

# REPORTS IN INFORMATICS

ISSN 0333-3590

A test of QUADPACK and Four Doubly  
Adaptive Quadrature Routines

Terje O. Espelid

REPORT NO 281

October 2004



*Department of Informatics*  
**UNIVERSITY OF BERGEN**  
*Bergen, Norway*

This report has URL <http://www.ii.uib.no/publikasjoner/texrap/ps/2004-281.ps>

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at  
<http://www.ii.uib.no/publikasjoner/texrap/>.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,  
P.O. Box 7800, N-5020 Bergen, Norway

# A Test of QUADPACK and Four Doubly Adaptive Quadrature Routines

Terje O. Espelid

October 2004

## Abstract

Recently several new doubly adaptive quadrature codes (Matlab) have been published. Through extensive testing we have demonstrated that these codes are superior to Matlab's best currently available quadrature software `quadl` both with respect to efficiency and reliability. In this paper we want to test and compare the new codes to the general purpose codes in QUADPACK (`dqag(e)`, FORTRAN 77). We include the results from extensive testing using both a Lyness-Kaganove testing technique and a battery test.

## 1 Introduction.

Automatic algorithms are now used widely for the numerical calculation of integrals. Since the first such algorithm was given by McKeeman [15] in 1962, many new and sophisticated algorithms, both adaptive and non-adaptive, have been developed, among these [3, 4, 12, 16, 18, 10, 5, 6, 7].

Recently Espelid [5, 6, 7] have published several papers describing a number of new doubly adaptive quadrature codes written in Matlab. Two of these codes, `coteda` and `coteglob`, were compared to Gander and Gautschi's Matlab codes `adaptsim` and `adaptlob`, [9, 10], through a Lyness-Kaganove testing technique, [14], and a battery test, in [5, 6]. Through these tests `coteda` and `coteglob` demonstrate both better reliability and better efficiency than `adaptsim` and `adaptlob`. Especially for moderate to high accuracy requests the two new codes are superior comparing the number of function values needed to meet the accuracy request.

The two codes `adaptsim` and `adaptlob` have, in slightly modified versions, since 2002 replaced Matlab's current quadrature software and is available now as `quad` and `quadl` in Matlab.

The two doubly adaptive codes `coteda` and `coteglob` are both based on a sequence of two Newton-Cotes rules using 5 and 9 points and the difference between these two codes are *local* versus *global* adaptivity. In 2004 Espelid, [7], extended these ideas using up to four rules in the sequence and designed four different experimental codes `da1glob`, `da2glob`, `da3glob` and `da4glob`, all globally doubly adaptive. The major difference between these four codes are the size of the 2-norm to the basic quadrature rules' weights. These codes are using a sequence of up to four rules based on equidistant point sets. The sets chosen have 5, 9, 17 and 33 points.

In [7] these rules are compared to `da0glob` ( a modified version of `coteglob`) and `quadl` through a battery test and a Lyness-Kaganove test. These tests clearly demonstrate that all these five doubly adaptive codes are far better than `quadl` both with respect to reliability and efficiency. Furthermore it is revealed that the four extended codes all are able to retain `da0glob` qualities on problems where adaptability is the essential challenge. In addition these four codes do benefit from having rules of higher polynomial degrees in the sequence when such issues are important in a problem class.

Gander and Gautschi state that their two codes, `adaptsim` and `adaptlob`, are better than other codes available in different software packages in 2/3 of all cases in their battery test. Now, in the tests performed in [5, 6] it becomes clear that `adaptsim` (and thus `quad`) must be considered as a very unreliable code. Removing `adaptsim` from Gander and Gautschi's battery test will influence the major conclusion of being better than other codes in 2/3 of all cases. Based on this fact it is reasonable to compare the new doubly adaptive codes to other reliable quadrature codes available.

In 1986, using a Lyness-Kaganove test, Berntsen [1] demonstrated that the general purpose code `dqag(e)` in QUADPACK (FORTRAN 77), [16], was the most reliable general purpose code available at that time and this code is in addition quite efficient. Thus, this code represents the state of the art in the mid-eighties and as far as I can judge it is still an important milestone in designing reliable quadrature software. Based on this I find it important to run a test of comparison of four of these five doubly adaptive codes (we do not include `da4glob` in this test since this code in general is less efficient than the other three extended codes in the test reported in [7]) and some of QUADPACK's general purpose codes.

The code `dqag(e)` has the option of using 6 different Gauss-Kronrod rules with 15, 21, 31, 41, 51 and 61 points respectively. We choose to compare only two of these six codes with the four doubly adaptive Matlab codes: namely `dqk15` and `dqk21`. The reason for this choice is that for some of the test functions adaptability is essential, thus using few evaluation points (15), and for some test functions high degree may be important when asking for high accuracy results. Picking the 21 point rule implies a degree of precision equal to 30 which is higher than the best rule used in the extended doubly adaptive codes. It is indeed the case that for some problems higher degree rules are an even better choice, however for the difficulty level and test functions in our tests these two choices are the overall best.

## 2 Testing of the six codes.

We report tests on the following six codes: the four first are Matlab-codes while the two last are FORTRAN 77 codes

- `da0glob`: Globally doubly adaptive code based on two rules of degrees 5 and 9.
- `da1glob`: Globally doubly adaptive code based on three rules of degrees 5, 9 and 17.
- `da2glob`: Globally doubly adaptive code based on four rules of degrees 5, 9, 17 and 27.
- `da3glob`: Globally doubly adaptive code based on four rules of degrees 5, 9, 15 and 23.
- `dqk15` : Globally adaptive code (QUADPACK) based on a 15 point Gauss-Kronrod rule.
- `dqk21`: Globally adaptive code (QUADPACK) based on a 21 point Gauss-Kronrod rule.

`da0glob` is just a minor modification of `coteglob` [6]. The four Matlab codes are available on the Web via T. O. Espelid's homepage: <http://www.ii.uib.no/~terje/>.

### 2.1 Results from the Lyness-Kaganove tests.

The test families used in our Lyness-Kaganove testing are given in Table 1, and they are picked from [1], [14] and [17]. In our experiments we have chosen the difficulty parameters  $\alpha_i, i = 1, \dots, 6$ , to be (numbered from family 1 to 6):  $\underline{\alpha} = (-0.5, 0.5, 2.0, -4.0, -2.0, 2.0)$ . The parameters,  $\lambda$  (or  $\lambda_i, i = 1, \dots, 4$ , for test family 5), are picked from the region of integration using a uniform random number generator, `rand`, written by W. Fullerton [8] (a FORTRAN 77 code found in National Institute of Standards and Technology's DMLIB).

We have tested the codes for absolute error tolerances  $\text{tol} = 10^{-1}, 10^{-2}, \dots, 10^{-12}$ . (For the Test family 1 we stop at  $10^{-5}$ .) For these values of `tol` and for each test family we have asked all routines to compute the integrals for 1000 samples of random parameters, and in most cases all the six codes report that the returned values satisfies the error request. The 1000 samples are generated by `rand` with seed 0.0 while for test family 5 we used four such sequences with seeds 0.0,  $\pi$ ,  $\sqrt{2}$  and  $\sqrt{3}$  respectively. The reason for choosing `rand` is it's portability and the possibility to reproduce the results. We observe good agreement between these test results and the test results reported in [7] for the codes appearing in both tests. One exception is the results for Family 1 where there seems to be some misprints in the results reported in [7].

For families 5 and 6 two of the doubly adaptive routines give a warning when `tol = 10^{-12}` indicating that `tol` is below the global noise level and that these two routines therefore had to

Table 1: The six test families.

1.	$\int_0^1 ( x - \lambda )^{\alpha_1} dx$	Singularity
2.	$\int_0^1 f_2(x) dx$ where $f_2(x, y) = \begin{cases} 0 & \text{if } x \leq \lambda \\ \exp(\alpha_2 x) & \text{otherwise} \end{cases}$	Discontinuous
3.	$\int_0^1 \exp(-\alpha_3  x - \lambda ) dx$	$C_0$ function
4.	$\int_1^2 10^{\alpha_4} / ((x - \lambda)^2 + 10^{2\alpha_4}) dx$	One Peak
5.	$\int_1^2 \sum_{i=1}^4 10^{\alpha_5} / ((x - \lambda_i)^2 + 10^{2\alpha_5}) dx$	Four Peaks
6.	$\int_0^1 2B(x - \lambda) \cos(B(x - \lambda)^2) dx$ where $B = 10^{\alpha_6} / \max(\lambda^2, (1 - \lambda)^2)$	Non-linear Oscillatory

modify `tol` with the possible effect that the true error may be larger than the specified tolerance. For the complete test results we refer to the Appendix where we list six tables containing all the results from these tests.

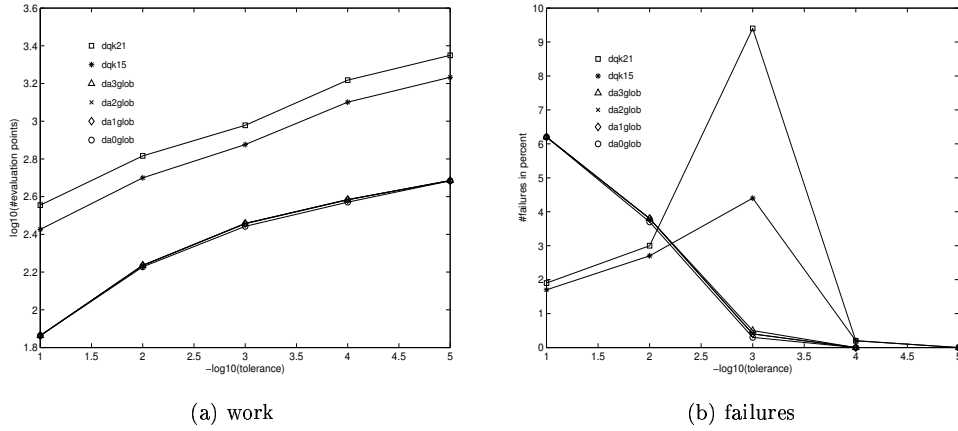
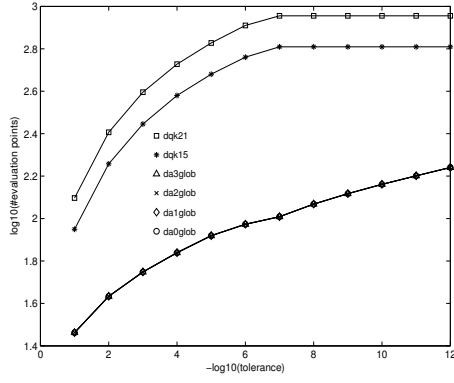
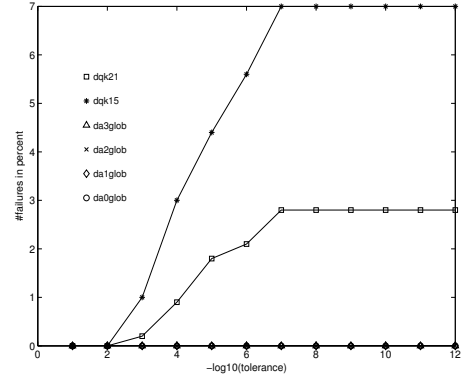


Figure 1: Test family 1 (Singularity).

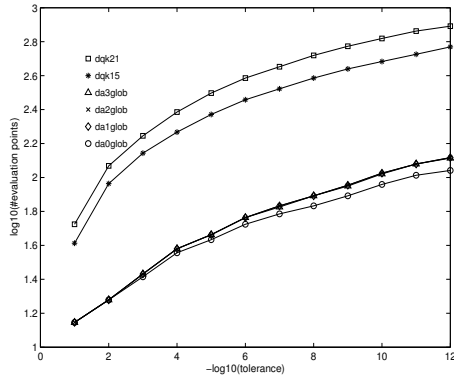


(a) work

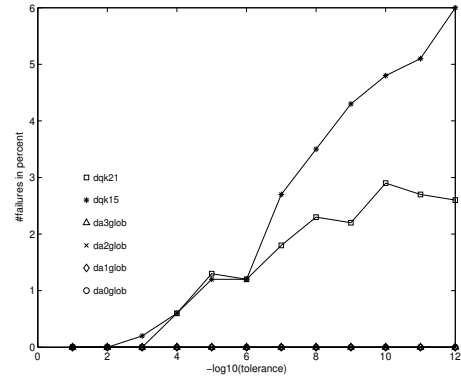


(b) failures

Figure 2: Test family 2 (Discontinuous).

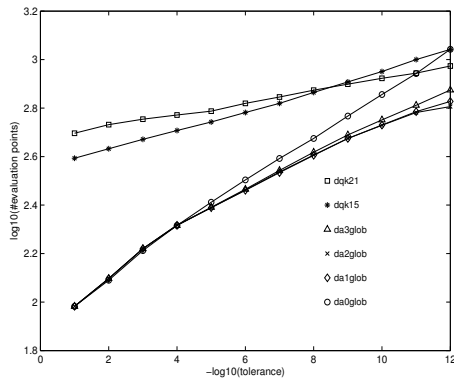


(a) work

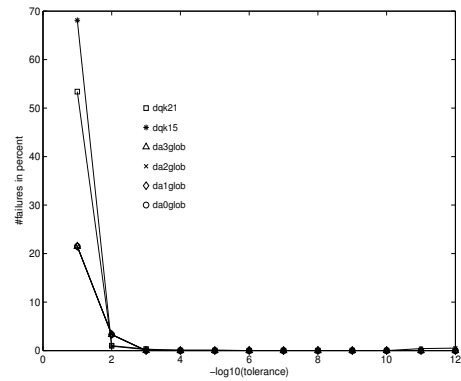


(b) failures

Figure 3: Test family 3 ( $C_0$  function).



(a) work



(b) failures

Figure 4: Test family 4 (One peak).

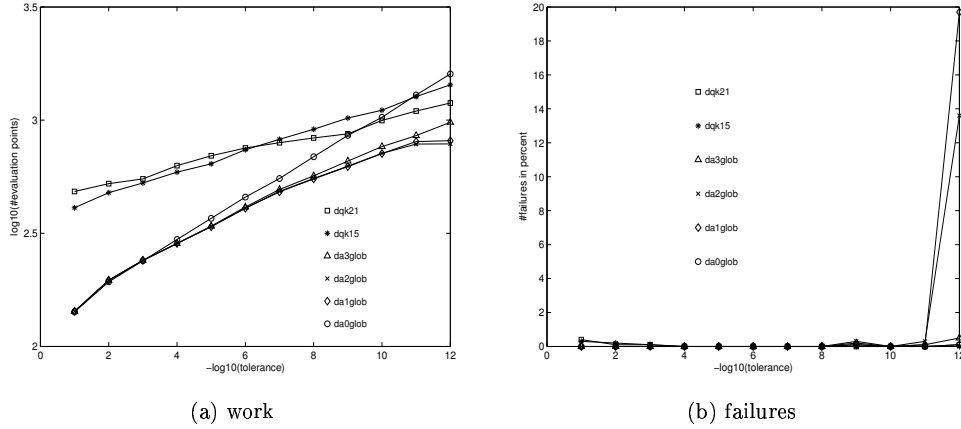


Figure 5: Test family 5 (Four peaks).

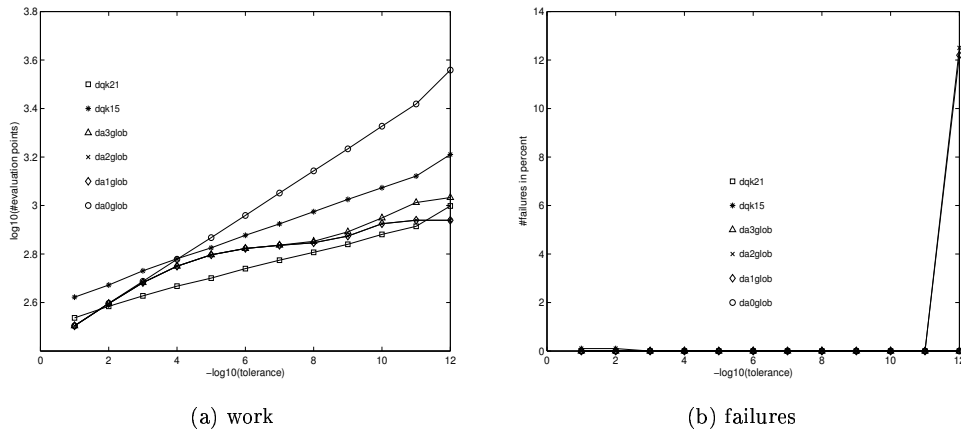


Figure 6: Test family 6 (Non-linear oscillatory).

We will give comments to these test results, given in the figures 1-6 and in the tables in the Appendix, for each family separately. We use `da1glob` as a reference code and relate both `dqk15` and `dqk21` to this code's performance.

Family 1 (Singularity, Figure 1): All codes have some failures for this family, `dqk21` has the highest reported number of failures with  $\approx 9\%$  when  $\tau_{ol}=10^{-3}$ . `dqk21` uses between 3.3 and 4.9 times the number of function evaluations as does `da1glob` while this ratio for `dqk15` is between 2.6 and 3.7. The main problem for the codes for this family is to detect and to shrink the interval containing the singularity. `da0glob` is marginally better than the other four codes indicating that the extended codes do not make much use of the higher degree rules and underlining the fact that few evaluation points in the rules are essential. Finally we observe that the average number of correct digits actually are slightly larger for the doubly adaptive codes in spite of the fact that these codes use much fewer function evaluations than the two QUADPACK codes.

Family 2 (Discontinuity, Figure 2): `dqk15` has between 0 and 2.7% failures while `dqk21` has between 0% and 7% failures for all tolerances compared to no failures for the four other codes. `dqk15` uses in addition from 3.1 to 6.3 times the number of function evaluations as does `da1glob` while this ratio for `dqk15` is between 4.3 and 8.8. The main problem for the codes for this family is to detect and to shrink the interval containing the discontinuity. No differences can be observed

between the four doubly adaptive codes which all behaves as `da1glob`. Neither `dqk15` nor `dqk21` give a warning that the tolerance may not be met and we observe that the average number of function evaluations (rounded to the nearest integer) is unchanged for the six last tolerances for both codes.

Family 3 ( $C_0$ -function, Figure 3): `dqk15` has between 0% and 6% failures while `dqk21` has between 0% and 2.9% failures compared to almost no failures for the four other codes. `dqk15` uses in addition from 2.9 to 5.1 times the number of function evaluations as does `da1glob` while this ratio for `dqk21` is between 3.8 and 6.8. The main problem for the codes for this family is to detect and to shrink the interval containing the  $C_0$ -part of the problem. Some differences can be observed between the four doubly adaptive codes with `da0glob` as the best, while the other doubly adaptive codes behave quite similar. The extended codes' attempts to use higher degree rules do not appear useful for this problem. However, the differences are small so the good behavior of `da0glob` is not ruined in the extensions.

Family 4 (One peak, Figure 4): `dqk15` has between 0% and 68.1% failures and `dqk21` between 0% and 53.4% failures compared to between 0 and 21.5% failures for the four other codes. The failures appear for all codes for low accuracies only:  $10^{-1}$  and  $10^{-2}$ . `dqk15` uses in addition from 1.6 to 4.1 times the number of function evaluations as does `da1glob` while this ratio for `dqk21` is between 1.4 and 5.2. Some differences can be observed between the four other codes with `da2glob` as the best: we clearly see that the extensions all benefit from the higher degree rules as the accuracy request increases compared to `da0glob`. The cases of failures for all codes have at least one digit wrong on the average. Finally we see that for low accuracy requests the doubly adaptive codes are far better than the QUADPACK codes while when we increase the accuracy request these two codes are more competitive.

An observation is that when integrating functions in Family 4 specifying a *relative* error request, then the six codes will have very few failures initially contrary to what we observe with an *absolute* error request with the same codes. The reason is simply that codes typically tend to underestimate the true integral initially for these problems and therefore, when using a relative error test, use a smaller number in the stopping criterion. This underestimation happens only initially in a globally adaptive code and the effect is a reliable code. Therefore, sometimes mere luck produces good results!

Family 5 (Four peaks, Figure 5): `dqk15` has between 0% and 0.3% failures while `dqk21` has between 0% and 0.4% failures. For these two codes these failures appear initially. `da0glob` and `da3glob` have a similar level of failures. `da1glob` and `da2glob` have very few failures for tolerances larger than  $10^{-12}$ . For  $\text{tol} = 10^{-12}$  both these two codes give warnings about too small tolerance value compared to the global noise level and therefore modify the stopping test. This results in failures in 19.7% of the cases for `da1glob` and 13.6% of the cases for `da2glob`. The reason for this is of course that both these codes have rules with 2-norms between 40 and 80 which implies that they will be effected by rounding noise at an earlier stage than the other codes in the test. The true values of these integrals are around 10 which implies a relative error  $\approx 10^{-13}$ . `dqk21` uses from 1.6 to 2.9 times the number of function evaluations as does `da1glob` while this ratio for `dqk15` is between 1.4 and 3.4. For accuracy requests between 1 to 11 decimals `da2glob` is the best code. Including all twelve accuracies `da3glob` is the best code.

Family 6 (Non-linear oscillatory, Figure 6 with  $\alpha_6 = 2$ ): all codes demonstrate very good reliability for most accuracies. For  $\text{tol} = 10^{-12}$  here too `da1glob` and `da2glob` give warnings about too small tolerance which in turn results in 12.2% and 12.5% of failures with the average number of wrong digits equal to 0.1 and 0.2. `dqk15` uses in addition from 1.1 to 1.9 times the number of function evaluations as does `da1glob` while this ratio for `dqk21` is between 0.8 and 1.1. `dqk21` is obviously the best overall code for this problem and we observe that since the nonlinear oscillations are spread over the whole interval the problem does not need adaptability, but will benefit from using rules of high polynomial degree.

Family 6 (Non-linear oscillatory, Figure 7 with  $\alpha_6 = 3$ ): let us increase the difficulty parameter,  $\alpha_6$ , from two to three and run the Lyness-Kaganove test (only 7 tolerances due to the work load) on this highly oscillatory problem class. We have tested three of the codes: `dqk15`, `dqk21` and `da1glob` on this test family. The other three doubly adaptive codes will behave quite similar to

`da1glob`. All doubly adaptive codes have the option to give a collection of intervals initially by giving a vector input: we have run `da1glob` on this problem both with the initial interval  $a=[0,1]$  and with the initial interval vector  $a=\text{linspace}(0,1,128)$  (in Figure 7 denoted `da1glob*`) and plot in Figure 7 the results of this test. We observe that `da1glob`, with the initial interval  $a=[0,1]$ , has a number of failures (severe) initially. The number of failures reduces as the tolerance increases, however all errors that remain seem to be severe (no correct decimals on the average) indicating that some subintervals are not resolved by this code. Replacing  $a=[0,1]$  by  $a=\text{linspace}(0,1,128)$  removes all these errors for this problem class. Similar failures have previously been reported by Espelid in [5, 6] for such doubly adaptive codes. Both `dqk15` and `dqk21` show good reliability on this problem, but we have observed that these two codes use a lot more function evaluations than the best doubly adaptive codes on the other problem families in this test. In the test done by Berntsen and Espelid [2] we observe that all the codes using different 21 point rules (Gauss, Lobatto, Clenshaw-Curtis, Gauss-Kronrod) perform at least as good as the `dqk21` rule on Family 6 with  $\alpha_6 = 3$ .

All these doubly adaptive codes do use a rule with as few as five nodes and this may be one reason for these failures. A different error source may be that rules using equidistant nodes might be more vulnerable to aliasing than rules using non-equidistant nodes. On the other hand, the fact that we use a basic rule with as few as five nodes is at the same time the reason for these codes excellent performance on problems where adaptability is important.

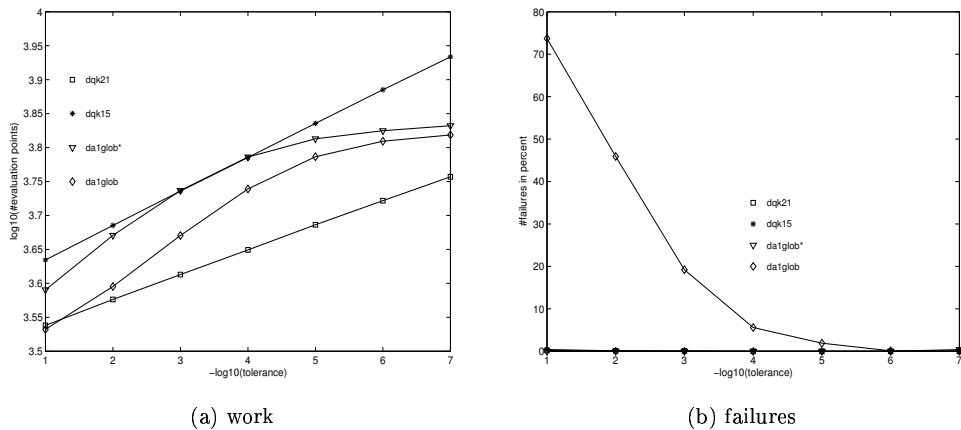


Figure 7: Test family 6 (Non-linear oscillatory with  $\alpha_6 = 3$ ).

## 2.2 Results from the battery test

We have also tested all six codes discussed in this paper on the 23 test problems, Table 2, used by Gander and Gautschi in their battery test. Gander and Gautschi [9, 10] picked 23 different test problems from two different sources: the 21 first comes from Kahaner [13] and the last two from [11]. We have tested the codes on twelve different *absolute* error tolerances  $\text{tol} = 10^{-1}, 10^{-2}, \dots, 10^{-12}$  and the results can be found in the Appendix. In order to summarize the results of this battery test we have constructed Table 3. Here we give, for each of the 23 problems, the following information:

1. An integer gives the number of cases out of the twelve tested accuracies where this code is both successful and uses the minimum number of function evaluations among all the successful codes in the test.
2. A number in parenthesis gives the number of cases out of the twelve tested accuracies where the accuracy request is not met for this code and problem.

Table 2: The battery test problems

---

1.	$\int_0^1 \exp(x) dx.$
2.	$\int_0^1 f(x) dx,$ where $f = 1$ if $x > 0.3$ else $f = 0.$
3.	$\int_0^1 \sqrt{x} dx.$
4.	$\int_{-1}^1 (\frac{23}{25} \cosh(x) - \cos(x)) dx.$
5.	$\int_{-1}^1 1/(x^4 + x^2 + 0.9) dx.$
6.	$\int_0^1 \sqrt{x^3} dx.$
7.	$\int_0^1 1/\sqrt{x} dx.$
8.	$\int_0^1 1/(1 + x^4) dx.$
9.	$\int_0^1 2/(2 + \sin(10\pi x)) dx.$
10.	$\int_0^1 1/(1 + x) dx.$
11.	$\int_0^1 1/(1 + \exp(x)) dx.$
12.	$\int_0^1 x/(\exp(x) - 1) dx.$
13.	$\int_{0.1}^1 \sin(100\pi x)/(\pi x) dx.$
14.	$\int_0^{10} \sqrt{50} \exp(-50\pi x^2) dx.$
15.	$\int_0^{10} 25 \exp(-25x) dx.$
16.	$\int_0^{10} 50/(\pi(2500x^2 + 1)) dx.$
17.	$\int_{0.01}^1 50(\sin(50\pi x)/(50\pi x))^2 dx.$
18.	$\int_0^\pi \cos(\cos(x) + 3 \sin(x) + 2 \cos(2x) + 3 \sin(2x) + 3 \cos(3x)) dx.$
19.	$\int_0^1 f(x) dx,$ if $x > 10^{-15}$ then $f = \log(x)$ else $f = 0.$
20.	$\int_{-1}^1 1/(1.005 + x^2) dx.$
21.	$\int_0^1 \sum_{i=1}^3 1/\cosh(20^i(x - 2i/10)) dx.$
22.	$\int_0^1 4\pi^2 x \sin(20\pi x) \cos(2\pi x) dx.$
23.	$\int_0^1 1/(1 + (230x - 30)^2) dx.$

---

Table 3: Summarizing the results for the battery test.

Problem	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
1	9	9	9	9	3	0
2	12	12	12	12	0	0
3	10	12	12	12	0	0
4	6	6	6	6	6	0
5	1	2	2	2	4	6
6	12	3	3	3	0	0
7	7	12	12	9	0	0
8	2	2	2	2	5	5
9	4	11	9	7	0	0
10	6	6	6	6	6	0
11	10	10	10	10	2	0
12	12	12	12	12	0	0
13	3	3	3	3	0	9
14	6	12	12	11	0	0
15	5	12	12	11	0	0
16	7	11	11	8	0	0
17	2(2)	3(2)	3(2)	3(2)	1	8
18	4	9	9	8	0	4
19	9	12	12	10	0	0
20	3	4	4	4	2	6
21	7(5)	3(9)	3(9)	3(9)	0(6)	0(9)
22	1	2	2	2	1	9
23	7	12	12	10	0	0
Sum	145(7)	180(11)	178(11)	163(11)	30(6)	47(9)

3. Finally we sum over all 23 problems both the number of cases where a code is the best and the number of failures out of the  $12 * 23 = 276$  cases totally.

According to Table 3 all these codes are very reliable with at most 11 failures out of the 276 cases and we observe that most of the failures occur in Problem 21.

`dqk15` has the fewest failures of all codes in the test but it is the best code in only 30 out of the 276 cases. `dqk21` is the best code in 47 of 276 cases. Furthermore we observe that the four matlab codes are all very competitive, each one being the best code in more than half the cases. `da1glob` is the best code in almost  $2/3$  of the cases.

### 3 Conclusions.

We conclude this testing by stating that which code to choose depends on the problem type at hand. The best overall code seems to be `da2glob` except for the weakness that when the accuracy request is high it may suffer from the rules' large 2-norms. Now, `da0glob` and all three modifications are monitoring the global noise level and will change the stopping criteria when `tol` is below the noise level. The point is that this happens at an earlier stage for `da1glob` and `da2glob` than for the other two doubly adaptive codes.

Furthermore the code `da3glob` is quite good and in the tests a similar weakness is not observed for this code. This code therefore seems well fitted for high accuracy requests too, but it is in general less efficient than `da2glob` for moderate accuracy requests. `da0glob` is still a very good code with a strength in detecting and isolating non-smooth problem areas. Asking for high accuracy this code is in many cases much less efficient than the extensions.

Both `dqk15` and `dqk21` confirms the impression of being reliable codes but both codes is in

general much less efficient than the three extended doubly adaptive codes tested in this paper. These two codes' strength, compared to the other codes in this test, is problem classes where adaptability is not important (e. g. oscillations over the whole interval as in Family 6) and the functions in the class is smooth implying that the rules' high polynomial degree does matter.

## References

- [1] J. Berntsen. A test of some well known quadrature routines. Reports in Informatics 20, Dept. of Informatics, Univ. of Bergen, 1986.
- [2] J. Berntsen and T. O. Espelid. Error Estimation in Automatic Quadrature Routines. *ACM Trans. Math. Softw.*, 17(2):233–252, 1991.
- [3] J. Berntsen, T. O. Espelid, and A. Genz. An Adaptive Algorithm for the Approximate Calculation of Multiple Integrals. *ACM Trans. Math. Softw.*, 17(4):437–451, 1991.
- [4] C. de Boor. On writing an automatic integration algorithm. In J. R. Rice, editor, *Mathematical Software*, New York, 1971. Academic Press.
- [5] T. O. Espelid. Doubly Adaptive Quadrature Routines based on Newton-Cotes Rules. Reports in Informatics 229, Dept. of Informatics, Univ. of Bergen, 2002.
- [6] T. O. Espelid. Doubly Adaptive Quadrature Routines based on Newton-Cotes Rules. *BIT*, 43:319–337, 2003.
- [7] T. O. Espelid. Extended doubly Adaptive Quadrature Routines. Reports in Informatics 266, Dept. of Informatics, Univ. of Bergen, 2004.
- [8] W. Fullerton. Rand: a uniform random number generator. Core Math. LIBrary, National Institute of Standards and Technology, 1977.
- [9] G. Gander and W. Gautschi. Adaptive Quadrature - Revisited. Report 306, Dept. Informatik., ETH, Zurich, 1998.
- [10] G. Gander and W. Gautschi. Adaptive Quadrature - Revisited. *BIT*, 40(1):84–101, 2000.
- [11] S. Garriba, L. Quartapelle, and G. Reina. Algorithm 36 - SNIFF: Efficient self-tuning algorithm for numerical integration. *Computing*, 20:363–375, 1978.
- [12] A.C. Genz and A.A. Malik. An adaptive algorithm for numerical integration over an N-dimensional rectangular region. *J. Comp. Appl. Math.*, 6(4):295–302, 1980.
- [13] D.K. Kahaner. Comparison of numerical quadrature formulas. In J. Rice, editor, *Mathematical Software*, pages 229–259, New York, 1971. Academic Press.
- [14] J.N. Lyness and J.J. Kaganove. A technique for comparing automatic quadrature routines. *Computer Journal*, 20:170–177, 1977.
- [15] W. M. McKeeman. Algorithm 145, adaptive numerical integration by Simpson's rule. *CACM*, 5(12):604, 1962.
- [16] R. Piessens, E. de Doncker-Kapenga, C.W. Überhuber, and D.K. Kahaner. *QUADPACK, A Subroutine Package for Automatic Integration*. Series in Computational Math., 1. Springer-Verlag, Berlin, 1983.
- [17] T. Sjørevik. Reliable and Efficient Algorithms for Adaptive Quadrature. Technical report, Thesis for the degree Doctor Scientiarum, Department of Informatics, University of Bergen, 1988.
- [18] P. van Dooren and L. de Ridder. An adaptive algorithm for numerical integration over an N-dimensional cube. *J. Comp. Appl. Math.*, 2(3):207–217, 1976.

## Appendix: the numerical experiments.

We report tests on the four Matlab routines: `da0glob`, `da1glob`, `da2glob`, `da3glob` and two general purpose codes from QUADPACK (FORTRAN 77) `dqk15` and `dqk21` from the code `dqag`. These two codes are based on 15 point and 21 point Gauss-Kronrod rules respectively and are the two QUADPACK rules using fewest points in the basic rule and are thus the most adaptable candidates from this code. (The other possible rules are also Gauss-Kronrod with 31, 41, 51 and 61 points respectively).

### The Lyness-Kaganove test.

The testing technique we have used in this part is due to Lyness and Kaganove [14]. This technique is based on a selection of test families of integrands. Each test family has a special attribute, a difficulty parameter and one or more random parameters. The test families used in these experiments are given in Table 1, and they are picked from [1], [14] and [17]. These test families have furthermore been used by Berntsen and Espelid in [2].

In our experiments we have chosen the difficulty parameters  $\alpha_i, i = 1, \dots, 6$ , to be (numbered from family 1 to 6):  $\underline{\alpha} = (-0.5, 0.5, 2.0, -4.0, -2.0, 2.0)$ . The random parameters,  $\lambda$  (or  $\lambda_i, i = 1, \dots, 4$ , for test family 5), are picked randomly from the region of integration using the FORTRAN routine `rand`, [8] (in these tests the routine has been modified to give a double precision floating point number in the interval  $[0, 1]$ ). All six codes are designed to accept a maximum number of function values and this value is set in all experiments in this paper equal to 10 000. We have tested the codes for absolute error tolerances  $\text{tol} = 10^{-1}, 10^{-2}, \dots, 10^{-12}$ . (For test family 1 we stop at  $10^{-5}$ .) For these values of  $\text{tol}$  and for each test family we have asked all routines to compute the integrals for 1000 samples of random parameters, and in all but a few cases the routines report that the returned values satisfies the error request. The experiments have been run on a Dell Optiplex GX260 computer.

In the tables below we report on the performance of the different routines for each test family and for each error request. For each routine the four numbers from left to right are:

1. The average number of integrand evaluations. Only the cases where the error request is satisfied are included in the average.
2. The average number of correct digits in the returned value. Here too, only the cases where the error request is satisfied are included in the average.
3. Failures in percent: that is cases where the actual error is greater than the error tolerance.
4. The average number of wrong digits in the cases described above.

The number of wrong digits is computed by

$$\text{Wrong digits} = | -\log_{10}(\text{tol}) + \log_{10}(\epsilon_{act}) |,$$

where  $\epsilon_{act}$  is the actual absolute accuracy in the returned value.

tol	da0glob				da1glob				da2glob			
$10^{-1}$	73	2.5	6.2	0.3	73	2.5	6.2	0.3	73	2.5	6.2	0.3
$10^{-2}$	169	3.8	3.7	0.4	172	3.8	3.8	0.4	172	3.8	3.8	0.4
$10^{-3}$	277	5.1	0.3	0.5	287	5.3	0.4	0.4	287	5.3	0.4	0.4
$10^{-4}$	372	5.8			384	6.2			384	6.2		
$10^{-5}$	483	6.6			483	6.9			485	6.9		
tol	da3glob				dqk15				dqk21			
$10^{-1}$	73	2.5	6.2	0.3	267	1.8	1.7	0.3	359	1.9	1.9	0.3
$10^{-2}$	172	3.8	3.8	0.4	501	2.9	2.7	0.4	655	2.9	3.4	0.5
$10^{-3}$	286	5.2	0.5	0.3	752	3.6	4.4	0.4	951	3.7	9.4	0.2
$10^{-4}$	383	6.0			1262	5.5	0.2	3.2	1651	5.6	0.2	2.4
$10^{-5}$	485	6.8			1711	6.5			2236	6.6		

Test Family 1: Singularity.

tol	da0glob		da1glob		da2glob					
$10^{-1}$	29	2.9	29	2.9	29	2.9				
$10^{-2}$	43	3.9	43	3.9	43	3.9				
$10^{-3}$	56	5.0	56	5.0	56	5.0				
$10^{-4}$	69	5.9	69	5.9	69	5.9				
$10^{-5}$	83	6.8	83	6.8	83	6.8				
$10^{-6}$	94	7.8	94	7.8	94	7.8				
$10^{-7}$	102	8.4	102	8.4	102	8.4				
$10^{-8}$	117	9.3	117	9.3	117	9.3				
$10^{-9}$	131	10.4	131	10.4	131	10.4				
$10^{-10}$	145	11.3	145	11.3	145	11.3				
$10^{-11}$	159	12.3	159	12.3	159	12.3				
$10^{-12}$	174	13.5	174	13.5	174	13.5				
tol	da3glob		dqk15				dqk21			
$10^{-1}$	29	2.9	89	2.6			125	2.7		
$10^{-2}$	43	3.9	181	3.5			255	3.7		
$10^{-3}$	56	5.0	279	4.5	1.0	0.4	394	4.7	0.2	0.3
$10^{-4}$	69	5.9	380	5.5	3.0	0.8	534	5.7	0.9	0.8
$10^{-5}$	83	6.8	479	6.7	4.4	1.4	672	6.9	1.8	1.1
$10^{-6}$	94	7.8	576	9.6	5.6	1.9	813	10.2	2.1	1.8
$10^{-7}$	102	8.4	645	15.4	7.0	2.5	903	15.4	2.8	2.3
$10^{-8}$	117	9.3	645	15.4	7.0	3.5	903	15.4	2.8	3.3
$10^{-9}$	131	10.4	645	15.4	7.0	4.5	903	15.4	2.8	4.3
$10^{-10}$	145	11.3	645	15.4	7.0	5.5	903	15.4	2.8	5.3
$10^{-11}$	159	12.3	645	15.4	7.0	6.5	903	15.4	2.8	6.3
$10^{-12}$	174	13.5	645	15.4	7.0	7.5	903	15.4	2.8	7.3

Test Family 2: Discontinuous.

tol	da0glob			da1glob			da2glob					
10 <sup>-1</sup>	14	3.2		14	3.2		14	3.2				
10 <sup>-2</sup>	19	4.0		19	4.0		19	4.0				
10 <sup>-3</sup>	26	5.0		27	5.0		27	5.0				
10 <sup>-4</sup>	36	6.1		38	6.1		38	6.1				
10 <sup>-5</sup>	43	7.0		46	7.0		46	7.0				
10 <sup>-6</sup>	53	8.1		58	8.1		58	8.1				
10 <sup>-7</sup>	61	9.0		68	9.1		68	9.1				
10 <sup>-8</sup>	69	10.3		78	10.3		78	10.3				
10 <sup>-9</sup>	78	11.0		90	11.0		90	11.0				
10 <sup>-10</sup>	91	12.1		106	12.1		106	12.1				
10 <sup>-11</sup>	103	13.1		120	13.1		120	13.1				
10 <sup>-12</sup>	110	14.2		131	14.2		131	14.2				
tol	da3glob			dqk15			dqk21					
10 <sup>-1</sup>	14	3.2		41	3.7		53	4.0				
10 <sup>-2</sup>	19	4.0		92	4.8		117	4.9				
10 <sup>-3</sup>	27	5.0		139	5.7	0.2	0.2	176	5.7			
10 <sup>-4</sup>	38	6.1		185	6.7	0.6	0.3	243	6.7	0.6	0.2	
10 <sup>-5</sup>	46	7.0		235	7.7	1.2	0.5	314	7.8	1.3	0.4	
10 <sup>-6</sup>	58	8.1		287	8.7	1.2	0.9	385	8.8	1.2	0.8	
10 <sup>-7</sup>	67	9.0		333	9.6	2.7	1.1	449	9.7	1.8	0.7	
10 <sup>-8</sup>	78	10.2		385	10.7	3.5	1.6	524	10.8	2.3	0.9	
10 <sup>-9</sup>	89	11.0		436	11.7	4.3	2.1	593	11.7	2.2	1.4	
10 <sup>-10</sup>	105	12.1		482	12.6	4.8	2.6	659	12.7	2.9	1.5	
10 <sup>-11</sup>	120	13.1	0.2	0.5	532	13.8	5.1	3.4	729	13.7	2.7	2.2
10 <sup>-12</sup>	130	14.2		588	14.5	6.0	3.8	779	14.3	2.6	3.2	

Test Family 3:  $C_0$  function.

tol	da0glob				da1glob				da2glob			
10 <sup>-1</sup>	96	3.5	21.5	1.5	96	3.5	21.5	1.5	96	3.5	21.5	1.5
10 <sup>-2</sup>	123	4.6	3.3	2.5	125	4.6	3.3	2.5	125	4.6	3.3	2.5
10 <sup>-3</sup>	163	5.5			166	5.5			166	5.5		
10 <sup>-4</sup>	207	6.5			207	6.4			208	6.4		
10 <sup>-5</sup>	258	7.1			245	7.2			245	7.2		
10 <sup>-6</sup>	319	8.3			289	8.2			290	8.2		
10 <sup>-7</sup>	391	9.3			342	9.2			344	9.2		
10 <sup>-8</sup>	473	10.0			403	10.0			404	10.0		
10 <sup>-9</sup>	585	10.8			473	10.9			473	10.9		
10 <sup>-10</sup>	718	12.1			537	12.1			536	12.1		
10 <sup>-11</sup>	876	12.8			610	13.1			605	13.1	0.4	0.0
10 <sup>-12</sup>	1103	13.4			672	13.9			640	13.4	0.5	1.0
tol	da3glob				dqk15				dqk21			
10 <sup>-1</sup>	96	3.5	21.5	1.5	392	6.8	68.1	1.5	497	7.4	53.4	1.5
10 <sup>-2</sup>	125	4.6	3.3	2.5	429	8.6	0.9	2.2	539	9.1	1.0	2.5
10 <sup>-3</sup>	166	5.4			469	9.9	0.2	3.5	568	10.4	0.3	3.5
10 <sup>-4</sup>	207	6.4			510	10.9	0.1	4.5	591	11.4		
10 <sup>-5</sup>	246	7.2			553	11.7	0.1	5.5	613	12.1		
10 <sup>-6</sup>	292	8.1			605	12.7			660	12.8		
10 <sup>-7</sup>	349	9.1			660	13.0			701	12.9		
10 <sup>-8</sup>	415	9.9			732	13.1			748	12.9		
10 <sup>-9</sup>	489	10.8			809	13.3			792	13.0		
10 <sup>-10</sup>	564	11.9			893	13.3			837	13.1		
10 <sup>-11</sup>	648	12.8			1000	13.3			879	13.2		
10 <sup>-12</sup>	749	13.7			1102	13.3			941	13.4		

Test Family 4: One Peak.

tol	da0glob				da1glob				da2glob			
10 <sup>-1</sup>	142	3.6			143	3.5			143	3.5		
10 <sup>-2</sup>	193	4.6			196	4.5			196	4.5		
10 <sup>-3</sup>	239	5.3			240	5.3			240	5.3		
10 <sup>-4</sup>	297	6.3			285	6.3			285	6.3		
10 <sup>-5</sup>	368	7.2			338	7.2			339	7.2		
10 <sup>-6</sup>	457	7.9			408	8.1			408	8.1		
10 <sup>-7</sup>	552	8.9			482	8.9			485	8.9		
10 <sup>-8</sup>	689	10.5			550	10.2			552	10.2		
10 <sup>-9</sup>	856	10.8	0.1	0.4	623	10.8	0.2	0.3	626	11.0	0.3	0.3
10 <sup>-10</sup>	1029	11.5			712	11.6			713	12.0		
10 <sup>-11</sup>	1292	13.0			804	12.8	0.1	0.1	784	12.8	0.3	0.1
10 <sup>-12</sup>	1599	13.5	0.1	0.2	811	13.2	19.7	0.5	785	13.0	13.6	0.4
tol	da3glob				dqk15				dqk21			
10 <sup>-1</sup>	143	3.5			410	6.9	0.3	0.9	484	7.0	0.4	0.7
10 <sup>-2</sup>	196	4.5			478	8.1	0.2	1.8	524	8.8	0.1	0.9
10 <sup>-3</sup>	240	5.3			528	9.0	0.1	3.9	550	9.8	0.1	0.8
10 <sup>-4</sup>	285	6.3			589	10.8			629	10.8		
10 <sup>-5</sup>	340	7.2			641	12.0			696	11.9		
10 <sup>-6</sup>	413	8.0			740	13.0			753	13.1		
10 <sup>-7</sup>	494	8.9			823	13.8			795	14.1		
10 <sup>-8</sup>	567	10.1			911	14.3			834	14.7		
10 <sup>-9</sup>	659	10.8	0.1	0.4	1022	14.6			870	14.8		
10 <sup>-10</sup>	764	11.6			1106	14.7			997	14.8		
10 <sup>-11</sup>	854	12.6	0.1	0.1	1271	14.8			1097	14.7		
10 <sup>-12</sup>	977	13.5	0.5	0.2	1432	14.8			1190	14.7	0.1	0.7

Test Family 5: Four Peaks.

tol	da0glob				da1glob				da2glob			
10 <sup>-1</sup>	319	4.7			319	4.7			319	4.7		
10 <sup>-2</sup>	395	5.5			394	5.5			394	5.5		
10 <sup>-3</sup>	487	6.6			481	6.5			481	6.5		
10 <sup>-4</sup>	600	7.7			562	7.6			562	7.6		
10 <sup>-5</sup>	737	8.3			626	7.9			626	7.9		
10 <sup>-6</sup>	910	9.3			665	8.6			665	8.6		
10 <sup>-7</sup>	1126	10.2			685	11.0			685	11.0		
10 <sup>-8</sup>	1390	11.0			702	12.1			702	12.1		
10 <sup>-9</sup>	1713	12.2			748	12.4			748	12.4		
10 <sup>-10</sup>	2124	13.0			841	12.5			841	12.5		
10 <sup>-11</sup>	2623	13.8			870	12.5			869	12.5		
10 <sup>-12</sup>	3619	14.2			869	12.5	12.2	0.2	869	12.5	12.5	0.1
tol	da3glob				dqk15				dqk21			
10 <sup>-1</sup>	319	4.7			419	11.0	0.1	2.2	344	12.6		
10 <sup>-2</sup>	394	5.5			470	12.2	0.1	3.2	384	13.1		
10 <sup>-3</sup>	481	6.5			538	13.2			424	13.5		
10 <sup>-4</sup>	562	7.6			603	13.7			465	13.8		
10 <sup>-5</sup>	626	7.9			669	13.8			502	13.8		
10 <sup>-6</sup>	665	8.6			754	13.9			549	13.9		
10 <sup>-7</sup>	686	10.0			840	13.9			595	13.9		
10 <sup>-8</sup>	712	11.0			943	14.0			641	13.9		
10 <sup>-9</sup>	779	12.1			1060	14.1			692	13.8		
10 <sup>-10</sup>	889	13.0			1184	14.2			760	13.7		
10 <sup>-11</sup>	1029	13.5			1322	14.2			821	13.7		
10 <sup>-12</sup>	1079	13.6			1624	14.2			996	13.8		

Test family 6: Nonlinear Oscillatory.

## The battery test.

Gander and Gautschi [9, 10] have picked a total of 23 different test problems from two different sources: the 21 first comes from Kahaner [13] and the last two are picked from [11]. In the following 23 tables we report on the result of testing the six codes on these 23 problems. We specify twelve different absolute error tolerances  $\text{tol} = 10^{-1}, 10^{-2}, \dots, 10^{-12}$  and report on the number of function evaluations used by each of the six codes. A minus sign in front of this number indicates that the requested error bound was NOT met by the actual code in this particular case.

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	9	9	9	9	15	21
$10^{-3}$	9	9	9	9	15	21
$10^{-4}$	9	9	9	9	15	21
$10^{-5}$	9	9	9	9	15	21
$10^{-6}$	9	9	9	9	15	21
$10^{-7}$	9	9	9	9	15	21
$10^{-8}$	9	9	9	9	15	21
$10^{-9}$	9	9	9	9	15	21
$10^{-10}$	17	17	17	17	15	21
$10^{-11}$	17	17	17	17	15	21
$10^{-12}$	17	17	17	17	15	21

Test problem 1:  $\int_0^1 \exp(x) dx$ .

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	29	29	29	29	75	105
$10^{-2}$	45	45	45	45	195	273
$10^{-3}$	61	61	61	61	285	399
$10^{-4}$	69	69	69	69	375	525
$10^{-5}$	85	85	85	85	495	693
$10^{-6}$	101	101	101	101	585	819
$10^{-7}$	109	109	109	109	675	945
$10^{-8}$	125	125	125	125	795	1113
$10^{-9}$	141	141	141	141	885	1239
$10^{-10}$	149	149	149	149	975	1365
$10^{-11}$	165	165	165	165	1095	1533
$10^{-12}$	181	181	181	181	1185	1659

Test problem 2:  $\int_0^1 f(x) dx$ , where  $f = 1$  if  $x > 0.3$  else  $f = 0$ .

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	13	13	13	13	45	21
$10^{-3}$	25	25	25	25	105	105
$10^{-4}$	33	33	33	33	195	189
$10^{-5}$	45	45	45	45	255	273
$10^{-6}$	65	65	65	65	315	399
$10^{-7}$	81	81	81	81	375	483
$10^{-8}$	97	97	97	97	465	567
$10^{-9}$	129	129	129	129	525	651
$10^{-10}$	169	169	169	169	585	777
$10^{-11}$	205	201	201	201	645	861
$10^{-12}$	257	237	237	237	705	945

Test problem 3:  $\int_0^1 \sqrt{x} dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	9	9	9	9	15	21
$10^{-3}$	9	9	9	9	15	21
$10^{-4}$	9	9	9	9	15	21
$10^{-5}$	9	9	9	9	15	21
$10^{-6}$	9	9	9	9	15	21
$10^{-7}$	17	17	17	17	15	21
$10^{-8}$	17	17	17	17	15	21
$10^{-9}$	17	17	17	17	15	21
$10^{-10}$	33	33	33	33	15	21
$10^{-11}$	33	33	33	33	15	21
$10^{-12}$	33	33	33	33	15	21

Test problem 4:  $\int_{-1}^1 (\frac{23}{25} \cosh(x) - \cos(x)) dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	17	17	17	17	15	21
$10^{-3}$	17	17	17	17	15	21
$10^{-4}$	33	33	33	33	45	21
$10^{-5}$	33	33	33	33	45	21
$10^{-6}$	33	33	33	33	45	21
$10^{-7}$	41	33	33	33	45	63
$10^{-8}$	57	57	57	57	45	63
$10^{-9}$	65	65	65	65	45	63
$10^{-10}$	81	65	65	65	105	63
$10^{-11}$	105	65	65	65	105	63
$10^{-12}$	129	65	65	113	105	63

Test problem 5:  $\int_{-1}^1 1/(x^4 + x^2 + 0.9) dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	9	9	9	9	15	21
$10^{-3}$	9	9	9	9	15	21
$10^{-4}$	13	17	17	17	15	21
$10^{-5}$	17	25	25	25	45	21
$10^{-6}$	25	33	33	33	105	63
$10^{-7}$	33	49	49	49	135	105
$10^{-8}$	45	57	57	57	165	189
$10^{-9}$	57	65	65	65	225	231
$10^{-10}$	69	81	81	89	255	273
$10^{-11}$	93	105	105	105	285	357
$10^{-12}$	113	121	121	121	345	399

Test problem 6:  $\int_0^1 \sqrt{x^3} dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	69	69	69	69	225	315
$10^{-2}$	93	93	93	93	435	609
$10^{-3}$	141	141	141	141	615	861
$10^{-4}$	193	193	193	193	825	1155
$10^{-5}$	245	245	245	245	1035	1449
$10^{-6}$	305	305	305	305	1215	1701
$10^{-7}$	405	397	397	397	1425	1995
$10^{-8}$	509	509	509	509	1605	2289
$10^{-9}$	629	613	613	613	1815	2541
$10^{-10}$	797	717	717	725	2025	2835
$10^{-11}$	993	837	837	901	2205	3087
$10^{-12}$	1233	997	997	1109	2445	3381

Test problem 7:  $\int_0^1 1/\sqrt{x} dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	9	9	9	9	15	21
$10^{-3}$	17	17	17	17	15	21
$10^{-4}$	17	17	17	17	15	21
$10^{-5}$	17	17	17	17	15	21
$10^{-6}$	25	25	25	25	15	21
$10^{-7}$	33	33	33	33	45	21
$10^{-8}$	33	33	33	33	45	21
$10^{-9}$	41	33	33	33	45	21
$10^{-10}$	49	33	33	33	45	21
$10^{-11}$	65	33	33	33	45	21
$10^{-12}$	81	49	49	49	45	63

Test problem 8:  $\int_0^1 1/(1+x^4) dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	73	73	73	73	195	189
$10^{-2}$	89	89	89	89	225	315
$10^{-3}$	129	129	129	129	285	315
$10^{-4}$	177	177	177	177	345	315
$10^{-5}$	209	193	193	193	405	357
$10^{-6}$	249	241	241	241	405	399
$10^{-7}$	329	289	289	297	465	483
$10^{-8}$	385	353	353	353	585	483
$10^{-9}$	465	393	401	401	645	567
$10^{-10}$	593	425	433	449	645	567
$10^{-11}$	737	465	481	545	765	651
$10^{-12}$	865	593	577	641	825	735

Test problem 9:  $\int_0^1 2/(2 + \sin(10\pi x)) dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	9	9	9	9	15	21
$10^{-3}$	9	9	9	9	15	21
$10^{-4}$	9	9	9	9	15	21
$10^{-5}$	9	9	9	9	15	21
$10^{-6}$	9	9	9	9	15	21
$10^{-7}$	17	17	17	17	15	21
$10^{-8}$	17	17	17	17	15	21
$10^{-9}$	25	17	17	17	15	21
$10^{-10}$	25	17	17	17	15	21
$10^{-11}$	33	17	17	33	15	21
$10^{-12}$	41	33	33	33	15	21

Test problem 10:  $\int_0^1 1/(1+x)dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	9	9	9	9	15	21
$10^{-3}$	9	9	9	9	15	21
$10^{-4}$	9	9	9	9	15	21
$10^{-5}$	9	9	9	9	15	21
$10^{-6}$	9	9	9	9	15	21
$10^{-7}$	9	9	9	9	15	21
$10^{-8}$	9	9	9	9	15	21
$10^{-9}$	9	9	9	9	15	21
$10^{-10}$	9	9	9	9	15	21
$10^{-11}$	17	17	17	17	15	21
$10^{-12}$	17	17	17	17	15	21

Test problem 11:  $\int_0^1 1/(1+\exp(x))dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	9	9	9	9	15	21
$10^{-3}$	9	9	9	9	15	21
$10^{-4}$	9	9	9	9	15	21
$10^{-5}$	9	9	9	9	15	21
$10^{-6}$	9	9	9	9	15	21
$10^{-7}$	9	9	9	9	15	21
$10^{-8}$	9	9	9	9	15	21
$10^{-9}$	9	9	9	9	15	21
$10^{-10}$	9	9	9	9	15	21
$10^{-11}$	9	9	9	9	15	21
$10^{-12}$	9	9	9	9	15	21

Test problem 12:  $\int_0^1 x/(\exp(x)-1)dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	505	505	505	505	675	525
$10^{-2}$	513	513	513	513	855	651
$10^{-3}$	641	641	641	641	945	651
$10^{-4}$	889	889	889	889	945	651
$10^{-5}$	977	977	977	977	945	651
$10^{-6}$	1065	1025	1025	1025	975	945
$10^{-7}$	1681	1305	1305	1305	1575	1197
$10^{-8}$	1905	1521	1521	1521	1785	1281
$10^{-9}$	2017	1617	1617	1617	1845	1323
$10^{-10}$	3113	1657	1657	1665	1875	1323
$10^{-11}$	3753	1841	1841	1953	1905	1323
$10^{-12}$	3977	2017	2017	2049	1905	1323

Test problem 13:  $\int_{0.1}^1 \sin(100\pi x)/(\pi x) dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	33	33	33	33	135	147
$10^{-2}$	37	37	37	37	165	189
$10^{-3}$	37	37	37	37	165	231
$10^{-4}$	45	45	45	45	195	231
$10^{-5}$	53	53	53	53	195	231
$10^{-6}$	73	57	57	57	195	231
$10^{-7}$	93	77	77	77	195	231
$10^{-8}$	97	97	97	97	225	231
$10^{-9}$	121	105	105	105	225	273
$10^{-10}$	149	117	117	117	255	273
$10^{-11}$	173	125	125	125	285	273
$10^{-12}$	193	137	137	149	315	273

Test problem 14:  $\int_0^{10} \sqrt{50} \exp(-50\pi x^2) dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	33	33	33	33	135	147
$10^{-2}$	37	37	37	37	135	147
$10^{-3}$	41	41	41	41	135	147
$10^{-4}$	49	49	49	49	165	189
$10^{-5}$	69	69	69	69	165	189
$10^{-6}$	77	69	69	69	165	189
$10^{-7}$	89	81	81	81	165	189
$10^{-8}$	113	89	89	89	195	189
$10^{-9}$	133	101	101	101	195	189
$10^{-10}$	165	109	109	109	225	231
$10^{-11}$	209	133	133	133	225	231
$10^{-12}$	249	145	145	161	255	231

Test problem 15:  $\int_0^{10} 25 \exp(-25x) dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	41	41	41	41	195	231
$10^{-2}$	41	41	41	41	225	273
$10^{-3}$	61	65	65	65	225	273
$10^{-4}$	77	77	77	77	255	315
$10^{-5}$	81	81	81	81	255	315
$10^{-6}$	105	105	105	105	255	315
$10^{-7}$	137	137	137	137	285	357
$10^{-8}$	169	161	161	161	285	357
$10^{-9}$	201	161	161	177	285	357
$10^{-10}$	249	177	177	177	285	357
$10^{-11}$	305	209	209	257	315	357
$10^{-12}$	377	273	273	289	345	399

Test problem 16:  $\int_0^{10} 50/(\pi(2500x^2 + 1))dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	93	93	93	93	135	147
$10^{-2}$	241	241	241	241	165	189
$10^{-3}$	261	261	261	261	495	315
$10^{-4}$	-297	-297	-297	-297	885	609
$10^{-5}$	-417	-417	-417	-417	945	651
$10^{-6}$	937	921	921	921	945	651
$10^{-7}$	1049	1009	1009	1009	1065	903
$10^{-8}$	1313	1137	1137	1137	1365	1197
$10^{-9}$	1777	1465	1465	1473	1725	1281
$10^{-10}$	2001	1649	1649	1657	1845	1323
$10^{-11}$	2369	1729	1729	1745	1875	1323
$10^{-12}$	3353	1825	1825	1857	1905	1323

Test problem 17:  $\int_{0.01}^1 50(\sin(50\pi x)/(50\pi x))^2 dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	41	41	41	41	105	63
$10^{-2}$	57	57	57	57	105	105
$10^{-3}$	73	73	73	73	105	105
$10^{-4}$	97	97	97	97	105	147
$10^{-5}$	129	121	121	121	135	147
$10^{-6}$	161	129	129	129	165	147
$10^{-7}$	177	153	153	153	195	147
$10^{-8}$	225	153	153	169	225	147
$10^{-9}$	273	185	185	185	225	189
$10^{-10}$	337	217	217	217	285	189
$10^{-11}$	401	241	241	265	315	273
$10^{-12}$	497	273	273	273	375	273

Test problem 18:  $\int_0^\pi \cos(\cos(x) + 3\sin(x) + 2\cos(2x) + 3\sin(2x) + 3\cos(3x))dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	45	45	45	45	105	147
$10^{-2}$	53	53	53	53	225	315
$10^{-3}$	65	65	65	65	315	441
$10^{-4}$	89	89	89	89	405	567
$10^{-5}$	117	117	117	117	525	735
$10^{-6}$	145	145	145	145	615	861
$10^{-7}$	173	173	173	173	705	987
$10^{-8}$	229	229	229	229	825	1155
$10^{-9}$	285	285	285	285	915	1281
$10^{-10}$	349	337	337	337	1005	1407
$10^{-11}$	441	389	389	397	1125	1575
$10^{-12}$	557	457	457	505	1215	1701

Test problem 19:  $\int_0^1 f(x)dx$ , if  $x > 10^{-15}$  then  $f = \log(x)$  else  $f = 0$ .

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	9	9	9	9	15	21
$10^{-2}$	9	9	9	9	15	21
$10^{-3}$	17	17	17	17	15	21
$10^{-4}$	17	17	17	17	45	21
$10^{-5}$	25	25	25	25	45	21
$10^{-6}$	33	33	33	33	45	21
$10^{-7}$	33	33	33	33	45	21
$10^{-8}$	49	33	33	33	45	63
$10^{-9}$	57	57	65	65	45	63
$10^{-10}$	65	65	65	65	105	63
$10^{-11}$	89	65	65	65	105	63
$10^{-12}$	121	65	65	65	105	63

Test problem 20:  $\int_{-1}^1 1/(1.005 + x^2)dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	37	37	37	37	75	105
$10^{-2}$	57	57	57	57	165	231
$10^{-3}$	73	73	73	73	225	273
$10^{-4}$	-85	-85	-85	-85	-255	-273
$10^{-5}$	-121	-121	-121	-121	-255	-315
$10^{-6}$	-157	-161	-161	-157	-255	-315
$10^{-7}$	-185	-185	-185	-185	-345	-357
$10^{-8}$	-233	-233	-233	-233	-375	-357
$10^{-9}$	437	-249	-249	-249	-435	-357
$10^{-10}$	565	-289	-289	-297	885	-441
$10^{-11}$	717	-329	-337	-353	945	-525
$10^{-12}$	857	-369	-369	-393	1095	-525

Test problem 21:  $\int_0^1 \sum_{i=1}^3 1/\cosh(20^i(x - 2i/10))dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	129	129	129	129	225	147
$10^{-2}$	169	169	169	169	225	147
$10^{-3}$	233	233	233	233	225	147
$10^{-4}$	257	257	257	257	225	147
$10^{-5}$	297	257	257	257	225	231
$10^{-6}$	425	297	297	297	375	315
$10^{-7}$	489	329	329	329	435	315
$10^{-8}$	513	345	345	345	465	315
$10^{-9}$	761	353	353	401	465	315
$10^{-10}$	897	481	481	497	465	315
$10^{-11}$	1001	513	513	513	465	315
$10^{-12}$	1417	513	513	513	765	315

Test problem 22:  $\int_0^1 4\pi^2 x \sin(20\pi x) \cos(2\pi x) dx$

tol	da0glob	da1glob	da2glob	da3glob	dqk15	dqk21
$10^{-1}$	25	25	25	25	45	63
$10^{-2}$	41	41	41	41	45	63
$10^{-3}$	53	53	53	53	225	231
$10^{-4}$	73	73	73	73	255	273
$10^{-5}$	93	93	93	93	255	273
$10^{-6}$	113	113	113	113	255	357
$10^{-7}$	137	137	137	137	315	399
$10^{-8}$	189	169	169	169	345	399
$10^{-9}$	217	201	201	201	345	399
$10^{-10}$	281	225	225	225	375	399
$10^{-11}$	337	249	249	257	435	441
$10^{-12}$	401	305	305	337	435	483

Test problem 23:  $\int_0^1 1/(1 + (230x - 30)^2) dx$