integrals of the form

$$\int_0^\infty \int_0^\infty f(x, y)\,\mathrm{d}y\,\mathrm{d}x \quad \text{and} \quad \int_{-\infty}^\infty \int_{-\infty}^\infty f(x, y)\,\mathrm{d}y\,\mathrm{d}x.$$

After summarizing the quite different strategies employed by the two algorithms, we introduce new sets of parameterized integral test families over nonfinite domains and use these to compare the effectiveness of those strategies.

2. Strategies employed by *Cubpack++* and *r2d2lri*

2.1. *Cubpack++*

Cubpack++ is a C++ algorithm for integration over compound two-dimensional domains. It uses transformations to render the domain a union of primitive regions, then applies a global adaptive subdivisional scheme with rules specific to each primitive region.

In the case of the plane, *Cubpack++* first applies a 36-point degree 13 symmetric rule due to Cools and Haegemans [1]. If a successful result is not achieved (which is normally the case), the plane is mapped onto a circle of radius R using what we will call a radial transformation, the value of R being determined in the course of evaluating the original 36-point rule for the plane. After translation and scaling, a 36-point degree 13 symmetric rule for the unit circle (also due to Cools and Haegemans [1]) is then applied and if this does not produce the desired accuracy, the circle is adaptively subdivided into a smaller circle and four polar rectangles, the latter being the subject of further translation into parallelograms so that subsequent cubatures can be performed.

For integration over the quarter-plane $[0, \infty)^2$, *Cubpack++* performs the sequence of domain transformations

$$\textit{Plane sector} \Rightarrow \textit{Semi-infinite strip} \Rightarrow \textit{Infinite strip} \Rightarrow \textit{Plane}$$

and then proceeds as for integration over the plane.

2.2. *r2d2lri*

r2d2lri [6] is designed for integrals of the form

$$I(x, y) = \int_a^b \int_{g(x)}^{h(x)} f(x, y)\,\mathrm{d}y\,\mathrm{d}x,$$

where the limits of integration a, b, $g(x)$ and $h(x)$ may be finite or infinite. (An earlier version of the algorithm, named *d2lri*, is described in [5].) The algorithm first maps the integration region onto $[0, 1]^2$ and then applies a non-adaptive sequence of embedded lattice rules, coupled with a sixth-order Sidi transformation [7]. The seed lattice for the embedded sequence is the 89-point Fibonacci lattice.

When dealing with integrals over nonfinite domains, *r2d2lri* applies an ordered succession of up to three transformations until it is determined that the requested accu-

Table 1
Transformations for mapping the plane onto $[0, 1]^2$.

1. First attempt: Radial transformation

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\,\mathrm{d}y\,\mathrm{d}x = 2\pi R^2 \int_0^1\int_0^1 v f(Rv\cos(2\pi u), y\sin(2\pi u)) + \frac{1}{v^3} f\Big(\frac{R\cos(2\pi u)}{v}, \frac{R\sin(2\pi u)}{v}\Big)\,\mathrm{d}v\,\mathrm{d}u$$

2. Second attempt: Rational transformation

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\,\mathrm{d}y\,\mathrm{d}x = \int_0^1\int_0^1 \Big(\frac{1}{(1-u)^2}+\frac{1}{u^2}\Big)\Big(\frac{1}{(1-v)^2}+\frac{1}{v^2}\Big) f\Big(\frac{1}{1-u}-\frac{1}{u}, \frac{1}{1-v}-\frac{1}{v}\Big)\,\mathrm{d}v\,\mathrm{d}u$$

3. Third attempt: Logarithmic transformation

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\,\mathrm{d}y\,\mathrm{d}x = \int_0^1\int_0^1 \frac{1}{u(1-u)}\,\frac{1}{v(1-v)}\, f\Big(\log\Big(\frac{u}{1-u}\Big), \log\Big(\frac{v}{1-v}\Big)\Big)\,\mathrm{d}v\,\mathrm{d}u$$

Table 2
Transformations for mapping the quarter plane onto $[0, 1]^2$.

1. First attempt: Rational transformation

$$\int_0^\infty \int_0^\infty f(x, y)\,\mathrm{d}y\,\mathrm{d}x = \int_0^1 \int_0^1 \frac{1}{u^2}\frac{1}{v^2} f\Big(\frac{1}{u} - 1, \frac{1}{v} - 1\Big)\,\mathrm{d}v\,\mathrm{d}u$$

2. Second attempt: Logarithmic transformation

$$\int_0^\infty \int_0^\infty f(x, y)\,\mathrm{d}y\,\mathrm{d}x = \int_0^1 \int_0^1 \frac{1}{1-u}\frac{1}{1-v} f\Big(\log\Big(\frac{1}{1-u}\Big), \log\Big(\frac{1}{1-v}\Big)\Big)\,\mathrm{d}v\,\mathrm{d}u$$

racy has been (or cannot be) attained. The speed of the non-adaptive core integrator in the algorithm makes this approach feasible.

Table 1 gives the succession of transformations used to map the plane onto $[0, 1]^2$. What we term the radial transformation is based on the transformation used in *Cubpack++* for integration over a plane, with the value of R determined using an initial application of the 36-point Cools–Haegemans rule for a plane. If this transformation does not result in a cubature of the requested accuracy, further cubatures are attempted using the rational transformation and then, if required, the logarithmic transformation.

Table 2 gives the succession of transformations used to map the quarter-plane onto $[0, 1]^2$.

3. Test integral families for nonfinite domains

Genz [3,4] defined a set of parameterized integral test families over $[0, 1]^s$, $s = 1, 2, 3, \ldots$, incorporating the use of both translation parameters and scaled difficulty parameters. Adapting Genz's approach, we present two sets of parameterized integral test families, over the domains $[0, \infty)^2$ and $(-\infty, \infty)^2$, respectively, exhibiting polynomial, rational or exponential integrand decay, or hybrid forms of these decay rates. The integrand parameters are characterized as follows:

- Translation parameters: $0 < w_x, w_y < 1$, uniformly distributed on $[0, 1]$.
- Difficulty parameters: $c_x, c_y > 0$, generated from chosen h and e, and numbers $0 < c'_x, c'_y \leqslant 1$, which are uniformly distributed on $[0, 1]$. The factor $\gamma = h/(2^e(c'_x + c'_y))$ is used to generate $c_x = \gamma c'_x$ and $c_y = \gamma c'_y$.

The new integral families are listed in tables 3 and 4. The auxiliary functions defining the parameters used in the test families have been chosen to scale the difficulty of the problems to an appropriate level when $e = 1$, and to maintain the property that for fixed e, the difficulty increases for increasing h.

4. Evaluation tools and coding considerations

A cubature algorithm performance evaluation program has been developed, incorporating the new sets of integral test families (as well as the original Genz families over

Table 3
Test families for the quarter plane: $[0, \infty)^2$.

Family QP1: Polynomial decay, peak at origin

$$\int_0^\infty \int_0^\infty \frac{1}{(x^2 + y^2 + c^{-1})^2}\,\mathrm{d}y\,\mathrm{d}x = \frac{c\pi}{4}, \quad c = \frac{1}{200(c_x + c_y + 1)}.$$

Family QP2: Rational decay, displaced peak

$$\int_0^\infty \int_0^\infty \frac{1}{(x^2 + a^2)^u}\,\frac{1}{(y^2 + b^2)^v}\,\mathrm{d}y\,\mathrm{d}x = \frac{a^{1-2u}\,b^{1-2v}\pi\,\Gamma(u - 1/2)\,\Gamma(v - 1/2)}{4\,\Gamma(u)\,\Gamma(v)},$$

$$a, b, u, v > 0, \quad a = w_x, \quad b = w_y, \quad u = \frac{2}{\log(c_x + 3/2)}, \quad v = \frac{2}{\log(c_y + 3/2)}.$$

Family QP3: Rational decay

$$\int_0^\infty \int_0^\infty \frac{1}{(x + a)^u}\,\frac{1}{(y + b)^v}\,\mathrm{d}y\,\mathrm{d}x = \frac{a^{1-u}\,b^{1-v}}{(u - 1)(v - 1)},$$

$$a, b > 0, \quad 0 < u, v < 1, \quad a = w_x, \quad b = w_y, \quad u = \frac{4}{\log(c_x + 2)}, \quad v = \frac{4}{\log(c_y + 2)}.$$

Family QP4: Rational decay, boundary singularities

$$\int_0^\infty \int_0^\infty \frac{1}{(x + y + c)^a \sqrt{xy}}\,\mathrm{d}y\,\mathrm{d}x = \frac{c^{1-a}\pi}{a - 1}, \quad c \geqslant 0, \quad a > 1, \quad c = w_x + w_y,$$

$$a = \frac{9}{2} - \log(c_x + c_y + 1).$$

Family QP5: Exponential decay

$$\int_0^\infty \int_0^\infty \mathrm{e}^{-(x^2+y^2)/t}\,\mathrm{d}y\,\mathrm{d}x = \frac{\pi t}{4}, \quad t = 5(c_x + c_y + 1).$$

Family QP6: Exponential decay, oscillating

$$\int_0^\infty \int_0^\infty \cos(x^2 + y^2)\mathrm{e}^{-t(x^2+y^2)}\,\mathrm{d}y\,\mathrm{d}x = \frac{\pi t}{4}\Big(\frac{1}{1 + t^2}\Big), \quad t = \frac{20}{c_x + c_y + 1}.$$

Family QP7: Exponential decay

$$\int_0^\infty \int_0^\infty x^a y^b \mathrm{e}^{-(x/u+y/v)}\,\mathrm{d}y\,\mathrm{d}x = u^{a+1} v^{b+1}\,\Gamma(a + 1)\,\Gamma(b + 1),$$

$$a, b, u, v > 0, \quad a = \log(c_x + 1), \quad b = \log(c_y + 1), \quad u = 4a\Big(w_x + \frac{1}{10}\Big), \quad v = 4b\Big(w_y + \frac{1}{10}\Big).$$

Family QP8: Hybrid polynomial–exponential decay

$$\int_0^\infty \int_0^\infty \frac{\mathrm{e}^{-rx}}{(x + a)^2}\,\frac{\mathrm{e}^{-qy}}{(y + b)^2}\,\mathrm{d}y\,\mathrm{d}x = \frac{1}{a} + \frac{1}{b} + r\mathrm{e}^{ar}\mathrm{Ei}(-ar) + q\mathrm{e}^{bq}\mathrm{Ei}(-bq),$$

$$a, b, r, q > 0, \quad a = w_x, \quad b = w_y, \quad r = \Big(c_x + \frac{1}{10}\Big)^{-1}, \quad q = \Big(c_y + \frac{1}{10}\Big)^{-1}.$$

$[0, 1]^2$ and extensions to them). The architecture of the software is sufficiently flexible to accommodate, with only minor changes, many different types of cubature algorithms. The values of e and h are supplied at runtime. An explanation of the information collected appears on the first page of the output generated by the evaluation software and is reproduced in figure 1.

Table 4
Test families for the plane: $(-\infty, \infty)^2$.

Family P1: Polynomial decay, peak at origin

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{(x^2+y^2+c^{-1})^2}\,\mathrm{d}y\,\mathrm{d}x = c\pi, \quad c = \frac{c_x + c_y + 1}{30}.$$

Family P2: Polynomial decay, displaced peak

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{((x+a)^2+(y+b)^2+c^{-1})^2}\,\mathrm{d}y\,\mathrm{d}x = c\pi, \quad c = \frac{c_x + c_y + 1}{30}, \quad a = 2w_x - 1,$$
$$b = 2w_y - 1.$$

Family P3: Rational decay

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{(x^2+a^2)^u}\,\frac{1}{(y^2+b^2)^v}\,\mathrm{d}y\,\mathrm{d}x = \frac{a^{1-2u}\,b^{1-2v}\pi\,\Gamma(u-1/2)\,\Gamma(v-1/2)}{\Gamma(u)\,\Gamma(v)},$$
$$a, b, u, v > 0, \quad a = c_x + 1, \quad b = c_y + 1, \quad u = 2/\log(c_x + 3/2), \quad v = 2/\log(c_y + 3/2).$$

Family P4: Exponential decay

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-(x^2+y^2)/t}\,\mathrm{d}y\,\mathrm{d}x = \pi t, \quad t = 5(c_x + c_y + 1).$$

Family P5: Exponential decay, displaced point of symmetry

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-((x-a)^2+(y-b)^2)/t}\,\mathrm{d}y\,\mathrm{d}x = \pi t, \quad t = 5(c_x + c_y + 1), \quad a = 4w_x - 2, \quad b = 4w_y - 2.$$

Family P6: Exponential decay, oscillating

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \cos(x^2+y^2)\mathrm{e}^{-t(x^2+y^2)}\,\mathrm{d}y\,\mathrm{d}x = \frac{\pi t}{1+t^2}, \quad t = \frac{8}{c_x + c_y + 1}.$$

Family P7: Exponential decay, oscillating, displaced point of symmetry

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \cos((x-a)^2+(y-b)^2)\,\mathrm{e}^{-t((x-a)^2+(y-b)^2)}\,\mathrm{d}y\,\mathrm{d}x = \frac{\pi t}{1+t^2},$$
$$t = 8/(c_x + c_y + 1), \quad a = 4w_x - 2, \quad b = 4w_y - 2.$$

Family P8: Exponential decay

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^\alpha y^\beta \mathrm{e}^{-(x^2+y^2)}\,\mathrm{d}y\,\mathrm{d}x = \Gamma\Big(\frac{\alpha+1}{2}\Big)\,\Gamma\Big(\frac{\beta+1}{2}\Big), \quad \alpha = 2\lfloor 5\log(c_x+1)\rfloor,$$
$$\beta = 2\lfloor 5\log(c_y+1)\rfloor.$$

Family P9: Exponential decay, displaced point of symmetry

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-a)^\alpha (y-b)^\beta \mathrm{e}^{-((x-a)^2+(y-b^2))}\,\mathrm{d}y\,\mathrm{d}x = \Gamma\Big(\frac{\alpha+1}{2}\Big)\,\Gamma\Big(\frac{\beta+1}{2}\Big),$$
$$a = 4w_x - 2, \quad b = 4w_y - 2, \quad \alpha = 2\lfloor 5\log(c_x+1)\rfloor, \quad \beta = 2\lfloor 5\log(c_y+1)\rfloor.$$

Family P10: Exponential decay

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^\alpha y^\beta \mathrm{e}^{-\sqrt{x^2+y^2}}\,\mathrm{d}y\,\mathrm{d}x = 2(\alpha+\beta+1)!\,\frac{\Gamma((\alpha+1)/2)\,\Gamma((\beta+1)/2)}{\Gamma((\alpha+\beta+2)/2)}, \quad \alpha = 2\lfloor 2\sqrt{c_x}\rfloor,$$
$$\beta = 2\lfloor 2\sqrt{c_y}\rfloor.$$

Table 4
(Continued).

Family P11: Exponential decay, displaced point of symmetry

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-a)^{\alpha}(y-b)^{\beta}\mathrm{e}^{-\sqrt{(x-a)^2+(y-b)^2}}\,\mathrm{d}y\,\mathrm{d}x =$$

$$2(\alpha+\beta+1)!\frac{\Gamma((\alpha+1)/2)\Gamma((\beta+1)/2)}{\Gamma((\alpha+\beta+2)/2)},$$

$$a = 4w_x - 2, \quad b = 4w_y - 2, \quad \alpha = 2\lfloor 2\sqrt{c_x}\rfloor, \quad \beta = 2\lfloor 2\sqrt{c_y}\rfloor.$$

The integrand in each test family has been conservatively coded, which, whilst increasing the time required to perform each cubature, ensures that the integral itself does not contribute any floating-point errors. The elimination of this source of potential runtime error from the cubature evaluation is essential. If such errors do occur, their source is the code for the algorithm being evaluated.

5. Results

For each of the test families in tables 3 and 4, the two algorithms were requested to return results correct to $1, 2, \ldots, 13$ significant figures. For *Cubpack++*, the majority of the tests were performed with the maximum number of points set at 100,000. Specific reference is made where a higher maximum number of points was used. With *r2d2lri*, the maximum number of points is 45,568 for the quarter plane and 91,172 for the plane.

All data was generated using code compiled with Borland C++ 5.02 running under Windows 98SE, on an 850 MHz Pentium III machine with 384 MB of memory.

Examples of the output from the evaluation code are provided in tables 3 and 4. (The definitions of the column headings are given in figure 1.) Such tables were generated for each of *Cubpack++* and *r2d2lri* with $e = 1$ and $h = 10(10)50$ over the two regions $[0, \infty)^2$ and $(-\infty, \infty)^2$. A selection of the results are depicted in figures 2–5 which show the number of function values and time needed (in milliseconds) to successfully return the requested accuracies for four of the families. In addition, the performances of *Cubpack++* and *r2d2lri* over the two regions with $e = 1$ and $h = 10$ and $h = 50$ are summarized in tables 7 and 8. These tables report the maximum number of requested significant figures successfully returned in at least 90% of cases (up to 13 significant figures) by the respective algorithms. Cases where unreliable results are returned have been appropriately identified.

In many instances, the imposition on *Cubpack++* of a maximum of 100,000 points limits the accuracy that this algorithm can successfully return. Further testing was performed with the integral families over a plane, using more than the original 100,000 points. Table 9 contains representative data from these tests.

TEST DETAILS
Platform: Pentium III 850 MHz, W98SE, Borland C++ 5.02
Algorithm tested: r2d2lri
Number of dimensions: 2
Integration domain: [0,INFINITY)^2
Date of test: 4/4/2001
Test integrals version number: 3.6, 5/8/2000
Evaluation code version number: 4.5.2 (xxd2lri), 7/4/2000
Input file name: parameter_values(1)
Output file name: parameter_values(1)_analysis
The number of test integrals per family: 20
The number of times each integral is computed (REPEATS): 200
Difficulty parameters: $h = 30$, $e = 1$
The definitions of e and h are given in Chapter 11 of "LATTICE RULES FOR MULTIPLE INTEGRATION"
by Sloan and Joe, Clarendon Press, 1994.

EXPLANATION OF OUTPUT:
For each integral family and each requested relative error
$0.5 \cdot 10^{(-k)}$, $k = 1, 2, \ldots, 13$, the following data is collected:

- The median number of function values used by the cubature algorithm.
- The mean number of function values used by the cubature algorithm.
- The interquartile range of the number of function values used by the cubature algorithm.
- The quality indicators:
  - EFF = EFFECTIVENESS: The percentage of times the algorithm returns a cubature with less than or equal to the requested error and signals that it has succeeded.
  - EEP = ERROR ESTIMATE PERFORMANCE: The percentage of times the algorithm returns an error estimate that is less than the actual error in the cubature.
  - RBST = ROBUSTNESS: The percentage of times the algorithm returns a sound result. Either it returns a cubature with the actual and estimated errors less than or equal to the requested error, or it signals that it has failed because it decides the error requirements have not been met and/or it has reached the maximum permitted number of function evaluations.
  - UNRm = UNRELIABILITY (minor): The percentage of times the algorithm signals a success but returns a marginally incorrect result. (The cubature error is greater than the requested error by less than 1 order of magnitude.)
  - UNRM = UNRELIABILITY (major): The percentage of times the algorithm signals a success but returns a clearly incorrect result. (The cubature error is greater than the requested error by at least 1 order of magnitude.)

Figure 1. Sample first page of the output from the cubature algorithm performance evaluation software.

- The median cubature evaluation time in milliseconds. The accuracy of this figure increases with the value of REPEATS. REPEATS = 1 indicates that the test was performed only to assess the accuracy and reliability of the algorithm under test.
- The mean cubature evaluation time in milliseconds. The accuracy of this figure is affected by the value of REPEATS in the same way as for the median cubature evaluation time.

Figure 1. (Continued.)

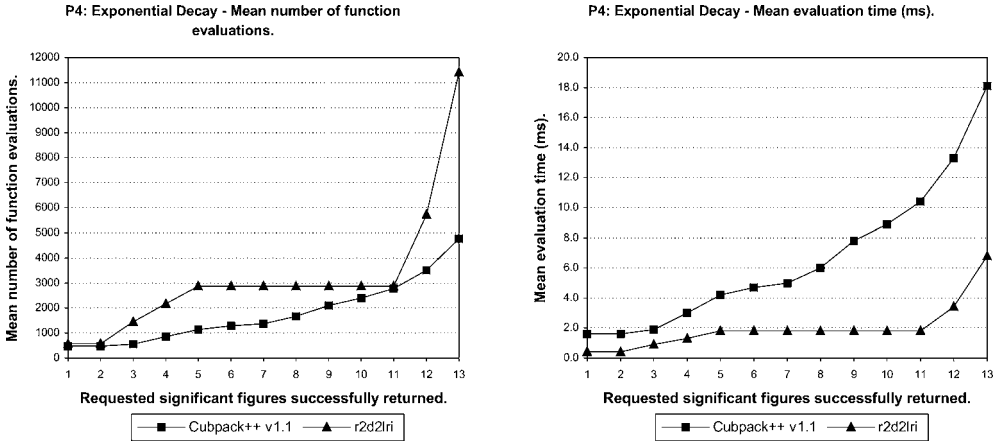

Figure 2. Performance results for family P4: $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-(x^2+y^2)/t}\,\mathrm{d}y\,\mathrm{d}x$.

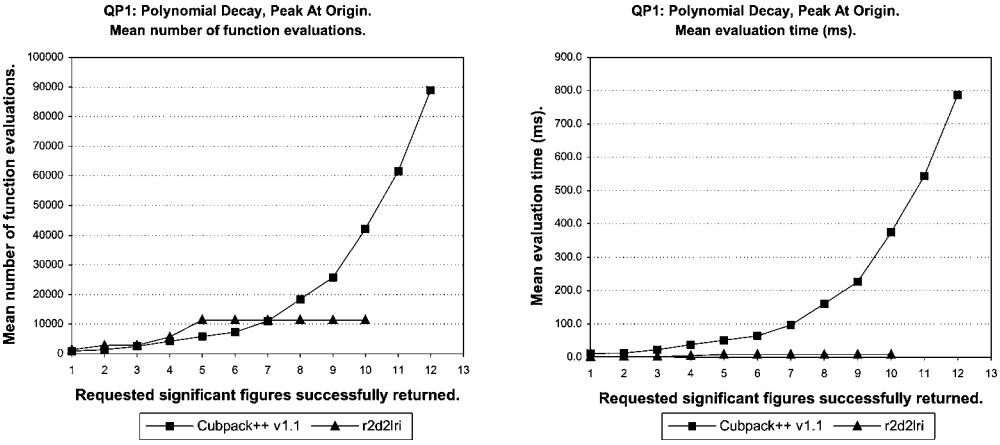

Figure 3. Performance results for family QP1: $\int_0^{\infty}\int_0^{\infty} 1/(x^2+y^2+c^{-1})^2\,\mathrm{d}y\,\mathrm{d}x$.

6. Observations

Based on the results of this investigation, the strategy employed in *Cubpack++* for evaluating integrals over (the two studied) infinite regions is generally effective and

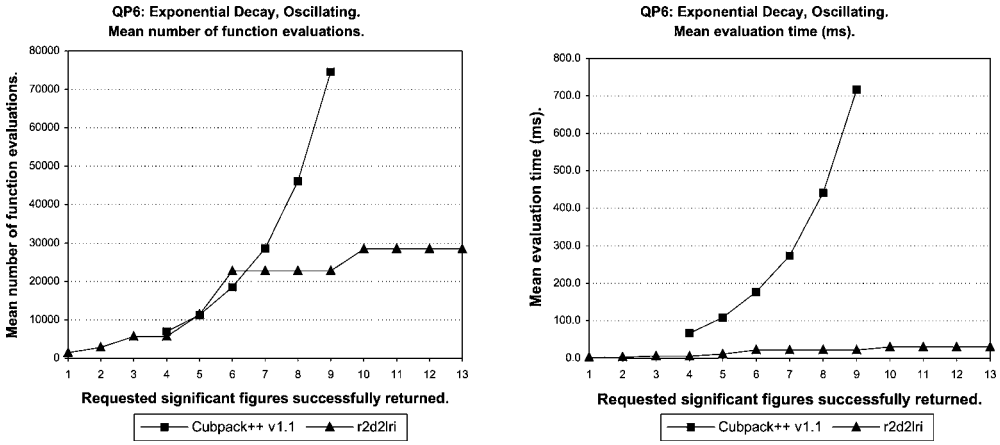

Figure 4. Performance results for family QP6: $\int_0^\infty \int_0^\infty \cos(x^2 + y^2)\mathrm{e}^{-t(x^2+y^2)}\,\mathrm{d}y\,\mathrm{d}x$.

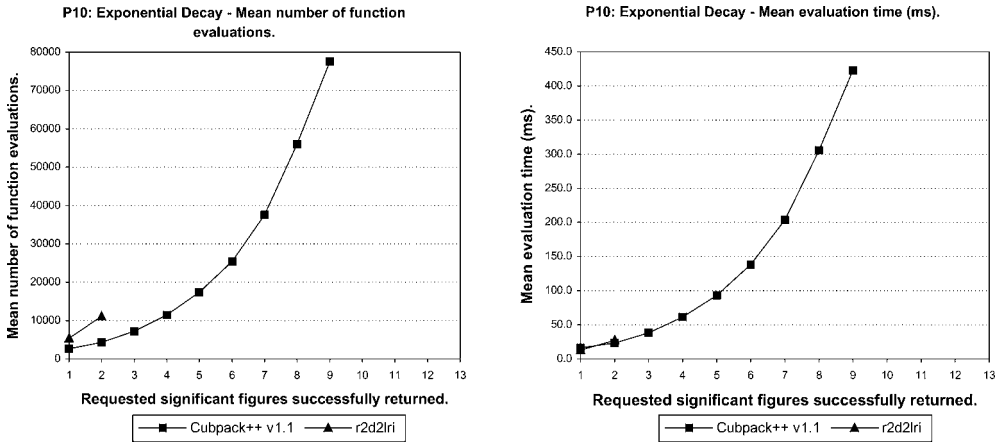

Figure 5. Performance results for family P10: $\int_{-\infty}^\infty \int_{-\infty}^\infty x^\alpha y^\beta \mathrm{e}^{-\sqrt{x^2+y^2}}\,\mathrm{d}y\,\mathrm{d}x$.

reliable. With few exceptions, provided the maximum number of function evaluations permitted is set at a high enough level, the algorithm successfully returns all requested accuracies up to 13 figures.

The obvious exceptions are those integrals displaying very slow (rational) decay (QP2, QP3, QP4 and P3), in which cases, only very low accuracies are achievable or the algorithm fails due to the production of floating-point errors (resulting from the generation of *Infs* in the internal transformations). Also, some unreliable results are returned when low accuracies are requested and the integrand is oscillatory in nature (QP6, P6) or, among these test integrals, of low to moderate difficulty.

In terms of effectiveness, *Cubpack++* is relatively insensitive to translation of the point of symmetry for integrations over the plane, though its efficiency is generally adversely affected by such translations. (The loss of efficiency could be avoided by

Table 5
Sample output from the evaluation software (*Cubpack++*).

Algorithm tested: *Cubpack++* v1.1
Integration domain: [0, Infinity)^2
The number of times each integral is computed (REPEATS): 10
Difficulty parameters: $h = 30$, $e = 1$
FAMILY QP8: Hybrid polynomial–exponential decay, 20 test integrals

Req rel error	# of function values		Interquartile range		Quality					Time	
	Median	Mean			EFF	EEP	RBST	UNRm	UNRM	Median	Mean
5.0 e–02	200	250	108	292	100	10	100	0	0	1.8	2.3
5.0 e–03	909	844	759	983	95	5	95	5	0	9.0	10.7
5.0 e–04	2082	2021	1890	2127	100	0	100	0	0	20.0	19.7
5.0 e–05	3481	3536	3296	3681	100	0	100	0	0	33.8	34.2
5.0 e–06	5590	5725	5442	5886	100	0	100	0	0	54.3	55.4
5.0 e–07	8920	9171	8772	9374	100	0	100	0	0	87.3	89.0
5.0 e–08	14972	14978	14404	15608	100	0	100	0	0	142.8	144.5
5.0 e–09	24005	24173	23549	24562	100	0	100	0	0	231.5	233.6
5.0 e–10	36335	36502	34931	37958	100	0	100	0	0	352.8	353.8
5.0 e–11	54321	54652	52469	57325	100	0	100	0	0	526.8	529.6
5.0 e–12	80108	81342	75693	85643	100	0	100	0	0	779.1	789.7
5.0 e–13	100057	100064	100029	100101	0	0	100	0	0	972.1	971.3
5.0 e–14	100057	100064	100029	100101	0	0	100	0	0	972.1	971.3

Table 6
Sample output from the evaluation software (*r2d2lri*).

Algorithm tested: *r2d2lri*
Integration domain: [0, Infinity)^2
The number of times each integral is computed (REPEATS): 200
Difficulty parameters: $h = 30$, $e = 1$
FAMILY QP8: Hybrid polynomial–exponential decay, 20 test integrals

Req rel error	# of function values		Interquartile range		Quality					Time	
	Median	Mean			EFF	EEP	RBST	UNRm	UNRM	Median	Mean
5.0 e–02	265	265	265	265	100	0	100	0	0	0.4	0.4
5.0 e–03	265	274	265	265	100	0	100	0	0	0.4	0.4
5.0 e–04	265	367	265	353	100	0	100	0	0	0.4	0.5
5.0 e–05	709	779	353	1063	100	0	100	0	0	1.0	1.1
5.0 e–06	1417	1258	1417	1417	100	0	100	0	0	2.0	1.8
5.0 e–07	1417	1328	1417	1417	100	0	100	0	0	2.0	1.9
5.0 e–08	1417	1542	1417	1417	100	0	100	0	0	2.0	2.2
5.0 e–09	2841	2910	1417	2841	100	0	100	0	0	4.0	4.1
5.0 e–10	4261	3691	1417	5681	100	0	100	0	0	5.9	5.2
5.0 e–11	5681	4047	1417	5681	100	0	100	0	0	7.9	5.7
5.0 e–12	5681	4473	2841	5681	100	0	100	0	0	7.9	6.3
5.0 e–13	5681	5255	5681	5681	100	0	100	0	0	7.9	7.4
5.0 e–14	5681	5966	5681	5681	100	0	100	0	0	7.9	8.4

Table 7
Highest accuracies successfully returned for test families QP1–QP8.

		QP1	QP2	QP3	QP4	QP5	QP6	QP7	QP8
$e = 1, h = 10$	*Cubpack++*	12	#	#	#	11	11*	12	11
	r2d2lri	12	13	13	6	13	13	12	13
$e = 1, h = 50$	*Cubpack++*	12*	#	#	#	11	8*	12	11
	r2d2lri	9	5	5	1	5	13	9	13

Cubpack++ cannot complete these tests due to the occurrence of floating-point errors.
*Unreliable results returned when low accuracies are requested.

Table 8
Highest accuracies successfully returned for test families P1–P11.

		P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
$e = 1, h = 10$	*Cubpack++*	13	13	3	13*	13*	11*	9	9	9*	10	10
	r2d2lri	13	12	13	13	13	13	13	13	13*	8	7
$e = 1, h = 50$	*Cubpack++*	13	13	0	13	13	12	6	9	9	10	10
	r2d2lri	13	4	4*	13	13	13	2	13	12	0	0

*Unreliable results returned when low accuracies are requested.

Table 9
Highest accuracies successfully returned by *Cubpack++* ($e = 1, h = 30$).

Test family	Maximum points	Average number of points used$^+$	Returned accuracy$^+$	Average time (sec)
P3*	1,000,000	25,848	1	0.15
P7	1, 000,000	565,385	13	2.38
P8	400,000	340,584	13	1.99
P9	400,000	341,823	13	2.00
P10	400,000	321,745	13	1.78
P11	400,000	332,323	13	1.82

$^+$The average number of points required to reliably return the number of significant figures of accuracy tabulated in the column Returned accuracy in at least 90% of the integrals tested.
*No improvement beyond that when a maximum of 100,000 points are used. Requests for more than 1 significant figure of accuracy sees *Cubpack++* consistently return unreliable results. For requests for 10 or more significant figures of accuracy, *Cubpack++*'s maximum of 1,000,000 points is often used.

using a *Cubpack++* feature which allows information about the point of symmetry to be specified.) This relative insensitivity can be explained by *Cubpack++*'s adaptive approach which enables cubature points to be concentrated where they are most needed.

r2d2lri is also generally effective for the integrals tested when moderate to high accuracies are requested. Its effectiveness lessens as the rate of decay in the integrand slows, though a high degree of reliability is maintained (failures to achieve the requested accuracy are correctly flagged). Both in efficiency and its ability to return high accuracies, the algorithm is adversely affected by translation of the point of symmetry for integrations over the plane.

Significantly, *r2d2lri* encounters no floating-point problems for the integrals tested and can achieve moderate to high accuracies for the slowly decaying integrands QP2, QP3, QP4 and P3. A contributant to this outcome is likely to be the conservatism in the choice of transformations used in the algorithm. First, Sidi transformations are known to have polynomial-like behaviour near 0 and 1 and to generate points not nearly so close to the boundaries of the integration region as do other periodizing transformations such as the IMT and tanh rules [7]. Furthermore, tuning of the algorithm to standard double precision arithmetic resulted in

1. the choice of the sixth-order Sidi transformation over higher-order Sidi transformations, and
2. conservative choices being made for the rational and logarithmic functions used for mapping infinite regions onto the unit square.

A clear outcome of the testing is that *r2d2lri* is very much faster than *Cubpack++* in almost all cases, often by between one and two orders of magnitude. This is a direct result of the simplicity of the algorithm and especially the use of pre-computed cubature nodes and weights.

Scaling can become an issue when performing numerical integration over nonfinite regions. Of the new test families presented here, only QP5, P4 and P5 exhibit some form of axis scaling as the difficulty parameter changes. For example, note that in P4

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-(x^2+y^2)/t}\,\mathrm{d}y\,\mathrm{d}x = t\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-(x^2+y^2)}\,\mathrm{d}y\,\mathrm{d}x,$$

so a change in the difficulty parameter t has the effect of re-scaling the integrand by equal amounts in the x and y directions. The evaluation of P4 for large values of h, e.g., $h = 500$ (where the average value of t is 125) poses no problems for either *Cubpack++* or *r2d2lri*, the form of the integral being particularly amenable to both algorithms. Even the random displacement of the point of symmetry (that becomes further from the origin with increasing h) in P5 does not present difficulties for either algorithm.

To collect data with more potential to reflect any problems that might arise when evaluating ill-scaled integrands, we have created two examples based on members of the set of RD test integrals [5], which we have numbered

$$\text{RD30a:}\quad \int_0^{\infty}\int_0^{\infty} \frac{1}{(px+qy+1)^2\sqrt{pqxy}}\,\mathrm{d}y\,\mathrm{d}x = \frac{\pi}{pq}, \quad p,q>0,$$

and

$$\text{RD34a:}\quad \int_0^{\infty}\int_0^{\infty} (px)^{-2/3}(qy)^{-1/3}\mathrm{e}^{-(px+qy+1/(pqxy))}\,\mathrm{d}y\,\mathrm{d}x = \frac{2\pi}{\mathrm{e}^3 pq\sqrt{3}}, \quad p,q>0.$$

Depending on the sizes of the coefficients p and q, RD30a and RD34a range from being *difficult* to *very difficult* integrals. To illustrate the effect on the returned cubatures of varying the coefficients p and q, we present in table 10 data collected from *Cubpack++* and *r2d2lri* when $p = q \in \{1, 100, 1000\}$. *Cubpack++* was initialized to

Table 10
Performance on test integrals RD30a and RD34a.

Algorithm	p, q	RD30a	RD34a
Cubpack++	1	8 (71681) (500109)	10* (47308)
	100	8 (81091) (500005)	13 (124387)
	1000	8 (100835) (500022)	13 (135730)
r2d2lri	1	7 (11377) (17042)	13 (11377)
	100	6 (1417) (22730)	6 (22753) (45442)
	1000	5 (5681) (39770)	4 (22753) (45442)

For the entries $a(b)(c)$ in the RD30a and RD34a columns:
a is the number of significant figures returned by the algorithm as signalled by the algorithm's error estimate.
b is the number of points used by the algorithm to obtain the accuracy a.
c is the number of points used by the algorithm before it halts in its attempt to obtain more than a significant figures.
*For higher requested accuracies, *Cubpack++* returned cubatures with error estimates that were incorrect by up to 2 significant figures.

permit the evaluation of a maximum of 500,000 points before halting. As with the other tests in this paper, requests for accuracy were from 2 to 13 significant figures.

For these test integrals, *Cubpack++* suffers least from the scaling of the variables, and can return the most accurate cubatures, if one allows the algorithm to use sufficiently large numbers of points. Conversely, *r2d2lri* uses fewer points and returns fewer numbers of significant figures, but is very reliable.

Both algorithms, but *r2d2lri* in particular, return more significant figures more quickly if the integrand can be re-scaled to minimize the demands on the computation. For example, a logical strategy when evaluating

$$\int_0^\infty \int_0^\infty \left(10^3 x\right)^{-2/3} \left(10^3 y\right)^{-1/3} \mathrm{e}^{-(10^3 x + 10^3 y + 1/(10^6 xy))} \,\mathrm{d}y \,\mathrm{d}x$$

is to change variables, then evaluate

$$\int_0^\infty \int_0^\infty u^{-2/3} v^{-1/3} \mathrm{e}^{-(u+v+1/(uv))} \,\mathrm{d}v \,\mathrm{d}u$$

and multiply the result by 10^{-6}.

7. Concluding remarks

In this study, we have considered only integration over the nonfinite domains, $[0, \infty)^2$ and $(-\infty, \infty)^2$. For these domains, provided the integrand decay is not too slow and a sufficiently high maximum number of function values is set, *Cubpack++* is an effective and reliable cubature algorithm. In general, *r2d2lri* is also effective and reliable, though in cases where the requested accuracy cannot be achieved, increasing the maximum number of function values allowed (beyond the current default maximum) is not possible.

For a one-off integral calculation where computing time is not an issue, *Cubpack++* is probably the algorithm of first choice for the regions considered here. However, given the significant speed advantage of *r2d2lri*, if many like integrals are to be evaluated and/or a fast execution time is important, the best strategy would be to try *r2d2lri* first. The likely outcome is that an accurate result will be returned very quickly. However, if sufficient accuracy is not achieved, *Cubpack++* could then be tried with a high maximum number of function values set. In cases where it is necessary to try both algorithms, the cost of first running *r2d2lri* would be insignificant in relative terms.

Improvements to both algorithms may result from the outcomes of a detailed investigation of (and/or experimentation with) different functions for transforming infinite regions onto finite ones. The speed of the core integrator in *r2d2lri* makes viable the addition of more transformations to the selection currently attempted. With the addition of an optional parameter for user-provided information on integrand type, it may also be possible to incorporate a better matching of transformation function to integrand type in both algorithms.

Relatively straightforward improvements to the algorithms could be achieved by the inclusion of exception handling in *Cubpack++* (to improve robustness rather than effectiveness) and provision in *r2d2lri* for user specification of the location of key points such as symmetry points or peaks.

When performing cubatures over a plane, *Cubpack++* has the flexibility to accept user-provided information that will optimize axis scaling. However, neither algorithm can accept user input to optimize axis scaling for other domains. From the data collected here, it is reasonable to suggest that the performance of both algorithms degrades when the integration variables are poorly scaled (though more so for the non-adaptive algorithm). The implied common sense strategy is, where possible, to optimize integrals for axis scaling before using a cubature algorithm to evaluate the integral.

References

[1] R. Cools and A. Haegemans, Construction of fully symmetric cubature formulae of degree $4k - 3$ for fully symmetric planar regions, J. Comput. Appl. Math. 17 (1987) 173–180.

[2] R. Cools, D. Laurie and L. Pluym, Algorithm 764: Cubpack++: A C++ package for automatic two-dimensional cubature, ACM Trans. Math. Software 23(1) (1997) 1–15.

[3] A.C. Genz, Testing multidimensional integration routines, in: *Tools, Methods and Languages for Scientific and Engineering Computation*, eds. B. Ford, J.C. Rault and F. Thomasset (North-Holland, Amsterdam, 1984) pp. 81–94.

[4] A.C. Genz, A package for testing multiple integration subroutines, in: *Numerical Integration: Recent Developments, Software and Applications*, eds. P. Keast and G. Fairweather (Reidel, Dordrecht, 1987) pp. 337–340.

[5] M. Hill and I. Robinson, d2lri: A non-adaptive algorithm for two-dimensional cubature, J. Comput. Appl. Math. 112 (1999) 121–145.

[6] I. Robinson and M. Hill, r2d2lri: An algorithm for automatic two-dimensional cubature, ACM Trans. Math. Software 27(3) (2001) 73–89.

[7] A. Sidi, A new variable transformation for numerical integration, in: *International Series of Numerical Mathematics*, Vol. 112 (1993) pp. 359–373.

[8] I.H. Sloan and S. Joe, *Lattice Methods for Multiple Integration* (Clarendon Press, Oxford, 1994).