# ALGORITHM 691
# Improving QUADPACK Automatic Integration Routines

PAOLA FAVATI
Istituto di Elaborazione dell'Informazione, CNR

and

GRAZIA LOTTI and FRANCESCO ROMANI
Universita' di Pisa

Two automatic adaptive integrators from QUADPACK (namely, QAG and QAGS) are modified by substituting the Gauss–Kronrod rules used for local quadrature with recursive monotone stable (RMS) formulas. Extensive numerical tests, both for one-dimensional and two-dimensional integrals, show that the resulting programs are faster, perform less functional evaluations, and are more reliable.

Categories and Subject Descriptors: G.1.4 [**Numerical Analysis**]: Quadrature and Numerical Differentiation—*adaptive quadrature*; G.4 [**Mathematics of Computing**]: Mathematical Software; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN*

General Terms: Algorithms

Additional Key Words and Phrases: Integration, interpolatory quadrature, program testing

## 1. INTRODUCTION

QUADPACK [19] is a collection of FORTRAN programs designed for the numerical evaluation of one-dimensional integrals and suitable for the treatment of some multidimensional problems.

In a companion paper [7], the authors introduced a new class of quadrature rules, called recursive monotone stable (RMS) formulas; these formulas allow the composition without wasting previously computed functional values.

Two globally adaptive integrators of QUADPACK, QAG and QAGS, based on Gauss–Kronrod rules, have been improved by substituting Gauss-

Authors' addresses: P. Favati, Istituto di Elaborazione dell'Informazione, CNR, Pisa, Italy; G. Lotti and F. Romani, Dipartimento di Informatica, Corso Italia 40, 56100 Pisa, Italy.

Kronrod rules with RMS ones. Extensive numerical tests show that the improved routines exhibit better performance and more reliability.

This package contains the improved programs in both single- (QXG/QXGS) and double-precision (DQXG/DQXGS) versions together with service routines and simple test drivers.

The source code of QUADPACK, used to derive the modified code, comes from the SLATEC library and has been received from the editor. The resulting code is self-contained and allows full compatibility with the other QUADPACK routines in the SLATEC library.

## 2. THE IMPROVED QUADRATURE PROGRAMS

QAG is a simple globally adaptive integrator that uses, as local quadrature module (LQM), a pair of Gauss–Kronrod integration formulas with $k$ and $2k + 1$ points. The user can choose between $k = 7$, 10, 15, 20, 25, or 30 by setting the variable KEY. The adaptive strategy attempts to reduce the error by means of the $\epsilon$-algorithm [21, 22], and its purpose is the elimination of the effects of integrand singularities of several types [17].

QAGS is an integrator based on globally adaptive interval subdivision in connection with extrapolation. The LQM uses a pair of Gauss–Kronrod integration formulas with 10 and 21 points. The extrapolation is carried out by means of the $\epsilon$-algorithm [21, 22], and its purpose is the elimination of the effects of integrand singularities of several types [17].

Both routines use an ordered list to retain information about the intervals (lower and upper limits, approximate integral, error estimate). The LQM used by QAGS and QAG (with $k = 10$) is named QK21 and is fully described and listed in [19].

In this section we briefly describe the modified LQM (called QXLQM) used by the improved automatic quadrature routines. QXLQM uses four RMS formulas, with 13, 19, 27, and 41 nodes, denoted by $Q_{13}$, $Q_{19}$, $Q_{27}$, $Q_{41}$, respectively. These formulas show the best performance among the families of RMS formulas; the location of nonnegative abscissas is given in Table I.

Let

$$I = \int_a^b f(x)\,dx, \qquad J = \int_a^b |f(x)|\,dx, \qquad M = \int_a^b |f(x) - I/(b-a)|\,dx.$$

The input parameters of QXLQM are the integrand function, the integration interval, and a control variable KEY which tells which formulas have to be used. In the following we assume KEY = 2 (the value used by QXGS). The meaning of the other values of KEY (used by QXG) are discussed later. The routine QXLQM computes the following:

| | |
|---|---|
| RESULT | Real approximation to the integral $I$. |
| RESABS | Real approximation to the integral $J$. |
| RESASC | Real approximation to the integral $M$. |
| ABSERR | Real estimate of $\lvert I - \text{RESULT}\rvert$. |

Table I. Values of Nonnegative Abscissas of the Rules $Q_{13}$, $Q_{19}$, $Q_{27}$, $Q_{41}$

| $Q_{13}$ | $Q_{19}$ | $Q_{27}$ | $Q_{41}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1/4 | 1/4 | 1/8 | 1/8 |
| 2/4 | 3/8 | 2/8 | 3/16 |
| 3/4 | 4/8 | 3/8 | 4/16 |
| 7/8 | 5/8 | 4/8 | 5/16 |
| 15/16 | 6/8 | 5/8 | 6/16 |
| 1 | 7/8 | 11/16 | 7/16 |
| | 15/16 | 12/16 | 8/16 |
| | 31/32 | 13/16 | 9/16 |
| | 1 | 14/16 | 10/16 |
| | | 15/16 | 11/16 |
| | | 31/32 | 12/16 |
| | | 63/64 | 13/16 |
| | | 1 | 27/32 |
| | | | 28/32 |
| | | | 29/32 |
| | | | 30/32 |
| | | | 31/32 |
| | | | 63/64 |
| | | | 127/128 |
| | | | 1 |

Let EPMACH be the machine precision and UFLOW be the smallest positive real in the arithmetic used. QXLQM uses the pair $Q_{13}$, $Q_{19}$ if the estimated relative error is less than $1000 * \text{EPMACH}$; otherwise it tries the pair $Q_{19}$, $Q_{27}$. The computation proceeds with the pair $Q_{27}$, $Q_{41}$ only if the estimated relative error is greater or equal to $1000 * \text{EPMACH}$ and the empirical test ABSERR/ABSOLD $< 0.16$ ensures the necessary smoothness of the integrand function. The constant 0.16 has been determined experimentally. A lower bound $\alpha * \text{EPMACH} * \text{RESABS}$ to ABSERR takes into account the influence of roundoff errors. In order to determine a suitable value of $\alpha$, the extended precision value QRES of RESULT has been computed for many functions and tolerances. The maximum value of the ratio $|\text{RESULT} - \text{QRES}|/\text{EPMACH} * \text{RESABS}$ results in a value less than 6. We decided to use $\alpha = 10$. Apart from this change, the same, very conservative, QUADPACK error estimation procedure was retained.

The computation (with KEY = 2) is structured as follows:

```
**Compute** RESOLD by using $Q_{13}$,
**Compute** RESULT, RESABS, RESASC by using $Q_{19}$,
**If** RESASC = 0 **then** ABSERR := |RESULT − RESOLD|
  **else** ABSERR := RESASC * min[1, (200 * | RESULT − RESOLD | / RESASC)$^{3/2}$];
**If** RESABS > UFLOW / (10 * EPMACH) **then** ABSERR := max[10 * EPMACH * RESABS, ABSERR],
**If** ABSERR < 1000 * EPMACH * RESABS **then exit**;
ABSOLD := ABSERR; RESOLD := RESULT;
**Compute** RESULT by using $Q_{27}$;
**If** RESASC = 0 **then** ABSERR := |RESULT − RESOLD|
  **else** ABSERR := RESASC * min[1, (200 * |RESULT − RESOLD| / RESASC)$^{3/2}$];
**If** RESABS > UFLOW / (10 * EPMACH) **then** ABSERR := max[10 * EPMACH * RESABS, ABSERR];
**If** (ABSERR < 1000 * EPMACH * RESABS) **or** (ABSERR > 0.16 * ABSOLD) **then exit**;
RESOLD := RESULT;
**Compute** RESULT by using $Q_{41}$;
**If** RESASC = 0 **then** ABSERR := |RESULT − RESOLD|
  **else** ABSERR := RESASC * min[1, (200 * |RESULT − RESOLD| / RESASC)$^{3/2}$],
**If** RESABS > UFLOW / (10 * EPMACH) **then** ABSERR = max [10 * EPMACH * RESABS, ABSERR].
```

The routine QXLQM saves the computed functional values together with the other information on the interval by using the same ordered list technique of QAG, QAGS. At any bisection the available functional values are reordered and the missing values are computed. No functional evaluations are performed twice for the same argument.

The variable KEY, which is used by QXG, allows the user to control which formulas can be used. The action of KEY is the following:

| | |
|---|---|
| KEY $\leq$ 0. | Only the pair $Q_{13}$, $Q_{19}$ is used. |
| KEY = 1. | The pair $Q_{13}$, $Q_{19}$ is used for the first error estimate, then if ABSERR $\geq$ 1000 * EPMACH * RESABS, $Q_{27}$ is used. |
| KEY = 2. | All the four formulas can be used, according to the algorithm (see above). |
| KEY = 3. | The pair $Q_{19}$, $Q_{27}$ is used for the first error estimate, then if ABSERR $\geq$ 1000 * EPMACH * RESABS, $Q_{41}$ is used. |
| KEY $\geq$ 4. | Only the pair $Q_{27}$, $Q_{41}$ is used. |

The routine QXGS differs from QAGS in the following ways:

(1) QXLQM (with KEY = 2) is called instead of QK21.
(2) Handling of the computed functional values is performed.
(3) The accuracy requirement over the set of big intervals is made stricter in order to correctly perform extrapolation.

The routine QXG differs from QAG in the following ways:

(1) QXLQM (with KEY set by the user) is called instead of QK15, QK21, . . . , QK61.

(2) Handling of the computed functional values is performed.

## 3. ONE-DIMENSIONAL INTEGRATION TESTS

The problem of testing general purpose quadrature routines is widely discussed in the literature. Two different approaches are discussed:

(1) battery experiments [1, 5, 6, 9, 11, 18, 20], where several test functions exhibiting different behaviors are used to measure the performance of quadrature algorithm under consideration;

(2) parameter studies [3, 14, 15, 16], where a family of integrand functions depending on some parameters is used to perform the test.

Both approaches have been used in our tests.

The battery test uses sets of functions obtained by the union of the ones proposed in the literature. Moreover some of our functions are obtained from a single family by varying the parameters. The subroutine defining the test functions assigns the value zero at points where a function assumes an infinite value and it assigns the true limiting value where a function has an indeterminate form.

Following Robinson [20] the integrands have been grouped, on the grounds of their analytical properties, into five classes.

*Class* A (*smooth*). Class A contains 43 well-behaved functions, that is, continuous and not greatly oscillatory functions with continuous first derivative. Examples include $\exp(x)$ in [0, 1] and $x/(\exp(x) - 1)$ in [0, 1].

*Class* B (*singularities*). Class B contains 84 functions having one or more integrable singularities in the function or in the first derivative within the integration interval. Examples include $|x - b|^a$ in [0, 1], where $a = 0.1$ (0.1) 0.5, $b = 1/2, 1/3, \pi/4$, and $\log \sin \pi x$ in [0, 1].

*Class* C (*peaks*). Class C contains 55 functions with one or more sharp peaks in the integration interval or with a singularity near the integration interval. Examples include $2^r/(1 + 2^r(x - b)^2)$ in [0, 1], where $r = 2$ (1) 9, $b = 1/2, 1/3$, and $\log(x + 0.001)$ in [0, 1].

*Class* D (*oscillating*). Class D contains 49 rapidly oscillating functions. Examples include $x \sin 30x \cos x$ in $[0, 2\pi]$ and $x^k \sin m\pi x$ in [0, 1], where $k = 0(1)3$, $m = 5, 10, 15, 20, 25, 100$.

*Class* E (*step*). Class E contains 29 functions with finite discontinuities in the function or in its first derivative within the integration interval. Examples include $[mx]x^{k-1}$ in [0, 1], where $k = 0, 1, 2$, $m = 5$ (5) 25, and $\{1/x\}$ in [0, 1].

Let $I$ denote the exact integral value, $I_{\text{comp}}$ the computed integral value, $E_{\text{abs}} = |I_{\text{comp}} - I|$ the absolute value of the absolute error, and $E_{\text{rel}} = E_{\text{abs}}/|I|$ the absolute value of the relative error. Most integration routines allow the user to state both absolute and relative error bounds. In Robinson [20], to make the results comparable, the different programs were tested with relative tolerances $10^{-t}$, $t = 2(2)10$. This choice is the simpler one, but does not allow zero values for $I$.

We prefer to use error tolerances $T = 2(2)14$ for the following logarithmic error measure:

$$E = -\log_{10}(\text{EPMACH} + E_{\text{abs}}/(1 + |I|)),$$

where EPMACH guarantees an upper bound to $E$. Note that $E$ is a reasonable measure since it is a continuous, monotone function of $E_{\text{abs}}$ and it is defined when $I = 0$, as well. Moreover, the use of a logarithmic measure allows the estimate of a reasonable mean value of the error for different test functions.

Once a tolerance $T$ for our error measure is fixed, the following tolerances for $E_{\text{abs}}$ and $E_{\text{rel}}$ result in

$$\text{Tol}_{\text{abs}} = (10^{-T} - \text{EPMACH})(1 + |I|)$$
$$\text{Tol}_{\text{rel}} = \text{Tol}_{\text{abs}}/|I|, \qquad \text{defined if } I \neq 0.$$

These tolerances are the error bounds given to the integration routines.

This procedure needs the a priori knowledge of the exact integral I and can be followed for test purposes only.

The computations have been carried out for the double-precision version of the programs by using system IBM/370 computers (IBM 3081 and IBM 3090) for which EPMACH = $16^{-13} \cong 2.22 \times 10^{-16}$. The maximum number of subintervals allowed in the subdivision process was limited to 1000. The parameters taken into account are the following:

(1) the percentage of successes;

(2) the mean number of functional evaluations;

(3) the total execution time.

Let $E_{\text{est}}$ and $E_{\text{act}}$ be the estimated and actual values of the measure $E$, respectively, obtained by the absolute error estimate of the program and by the true error.

The computation is considered *successful* if either $E_{\text{est}} > E_{\text{act}} > T$ (which is the "classical" notion of success) or $E_{\text{act}} > E_{\text{est}} > T$ (in this case the goal of integrating within the required accuracy has been reached but the error estimate is too optimistic). The computation is considered *failed* if $E_{\text{est}} > T$ and $E_{\text{act}} < T$; otherwise the program is assumed to *quit*. The output error flag of the programs are not taken into consideration.

Preliminary tests suggested using QAG with KEY = 2 and QXG with KEY = 1, since for these values the routines exhibit the best reliability.

Figure 1 shows the percentage of successes for each program and for each class. Results concerning CADRE have been included for comparison purposes. QUADPACK routines are clearly more robust than CADRE and improved routines are slightly better than the original ones.

Figures 2–6 show the mean functional evaluations for the five classes. In each case improved routines are better than original ones and extrapolated routines are better than the corresponding nonextrapolated ones. The only exception is for peaked functions where no advantages from extrapolation are apparent and, for the highest value of $T$, QAG is better than QAGS.

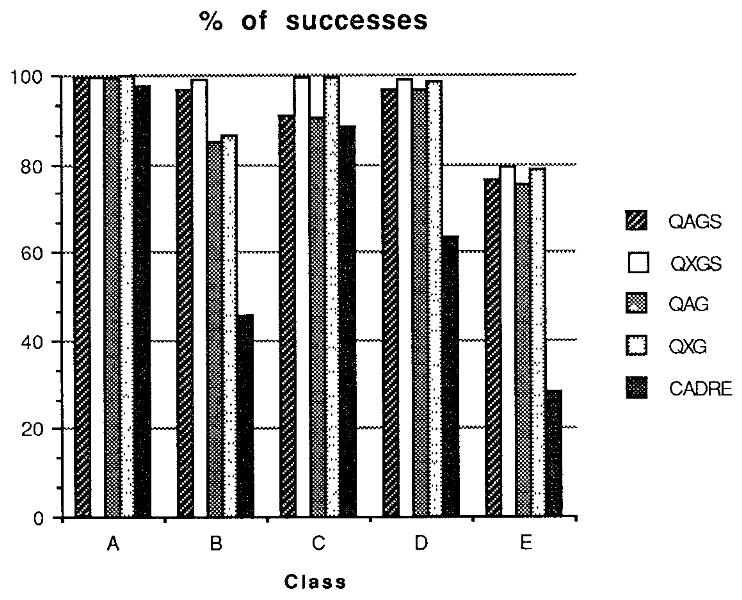

Fig. 1. Percentages of successes.

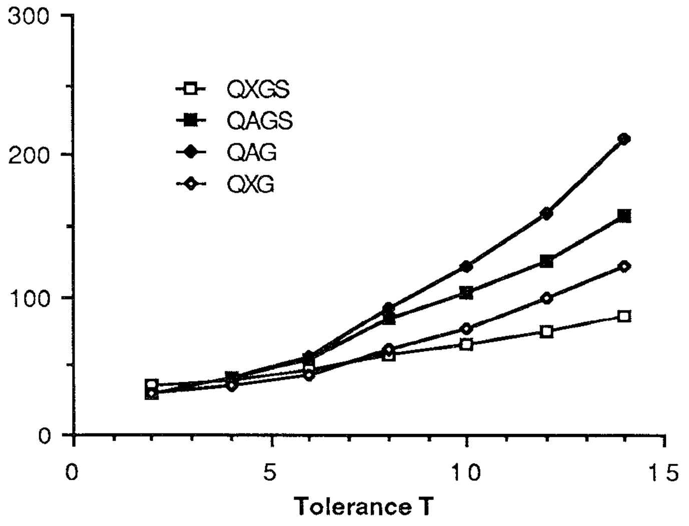

Fig. 2. Mean evaluations Class A (smooth).

Figure 7 shows the total execution time for QAGS, QXGS, QAG, and QXG, for the classes A–D. The improved routines require both an evaluation time and a total time shorter than the original ones.

For what concerns memory occupation, the maximum number of subintervals produced in the subdivision process has been computed for the various

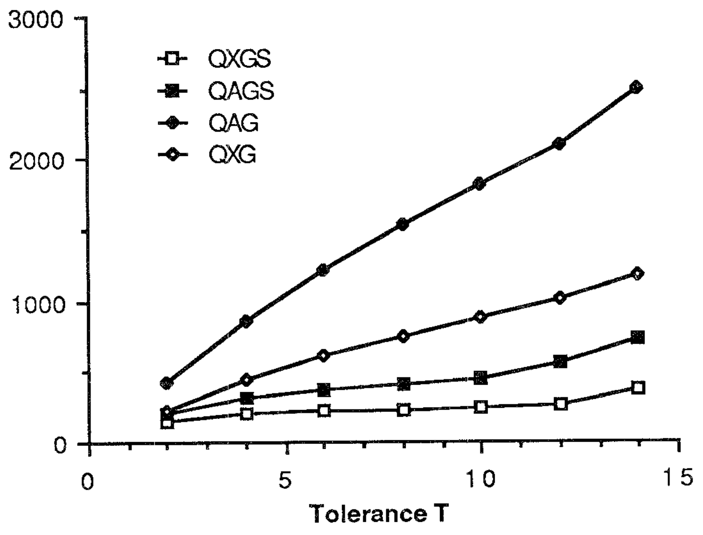

Fig. 3. Mean evaluations Class B (singularities).

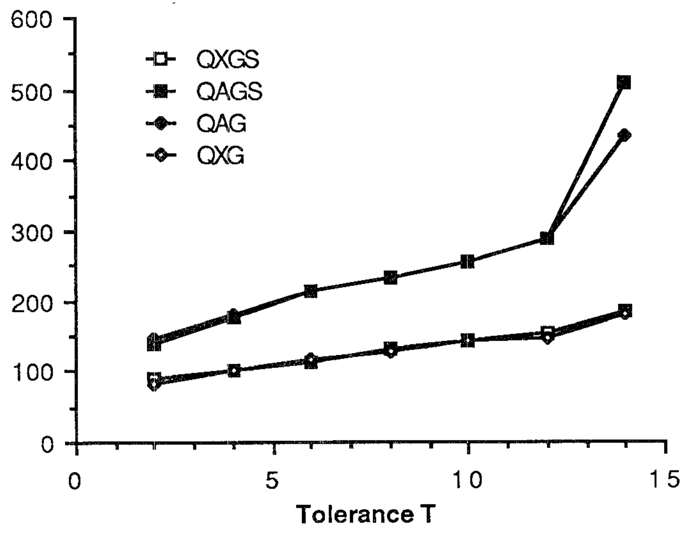

Fig. 4. Mean evaluations Class C (peaks).

classes of functions.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| QXGS | 14 | 70 | 36 | 52 | 1000 |
| QAGS | 15 | 48 | 26 | 103 | 1000 |
| QXG | 41 | 233 | 34 | 78 | 1000 |
| QAG | 26 | 229 | 26 | 104 | 1000 |

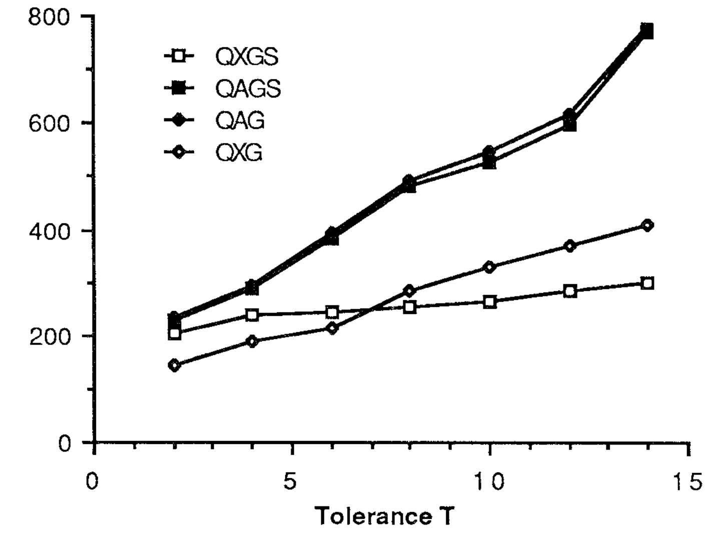

Fig. 5. Mean evaluations Class D (oscillating).

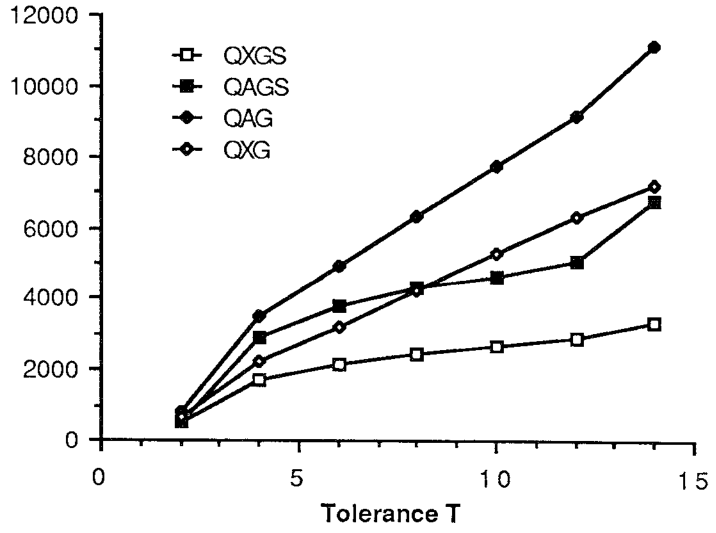

Fig. 6. Mean evaluations Class E (step).

These results show that, for improved routines, less than 100 subintervals suffice to integrate correctly functions in the classes A–D (the only exception being QXG applied to functions with singularities).

The parameter studies have been performed to test the sensitivity of the quadrature programs to the location of discontinuities.

The first problem taken into consideration is the computation of the integral of the step function, namely,

$$\int_0^1 f(x)\,dx,$$

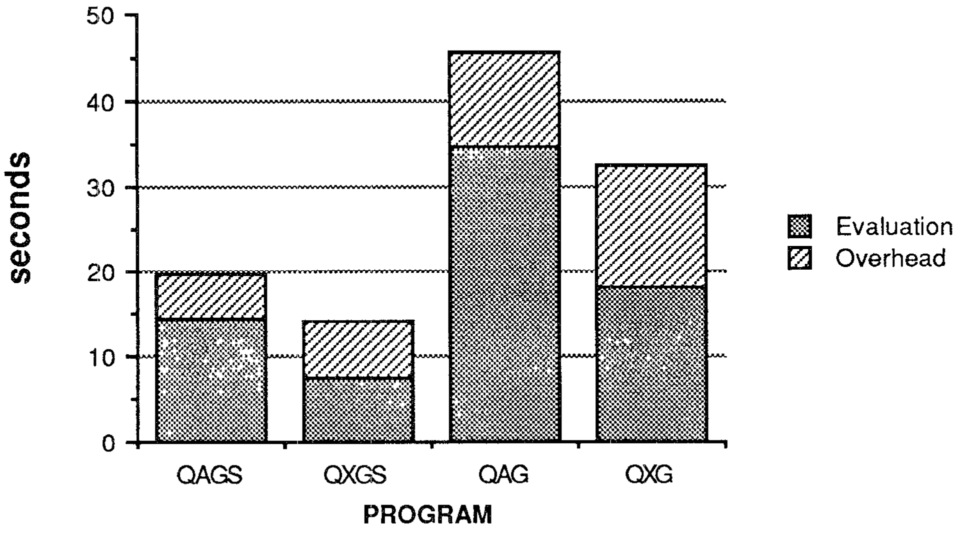

Fig. 7. Total execution time, overhead time, and evaluation time of classes A–D.

where

$$f(x) = \begin{cases} 1, & \text{if } x < y, \\ 0, & \text{otherwise,} \end{cases}$$

with $y$ varying in [0, 1].

It is easy to imagine that automatic routines using bisection, like the four under consideration, will have a special behavior when $y$ is near 1/2, 1/4, 3/4, . . . , etc. The more evident phenomena appear in the neighborhood of $y = 1/2$. The use of open basic rules presents serious difficulties when $y$ is close to 1/2 because these rules see a too simple integrand function, ignoring the singularity and failing the integration. This fact is shown in Figures 8 and 9. Apparently there are two sources of problems: the use of open rules for QAG and QAGS and the extrapolation for QAGS and QXGS. QXG, which uses closed rules without extrapolation, seems to be quite insensitive to the location of the discontinuity.

The second problem taken into consideration is the computation of the integral

$$\int_0^1 f(x)\,dx,$$

where

$$f(x) = |x - y|,$$

with $y$ varying in [0, 1]. In this case the discontinuity is on the first derivative. The same phenomena due to the use of open formulas are expected. The plots for this function are very similar to the previous ones and are not presented for the sake of brevity. The difficulties are present but less apparent. The routine QXG seems to be quite insensitive to the location of the discontinuity, as in the previous case.

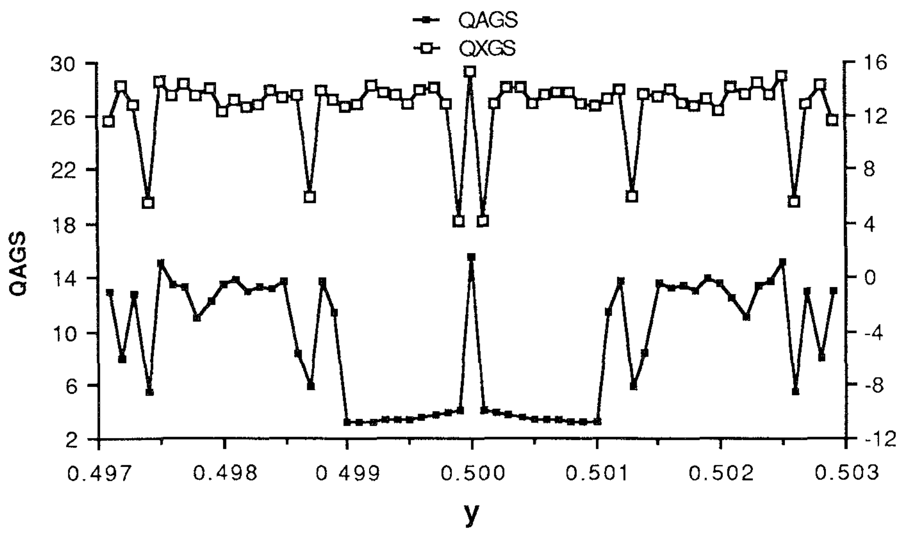

Fig. 8. Actual error $E_{act}$ for QAGS and QXGS near $y = 1/2$ ($T = 10$).

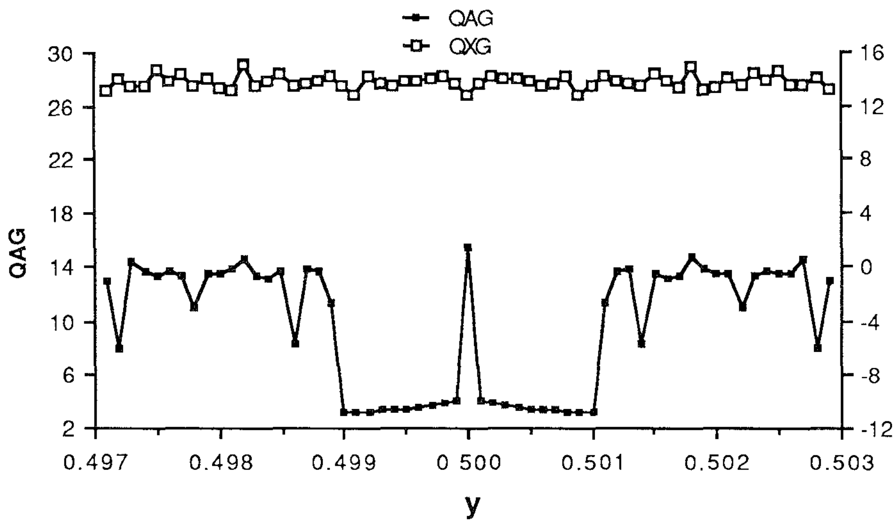

Fig. 9. Actual error $E_{act}$ for QAG and QXG near $y = 1/2$ ($T = 10$).

A third test, with a function presenting discontinuities in the second derivative (namely, $|x - y|^2$), shows that the four routines are insensitive to this type of discontinuity. The same result has been obtained with the highly oscillating parametric integrand ($\cos \alpha\pi x + 1$) used by DeBoor [3] for testing CADRE.

## 4. TWO-DIMENSIONAL INTEGRATION TESTS

As suggested in the reference manual [19, p. 113], QUADPACK routines, like all the one-dimensional integrators, can be used to handle double integrals of the form

$$I_1 = \int_a^b dx \int_{c(x)}^{d(x)} f(x, y)\, dy = \int_a^b I_2(x)\, dx.$$

The test computation of a double integral with a pair of one-dimensional integrators, say INT1, INT2, is structured as follows:

(1) Given the exact value $I_1$ and the tolerance $T$, compute the absolute tolerance $\mathrm{Tol}_{\mathrm{abs}} = (10^{-T} - \mathrm{EPMACH})(1 + |I_1|)$.
(2) Assign to INT1 the tolerance $\mathrm{Tol}_{\mathrm{abs1}} = 0.8\ \mathrm{Tol}_{\mathrm{abs}}$.
(3) Assign to INT2 the tolerance $\mathrm{Tol}_{\mathrm{abs2}} = \mathrm{Tol}_{\mathrm{abs}}/(20\,|b - a|)$.
(4) Compute $C_1$ with the integrator INT1 by using the result $C_2(x)$ of INT2 as the integrand function.

The test functions have been selected from those presented in literature for testing double integrals. A first group of 8 well-behaved integrals has been used for a battery test with tolerances $T = 4, 6, 9, 12$. Examples are

$$\int_0^1 \int_0^1 x^2 e^{xy}\, dx\, dy, \quad \int_0^{3\pi} \int_0^{3\pi} \cos(x + y)\, dx\, dy, \quad \int_{-1}^1 \int_0^{\pi/2} x^2 \sin(x + y)\, dx\, dy,$$

$$\int_0^1 \int_0^1 (x^2 + 0.0001)^{-1}((y + 1/4)^2 + 0.0001)^{-1}\, dx\, dy.$$

The test was completed with five more difficult integrals, tested with tolerances 3, 4, 5, 6, namely,

$$\int_0^1 \int_0^1 |x^2 - y|\, dx\, dy, \qquad \int_0^1 \int_0^1 e^{|x+y-1|}\, dx\, dy,$$

$$\int_0^1 \int_0^1 (1 - xy)^{-1}\, dx\, dy, \qquad \int_0^1 \int_0^1 f(x, y)\, dx,$$

where

$$f(x, y) = \begin{cases} 1, & \text{if } x^2 + y^2 < 1 \\ 0, & \text{otherwise} \end{cases}$$

and

$$\int_0^1 \int_0^1 1/[(1 - x)(1 - y)]^{-1}\, dx.$$

Preliminary tests, using nonextrapolated routines used as INT1, exhibit a very bad performance; for example, for the singular integrand $(1 - xy)^{-1}$, with tolerance $T = 5$ we get the following results

| INT1 | INT2 | $E_{\mathrm{act}}$ | Number of Evaluations |
|---|---|---|---|
| QAG | QAGS | 8.31 | 216279 |
| QAGS | QAGS | 13.66 | 21315 |

Table II. Results of Test with Battery Test of 13 Integrands

| INT1 | INT2 | Succeses | Fails | Quits | Mean Number of Evaluations | Time (s) |
|---|---|---|---|---|---|---|
| QAGS | QAGS | 50 | 2 | 0 | 76989 | 86 sec |
| QAGS | QAG | 50 | 2 | 0 | 66989 | 81 sec |
| QXGS | QXGS | 51 | 0 | 1 | 22544 | 36 sec |
| QXGS | QXG | 52 | 0 | 0 | 14593 | 28 sec |
| ADAPT | | 50 | 0 | 2 | 20200 | 20 sec |
| TWODQD | | 42 | 1 | 9 | 9000 | 11 sec |

Therefore only extrapolated routines have been used as INT1, and the following four combinations were tested:

$$\text{QAGS} - \text{QAG(KEY} = 2), \qquad \text{QAGS} - \text{QAGS},$$

$$\text{QXGS} - \text{QXG(KEY} = 1), \qquad \text{QXGS} - \text{QXGS}.$$

In Table II the complete set of results is presented, together with the performance of TWODQD and ADAPT, adaptive routines for automatic quadrature in many dimensions [10, 12, 13], tested for comparison purposes. From the test, the following considerations can be deduced.

(1) The QUADPACK routines are more reliable than TWODQD and ADAPT for high tolerances and can be applied to irregular regions too.
(2) The use of QXGS and QXG instead of QAGS and QAG routines increases the reliability, results in a lower number of functional evaluations, and decreases the evaluation time.

## 5. DESCRIPTION OF THE PACKAGE

The package is self-contained and consists of the single-precision version of the following routines written in standard FORTRAN.

(1) Test routines:

| | |
|---|---|
| TEST | test driver for QXGS and QXG; |
| F | integrand function for the test. |

(2) New code:

| | |
|---|---|
| QXGS | integration routine, modification of QUADPACK routine QAGS; |
| QXG | integration routine, modification of QUADPACK routine QAG; |
| QXGSE | integration routine, called by QXGS, modification of QUADPACK routine QAGSE; |
| QXGE | integration routine, called by QXG, modification of QUADPACK routine QAGE; |
| QXLQM | called by QXGSE, QXGE (replaces the QUADPACK routines QK10, QK21, . . . , QK61); |
| QXRUL | called by QXLQM, applies the quadrature rules; |
| QXRRD | reorders the computed functional values before the bisection of an interval; it is called by QXGSE, QXGE; |
| QXCPY | copies one array into another, called by QXGSE, QXGE (the overall efficiency can be improved by coding this routine into machine language). |

(3) Service routines shared with original QUADPACK code:

| | |
|---|---|
| QELG | QUADPACK service routine (SLATEC version); performs $\epsilon$-algorithm extrapolation, called by QXGSE; |
| QPSRT | QUADPACK service routine (SLATEC version); performs the ordered list management, called by QXGSE, QXGE; |
| R1MACH | SLATEC service routine [8] handles machine constants; it is machine dependent and should be modified depending on the machine characteristics. |

The double-precision version of the package contains routines whose names begin with D (e.g., DQXG) and uses the service routine D1MACH. In R1MACH and D1MACH the calls for error messages have been eliminated in order to obtain a stand-alone package.

These new quadrature programs save time by retaining functional values; it is clear that the storage requirements are much harder than the original QUADPACK routines. In fact, QXGS and QXG require two work arrays: (1) WORK consisting of LENW = 46 * LIMIT real words, and (2) IWORK consisting of LENIW = 3 * LIMIT integer words, where LIMIT is the maximum number of subintervals allowed. The experiments presented in Section 3 suggest that LIMIT = 100 will work reasonably (except for step functions). On a 32-bit computer with 8-byte double precision this means that, in most cases, less than 40,000 bytes of memory are needed to retain functional computed values.

In order to adapt the package to an existing environment there are three possibilities:

(1) Users may allow the package to stand alone. R1MACH and D1MACH should be changed according to the used computer; all the routines in the package are installed. If an error handling routine is wanted, one can adapt the calls to XERROR in QXGS and QXG.

(2) SLATEC users may uncomment the calls to XERROR in QXGS and QXG and install only the new code.

(3) QUADPACK users, out of SLATEC environment, should install only the new code. The local version of QELG, QPSRT, and R1MACH should be checked for compatibility with the new code.

## REFERENCES

1. Casaletto, J., Picket, M., and Rice, J. R. A comparison of some numerical integration programs, *Signum Newsletter 4* (1969), 30-40.
2. Davis, P. J., and Rabinowitz, P. *Methods of Numerical Integration*. Academic Press, New York, 1984.
3. DeBoor, C.. CADRE: An algorithm for numerical quadrature. In *Mathematical Software*, J. R. Rice, Ed., Academic Press, New York, 1971, pp. 417-449.
4. Dongarra, J. J., and Grosse, E.. Distribution of mathematical software via electronic mail. *Signum Newsletter 20* (1985), 45-47.
5. Engels, H. *Numerıcal Quadrature and Cubature*. Academic Press, New York, 1980.
6. Fairweather, G., and Keast, P. An investigation of Romberg quadrature. *ACM Trans. Math. Softw. 4*, 4 (Dec. 1978), 316-322.
7. Favati, P., Lotti, G., and Romani, F. Interpolatory integration formulas for optimal composition. Thıs issue, pp. 207-217.
8. Fox, P. A., Hall, A. D., and Schryer, N. L. ALGORITHM 528: Framework for a portable library. *ACM Trans. Math. Sotfw. 4*, 2 (June 1978), 177-188.
9. Gentleman, W M. Implementing Clenshaw-Curtis quadrature, I Methodology and experience; II Computing the cosine transformation. *Commun. ACM 15*, 5 (May 1972), 337-360.
10. Genz, A. C., and Malik, A. A. Remarks on Algorithm 6: An adaptive algorithm for numerical integrations over an $N$-dimensional rectangular region. *J. Comput. Appl. Math. 6* (1980), 295-302.
11. Kahaner, D. K. Comparison of numerical quadrature formulas. In *Mathematical Software*, J. R. Rice, Ed., Academic Press, New York, 1971, pp. 229-259.
12. Kahaner, D. K., and Rechard, O. W. TWODQD: An adaptive routine for two-dimensional quadrature. *J. Comput. Appl. Math. 17* (1987), 215-234.
13. Kahaner, D. K., and Wells, M. B. An experimental algorithm for $N$-dimensional adaptive quadrature. *ACM Trans. Math. Softw. 5*, 1 (Mar. 1979), 86-96.
14. Lyness, J. N. When not to use an automatic quadrature routine. *SIAM Rev. 25* (1983), 63-87.
15. Lyness, J. N., and Kaganove, J. J. A technique for comparing automatic quadrature routines. *Comp. J. 20* (1975), 170-177.
16. Lyness, J. N., and Kaganove, J. J. Comments on the nature of automatic quadrature routines. *ACM Trans. Math. Softw. 2*, 1 (Mar. 1976), 65-81.
17. Lyness, J. N., and Ninham, B. W. Numerical quadrature and asymptotic expansions. *Math. Comput. 21* (1967), 162-178.
18. Piessens, R. An algorithm for automatic integration. *Angewandte Informatik 9* (1973), 399-401.
19 Piessens, R , et al. *QUADPACK: A Subroutine Package for Automatic Integration*. Springer-Verlag, Berlin, 1983.
20. Robinson, I. A comparison of numerical integration programs, *J. Comput. Appl. Math. 5* (1979), 207-223.
21. Shanks, D. Nonlinear transformations of divergent and slowly convergent sequences. *J. Math. Phys. 34* (1955), 1-42.
22. Wynn, P. On a device for computing the $e_m(S_n)$ transformation. *Math. Comput. 10* (1956), 91-96.

Received June 1988; revised July 1989; accepted May 1990