Self-Validating Integration and Approximation of Piecewise Analytic Functions

Knut Petras *
Institut für Angewandte Mathematik
Technische Universität Braunschweig
Pockelsstr. 14, D-38106 Braunschweig, Germany

Abstract. Let an analytic or a piecewise analytic function on a compact interval be given. We present algorithms that produce enclosures for the integral or the function itself. Under certain conditions on the representation of the function, this is done with the minimal order of numbers of operations. The integration algorithm is implemented and numerical comparisons to non-validating integration software are presented.

1. Introduction

Our purpose is to integrate or to approximate analytic or piecewise analytic functions f on an interval $[a, b]$. If we know that f is analytic, then there are algorithms for integration and approximation based on n function evaluations, that converge exponentially with increasing n. Let now f be given by a function term. Then, using complex interval arithmetic, it is often possible to verify the analyticity. In this case, a self-validating algorithm may easily be constructed, such that the convergence rate is also exponentially.

Suppose we only hope that the underlying function is analytic and therefore use an algorithm which converges exponentially in this case. The convergence rate is in general reduced drastically if the analyticity is hurt at some points. If we knew the location of such breakpoints between intervals of analyticity, we could subdivide the basic interval and apply the algorithm on each subinterval. We want to consider the case that we do not know the number and location of the breakpoints, in particular, that we do not know in advance whether there is a breakpoint or not. Note that, e.g., f defined by $f(x) = |5J_0(4x) + 2|^{1/2}$ (J_0 denotes the Bessel function, see Abramowitz and Stegun [1, Chapter 9]) is analytic, while g defined by $g(x) = |5J_0(4x)+1.99|^{1/2}$ has 10 singularities in the interval $[1, 10]$. We want to show that simple self-validating algorithms yield, in some sense, the optimal rate of convergence in all the mentioned cases. The optimal rate for integrating analytic functions is exponential convergence. What may be regarded as the optimal rate for integrating piecewise analytic functions?

Let $S = \{s_0, \ldots, s_{m+1}\} \subset \mathbb{R}$, where $\{s_1, \ldots, s_m\} \subset [a, b]$, be a set of points such that the integrand f is analytic and bounded by a constant M on each circular disc with midpoint $(s_\nu + s_{\nu-1})/2$ and radius $(s_\nu - s_{\nu-1})/2$:

* *The author is supported by a Heisenberg scholarship of the Deutsche Forschungsgemeinschaft*

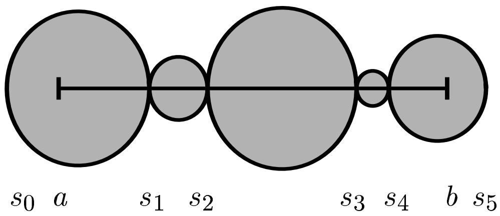

Figure 1.

Then, the optimal integration formula for the class of all such functions has a maximal error of about $\exp(-\operatorname{const}\cdot\sqrt{n})$, where const > 0 and n is the number of evaluation points of f (see Bojanov [3]). Note that we have to know the location of the s_ν in advance. Of course, the problem of uniform approximation cannot be easier. In the following, we consider assumptions, which are weaker in some sense and stronger in another sense. We require that we have a region of analyticity of the form

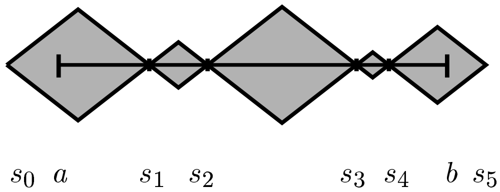

Figure 2.

(this is a weaker assumption) and that the analyticity and boundedness on such a region may in some sense be recognized by complex interval arithmetic (this is a stronger assumption). Then, we also obtain the convergence rate $\exp(-\operatorname{const}\cdot\sqrt{n})$, where const > 0 and n is the number of arithmetic operations. Here, we obtain this rate of convergence without knowing the s_ν.

The situation is therefore similar to the integration or approximation of differentiable functions having unknown singularities. There, singularities may reduce the rate of convergence of 'usual' algorithms considerably, while the rate of convergence is not influenced by singularities for appropriate self-validating algorithms (see Petras [11]).

Self-validating algorithms are, of course, already existing. Some of them (cf., e.g., Corliss [5] or Corliss and Rall [6]) are using automatic differentiation in order to get the necessary information for error estimation. The algorithm of Eiermann [9] is obviously similar to our proposal. However, there are some differences which might influence the theoretical statements taht will be made below.

In Section 2, we recall some simple facts about complex interval arithmetic. In Section 3 and 4, we define our algorithms and prove our main results. Section 5 gives a short description of an implementation and Section 6 some corresponding numerical examples.

2. Complex interval arithmetic

In the literature (see Alefeld and Herzberger [2]), there are mainly two proposals for a complex interval arithmetic. The first one considers circles and the second one rectangles with sides parallel to the coordinate axes as complex intervals. For simplicity but not for principal reasons we decide on rectangular interval arithmetic. Applying a function (a binary operation) to a rectangle (or two rectangles) in the complex plain, the result is defined as the smallest complex interval that contains the image. Our algorithms require a complex interval arithmetic, which is able to see if an expression represents an analytic function on a rectangle at least if the rectangle is sufficiently small relative to the region of analyticity. This is not difficult for functions which are composed from standard functions, i.e., from functions whose regions of analyticity are known. The arithmetic may estimate the modulus of an analytic function on a rectangle, either by simply applying the arithmetic to the whole rectangle (which saves time) or by applying the arithmetic to the four edges respectively (which increases accuracy). The complex interval arithmetic shall be applicable to f in a certain region containing $[a, b] \setminus S$.

Remark. *Ideally, the complex interval arithmetic should do more. Consider*

$$f(x) = |\sin x|, \qquad x \in [-1, 1]$$

Our algorithms use subdivision and we always consider a real subinterval $[\alpha, \beta]$ *and a rectangle of analyticity, which contains the subinterval. Suppose* $[\alpha, \beta] = [0, 1]$*. Then* f *is not analytic on any complex interval* ϱ *containing* $[\alpha, \beta]$ *in its interior. However,* $f\big|_{[\alpha,\beta]}$ *may easily be extended to an analytic function on* $\mathbb{C}$*. The arithmetic should be able to recognize such situations. This is desirable but is not necessary for the theorems below.*

One step in our approximation algorithm is the estimation of the Lipschitz-constant of the integrand. This can be done efficiently by using (real) interval arithmetic and automatic differentiation (see Rall [13]).

3. An integration algorithm and its rate of convergence

Our purpose is to integrate piecewise analytic functions on an interval $[a, b]$. More precisely, we require that the integrand f is bounded on $[a, b]$ and that there is a set $S = \{s_0, \ldots, s_m\}$ such that $a \le s_0 < s_1 < \ldots < s_m \le b$ and f is analytic in each open interval $(s_\nu, s_{\nu+1})$. For given $\varepsilon > 0$, the result of our computation shall be a number q such that we can guarantee

$$\int_a^b f(x)\,dx \in (q - \varepsilon, q + \varepsilon). \tag{1}$$

We start with some remarks on classical quadrature formulas. For fixed $A > 1$ and $B > 0$, let the rectangle $\varrho(\alpha, \beta) := \varrho(\alpha, \beta, A, B)$ in the complex plane be defined by

$$\varrho(\alpha, \beta) = \{x + iy \mid |x - \frac{\beta+\alpha}{2}| \le A \cdot \frac{\beta-\alpha}{2},\ \ |y| \le B \cdot \frac{\beta-\alpha}{2}\}$$

and let m be a functional satisfying

$$m(f; \alpha, \beta) \ge \sup_{z \in \varrho(\alpha,\beta)} |f(z)|.$$

For the integration problem, choose for example $A = \frac{5}{4}$ and $B = \frac{3}{4}$. Then, since the ellipse with foci α, β and lengths $A(\beta - \alpha)$ and $B(\beta - \alpha)$ of its axes is completely contained in $\varrho(\alpha, \beta)$, we have, for example,

$$|R^G_{n;[\alpha,\beta]}[f]| \le 2 \cdot 4^{-n} \cdot (\beta - \alpha) m(f; \alpha, \beta) \text{ and } |R^{CC}_{n;[\alpha,\beta]}[f]| \le 3 \cdot 2^{-n} \cdot (\beta - \alpha) m(f; \alpha, \beta) \tag{2}$$

for functions being analytic on $\varrho(\alpha,\beta)$ (see Petras [11] and Brass [4] respectively). The notation is as follows:

- R_n is the error of a quadrature formula

$$Q_n[f] = \sum_{\nu=1}^{n} c_\nu f(x_\nu)$$

  for the evaluation of an integral over $[-1,1]$. $Q_{n;[\alpha,\beta]}$ is the same quadrature formula transformed onto the interval $[a,b]$ (see Brass [4, p.14]) and $R_{n;[\alpha,\beta]}$ is the corresponding error.
- R_n^G is the error of the Gaussian quadrature formula (see Brass [4, p.90]).
- R_n^{CC} is the error of the Clenshaw-Curtis quadrature formula (see Brass [4, p.117]).

The first error estimate in (2) is slightly improvable, but only by a constant factor, while the second one is at most improvable by factors, which do not decrease exponentially with increasing n. So, the Gaussian formula seems to be far superior. However, the known algorithms for the determination of the nodes and weights in the Gaussian formula with accuracy at least ε require at least $n^2 \cdot |\ln\varepsilon|^\delta$ arithmetic operations with some $\delta > 0$, while, using the Fast Fourier Transform, we need only $O(n \cdot \ln n)$ operations for the Clenshaw-Curtis formula if $n = 2^d + 1$, $d \in \mathbb{N}$. If the quadrature formula in use is stored, then one should prefer the Gaussian formula. If it has to be computed at run-time, the Clenshaw-Curtis formula is preferable. We recommend to store some Gaussian formulas up to a certain number of nodes. If a quadrature formula with higher precision is required, then we could calculate an appropriate Clenshaw-Curtis formula.

In order to have any scaling factor that will be used as a stopping criterion in our algorithm defined below, we estimate the integrand in the whole interval of integration. Therefore, let f be given by an expression, such that the (real) interval evaluations are all bounded by a fixed number if the widths of the real input intervals are less than a fixed number. This ascertains that we can obtain an upper bound for $|f|$ by a simple subdivision algorithm using interval arithmetic. If f is recognized as an analytic function on $\varrho(\alpha,\beta)$, complex interval arithmetic may yield an upper bound $m(f;\alpha,\beta)$ for $|f|$ on this rectangle.

Algorithm.

1.) *Choose a number $c > 1$ and a sequence $\left(Q_n\right)_{n\in\mathbb{N}}$ of quadrature formulas that satisfy error estimates of the form*

$$|R_{n;[\alpha,\beta]}[f]| \leq D \cdot E^{-n} \cdot (\beta - \alpha) m(f;\alpha,\beta)$$

with $E > 1$.

2.) *Calculate an upper bound M for $\|f\|_\infty$.*

3.) *By repeated bisection, determine a subdivision*

$$[a,b] = [a_0,a_1] \cup [a_1,a_2] \cup [a_2,a_3] \cup \ldots \cup [a_{k-1},a_k]$$

such that there is an index set $J \subset \{1,\ldots,k\}$ satisfying

$$m(f;a_{j-1},a_j) \leq c \cdot M \qquad \text{if } j \in J \tag{3}$$

and

$$\sum_{j\notin J} |a_j - a_{j-1}| \le \frac{\varepsilon}{2M}.$$

The interval to be bisected shall always be one of those with greatest lengths, on which condition (3) is not yet satisfied.

4.) *Choose*

$$n \ge \frac{1}{\ln E} \cdot \ln \frac{2(b-a)cDM}{\varepsilon}$$

(which means that by assumption $|R_{n;[\alpha,\beta]}[f]| \le (\beta-\alpha)\frac{\varepsilon}{2(b-a)}$*) and set*

$$q := \sum_{j\in J} Q_{n;[a_{j-1},a_j]}[f].$$

Remarks.

1.) *Because it is natural to calculate the information about the behaviour of* f *in the complex plane by subdivision, our proposal is of course similar to that used by Eiermann [9]. However, there are some differences that probably influence the order optimal behaviour proved below. Main differences are*

   a) *We use a (in principle, see Section 5.1) whole sequence of quadrature formulas.*

   b) *The exit criteria are different. They are designed here to provide the desired enclosures also if we have several singularities. In particular, we have no exit criterion that the number of function evaluations is too high (in the implementation, we exit, if subintervals cannot be subdivided further on the computer).*

2.) *If there is a number* $h > 0$ *such that* $m(f;\alpha,\beta) \le c\cdot M$ *whenever* $\beta-\alpha \le h$*, the algorithm stops after* $O(|\ln\varepsilon|)$ *arithmetic operations, i.e., we have exponential convergence. This can easily be seen, since after finitely many steps, we obtain the bound* $c\cdot M$ *for the integrand in a domain containing the basic interval. Then, the choice of* n *in the algorithm gives the assertion.*

3.) *Suppose we have a piecewise analytic function and suppose further that the integrand on each of the intervals* $(s_{\nu-1}, s_\nu)$ *may be continued in the sense of Remark 1 to an analytic function on a rectangle containing* $[s_{\nu-1}, s_\nu]$ *in its interior respectively. Then, it can be shown that we also have exponential convergence.*

Theorem 1. *Let* $S = \{s_1,\ldots,s_m\} \subset \mathbb{R}$*. If, for the chosen* A*,* B *and* c*, there is some* $\gamma = \gamma(c)$ *such that*

$$m(f;\alpha,\beta) \le c\cdot M$$

whenever

$$\varrho(\alpha,\beta) \subset V_\gamma := \{z \mid |\Im(z)| \le \gamma\cdot \operatorname{dist}\left(\Re(z), S\right)\},$$

then, for each $\varepsilon \in (0,1/2]$*, the described algorithm stops after the determination of* Q_n *plus at most* $O(\ln^2\varepsilon)$ *arithmetic operations and yields a number* q *guaranteeing (1).*

Remark.

1.) *According to our introduction, the theorem shows the order optimality of the given algorithm.*

2.) V_γ is a region as described in Figure 2.

3.) If we choose the Clenshaw-Curtis formulas as the basic sequence of quadrature formulas, then, the calculation of the nodes and weights of the appropriate formula in step 4 requires $O(|\ln\varepsilon|\,|\ln|\ln\varepsilon||)$ operations. Namely we choose the smallest possible n, which is greater than the lower bound in step 4 and has the representation $n = 2^k + 1$ with an integer k.

The proof of Theorem 1 requires the estimation of subinterval widths that can be produced by Algorithm 1

Lemma. *Let the assumptions of Theorem 1 be satisfied. Suppose that the algorithm has produced the subinterval*

$$[a_j, a_{j+1}] \subset [s_\nu, s_{\nu+1}] \qquad \text{with} \qquad \frac{a_j + a_{j+1}}{2} \leq \frac{s_\nu + s_{\nu+1}}{2}.$$

Then,

$$a_{j+1} - a_j > \frac{\gamma}{B + A\gamma} \cdot (a_j - s_\nu).$$

Proof: The interval $[a_j, a_{j+1}]$ has been produced by a bisection of one of the intervals $[a_j - (a_{j+1} - a_j), a_{j+1}]$ or $[a_j, a_{j+1} + (a_{j+1} - a_j)]$. Therefore, at least one of the conditions

$$\varrho(a_j - (a_{j+1} - a_j), a_{j+1}) \not\subset V_\gamma \qquad \text{or} \qquad \varrho(a_j, a_{j+1} + (a_{j+1} - a_j)) \not\subset V_\gamma$$

is satisfied. The first one is weaker since $\frac{a_j + a_{j+1}}{2} \leq \frac{s_\nu + s_{\nu+1}}{2}$. This condition means

$$a_j - A(a_{j+1} - a_j) + i \cdot B(a_{j+1} - a_j) \notin V_\gamma$$

or, equivalently,

$$B(a_{j+1} - a_j) > \gamma\Big(a_j - s_\nu - A(a_{j+1} - a_j)\Big),$$

which is the assertion of the lemma. □

Proof of the theorem. Assume that, for the first time during the algorithm, a subinterval of length $\delta/2$ would be produced in the next step. Then it would originate from a bisection of a subinterval $[\alpha_0, \beta_0]$ of length δ, for which $\varrho(\alpha_0, \beta_0) \not\subset V_\gamma$.

From this moment on, no interval $[\alpha, \beta]$ is subdivided, which satisfies

$$\max\{\mathrm{dist}(\{\alpha\}, S), \mathrm{dist}(\{\beta\}, S)\} > L\delta + \delta, \qquad \text{where} \qquad L = \frac{1}{2}(\frac{B}{\gamma} + A - 1).$$

Either such an interval would be of length $\geq 2\delta$ or it would be of length δ. The first case cannot occur since such an interval would be subdivided before an interval of length $\delta/2$ would be produced. The second case is not possible because the lemma shows that an interval that is further away from S than $L\delta$ has length $> \delta$.

The union of subintervals, whose indicdes are not in the set J, therefore has measure $\leq 2m\delta(L+1)$. If this is less than $\frac{\varepsilon}{2M}$, our assumption must have been wrong, because otherwise, our algorithm would have stopped according to step 3. We conclude that

$$\delta \geq \frac{\varepsilon}{4mM(L+1)}.$$

Furthermore, the $(L+1)\delta$-neighbourhood U of S is completely covered by at most $2m(L+2)$ subintervals, since all intervals have widths $\geq \delta$.

In the following, we count the subintervals covering the complement of U, where we consider without restriction $W = [s_0, \frac{s_0+s_1}{2}] \setminus U$. Denote by $[a_{\mu-1}, a_\mu]$, $[a_\mu, a_{\mu+1}]$, $\ldots$, $[a_{\mu+\lambda-1}, a_{\mu+\lambda}]$ the produced subintervals intersecting W. By the lemma, we have

$$a_{\mu+\lambda} \geq s_0 + (1+\phi)^\lambda (\alpha_\mu - s_0) \qquad \text{where} \qquad \phi = \frac{\gamma}{B + A\gamma}$$

and obtain

$$\lambda \leq \left\lceil \frac{1}{\ln(1+\phi)} \ln \frac{s_1 - s_0}{2(a_\mu - s_0)} \right\rceil .$$

Using

$$a_\mu - s_0 \geq (L+1)\delta \geq \frac{\varepsilon}{4kM},$$

we see that at most $O(|\ln \varepsilon|)$ subdivisions are performed.

On each of the produced subintervals (a_{j-1}, a_j) with $j \in J$ we apply the same quadrature formula with $n = O(|\ln \varepsilon|)$ nodes. We obtain the required precision since the total error on all the intervals (a_{j-1}, a_j) with $j \in J$ and with $j \notin J$ is less than $\frac{\varepsilon}{2}$ respectively. □

4. An approximation algorithm and its rate of convergence

We now want to approximate a Lipschitz continuous piecewise analytic function by piecewise polynomials. Given $\varepsilon > 0$, we want to produce an approximation g satisfying

$$\max_{a \leq x \leq b} |f(x) - g(x)| < \varepsilon.$$

For our purpose it is necessary to estimate the error of interpolation operators $L_{n;[\alpha,\beta]} : C[\alpha,\beta] \to \mathbb{P}_n$ applied to functions being analytic on certain rectangles. Here, let L_n be such an interpolation operator with $n+1$ nodes in $[-1,1]$ and let $L_{n;[\alpha,\beta]}$ be the operator with the nodes of L_n affinely transformed onto $[\alpha,\beta]$. The error of $L_{n;[\alpha,\beta]}$ is defined by

$$\text{err}_{[\alpha,\beta]}[f](x) = f(x) - L_{n;[\alpha,\beta]}[f](x).$$

We consider two types of interpolations. The first one is an interpolation using function values at the boundary points of the interval. This yields continuity of the compound approximation. On $[-1,1]$, we recommend the extreme values of the Chebyshev polynomial T_n, i.e., the zeros of $p = T_{n+1} - T_{n-1}$ and we exemplarily require analyticity on the rectangle

$$\varrho(\alpha,\beta) := \varrho(\alpha,\beta,\frac{5}{4},\frac{3}{4})$$

The interpolation error $\text{err} = \text{err}_{[-1,1]}[f]$ has the representation (see Davis [7, p.68])

$$\text{err}(x) = \frac{1}{2\pi i} \int_{\partial\varrho(-1,1)} \frac{p(x) f(z)}{p(z)(z-x)}\, dz,$$

i.e., if $|f(z)| \leq 1$ on $\varrho(-1,1)$, we obtain

$$|\,\text{err}(x)| \leq \frac{|p(x)|}{2\pi} \int_{\partial\varrho(-1,1)} |k(x,z)|\,|dz|, \qquad \text{where} \qquad k(x,z) = \frac{1}{p(z)\cdot(z-x)}.$$

From the representation of T_ν in the complex plane,

$$T_\nu(z) = \frac{1}{2}\left(w^\nu + \frac{1}{w^\nu}\right), \qquad \text{where } z = \tfrac{1}{2}\left(w + \tfrac{1}{w}\right),$$

we obtain

$$k(z,x) = \left(w^n - \frac{1}{w^n}\right)^{-1} \cdot \frac{1}{(z-x)\sqrt{z^2-1}}.$$

Let $|w| \geq r$, i.e., let z lie outside the ellipse with foci ± 1 and lenghts of axes $r \pm 1/3$. Then,

$$|\operatorname{err}(x)| \leq \frac{|p(x)|}{2\pi}\left(r^n - \frac{1}{r^n}\right)^{-1} \cdot C(x),$$

where

$$C(x) = \int_{\partial\varrho(-1,1)} \frac{|dz|}{|z-x|\sqrt{|z^2-1|}}.$$

We have

$$|z-x|\sqrt{|z^2-1|} = \begin{cases} \sqrt{\dfrac{9}{16} + (t-x)^2} \cdot \sqrt[4]{\dfrac{9}{4} + \left(t^2 - \dfrac{7}{16}\right)^2} & \text{for } z = t + \frac{3i}{4} \\ \sqrt{\left(\dfrac{5}{4} - x\right)^2 + t^2} \cdot \sqrt[4]{\dfrac{81}{256} + \dfrac{41}{8}t^2 + t^4} & \text{for } z = \frac{5}{4} + it \end{cases}$$
$$\geq \begin{cases} \sqrt{\dfrac{3}{2}} \cdot \sqrt{\dfrac{9}{16} + (t-x)^2} & \text{for } z = t + \frac{3i}{4} \\ \dfrac{3}{4} \cdot \sqrt{\left(\dfrac{5}{4} - x\right)^2 + t^2} & \text{for } z = \frac{5}{4} + it. \end{cases}$$

Explicit integration of the reciprocals of these bounds yields

$$C(x) < 4\Bigg(\frac{4}{3}\ln\left(\frac{3}{4} + \sqrt{\left(\frac{5}{4} - x\right)^2 + \frac{9}{16}}\right) - \frac{4}{3}\ln\left(\frac{5}{4} - x\right)$$
$$+ \sqrt{\frac{2}{3}}\ln\left(x + \sqrt{\frac{9}{16} + x^2}\right) - \sqrt{\frac{2}{3}}\ln\left(x - \frac{5}{4} + \sqrt{\frac{9}{16} + \left(x - \frac{5}{4}\right)^2}\right)\Bigg) \leq \frac{8.36}{\sqrt{1-x^2}}.$$

From $|p(x)| \leq 2\sqrt{1-x^2}$ and a transformation onto an arbitrary interval $[a,b]$, we obtain

$$|\operatorname{err}_{[\alpha,\beta]}(x)| \leq \frac{8}{3} \cdot 2^{-n} \cdot \left(1 - \frac{1}{4^n}\right)^{-1} \cdot \sup_{z \in \varrho(\alpha,\beta)} |f(z)|.$$

If we want to have differentiability of order k throughout main parts of the approximation interval and if automatic differentiation (see Rall [13]) is available, we should additionally use the first k derivatives in our approximation, i.e., the nodal polynomial is

$$p(x) = \left(1 - x^2\right)^k \cdot (T_{n+1}(x) - T_{n-1}(x))$$

For example, for the rectangles

$$\varrho(\alpha,\beta) := \varrho(\alpha,\beta,\sqrt{2},1)$$

we obtain analogously to the case $k = 0$,

$$|\operatorname{err}_{[\alpha,\beta]}(x)| \le \frac{14}{9} \cdot (1+\sqrt{2})^{-n} \cdot \left(1 - \frac{1}{(1+\sqrt{2})^{2n}}\right)^{-1} \cdot \sup_{z \in \varrho(\alpha,\beta)} |f(z)| \qquad \text{if } k \ge 1.$$

For our algorithm it is only important that we can choose a form of a rectangle and a corresponding sequence of interpolation operators L_n using $n+1$ nodes respectively such that for $x \in [\alpha, \beta]$,

$$|\operatorname{err}_{[\alpha,\beta]}(x)| \le C \cdot E^{-n} \sup_{z \in \varrho(\alpha,\beta)} |f(z)| \qquad \text{with } E > 1.$$

Algorithm.

1.) *(Initialization) Determine rough upper bounds M_0 for $\|f\|_\infty$ and M_1 for $\|f'\|_\infty$ (for example by using interval arithmetic and automatic differentiation). Furthermore, choose $A > 1$, $B > 0$, $c_0 > 0$ and a positive integer c_1 (all that independently of ε). The basic interval $[a, b]$ forms the initial set of subintervals.*

2.) *As long as the greatest subinterval $[\alpha, \beta]$ on which $m(f; \alpha, \beta) > c_0 \cdot M_0$ has length $\ge \frac{c_1\varepsilon}{2M_1}$, bisect this subinterval.*

3.) *Calculate the maximum μ of all values $m(f; \alpha, \beta) < c \cdot M_0$ on generated subintervals and choose*

$$n \ge \frac{\ln(C\mu) - \ln\varepsilon}{\ln E}.$$

4.) *On each subinterval $[\alpha, \beta]$ with $m(f; \alpha, \beta) < c \cdot M_0$ apply L_n and divide the remaining subintervals into $\le c_1$ smaller subintervals respectively such that for each new subinterval $[\alpha, \beta]$, we have $2M_1(\beta - \alpha) \le \varepsilon$. Apply then linear interpolation at α and β on these subintervals.*

Theorem 2. *Let $S = \{s_1, \ldots, s_m\} \subset \mathbb{R}$. If, for the chosen c_0, there is a number $\gamma = \gamma(c_0)$, such that*

$$m(f; \alpha, \beta) \le c_0 \cdot M_0 \tag{4}$$

whenever

$$\varrho(\alpha, \beta) \subset V_\gamma := \{z \mid |\Im(z)| \le \gamma \cdot \operatorname{dist}(\Re(z), S)\},$$

then, for each $\varepsilon \in (0, 1/2]$, we obtain the precision ε by using at most $O(\ln^2 \varepsilon)$ function evaluations and $O(|\ln \varepsilon|)$ complex interval evaluations of f. The approximation is piecewise a polynomial of degree $O(|\ln \varepsilon|)$.

Sketch of the Proof. The first part of the proof is completely analogous to the proof of Theorem 1. Suppose that the algorithm stops, i.e., for the first time, the greatest subinterval, on which condition (4) is not satisfied, has length $\le \frac{c_1\varepsilon}{2M_1}$. Then, in the preceding step, an interval of length $> \frac{c_1\varepsilon}{2M_1}$ must have been subdivided. As in the proof of Theorem 1, we conclude that the $(L+1)\frac{c_1\varepsilon}{2M_1}$-neighbourhood U of S is completely covered by at most $2m(L+2)$ subintervals (here, L is defined as in the proof of Theorem 1). Each of these subintervals may be subdivided into at most c_1 subintervals such that linear interpolation or the operator L_n yields the required precision. If a subinterval $[\alpha, \beta]$ does not intersect U and does not contain the midpoint between two consecutive singularities, we know that by the lemma in Section 3, it is of length $\ge (1 + \frac{\gamma}{B+A\gamma}) \operatorname{dist}([\alpha, \beta], S)$. Again, as in the proof of Theorem 1, we see that at most $O(|\ln \varepsilon|)$ subintervals have been produced by the algorithm. The choice of n completes the proof. $\square$

5. Implementation

In this section, we describe some details of an implementation of the integration algorithm. We chose the programming language PASCAL-XSC (see `http://www.xsc.de` or Klatte et al. [10]) on a Pentium II, 400 MHz under LINUX*. The prototype of the program, `cinte`, is available (with description) via `http://www.tu-bs.de:/~petras/software.html`.

5.1. Quadrature formulas

For the following reasoning (cf. Petras [11]), we choose the Gaussian formulas. It is known that Gaussian formulas are almost optimal for classes of analytic functions being bounded on certain ellipses in the complex plane. Via the (complicated) functions that map these ellipses conformally onto our used rectangles, we could also construct almost optimal formulas for analytic functions on rectangles. However, these formulas would be different for different choices of the shape of the rectangle. In order to increase flexibility and simplicity, we chose the Gaussian quadrature formulas Q_n^G. We furthermore chose rectangles with side lengths $(r+1/r)(a_j - a_{j-1})/2$ in $\Re$-direction and $(r-1/r)(a_j - a_{j-1})/2$ in $\Im$-direction. The sides of these rectangles are tangential to ellipses that are connected to the almost optimality of the Gaussian quadrature formulas. There is some numerical evidence that, by applying Gaussian quadrature formulas instead of the optimal ones, we loose less that a factor const $\cdot 1.08^n$ in the error estimate. (Note that quadrature formulas like the Clenshaw-Curtis formulas would yield factors of asymptotically more than 2^n).

We provide $Q_2^G, Q_4^G, Q_7^G, Q_{11}^G, Q_{16}^G, Q_{22}^G, Q_{29}^G, Q_{38}^G, Q_{47}^G, Q_{57}^G, Q_{68}^G, Q_{80}^G, Q_{93}^G, Q_{108}^G$ and Q_{128}^G for the algorithm. Theoretically, we would need an infinite sequence of Gaussian formulas. However, it makes practically no sense to store quadrature formulas whose error estimates yield less than the smallest positive machine number for a function, whose calculated upper bound on the rectangle is the largest machine number. Since we usually choose $r > 2$, the mentioned Gaussian formulas almost suffice. In rare cases, additional subdivision can be necessary. However, the effort for this is small compared to the application of the necessary quadrature formulas. In our opinion, these cases do not justify storing further Gaussian formulas.

5.2. Data structures

After having chosen all parameters and quadrature formulas, $f([a,b])$ is estimated in the second phase of the algorithm. For this purpose, we need interval evaluations. We may start subdivision with a large initial interval. In particular in this case, interval arithmetic can produce large overestimations. Therefore, there is often the danger of exits due to overflow. Hence, before applying an operation, we check in advance, whether this operation might yield an overflow (or underflow). For multiplication, e.g., we simply have to add the exponents and check, whether the sum is greater than the maximal representable exponent. This information is stored into an extra boundedness bit. After a detected possible unboundedness (by a machine number), we have to apply no further operations and the final result of the corresponding expression is just the information that the boundedness bit is 1. Our first data structure therefore consists of an interval and the boundedness bit.

* The program can be compiled without problems using the GNU C-library gcc-lib version 2.7.2.1. Newer versions may produce run-time errors due to some problems in the co-operation of PASCAL-XSC and the C-library.

For the third phase of the algorithm, where we try to prove analyticity (or analytic extendability), an analyticity bit is used analogously to the boundedness bit in phase 2. In order to integrate f over the small intervals $A_j = [a_{j-1}, a_j]$, where we have not yet analyticity, it turned out to be helpful but not costly to calculate also bounds for f and f' (if f' is bounded) on A. The estimation of f' is done with the rules of automatic differentiation (see, e.g., Rall [13]). Therefore, two real intervals with corresponding boundedness bits are incorporated.

Boundedness of f' is then used by applying the corresponding optimal quadrature formulas, the midpoint formulas.

5.3. Input of the integrand

The integration routine requires access to routines for the integrand using three different data types for the phases 2, 3 and 4 respectively. One could collect all the corresponding data types into one type and write only one routine for f. The correct data type for the respective function call could be selected by an integer. However, this solution, that is also implemented, more time consuming in the final phase 4, where only (usually many) real interval evaluations of f are needed. Hence, we decided to provide 3 almost identical copies of the routine for f with only different data types. If we would have chosen an interval package for C++, e.g., we could have used templates, which are not available in PASCAL-XSC. More details for the use are given on the above-mentioned web-page.

6. Numerical Examples

One objection against 'usual' integration software is that it can be 'fooled'. This means that we can find examples, where the program gives a 'wrong' answer. The usual objection against validating algorithms is that they are slow.

The response to the first objection is often that 'usually, such examples do not occur in practice'. However, no one can know all present and future applications, i.e., the whole practice.

It is not the purpose of this paper to discredit this first type of software. It often yields satisfactory results very fast, in particular, if integrands are smooth in larger parts of the interval of integration. Our purpose is to provide a fast alternative for users who want to be sure about the result. It turns out that sometimes, validation may even be an advantage with respect to speed although there are faster interval packages than the one used.

We compare our program with THE standard software for general purpose integration, i.e., with the routine `dqags` of the package QUADPACK (see Piessens et al. (1983)). This is for example part of the Nag- and the IMSL-library and is also contained in the netlib (see `http://www.netlib.org/`)

We first give two examples that `dqags` may fail.

Example 1. *The integral*

$$\int_0^\pi 5\sin x + \frac{(9x-4)(9x-8)(3x-4)(9x-10)(\pi-2x)}{1+(90x-110)^4}\,dx$$

gives the estimate 9.9998477865612 *if we choose the required precision* $\varepsilon = 10^{-8}$ *and the estimate* 9.8806414386056 *if* $\varepsilon = 10^{-13}$. *Obviously, one of the estimates is wrong. The reason for the failure for this analytic integrand is that it almost coincides with* $5\sin x$ *at the nodes of the first* *two quadrature formula used by* `dqags`. *If the required precision is* $\varepsilon = 10^{-8}$, *the integrand is interpreted as* $5\sin x$ *within the required accuracy and the first two integral estimates almost coincide. Therefore, the algorithm stops. Of course, by simply looking at the results, we can not decide, which of the* ε *(or if both* ε*) leads to an error. We may trust more in* $\varepsilon = 10^{-13}$ *but it this is no guarantee. With more effort, we could of course find an example, where a larger* ε *gives a smaller error.*

Example 2. *(Petras [11]) The following example seems to be more realistic than Example 1. Calculate*

$$I_z := \int_0^1 \left(\sin x + \frac{1}{8}|x - z|^{3/2}\right) dx$$

for different values of z. *First we require a relative accuracy of* 10^{-8} *and choose* $z = z_i = (2i-1)/200$, $i = 1, \ldots, 100$. *In most cases,* `dqags` *calculates* I_z *with the given accuracy, while in two cases, the error is more than 100 times the permitted error. Requiring an accuracy of* 10^{-9} *and choosing* $z = z_i = (2i-1)/2000$, $i = 1, \ldots, 1000$ *there is one* z *such that the error is more than 10000 times the permitted error.*

These examples indicate the need for safer algorithms. However, these algorithms should not be too slow compared to `dqags`. In order to assess the speed of the validating algorithm, we should first have a look on the speed of the used interval arithmetic package. In Table 1, we list some ratios of the time used to perform an interval operation (or function) in PASCAL-XSC and the time for the corresponding floating point operation (or function) in FORTRAN.

$+/-$	$\times$	$/$	$\sqrt{\ }$	exp	log	sin	tan
15	15.6	5.3	6.7	3.5	4.5	6.8	6.2

Table 1. time(PASCAL-XSC interval function)/ time(FORTRAN function)

Table 2 is analogous to Table 1 but with complex interval operations instead of real interval operations.

$+/-$	$\times$	$/$	$\sqrt{\ }$	exp	log	sin	tan
48	607	8900	208	21	78	44	300

Table 2. time(PASCAL-XSC complex interval function)/ time(FORTRAN function)

One reason for the slow complex interval arithmetic is the aim to compute best possible enclosures for the result of the operation. Since we only want to have rough enclosures for f, a rough complex interval arithmetic should be sufficient. Furthermore, because we deal with functions that are real on the real line, we know that all calculated complex intervals are symmetric with respect to the real line. Both observations lead to a faster complex interval arithmetic:

$+/-$	$\times$	$/$	$\sqrt{\ }$	exp	log	sin	tan
43	73	52	12	17.5	19	36	40

Table 3. time(new PASCAL-XSC complex interval function)/ time(FORTRAN function)

With this faster arithmetic, we obtain the following examples:

Example 3. *We consider the integrals from Examples 1 and 2.* `dqags` *does not calculate all the integrals accurately.* `cinte` *is much slower but gives correct answers. Also the number of function evaluations is higher, but this extra effort seems to be necessary in order to guarantee the results.*

$$I = \int_0^\pi 5\sin x + \frac{(9x-4)(9x-8)(3x-4)(9x-10)(\pi-2x)}{1+(90x-110)^4}\,dx$$

$$II = \int_0^1 4\sin x + \frac{1}{4}|x-z|^{3/2}, \quad z = \frac{2i-1}{4000},\ i = 1,\ldots,1000$$

	`dqags`		`cinte`	
integral	*time (ms)*	*# f-calls*	*time (ms)*	*# f-calls*
$I,\ \varepsilon = 10^{-8}$	0.08*)	*21*	*40.3*	*605*
$I,\ \varepsilon = 10^{-12}$	0.75**)	*525*	*45.7*	*807*
$II,\ \varepsilon = 10^{-9}$	330***)	*290388*	*15670*	*378213*

*) Result: 9.9998477865612

**) Result: 9.8806414386056

***) $1\times$ error $> 10000\varepsilon$
$4\times$ error $> 100\varepsilon$,...

Example 4. *The following examples shows first a simple integral, where* `dqags` *is about 5 times faster than* `cinte`*. The next two integrands show irregular oscillations. Here,* `cinte` *is as fast as or even faster than* `dqags`*. In these examples,* `cinte` *requires far less function evaluations than* `dqags`*.*

integral	`dqags` *time (ms)*	`dqags` *# f-calls*	`cinte` *time (ms)*	`cinte` *# f-calls*
$\int_0^7 e^{-x^2}\,dx$	*0.1*	*105*	*0.49*	*29*
$\int_{10}^{40} e^x \sin x^2\,dx$	*10.8*	*7077*	*11.5*	*845*
$\int_0^{10} \sin e^x\,dx$	*748*	*121821*	*109*	*9768*

Example 5. *Finally, we tested examples with bounded integrands from the QUADPACK-book [12]. In this book, special routines are recommended for the different integrals. We compared their speed with our general purpose method respectively. The parameters α and b attain the values given in [12]. Although our program is not adapted to the special types of integrands, it is slower by only about the factors appearing in Table 1. Again, we have less function evaluations for* `cinte` *than for* `dqags`*.*

integral	QUADPACK time (ms)	QUADPACK # f-calls	`cinte` time (ms)	`cinte` # f-calls
$\int_0^1 \frac{4^{-\alpha}\,dx}{(x-\pi/4)^2+16^{-\alpha}}$	30.5	39075	330	15900
$\int_0^{\pi} \cos(2^\alpha \sin x)\,dx$	10.5	10919	38.1	2695
$\int_0^1 e^{20x-20}\sin(2^\alpha x)\,dx$	3.1	1952	11.5	551
$\int_0^b x^2 \exp(-2^{-\alpha}x)\,dx$	0.64	366	3.5	174

The examples indicate that `cinte` is often not much slower than the QUADPACK-routines. Moreover, we cannot know in advance, which of the programs is the fastest but we know that (if there is no error in the program) `cinte` always yields a correct answer.

Acknowledgement. *The author would like to express his gratitude to Rudolf Lohner for giving so many useful hints concerning PASCAL-XSC to Walter Krämer for providing standard functions in complex interval arithmetic and to Werner Hofschuster for his help during programming error detection.*

References

[1] Abramowitz, M. and Stegun, I.A., *Handbook of mathematical functions with formulas, graphs, and mathematical tables.* National Bureau of Standards, Washington, D.C., 1964.

[2] Alefeld, G. and Herzberger, J., *Introduction to Interval Computations.* Academic Press, New York, 1983.

[3] Bojanov, B. D., Optimal rate of integration and ε-entropy of a class of analytic functions (in Russian). Mat. Zametki 14 (1973), 3-10 [English translation: Math. Notes 19, 551-556]

[4] Brass, H., *Quadraturverfahren.* Vandenhoeck & Ruprecht, 1977.

[5] Corliss, G.F., Computing narrow inclusions for definite integrals. In Kaucher, E., Kulisch, U., and Ullrich, C. (eds.), Computerarithmetic, Teubner, Stuttgart, 1987, pp. 150–169

[6] Corliss, G.F. and Rall, L.B., Adaptive, self-validating numerical quadrature. SIAM J. Sci. Statist. Comput. 8 (1987), 831–847.

[7] Davis, P.J., *Interpolation and Approximation.* Interpolation and approximation. Blaisdell Publ., New York, 1963.

[8] Davis, P.J. and Rabinowitz, P., *Methods of Numerical Integration*, London: Academic Press, 1984.

[9] Eiermann, M.C., Automatic, guaranteed integration of analytic functions, BIT 29 (1989), 270–282.

[10] Klatte, R., Kulisch, U., Neaga, M., Ratz, D. and Ullrich, C., *PASCAL-XSC-Language Reference with Examples.* Springer, Berlin, 1991.

[11] Petras, K., On the complexity of self-validating numerical integration and approximation of functions with singularities. J. Complexity 14 (1998), 302–318.

[12] Piessens, R., de Doncker-Kapenga, E., Überhuber, C.W. and Kahaner, D.K., *QUADPACK: A subroutine package for automatic integration*, Springer Ser. Comput. Math., v. 1, Springer, Berlin, 1983.

[13] Rall, L.B., *Automatic differentiation – Techniques and Applications*, Springer Lecture Notes in Computer Science, Vol. 120, Berlin, 1981.