Validated Evaluation of Special Mathematical Functions

Franky Backeljauw, Stefan Becuwe, and Annie Cuyt

Universiteit Antwerpen, Department of Mathematics and Computer Science,
Middelheimlaan 1, B-2020 Antwerpen, Belgium
{franky.backeljauw, stefan.becuwe, annie.cuyt}@ua.ac.be

Abstract. Because of the importance of special functions, several books and a large collection of papers have been devoted to the numerical computation of these functions, the most well-known being the Abramowitz and Stegun handbook [1]. But up to this date, no environment offers routines for the provable correct evaluation of these special functions.

We point out how series and limit-periodic continued fraction representations of the functions can be helpful in this respect. Our scalable precision technique is mainly based on the use of sharpened a priori truncation and round-off error upper bounds, in case of real arguments. The implementation is validated in the sense that it returns a sharp interval enclosure for the requested function evaluation, at the same cost as the evaluation.

1 Introduction

Special functions are pervasive in all fields of science and industry. The most well-known application areas are in physics, engineering, chemistry, computer science and statistics. Often encountered functions are the Gauss hypergeometric function ${}_2F_1(a, b; c; x)$, the Bessel functions of integer and half-integer order, the (complementary) error function to name just a few. Because of their importance, several books and a large collection of papers have been devoted to algorithms for the numerical computation of these functions.

Virtually all present-day computer systems, from personal computers to the largest supercomputers, implement the IEEE floating-point arithmetic standard, which provides 53 binary or approximately 16 decimal digits accuracy. For most scientific applications, this is more than sufficient. For instance, in electromagnetic simulation models the final required accuracy is usually in the order of only 2 to 3 significant digits.

However, for a rapidly expanding body of applications, 64-bit IEEE arithmetic is no longer sufficient. These range from some exploratory mathematical investigations to large-scale physical simulations performed on highly parallel supercomputers. In these applications, portions of the code typically involve numerically sensitive calculations, which produce results of questionable accuracy using conventional arithmetic. These inaccurate results may in turn induce other errors, such as taking the wrong path in a conditional branch.

S. Autexier et al. (Eds.): AISC/Calculemus/MKM 2008, LNAI 5144, pp. 206–216, 2008.

Such blocks of code benefit enormously from validated numerical techniques, possibly in combination with high-precision arithmetic. Indeed, a reliable numeric technique delivers a floating-point enclosure for the exact result rather than a computed estimate.

Up to this date, even environments such as Maple, Mathematica, MATLAB and libraries such as IMSL, CERN and NAG offer no routines for the provable correct evaluation of special functions. The following quotes concisely express the need for new developments in the evaluation of special functions:

- *"Algorithms with strict bounds on truncation and rounding errors are not generally available for special functions. These obstacles provide an opportunity for creative mathematicians and computer scientists."* Dan Lozier, general director of the DLMF project, and Frank Olver [2,3].
- *"The decisions that go into these algorithm designs — the choice of reduction formulae and interval, the nature and derivation of the approximations — involve skills that few have mastered. The algorithms that MATLAB uses for gamma functions, Bessel functions, error functions, Airy functions, and the like are based on Fortran codes written 20 or 30 years ago."* Cleve Moler, founder of MATLAB [4].

2 Validated Function Evaluation

Let us assume to have at our disposal a scalable precision IEEE 754-854 compliant [5] floating-point implementation of the basic operations, comparisons, base and type conversions, in the rounding modes upward, downward, truncation and round-to-nearest. Such an implementation is characterized by four parameters: the internal base β, the precision t and the exponent range $[L, U]$. Here we aim at least at implementations for $\beta = 2$ at precisions $t \geq 53$, and at implementations for use with $\beta = 2^i$ or $\beta = 10^i$ where $i > 1$.

We denote by $\oplus, \ominus, \otimes, \oslash$ the exactly rounded (to the nearest) floating-point implementation of the basic operations $+, -, \times, \div$ in the chosen base β and precision t. Hence these basic operations are carried out with a relative error of at most $1/2\ \beta^{-t+1}$ which is also called $1/2$ `ulp` in precision t:

$$\left| \frac{(x \circledast y) - (x * y)}{x * y} \right| \leq \frac{1}{2}\beta^{-t+1}, \qquad * \in \{+, -, \times, \div\}.$$

The realization of a machine implementation of a function $f(x)$ in that floating-point environment is essentially a three-step procedure:

1. For a given argument x, the evaluation $f(x)$ is often reduced to the evaluation of f for another argument $\tilde{x}$ lying within specified bounds and for which there exists an easy relationship between $f(x)$ and $f(\tilde{x})$. The issue of argument reduction is a topic in its own right and mostly applies to only the simplest transcendental functions [6]. In the sequel we skip the issue of argument reduction and assume for simplicity that $x = \tilde{x}$.

2. After determining the argument, a mathematical model F for f is constructed and a truncation error
$$\frac{|f(x) - F(x)|}{|f(x)|} \tag{1}$$
comes into play, which needs to be bounded. In the sequel we systematically denote the approximation $F(x) \approx f(x)$ by a capital italic letter.
3. When implemented, in other words evaluated as $\mathtt{F}(x)$, this mathematical model is also subject to a round-off error
$$\frac{|F(x) - \mathtt{F}(x)|}{|f(x)|} \tag{2}$$
which needs to be controlled. We systematically denote the implementation $\mathtt{F}(x)$ of $F(x)$ in capital typewriter font.

The technique to provide a mathematical model $F(x)$ of a function $f(x)$ differs substantially when going from a fixed finite precision context to a finite scalable precision context. In the former, the aim is to provide one optimal mathematical model, requiring as few operations as possible. Here optimal means that the model's complexity is minimal with respect to the truncation error bound imposed by the fixed finite precision. In the latter, the goal is to provide a generic technique, from which a mathematical model yielding the imposed accuracy, is deduced at runtime. Hence best approximants are not an option since these models have to be recomputed every time the precision is altered and a function evaluation is requested. At the same time the generic technique should generate an approximant of as low complexity as possible.

We aim, on the one hand, at a generic technique suitable for use in a multiprecision context, which on the other hand, is efficient enough to compete with the traditional hardware algorithms when the base β is set to 2 and the precision t to 53. We also want our implementation to be reliable, meaning that a sharp interval enclosure for the requested function evaluation is returned without any additional cost.

Besides series representations, as presented in Section 3, continued fraction representations of functions can be very helpful in the multiprecision context. A lot of well-known constants in mathematics, physics and engineering, as well as elementary and special functions enjoy very nice and rapidly converging continued fraction representations. In addition, many of these fractions are limit-periodic. Both, series and continued fraction representations, are classical techniques to approximate functions and there's a lot of literature describing implementations that make use of them [7]. But so far, no attempt is made at an efficient yet provable correct implementation.

It is well-known that the tail or remainder term of a convergent Taylor series expansion converges to zero. It is less well-known that the tail of a convergent continued fraction representation does not necessarily converge to zero. It does not even need to converge at all. A suitable approximation of the usually disregarded continued fraction tail may speed up the convergence of the continued fraction approximants. This idea is elaborated in Section 4.

3 Taylor Series Development

For simplicity, but without loss of generality, we assume that the Taylor series of $f(x)$ is given at the origin:

$$f(x) = \sum_{n=0}^{\infty} a_n x^n. \tag{3}$$

If we want the total error $|f(x) - \mathtt{F}(x)|/|f(x)|$ to be bounded by $\alpha\beta^{-t+1}$ we must determine N such that for $F(x) = p_N(x)$, the partial sum of degree N of (3), the truncation error

$$\left| \frac{f(x) - p_N(x)}{f(x)} \right| \leq \frac{\alpha}{2}\beta^{-t+1}$$

and evaluate $p_N(x)$, possibly in a working precision s slightly larger than the user precision t, such that the computed value $\mathtt{F}(x) = \mathtt{pN}(x)$ satisfies

$$\left| \frac{p_N(x) - \mathtt{pN}(x)}{f(x)} \right| \leq \frac{\alpha}{2}\beta^{-t+1}.$$

An upper bound for the error $|f(x) - p_N(x)|$ is obtained from the sequence of coefficients $\{a_n\}_n$. If this sequence is strictly decreasing with all $a_n > 0$ and with

$$r_n = a_n/a_{n-1} \leq R < 1, \qquad n \geq 1,$$

then

$$\sum_{n=N+1}^{\infty} a_n \leq a_N \sum_{n=0}^{\infty} R^n = \frac{a_N}{1-R}.$$

If the sequence is alternating with $\{(-1)^n a_n\}_n$ positive and decreasing and if N is odd, then

$$\sum_{n=N+1}^{\infty} a_n \leq a_{N+1}.$$

Furthermore in both cases $|f(x)| \geq |p_N(x)|$ and hence

$$\left| \frac{p_N(x) - \mathtt{pN}(x)}{f(x)} \right| \leq \left| \frac{p_N(x) - \mathtt{pN}(x)}{p_N(x)} \right|.$$

A standard method for the evaluation of the polynomial $p_N(x)$ is Horner's scheme, namely

$$p_N(x) = a_0 + x(a_1 + x(a_2 + x(\ldots + xa_N))). \tag{4}$$

Since the coefficients a_n of $p_N(x)$ are often related by a simple ratio $r_n = a_n/a_{n-1}$, Horner's scheme can be rewritten as

$$p_N(x) = a_0(1 + xr_1(1 + xr_2(1 + xr_3(\ldots + xr_N)))). \tag{5}$$

Let $\tilde{a}_0$ and $\tilde{r}_n$ denote the machine representations available for a_0 and r_n respectively. Let $\delta(\cdot)$ denote an upper bound for the relative error (expressed as a multiple of the working precision $1/2$ `ulp`) due to replacing the expression $(\cdot)$ by its floating-point counterpart. Hence

$$\begin{aligned}
\tilde{a}_0 &= a_0(1+\delta_0), \qquad |\delta_0| \le \frac{1}{2}\delta(a_0)\beta^{-s+1},\\
\tilde{r}_n &= r_n(1+\delta_n), \qquad |\delta_n| \le \frac{1}{2}\delta(r_n)\beta^{-s+1}, \qquad n = 1,\ldots,N.
\end{aligned}$$

A round-off error analysis of the nested scheme

$$\begin{aligned}
&q_N(x) = 1,\\
&q_n(x) = 1 \oplus x \otimes \tilde{r}_{n+1} \otimes q_{n+1}(x), \qquad n = N-1,\ldots,0,\\
&\mathtt{pN}(x) = \tilde{a}_0 \otimes q_0(x)
\end{aligned}$$

provides the bound [8, pp. 69]

$$\left|\frac{p_N(x) - \mathtt{pN}(x)}{p_N(x)}\right| \le \left|\frac{\epsilon(N)}{1-\epsilon(N)}\right| \frac{p_N^+(|x|)}{|p_N(x)|},$$

where

$$\begin{aligned}
p_N^+(x) &= \sum_{n=0}^{N} |a_n| x^n,\\
\epsilon(N) &= \frac{1}{2}\left(\delta(a_0) + N\left(3 + \delta(x) + \max_{n=1,\ldots,N}\delta(r_n)\right)\right)\beta^{-s+1}.
\end{aligned}$$

Note that the factor

$$\frac{p_N^+(|x|)}{|p_N(x)|} \ge 1. \tag{6}$$

It equals 1 if $a_n \ge 0$ for all n and $x \ge 0$, or if $(-1)^n a_n \ge 0$ for all n and $x \le 0$. Otherwise this factor can sometimes be arbitrarily large.

4 Continued Fraction Representation

Let us consider a continued fraction representation of the form

$$f = \frac{a_1}{1+\dfrac{a_2}{1+\ldots}} = \frac{a_1}{\left\lceil 1\right\rfloor} + \frac{a_2}{\left\lceil 1\right\rfloor} + \ldots = \sum_{n=1}^{\infty} \frac{a_n}{\left\lceil 1\right\rfloor}, \quad a_n := a_n(x), \quad f := f(x) \tag{7}$$

with $a_n \ge -1/4$. Here a_n is called the n-th partial numerator. The continued fraction is said to be limit-periodic if the limit $\lim_{n\to\infty} a_n$ exists (it is allowed to

be $+\infty$). We respectively denote by the N-th approximant $f_N(x; w_N)$ and N-th tail $t_N(x)$ of (7), the values

$$f_N(x; w_N) = \sum_{n=1}^{N-1} \frac{a_n}{\lceil 1 \rfloor} + \frac{a_N}{\lceil 1 + w \rfloor},$$
$$t_N(x) = \sum_{n=N+1}^{\infty} \frac{a_n}{\lceil 1 \rfloor}.$$

We restrict ourselves to the case where a sequence $\{w_n\}_n, w_n \neq 0$ can be chosen such that $\lim_{n\to\infty} f_n(x; w_n) = \lim_{n\to\infty} f_n(x; 0)$.

The tails $t_N(x)$ of a convergent continued fraction can behave quite differently compared to the tails of a convergent series which always go to zero. We illustrate the different cases with an example. Take for instance the continued fraction expansion

$$\frac{\sqrt{1+4x}-1}{2} = \sum_{n=1}^{\infty} \frac{x}{\lceil 1 \rfloor}, \qquad x \geq -\frac{1}{4}.$$

Each tail $t_N(x)$ converges to the value $\frac{1}{2}(\sqrt{1+4x}-1)$ as well and hence the sequence of tails is a constant sequence. More remarkable is that the even-numbered tails of the convergent continued fraction

$$\sqrt{2}-1 = \sum_{n=1}^{\infty} \left(\frac{(3+(-1)^n)/2}{\lceil 1 \rfloor} \right) = \frac{1}{\lceil 1 \rfloor} + \frac{2}{\lceil 1 \rfloor} + \frac{1}{\lceil 1 \rfloor} + \frac{2}{\lceil 1 \rfloor} + \dots$$

converge to $\sqrt{2}-1$ while the odd-numbered tails converge to $\sqrt{2}$ (hence the sequence of tails does not converge), and that the sequence of tails $\{t_N(x)\}_N = \{N+1\}_N$ of

$$1 = \sum_{n=1}^{\infty} \frac{n(n+2)}{\lceil 1 \rfloor}$$

converges to $+\infty$. When carefully monitoring the behaviour of these continued fraction tails, very accurate approximants $f_N(x; w_N)$ for f can be computed by making an appropriate choice for w_N. For instance, when $\lim_{n\to\infty} a_n = a < +\infty$ then an estimate of the N-th tail is given by $(\sqrt{1+4a}-1)/2$. The appearance of the square root explains the condition $a_n \geq -\frac{1}{4}$.

The relative truncation error $|f(x) - f_N(x; w_N)|/|f(x)|$ is bounded by the so-called interval sequence theorem [9]. Let the sequence of intervals $\{[L_n, R_n]\}_n$ with $-\frac{1}{2} \leq L_n \leq R_n < \infty$ be given such that we have for

$$b_n := (1 + \operatorname{sign}(L_n) \max(|L_n|, |R_n|)) L_{n-1},$$
$$c_n := (1 + \operatorname{sign}(L_n) \min(|L_n|, |R_n|)) R_{n-1},$$

that

$$b_n \leq a_n \leq c_n, \qquad 0 \leq b_n c_n.$$

Then

$$\left|\frac{f(x)-f_N(x;w_N)}{f(x)}\right| \leq \frac{R_N-L_N}{1+L_N}\prod_{n=1}^{N-1} M_n,$$

$$L_N \leq w_N \leq R_N, \qquad M_n = \max\left\{\left|\frac{L_n}{1+L_n}\right|, \left|\frac{R_n}{1+R_n}\right|\right\}.$$

The L_n and R_n are tails of continued fractions constructed with the entries b_n and c_n [9] which actually bound the floating-point uncertainty on a_n. If the partial numerators a_n of the continued fraction (7) satisfy $a_n \geq {}^{-1}\!/_{4}$, then we know that:

- in case all $a_n > 0$ and $w_N \leq t_N$, the even approximants satisfy $f_N(x;w_N) \leq f(x)$,
- in case all $a_n < 0$ and $w_N \leq t_N$, all approximants satisfy $f_N(x;w_N) \leq f(x)$.

Hence we obtain for the round-off error on the computed value $\mathtt{F}(x) = \mathtt{fN}(x;w_N)$:

$$\left|\frac{f_N(x;w_N)-\mathtt{fN}(x;w_N)}{f_N(x;w_N)}\right| \leq \left|\frac{f_N(x;w_N)-\mathtt{fN}(x;w_N)}{\mathtt{fN}(x;w_N)}\right|.$$

If the machine representation $\tilde{a}_n = a_n(1+\delta_n)$ with $|\delta_n| \leq {}^{1}\!/_{2}\,\delta(a_n)\beta^{-s+1}$ then [10, pp. 156–158] [11]

$$\left|\frac{f_N(x;w_N)-\mathtt{fN}(x;w_N)}{\mathtt{fN}(x;w_N)}\right| \leq \frac{1}{2}\,(4+\Delta)\left(1+M+\ldots+M^{N-1}\right)\beta^{-s+1},$$

$$\Delta = \max_{n=1,\ldots,N}\delta(a_n), \qquad M = \max_{n=1,\ldots,N} M_n,$$

where $s \geq t$ is the working precision.

5 Example: The Error Function

We consider the error function and the complementary error function

$$\operatorname{erf}(x) = \frac{2}{\sqrt{x}}\int_0^x e^{-t^2}dt,$$

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{x}}\int_x^\infty e^{-t^2}dt$$

for $x \in \mathbb{R}$. These functions are closely related to one another through

$$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x). \tag{8}$$

Furthermore, we can limit the discussion to $x > 0$ since

$$\operatorname{erf}(0) = 0,$$

$$\operatorname{erf}(-x) = -\operatorname{erf}(x),$$

$$\operatorname{erfc}(-x) = 2 - \operatorname{erfc}(x).$$

5.1 Series Implementation $0 < x \leq 1$

The Maclaurin series of $\mathrm{erf}(x)$ is defined by

$$\frac{\mathrm{erf}(x)}{2/\sqrt{\pi}} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)n!}. \tag{9}$$

Its coefficients are related by the ratio

$$r_n = -\frac{2n-1}{n(2n+1)},$$

which can be computed using one floating-point division, if we assume that N is such that $N(2N+1)$ remains exactly representable in the base β precision s floating-point system in use. Then $\max_{n=1,\ldots,N} \delta(r_n) = 1$. A sufficient lower bound for $\mathrm{erf}(x)$ is given by

$$e(x) = x - \frac{x^3}{3}$$

for which $\mathrm{erf}(x)/e(x) \leq 1.121$. The factor (6) is bounded by 2. To compute the series using (5) we replace x by x^2 and a_0 by x, with $\delta(x^2) = 1$ and $\delta(a_0) = 0$ given that x is a floating-point number.

In *Table 1* we display the evaluation of $\mathrm{erf}(x)$ in a scalable precision floating-point system with $\beta = 2$ and $t = 125$ for a number of x-values. We also list the degree N of the partial sum and the working precision s.

Table 1.

x	$\mathrm{erf}(x)$	N	s
0.125	0.14031620480 ...	15	139
0.250	0.27632639016 ...	19	139
0.375	0.40411690943 ...	21	139
0.500	0.52049987781 ...	25	139
0.625	0.62324088218 ...	27	140
0.750	0.71115563365 ...	29	140
0.875	0.78407506105 ...	31	140
1.000	0.84270079294 ...	35	140

5.2 Continued Fraction Implementation on $1 < x$

Using (8) in combination with

$$\mathrm{erfc}(x) = \frac{e^{-x^2}}{\sqrt{\pi}} \left(\cfrac{2x/(2x^2+1)}{1} + \sum_{n=2}^{\infty} \cfrac{\frac{-(2n-3)(2n-2)}{(2x^2+4n-7)(2x^2+4n-3)}}{1} \right)$$

the values in *Table 2* are obtained. Again $\beta = 2, t = 125$ and the approximant number N and the working precision s are listed.

Here for $n \geq 2$ all $a_n < 0$ with

$$\delta(a_n) = 7, \qquad M = 0.85.$$

We can safely put that the integers $4N - 3$ and $(2N - 3)(2N - 2)$ can be represented exactly and that $\delta(x^2) = 1$ since $\delta(x) = 0$. For the additional factors $\exp(-x^2)/\sqrt{\pi}$ in combination with $2x/(2x^2+1)$ a separate error analysis is made.

Table 2.

x	erfc(x)	N	s
1.750	$1.3328328780 \ldots e{-}2$	77	143
2.500	$4.0695201744 \ldots e{-}4$	41	142
3.250	$4.3027794636 \ldots e{-}6$	27	143
4.000	$1.5417257900 \ldots e{-}8$	20	142
4.750	$1.8485047721 \ldots e{-}11$	16	142
5.500	$7.3578479179 \ldots e{-}15$	14	143
6.250	$9.6722041318 \ldots e{-}19$	12	142
7.000	$4.1838256077 \ldots e{-}23$	11	144

6 Special Function Support

In Table 3 we indicate which functions and which argument ranges (on the real line) are covered by our implementation. For the definition of the special functions we refer to [10].

Table 3.

special function	series	continued fraction
$\gamma(a, x)$		$a > 0, x \neq 0$[1]
$\Gamma(a, x)$		$a \in \mathbb{R}, x \geq 0$
erf(x)	$\lvert x\rvert \leq 1$	identity via erfc(x)
erfc(x)	identity via erf(x)	$\lvert x\rvert > 1$
dawson(x)	$\lvert x\rvert \leq 1$	$\lvert x\rvert > 1$

[1] For $a > 0, a > x$ a faster implementation making use of series is under development.

Table 3. (*continued*)

special function	series	continued fraction
Fresnel $S(x)$	$x \in \mathbb{R}^2$	
Fresnel $C(x)$	$x \in \mathbb{R}^2$	
$E_n(x),\ n > 0$		$n \in \mathbb{N}, x > 0$[3]
${}_2F_1(a, n; c; x)$		$a \in \mathbb{R}, n \in \mathbb{Z},$ $c \in \mathbb{R} \setminus \mathbb{Z}_0^-, x < 1$
${}_1F_1(n; c; x)$		$n \in \mathbb{Z},$ $c \in \mathbb{R} \setminus \mathbb{Z}_0^-, x \in \mathbb{R}$
$I_n(x)$	$n = 0, x \in \mathbb{R}$	$n \in \mathbb{N}, x \in \mathbb{R}$
$J_n(x)$	$n = 0, x \in \mathbb{R}$	$n \in \mathbb{N}, x \in \mathbb{R}$
$I_{n+1/2}(x)$	$n = 0, x \in \mathbb{R}$	$n \in \mathbb{N}, x \in \mathbb{R}$
$J_{n+1/2}(x)$	$n = 0, x \in \mathbb{R}$	$n \in \mathbb{N}, x \in \mathbb{R}$

A similar implementation in the complex plane is the subject of future research.

References

1. Abramowitz, M., Stegun, I.: Handbook of mathematical functions with formulas, graphs and mathematical tables. U.S. Government Printing Office, NBS, Washington (1964)
2. Cipra, B.A.: A new testament for special functions. SIAM News 31(2) (March 1998)
3. Lozier, D.: NIST Digital Library of Mathematical Functions. Annals of Mathematics and Artificial Intelligence 38(1–3) (May 2003)
4. Moler, C.B.: Cleve's corner: The tetragamma function and numerical craftsmanship: MATLAB's special mathematical functions rely on skills from another era. Technical note, The MathWorks, Inc (2002)
5. Floating-Point Working Group: IEEE standard for binary floating-point arithmetic. SIGPLAN 22, 9–25 (1987)
6. Muller, J.M.: Elementary functions: Algorithms and implementation. Birkhäuser, Basel (1997)
7. Lozier, D., Olver, F.: Numerical Evaluation of Special Functions. In: Gautschi, W. (ed.) AMS Proceedings of Symposia in Applied Mathematics, vol. 48, pp. 79–125 (1994); Updated version (December 2000), `http://math.nist.gov/nesf/`
8. Higham, N.: Accuracy and stability of numerical algorithms. SIAM, Philadelphia (1996)

[2] When $|x| > 1$ the implementation is slow. A faster series version is planned in the near future.

[3] When $0 < x \leq 1$ the implementation is slow. A faster series version is planned in the near future.

9. Cuyt, A., Verdonk, B., Waadeland, H.: Efficient and reliable multiprecision implementation of elementary and special functions. SIAM J. Sci. Comput. 28, 1437–1462 (2006)
10. Cuyt, A., Brevik Petersen, V., Verdonk, B., Waadeland, H., Jones, W.: Handbook of Continued Fractions for Special Functions. Springer, Heidelberg (2008)
11. Jones, W., Thron, W.: Numerical stability in evaluating continued fractions. Math. Comp. 28, 795–810 (1974)