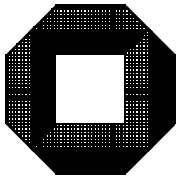


UNIVERSITÄT KARLSRUHE

Ein rechnergestützter Fehlerkalkül mit Anwendung auf ein genaues Tabellenverfahren

W. Hofschuster und W. Krämer

Preprint Nr. 96/5

Institut für Wissenschaftliches Rechnen
und Mathematische Modellbildung

76128 Karlsruhe

Anschrift der Verfasser:

Dipl.-Math. oec. Werner Hofschuster
Institut für Angewandte Mathematik

HDoz. Dr. Walter Krämer
Institut für Wissenschaftliches Rechnen und
Mathematische Modellbildung (IWRMM)

Universität Karlsruhe
Postfach 6980
76128 Karlsruhe

Bundesrepublik Deutschland

Das Postscript-File dieses Preprints sowie die vollständigen Quelltexte aller Programme sind über FTP unter der Adresse
`iamk4515.mathematik.uni-karlsruhe.de`
im Verzeichnis
`/pub/iwrmm/preprints`
erhältlich.

Ein rechnergestützter Fehlerkalkül mit Anwendung auf ein genaues Tabellenverfahren[1]

Werner Hofschuster und Walter Krämer

1 Einleitung

Einschließungsalgorithmen ermöglichen es, mathematisch gesicherte Aussagen mit dem Rechner zu gewinnen. Aufbauend auf einer mathematisch sauber definierten Rechnerarithmetik wurden in den letzten 20 Jahren zahlreiche Verfahren entwickelt, welche die Lösung der zugrundeliegenden Probleme in enge Schranken einschließen. Das Hilfsmittel für den Verifikationsprozeß ist das Vorhandensein einer (Maschinen-) Intervallarithmetik. Letztendlich kommt es darauf an, daß man mit dem Computer Wertebereichseinschließungen gewisser Funktionen über (i. a. mehrdimensionalen) Intervallen berechnen kann.

Geht man von linearen Problemen weg, so benötigt man nicht nur Intervall-Grundoperationen, sondern auch Intervallfunktionen wie Sinus, Kosinus, Logarithmus usw. Wertet man eine solche Maschinen-Intervallfunktion für ein gegebenes Intervallargument aus, so wird eine sichere, durch Maschinenzahlen begrenzte Einschließung des Wertebereiches über dem gegebenen Argumentintervall als Ergebnis zurückgegeben. Dabei ist man bei der Realisierung einer Intervallfunktion daran interessiert, den tatsächlichen Wertebereich nur unwesentlich zu überschätzen (hochgenaue Realisierung; bestmöglich, aber in der Regel nicht erreichbar, wäre der nach außen gerundete exakte Wertebereich). Zu grobe Überschätzungen können z. B. dazu führen, daß der Nachweis einer Abbildung eines Intervallvektors in sich (in Verbindung mit einem geeigneten Fixpunktsatz ist dies häufig der Kern des Verifikationsprozesses) nicht funktioniert.

In modernen Entwicklungsumgebungen für wissenschaftliches Rechnen, wie z. B. den sogenannten XSC-Entwicklungsumgebungen, sind bereits heute die in Programmiersprachen üblicherweise vorhandenen mathematischen Funktionen als Intervallfunktionen vorhanden. Allerdings verwenden diese Intervallfunktionen ein höhergenaues internes Gleitkommadatenformat, welches mit Hilfe von Ganzzahloperationen emuliert wird. Dies führt notgedrungen zu erheblichen Laufzeiteinbußen.

In einem gemeinsamen Projekt des Instituts für Wissenschaftliches Rechnen und Mathematische Modellbildung (IWRMM) und des Instituts für Angewandte Ma-

[1] Modifizierte Version vom 12.02.97

thematik wird daran gearbeitet, diesen Laufzeitnachteil zu eliminieren. So kann z.B. durch den Einsatz genauer Tabellenverfahren ein zweistelliger Beschleunigungsfaktor gegenüber den bisherigen Funktionsrealisierungen erzielt werden. Die Funktionen werden dabei aus Portabilitätsgründen in ANSI-C [1, 9] implementiert.

Um hochgenaue, verläßliche Fehlerschranken berechnen zu können, mußte hierzu ein Fehlerkalkül entwickelt werden, welcher es erlaubt, die sehr umfangreichen Fehlerabschätzungen zumindest halbautomatisch durchzuführen. Dieser Kalkül soll in den folgenden Abschnitten vorgestellt und auf ein Tabellenverfahren angewendet werden.

Dieser Kalkül[2] wird in Abschnitt 2 zunächst theoretisch vorgestellt. Eine konkrete Realisierung in PASCAL-XSC [10] – das entsprechende Programmlisting `eps_ari.p` findet sich auszugsweise im Anhang A – wird besprochen und an einem ersten Beispiel vorgeführt.

Bevor das Tabellenverfahren für $e^x - 1$ ausführlich erläutert wird, werden in Abschnitt 3 zwei später benötigte Approximationen angegeben und deren Fehlerschranken berechnet. Auf die verwendete Methode wird hier nur sehr knapp eingegangen, sie kann in [15] nachgelesen werden.

Abschnitt 4 beschreibt das verwendete Tabellenverfahren, Abschnitt 5 dessen konkrete Implementierung für Punktargumente. Der entsprechende ANSI-C Quelltext befindet sich im Anhang B.

Als Beispielfunktion wurde $e^x - 1$ gewählt, da diese Funktion im Zusammenhang mit der Realisierung der hyperbolischen Funktionen auftritt. So kann z.B. die Funktion $\sinh(x)$ numerisch günstig durch Anwendung der folgenden Gleichung berechnet werden:

$$\sinh(x) = \mathrm{sign}(x) \cdot 0.5 \cdot ((e^{|x|} - 1) + \frac{(e^{|x|} - 1)}{(e^{|x|} - 1) + 1}).$$

Eine Berechnung mit Hilfe der Funktion e^x würde aufgrund von Auslöschungseffekten zu unbrauchbaren Ergebnissen führen.

Schließlich wird in Abschnitt 6 die eigentliche Fehlerabschätzung durchgeführt. Hier wird der früher entwickelte Kalkül bzw. dessen Umsetzung mittels Intervallroutinen angewendet. Ein Auszug des entgültigen Programms zur worst-case-Fehlerabschätzung kann in Anhang C nachvollzogen werden. Man erhält als gesicherte Oberschranke $\varepsilon(\texttt{expm1})$ für den relativen Maximalfehler

$$\varepsilon(\texttt{expm1}) = 2.593 \cdot 10^{-16} \tag{1}$$

(die Maschinengenauigkeit beträgt $1.1102\ldots\cdot 10^{-16}$). Mit dieser Fehlerschranke wird schließlich in Abschnitt 7 die Realisierung der Intervallfunktion durchgeführt.

In der Arbeit verwendete Notation:

$S = S(b, l, \underline{e}, \overline{e})$	Gleitkommaraster mit Basis b, Mantissenlänge l und Exponent e mit $\underline{e} \le e \le \overline{e}$

[2] In dieser Arbeit betrachten wir nur die Grundoperationen. Der Kalkül kann aber auf weitere Operationen, z.B. einstellige Funktionsberechnungen, erweitert werden.

$S(2, 53, -1022, 1023)$	IEEE-Datenformat double
$\circ \in \{+, -, \bullet, /\}$	exakte reelle Operation
$\boxdot, \circ \in \{+, -, \bullet, /\}$	Gleitkommaoperator, Maschinenoperation mit Rundung zur nächstgelegenen Gleitkommazahl
$\bigtriangledown, \circ \in \{+, -, \bullet, /\}$	Maschinenoperation mit Rundung nach unten
$\bigtriangleup, \circ \in \{+, -, \bullet, /\}$	Maschinenoperation mit Rundung nach oben
$\lvert a \circ b - a \boxdot b \rvert := \lvert (a \circ b) - (a \boxdot b) \rvert$	Implizite Klammerung beachten!
`MinReal`	kleinste positive normalisierte Gleitkommazahl, für IEEE-Datenformat gilt: `MinReal`:=$2.22\ldots \cdot 10^{-308}$
`MaxReal`	größte normalisierte Gleitkommazahl, für IEEE-Datenformat gilt: `MaxReal`:=$1.78\ldots \cdot 10^{308}$
$\tilde{a}$	auf der Maschine berechnete, i.a. fehlerbehaftete Größe
ulp	eine Einheit in der letzten Mantissenstelle (unit last place)
ε^*	relative Maschinengenauigkeit, $\varepsilon^* := \frac{1}{2}2^{1-l} = 2^{-l}$
`eps52`	`eps52`:= $2^{1-53} = 2.22044\ldots \cdot 10^{-16}$
`eps53`	`eps53`:= $\frac{1}{2}2^{1-53} = 1.11022\ldots \cdot 10^{-16} = \varepsilon^*$
$\mathrm{succ}(x)$, $x \in S$	Gleitkommanachfolger von x
$\mathbb{R}^+$	Menge der positiven reellen Zahlen
$I\mathbb{R}$	Menge der abgeschlossenen Intervalle über den reellen Zahlen
IS	Menge der Maschinenintervalle $IS = \{[\underline{a}, \overline{a}] \mid \underline{a}, \overline{a} \in S, \underline{a} \leq \overline{a}\}$
$\lvert A \rvert, A \in I\mathbb{R}$	$\lvert A \rvert := \max_{a \in A} \lvert a \rvert$, Betragsmaximum
$<A>, A \in I\mathbb{R}$	$<A> := \min_{a \in A} \lvert a \rvert$, Betragsminimum
$\mathrm{diam}(A)$, $A \in I\mathbb{R}$	Durchmesser eines Intervalls, $\mathrm{diam}(A) := \sup(A) - \inf(A)$
$x\rvert_n$, $x \in S$	Die führenden n Mantissenbits von x

2 Fehlerkalkül

In diesem Abschnitt werden Techniken hergeleitet, welche es erlauben, Rundungsfehler in Gleitkommaalgorithmen sicher abzuschätzen. Auch die Fehlerfortpflanzung wird durch den hier vorgestellten Kalkül mit erfaßt.

Für den Fehlerkalkül wird vorausgesetzt, daß die Operationen auf der Maschine dem IEEE-Standard [2, 3] entsprechen. Das bedeutet insbesondere, daß für die Grundoperation $\circ \in \{+, -, \bullet, /\}$ und deren Maschinenäquivalent $\boxdot, \circ \in \{+, -, \bullet, /\}$

die folgende Beziehung gilt:

$$\bigwedge_{\circ \in \{+,-,\bullet,/\}} \quad \bigwedge_{\substack{a,b \in S \text{ mit} \\ |a \circ b| \in [\texttt{MinReal},\texttt{MaxReal}]}} \left| \frac{a \circ b - a \boxed{\circ} b}{a \circ b} \right| \leq \varepsilon^* \; . \tag{2}$$

Lemma 1 (Unterlaufbereich) Im Unterlaufbereich $U := (-\texttt{MinReal},\texttt{MinReal})$ gilt die folgende Abschätzung:

$$\bigwedge_{\circ \in \{+,-,\bullet,/\}} \quad \bigwedge_{\substack{a,b \in S \\ |a \circ b| < \texttt{MinReal}}} |a \boxed{\circ} b - a \circ b| \leq \texttt{diam}([0,\texttt{MinReal}]) = \texttt{MinReal} \tag{3}$$

.

Beweis:
Hier wird nur verwendet, daß die Vorzeichen von $a \boxed{\circ} b$ und $a \circ b$ übereinstimmen □

Die obigen Beziehungen sind richtig, falls als Rundungsmode „Rundung zur nächsten Maschinenzahl“ verwendet wird. Bei Verwendung von gerichtet gerundeten Operationen muß $\varepsilon^* = \frac{1}{2} b^{1-l}$ in Formel (2) durch $2\varepsilon^* = b^{1-l}$ ersetzt werden. Dies trifft auch dann zu, wenn man von der Maschinenarithmetik nur voraussetzt, daß sie „faithful“ ist (vgl. [5, 16]).

Im folgenden soll die Fehlerfortpflanzung für die Grundoperationen bei Anwendung auf bereits fehlerbehaftete Argumente untersucht werden.

Seien dazu $\circ \in \{+,-,\bullet,/\}$ und $a \in A \in I\!I\!R$, $b \in B \in I\!I\!R$,

$$\tilde{a} = a + \triangle_a \;\text{ mit }\; |\triangle_a| \leq \triangle(a),$$

$$\tilde{b} = a + \triangle_b \;\text{ mit }\; |\triangle_b| \leq \triangle(b).$$

Die Größen $A, B, \triangle(a)$ und $\triangle(b)$ seien dabei gegeben. Gesucht ist dann eine Schranke $\triangle(\circ) \in I\!R^+$, so daß

$$\bigwedge_{\substack{a \in A,\ b \in B \\ a \circ b, \tilde{a} \circ \tilde{b} \in [-\texttt{MaxReal},\texttt{MaxReal}]}} \quad \bigwedge_{\substack{|a - \tilde{a}| \leq \triangle(a) \\ |b - \tilde{b}| \leq \triangle(b)}} \left| a \circ b - \tilde{a} \boxed{\circ} \tilde{b} \right| \leq \triangle(\circ)$$

gilt.

Die folgenden Sätze beantworten diese Frage für die arithmetischen Grundoperationen. Generell sei dabei vorausgesetzt, daß $a \circ b, \tilde{a} \circ \tilde{b} \in [-\texttt{MaxReal},\texttt{MaxReal}]$, d.h. daß kein Überlauf auftritt. Die später angegebenen Intervallroutinen brechen bei Überlauf mit einer entsprechenden Fehlermeldung ab.

Satz 1 (Addition) Für die Addition $\circ := +$ gilt die Fehlerfortpflanzungsabschätzung

$$\bigwedge_{\substack{a \in A \\ b \in B}} \quad \bigwedge_{\substack{|a - \tilde{a}| \leq \triangle(a) \\ |b - \tilde{b}| \leq \triangle(b)}} \left| a + b - \tilde{a} \boxplus \tilde{b} \right| \leq \triangle(\text{add}) \quad \text{mit}$$

$$\triangle(\text{add}) := \varepsilon^*(|A + B|) + (1 + \varepsilon^*)(\triangle(a) + \triangle(b)) + \texttt{MinReal}$$

Beweis:
Es seien $a \in A, b \in B, \tilde{a} = a + \triangle_a \in A + [-\triangle(a), \triangle(a)], |\triangle_a| \leq \triangle(a)$ und $\tilde{b} = b + \triangle_b \in B + [-\triangle(b), \triangle(b)], |\triangle_b| \leq \triangle(b)$ beliebig aber fest gewählt.

I) Fehlerschranke für $|\tilde{a} + \tilde{b} - \tilde{a} \boxplus \tilde{b}|$:
In dieser Abschätzung werden nur gestörte Argumente berücksichtigt.

a) $\tilde{a} + \tilde{b} \in U$, d.h. die exakte Summe der gestörten Argumente liegt im Unterlaufbereich.
$\tilde{a} + \tilde{b} \in U \Longrightarrow \tilde{a} \boxplus \tilde{b} \in \overline{U} := U \cup \{-\text{MinReal}, \text{MinReal}\}$
$\overset{\text{Lemma 1}}{\Longrightarrow} |\tilde{a} + \tilde{b} - \tilde{a} \boxplus \tilde{b}| \leq \texttt{MinReal}$

b) $\tilde{a} + \tilde{b} \notin U$, d.h. der Betrag der exakten Summe der gestörten Argumente liegt in $|\tilde{a} + \tilde{b}| \in [\texttt{MinReal}, \texttt{MaxReal}]$:
$\tilde{a} + \tilde{b} \notin U \Longrightarrow \tilde{a} \boxplus \tilde{b} \notin U$
$\Longrightarrow |\tilde{a} + \tilde{b} - \tilde{a} \boxplus \tilde{b}| \leq \varepsilon^* |\tilde{a} + \tilde{b}| \leq \varepsilon^* \Big(|A + B| + \triangle(a) + \triangle(b) \Big)$.
Dabei wird z.B. verwendet, daß $\tilde{a} \in A + [-\triangle(a), \triangle(a)]$ und damit $|\tilde{a}| \leq |A| + \triangle(a)$ gilt.

II) Fehlerschranke für $|a + b - (\tilde{a} + \tilde{b})|$, d.h. Fehlerschranke bei Verwendung der exakten Operation $+$:
$|a + b - (\tilde{a} + \tilde{b})| \leq |a - \tilde{a}| + |b - \tilde{b}| \leq \triangle(a) + \triangle(b)$

Gesamtfehlerabschätzung:

$$
\begin{aligned}
& |a + b - \tilde{a} \boxplus \tilde{b}| \leq |a + b - (\tilde{a} + \tilde{b})| + |\tilde{a} + \tilde{b} - \tilde{a} \boxplus \tilde{b}| \\
\overset{\text{I),II)}{\leq} \quad & \triangle(a) + \triangle(b) + \begin{cases} \texttt{MinReal}, & \text{Fall I) a)} \\ \varepsilon^* \Big(|A + B| + \triangle(a) + \triangle(b) \Big), & \text{Fall I) b)} \end{cases} \\
\leq \quad & \begin{cases} \triangle(a) + \triangle(b) + \texttt{MinReal}, & \text{Fall I) a)} \\ \varepsilon^*(|A + B|) + (1 + \varepsilon^*)(\triangle(a) + \triangle(b)), & \text{Fall I) b)} \end{cases} \\
\leq \quad & \varepsilon^*(|A + B|) + (1 + \varepsilon^*)(\triangle(a) + \triangle(b)) + \texttt{MinReal} \\
= \quad & \triangle(\text{add}).
\end{aligned}
$$

Die Größen $a, b, \tilde{a}, \tilde{b}$ waren beliebig gewählt, so daß die Behauptung von Satz 1 unmittelbar folgt □

Satz 2 (Subtraktion) Für die Subtraktion gilt die folgende Fehlerschranke $\triangle(\text{sub})$:

$$\triangle(\text{sub}) := \varepsilon^*(|A - B|) + (1 + \varepsilon^*)(\triangle(a) + \triangle(b)) + \texttt{MinReal}.$$

Beweis:
Der Beweis wird analog zum Beweis von Satz 1 geführt.

Satz 3 (Multiplikation) Für die Multiplikation $\circ := \bullet$ gilt

$$\bigwedge_{\substack{a \in A \\ b \in B}} \quad \bigwedge_{\substack{|a - \tilde{a}| \leq \triangle(a) \\ |b - \tilde{b}| \leq \triangle(b)}} \Big| a \cdot b - \tilde{a} \boxdot \tilde{b} \Big| \leq \triangle(\text{mul}) \quad \text{mit}$$

$$\triangle(\text{mul}) := |A||B|\varepsilon^* + (|A| \triangle (b) + |B| \triangle (a) + \triangle(a) \triangle (b)) \cdot (1 + \varepsilon^*) + \texttt{MinReal}$$

Beweis:
Es seien $a \in A, b \in B, \tilde{a} = a + \triangle_a \in A + [- \triangle (a), \triangle(a)], | \triangle_a | \leq \triangle(a)$ und $\tilde{b} = b + \triangle_b \in B + [- \triangle (b), \triangle(b)], | \triangle_b | \leq \triangle(b)$ beliebig aber fest gewählt.

I) Abschätzung für $|\tilde{a} \cdot \tilde{b} - \tilde{a} \boxdot \tilde{b}|$:

a) $\tilde{a} \cdot \tilde{b} \in U \Longrightarrow \tilde{a} \boxdot \tilde{b} \in \overline{U} \overset{\text{Lemma 1}}{\Longrightarrow} |\tilde{a} \cdot \tilde{b} - \tilde{a} \boxdot \tilde{b}| \leq \texttt{MinReal}$

b) $\tilde{a} \cdot \tilde{b} \notin U \Longrightarrow |\tilde{a} \boxdot \tilde{b}| \notin U$
$\Longrightarrow |\tilde{a} \cdot \tilde{b} - \tilde{a} \boxdot \tilde{b}| \leq \varepsilon^* |\tilde{a}\tilde{b}| \leq \varepsilon^*(|A| + \triangle(a))(|B| + \triangle(b))$
$= \varepsilon^*(|A||B| + |A| \triangle (b) + |B| \triangle (a) + \triangle(a) \triangle (b)).$

II) Abschätzung für $|a \cdot b - \tilde{a} \cdot \tilde{b}|$:
$|a \cdot b - \tilde{a} \cdot \tilde{b}| = |a \cdot b - (a + \triangle_a)(b + \triangle_b)| = |a \triangle_b + b \triangle_a + \triangle_a \triangle_b|$
$\leq |A| \triangle (b) + |B| \triangle (a) + \triangle(a) \triangle (b).$

Gesamtfehlerabschätzung:

$$\begin{aligned}
& |a \cdot b - \tilde{a} \boxdot \tilde{b}| \leq |a \cdot b - \tilde{a} \cdot \tilde{b}| + |\tilde{a} \cdot \tilde{b} - \tilde{a} \boxdot \tilde{b}| \\
\overset{\text{I),II)}{\leq} & \begin{cases} |A| \triangle (b) + |B| \triangle (a) + \triangle(a) \triangle (b) + \texttt{MinReal}, & \text{Fall I) a)} \\ (|A| \triangle (b) + |B| \triangle (a) + \triangle(a) \triangle (b))(1 + \varepsilon^*) + |A||B|\varepsilon^*, & \text{Fall I) b)} \end{cases} \\
\leq & |A||B|\varepsilon^* + (|A| \triangle (b) + |B| \triangle (a) + \triangle(a) \triangle (b))(1 + \varepsilon^*) + \texttt{MinReal} \\
= & \triangle(\text{mul}) \ \square
\end{aligned}$$

Lemma 2 (Inverse)

$$\bigwedge_{|\eta| \leq \eta^* < 0.5} \left| \frac{1}{1 + \eta} \right| \leq 1 + \varepsilon(\text{inv})$$

mit $\varepsilon(\text{inv}) := (1 + 2\eta^*) \cdot \eta^*$.

Beweis:
Mit $|\eta| \leq \eta^* < 0.5$ gilt:

$$1 - \eta^* \leq \frac{1}{1 + \eta} \leq 1 + (1 + 2\eta^*) \cdot \eta^*.$$

Daraus folgt $\varepsilon(\text{inv}) = (1 + 2\eta^*) \cdot \eta^* \ \square$

Satz 4 (Division) Für die Division $\circ := /$ gilt

$$\bigwedge_{\substack{a \in A \\ b \in B}} \bigwedge_{\substack{|a - \tilde{a}| \leq \triangle(a) \\ |b - \tilde{b}| \leq \triangle(b)}} \left| a/b - \tilde{a} \boxplus \tilde{b} \right| \leq \triangle(\text{div}) \quad \text{mit}$$

$$\triangle(\text{div}) := \frac{1}{< B > - \triangle (b)} \Big(\triangle (a) + (|A| + \triangle(a)) \cdot (\varepsilon^* + \varepsilon(\text{inv})) \Big) + \texttt{MinReal}$$

und

$$\varepsilon(\text{inv}) := (1 + 2\frac{\triangle(b)}{<B>})\frac{\triangle(b)}{<B>}$$

Dabei wird vorausgesetzt, daß

$$\triangle(b) < \ 0.5 \ \cdot <B> \tag{4}$$

gilt.

Beweis:
Es seien $a \in A, b \in B, \tilde{a} = a + \triangle_a \in A + [-\triangle(a), \triangle(a)], \triangle_a \leq \triangle(a)$ und $\tilde{b} = b + \triangle_b \in B + [-\triangle(b), \triangle(b)], \triangle_b \leq \triangle(b)$ beliebig aber fest gewählt.

I) Abschätzung für $|\tilde{a}/\tilde{b} - \tilde{a} \boxplus \tilde{b}|$:

a) $\tilde{a}/\tilde{b} \in U \Longrightarrow \tilde{a} \boxplus \tilde{b} \in \overline{U} \overset{\text{Lemma 1}}{\Longrightarrow} |\tilde{a}/\tilde{b} - \tilde{a} \boxplus \tilde{b}| \leq \texttt{MinReal}$

b) $\tilde{a}/\tilde{b} \notin U \Longrightarrow |\tilde{a} \boxplus \tilde{b}| \notin U$
$\Longrightarrow |\tilde{a}/\tilde{b} - \tilde{a} \boxplus \tilde{b}| \leq \varepsilon^* |\tilde{a}/\tilde{b}| \leq \varepsilon^*(|A| + \triangle(a)) \cdot \frac{1}{<B> - \triangle(b)}$

Dabei wird z.B. verwendet, daß $\tilde{b} \in B + [-\triangle(b), \triangle(b)]$ und damit $\frac{1}{|\tilde{b}|} \leq \frac{1}{<B> - \triangle(b)}$ gilt.

II) Abschätzung für $|a/b - \tilde{a}/\tilde{b}|$:

$$\begin{aligned}
|a/b - \tilde{a}/\tilde{b}| &= |\frac{a}{b} - \frac{(a + \triangle_a)}{b} \cdot \frac{1}{1 + \frac{\triangle_b}{b}}| \\
&= |\frac{a}{b} - \frac{(a + \triangle_a)}{b} \cdot (1 + \varepsilon_{\text{inv}})| \\
&= \frac{1}{|b|} |\triangle_a + (a + \triangle_a) \cdot \varepsilon_{\text{inv}}| \\
&\overset{\text{mit (4)}}{\leq} \frac{1}{<B>} \Big(\triangle(a) + (|A| + \triangle(a)) \cdot \varepsilon(\text{inv}) \Big)
\end{aligned}$$

mit $|\varepsilon_{\text{inv}}| \leq \varepsilon(\text{inv}) := (1 + 2 \cdot \frac{\triangle(b)}{<B>}) \cdot \frac{\triangle(b)}{<B>}$ nach Lemma 2.

Gesamtfehlerabschätzung:

$$\begin{aligned}
& |a/b - \tilde{a} \boxplus \tilde{b}| \leq |a/b - \tilde{a}/\tilde{b}| + |\tilde{a}/\tilde{b} - \tilde{a} \boxplus \tilde{b}| \\
\overset{\text{I),II)}{\leq} \ & \frac{1}{<B>} \Big(\triangle(a) + (|A| + \triangle(a)) \cdot \varepsilon(\text{inv}) \Big) + \\
& \qquad \begin{cases} \texttt{MinReal}, & \text{Fall I) a)} \\ \varepsilon^*(|A| + \triangle(a)) \cdot \frac{1}{<B> - \triangle(b)}, & \text{Fall I) b)} \end{cases} \\
\leq \ & \frac{1}{<B> - \triangle(b)} \Big(\triangle(a) + (|A| + \triangle(a)) \cdot (\varepsilon^* + \varepsilon(\text{inv})) \Big) + \texttt{MinReal} \\
= \ & \triangle(\text{div}) \ \square
\end{aligned}$$

Die hergeleiteten Fehlerabschätzungen können nun mit Hilfe von Intervallrechnung in Intervallroutinen umgesetzt werden.

Das Modul `eps_ari`
In diesem Modul werden die Sätze 1 bis 4 in ein PASCAL-XSC Programm umgesetzt. Die Quelltexte sind im Anhang A auszugsweise angegeben. Mit den in diesem Modul realisierten Funktionen `DeltaAdd`, `DeltaSub`, `DeltaMul`, `DeltaDiv` kann dann die Fehlerfortpflanzung automatisch erfaßt werden. Z.B. liefert ein Aufruf wie

```
DeltaRes := DeltaAdd( A, B, DeltaA, DeltaB );
```

eine Maschinenzahl `DeltaRes` als Ergebnis, für welche gilt

$$\bigwedge_{\substack{a \in A \\ b \in B}} \bigwedge_{\substack{|a-\tilde{a}| \le \triangle(a) \\ |b-\tilde{b}| \le \triangle(b)}} \left| a+b-\tilde{a} \boxplus \tilde{b} \right| \le \triangle(\mathrm{add}) \le \texttt{DeltaRes}$$

(vergleiche Satz 1). Dabei sind $A, B \in IS$, DeltaA, DeltaB $\in S$. Entsprechendes gilt für die anderen Funktionen.

Ein Programm zur Fehlererfassung bei der Auswertung eines Polynoms $p(x) = \sum_{i=0}^{n} p_i x^i$ mittels des Hornerschemas kann z.B. für $x \in \texttt{X} \in IS, \tilde{x} = x + \triangle_x$ mit $| \triangle_x | \le \triangle(x) \le \texttt{DeltaX} \in S$ wie folgt programmiert werden:

```
H:= p[n]; DeltaH:= diam( H );
for i:= n-1 downto 0 do begin
   DeltaH:= DeltaMul( H, X, DeltaH, DeltaX );
   H:= H * X;
   DeltaH:= DeltaAdd( H, p[i], DeltaH, diam( p[i] ) );
   H:= H + p[i];
end;
PolX:= H;
AbsErr:= DeltaH;
```

Nach Ausführung dieses Programmstücks gilt dann

$$|p(x) - \tilde{p}(\tilde{x})| \le \texttt{AbsErr}$$

für jedes $x \in \texttt{X}$ und jedes $\tilde{x} = x + \triangle_x$ mit $| \triangle_x | \le \triangle(x)$. Weiter gilt $p(x) \in \texttt{PolX}$ für jedes $x \in \texttt{X}$ und jedes $\tilde{p}(\tilde{x}) = (\ldots((\tilde{p}_n \boxdot \tilde{x} \boxplus \tilde{p}_{n-1}) \boxdot \tilde{x} \boxplus \tilde{p}_{n-2}) \boxdot \tilde{x} \boxplus \ldots \boxplus \tilde{p}_1) \boxdot \tilde{x} \boxplus \tilde{p}_0$ mit $\tilde{p}_i \in p[i], i = 0(1)n$.

Numerisches Beispiel
Betrachtet wird das Polynom $p(x) = \sum_{i=0}^{15} p_i x^i := \sum_{i=0}^{15} \frac{1}{i!} x^i$. Als Polynomkoeffizienten p_i werden die engstmöglichen Maschinenintervalleinschließungen verwendet:

```
p[ 0] := [  1.000000000000000E+000,  1.000000000000000E+000 ];
p[ 1] := [  1.000000000000000E+000,  1.000000000000000E+000 ];
p[ 2] := [  5.000000000000000E-001,  5.000000000000000E-001 ];
...
p[13] := [  1.605904383682161E-010,  1.605904383682162E-010 ];
p[14] := [  1.147074559772972E-011,  1.147074559772973E-011 ];
p[15] := [  7.647163731819816E-013,  7.647163731819818E-013 ];
```

Mit dem oben vorgestellten Programmfragment erhält man folgenden Werte:

```
x = 1:
                PolX = [  2.718281828458994E+000,  2.718281828458995E+000 ]
              AbsErr =  6.402573517656651E-016
AbsErr/MinAbs(PolX) =  2.355375167734649E-016

x = -4:
                PolX = [     1.814980943022E-002,     1.814980943024E-002 ]
              AbsErr =  1.313450654637236E-014
AbsErr/MinAbs(PolX) =  7.236718708736003E-013
```

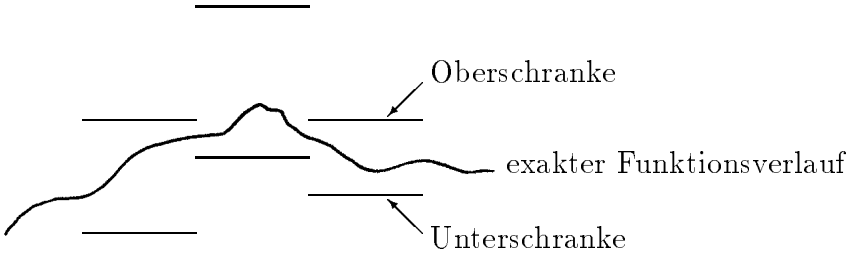
Im Fall $x = 1$ liegt die relative Fehlerschranke nahe der Maschinengenauigkeit (es werden nur positive Größen aufsummiert). Im Fall $x = -4$ ergibt sich aufgrund alternierender Vorzeichen und eines wesentlich kleineren Polynomwertes eine um drei Zehnerpotenzen größere Fehlerschranke.

3 Genauigkeit der Approximation

In diesem Abschnitt wird eine in [15] ausführlich beschriebene Methode verwendet, die es erlaubt, bei gegebenen Approximationskoeffizienten (i.a. ist dies ein Satz von Maschinenzahlen, welcher aus einem Satz von Langzahlwerten gewonnen wird) eine sichere Oberschranke für den absoluten/relativen maximalen Approximationsfehler zu bestimmen.

Zunächst wird mit Hilfe von Langzahlintervallrechnung (siehe [14]) ein Intervallpolynom berechnet, dessen Graphenmenge die tatsächliche Fehlerkurve mit Sicherheit einschließt. In einem zweiten Schritt werden für dieses Intervallpolynom, dessen Intervallkoeffizienten i.a. Maschinenintervalle von wenigen ulp Durchmesser sind, mittels gewöhnlicher Intervallrechnung (z.B. mittels eines Branch-and-Bound-Verfahrens [17]) die betragsgrößten Funktionswerte sicher eingeschlossen. Das Maximum der Oberschranken dieser Einschließungen ist dann eine sichere obere Schranke des Approximationsfehlers. Diese Schranke gilt für **alle** Argumente im vorgegebenen Approximationsintervall.

In [15] ist ebenfalls beschrieben, wie man den Graph des Approximationsfehlers mittels Treppenfunktionen einschließen kann. Die folgenden Abbildungen sind mit diesem Verfahren erzeugt. In jeder Spalte, die einer Bildpunktbreite entspricht, sind zwei Begrenzungspixel gezeichnet. Der wahre Verlauf der Fehlerkurve befindet sich für alle Argumente, die einer solchen Spalte zugeordnet sind, mit Sicherheit zwischen diesen beiden Begrenzungspunkten. Die Treppenstufen sind sehr klein gewählt, was den Eindruck einer stetigen Funktion vermittelt. Das trifft jedoch **nicht** zu. Vielmehr steht jeder gezeichnete Bildpunkt für eine der Spaltenbreite entsprechenden stückweise konstanten Funktion. Die folgende Vergrößerung soll dies noch einmal verdeutlichen:



Im vorliegenden Fall der Implementierung der Funktion $e^x - 1$ sind die beiden Funktionen

$$f(x) := \frac{e^x - 1 - x}{x^2} = \sum_{k=0}^{\infty} \frac{x^k}{(k+2)!}, \quad x \in [-\frac{\ln(2)}{64}, \frac{\ln(2)}{64}] =: \text{Bereich I} \tag{5}$$

$$g(x) := \frac{e^x - 1 - x - \frac{x^2}{2}}{x^3} = \sum_{j=0}^{\infty} \frac{x^j}{(j+3)!}, \quad x \in [\ln(\frac{3}{4}), \ln(\frac{5}{4})] =: \text{Bereich II} \tag{6}$$

polynomial zu approximieren.

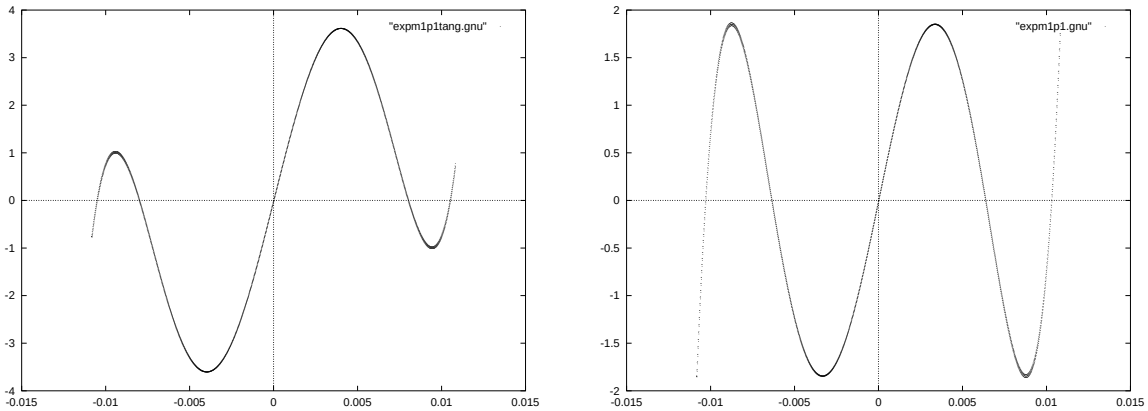


Abbildung 1: Fehlerkurven Bereich I

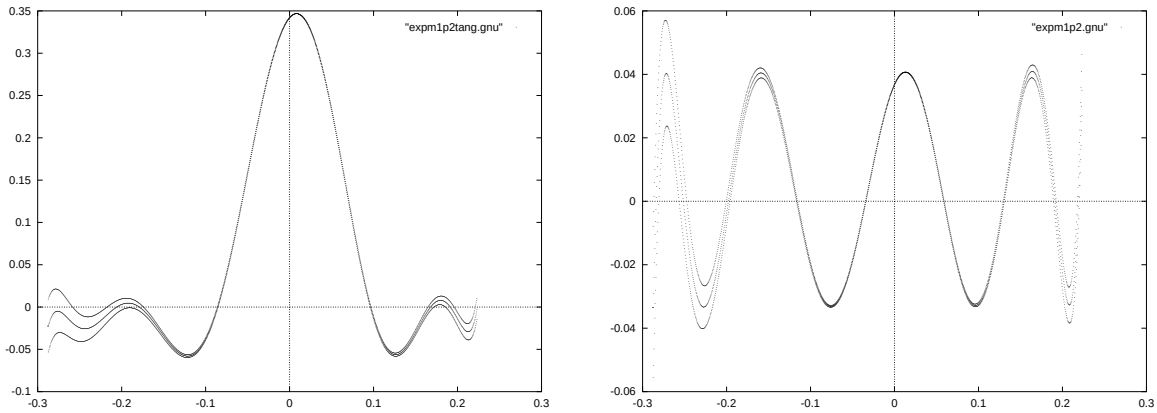


Abbildung 2: Fehlerkurven Bereich II

Die Abbildungen 1 und 2 zeigen jeweils links den Fehlerverlauf der Approximation, wie sie in [20] angegeben ist. Rechts ist der Fehlerverlauf einer mit Hilfe des Computeralgebrapakets Maple [7] berechneten Approximation abgebildet. Der maximale Fehler ist hier deutlich kleiner, so daß diese Approximation in der hier besprochenen Implementierung von $e^x - 1$ Verwendung findet. Man beachte, daß die Funktionswerte mit `PlotScale` $= 10^{15}$ skaliert sind (Programmlisting per FTP erhältlich). Als Schranken für die maximalen Approximationsfehler findet man

```
Maxi:  1.850454976079262E-015
```

im Fall (5) und

```
Maxi:  4.101904694867334E-017
```

im Fall (6). Dabei wird $f(x)$ approximiert durch $\sum_{k=0}^{4} a_k x^k$ mit

	hexadezimal	dezimal
a0	3FE0000000000000	5.000000000000000E-001
a1	3FC555555554DD45	1.666666666658136E-001
a2	3FA555555554B94D	4.166666666638950E-002
a3	3F811114F8A77AAA	8.333362425159880E-003
a4	3F56C1718E0F9DDC	1.388893979532449E-003

und $g(x)$ durch $\sum_{j=0}^{8} b_j x^j$ mit

	hexadezimal	dezimal
b0	3FC5555555555554	1.666666666666666E-001
b1	3FA5555555555503	4.166666666666610E-002
b2	3F81111111113FE1	8.333333333354122E-003
b3	3F56C16C16CA7FF7	1.388888889017890E-003
b4	3F2A01A0159D7CFF	1.984126964158297E-004
b5	3EFA019F817DAFAE	2.480157863209126E-005
b6	3EC71E05122BF5CB	2.755792722352050E-006
b7	3E928240725839F5	2.758025508816736E-007
b8	3E5A496317DE7DCF	2.448136759253856E-008

Im späteren Programm werden die hier hexadezimal angegebenen Koeffizienten verwendet.

Die mittleren Kurven in Abbildung 2 ergeben sich, wenn man zusätzlich zu den Begrenzungsbildpunkten noch einen Funktionswert der Fehlerkurve an einer zufällig gewählten Stelle in jedem einer Spalte entsprechenden Abszissenintervall einzeichnet. Die hier berechneten Oberschranken für die absoluten Approximationsfehler werden in Abschnitt 6 benötigt (vgl. auch Programmlisting im Anhang C).

4 Tabellenverfahren für $e^x - 1$

Es wird im folgenden ein sehr schnelles und gleichzeitig hochgenaues Tabellenverfahren zur Berechnung der Funktion $e^x - 1$ für $x \in S(2, 53, -1022, 1023)$ vorgestellt (siehe auch [20]).

4.1 Idee

Die gesamte Funktionsberechnung erfolgt im Zieldatenformat (IEEE double, 64 Bit), d.h. für Zwischenrechnungen wird kein höhergenaues Datenformat benutzt. Die Berechnung erfolgt wie üblich in drei Schritten:

I) Argumentreduktion

II) Polynomapproximation im reduzierten Bereich

III) Ergebnisanpassung, dabei Verwendung von tabellierten Konstanten

Im Intervall $[\ln(1-\frac{1}{4}), \ln(1+\frac{1}{4})]$ erfolgt die Berechnung mit Hilfe einer zweiten Polynomapproximation. In diesem Fall ist eine abschließende Ergebnisanpassung nicht notwendig. Dies sichert auch im Bereich um die Null (hier ist $e^x - 1 \approx 0$) die gewünschte hohe relative Ergebnisgenauigkeit.

4.2 Verfahren

Ausnahmefälle und Bereichsaufspaltung

Zunächst werden anhand des Eingabearguments x mehrere Sonderfälle abgeprüft. Für x = NAN (not a number) wird eine Fehlerbehandlungsroutine aufgerufen. In dieser Routine kann festgelegt werden, ob das Programm grundsätzlich abgebrochen und eine Fehlermeldung ausgegeben werden soll, oder ob eine Unterscheidung nach „signaling NAN“ bzw. „quiet NAN“ mit unterschiedlicher Fehlerbehandlung erfolgen soll. Für $x >$ `qExpT2c` $= 709.0895657128240$ erfolgt ein Programmabbruch, eine Fehlermeldung[3] wird ausgegeben. Der Fall $x = +\infty$ wird hier miterfaßt und muß nicht getrennt abgeprüft werden. Für $x < -37.42994775023704$ braucht nicht mehr gerechnet zu werden, als Ergebnis wird der Wert -1 zurückgegeben. Der Fall $x = -\infty$ wird hier ebenfalls miterfaßt. Für $x = 0$ sowie für kleine Eingabeargumente $0 \neq |x| < 2^{-54}$ wird x selbst zurückgegeben.

Liegt kein Sonderfall vor, so werden ausgehend vom Argument x die folgenden beiden Fälle unterschieden:

Bereich I: $\quad x \leq \ln(1-\frac{1}{4}) \quad$ oder $\quad \ln(1+\frac{1}{4}) \leq x$

Bereich II: $\quad \ln(1-\frac{1}{4}) < x < \ln(1+\frac{1}{4})$

Nur im Bereich I wird ein Tabellenverfahren verwendet.

Bereich I $\quad (x \leq \ln(1-\frac{1}{4})$ oder $\ln(1+\frac{1}{4}) \leq x)$

Ausgehend vom Argument x wird ein reduziertes Argument r mit $r \in [-\ln(2)/64, \ln(2)/64]$ berechnet. Dazu wird $m \in \mathbb{Z}$ und $j \in \{0, 1, \ldots, 31\}$ so gewählt, daß in der Darstellung

$$x = (32m + j)\ln(2)/32 + r \tag{7}$$

[3] Für $x > 709.78271289338399$ liegt das Ergebnis $e^x - 1$ im Überlaufbereich. Für $709.0895657128240 < x \leq 709.78271289338399$ ist für eine Zwischenrechnung eine zusätzliche Fallunterscheidung nötig, aus Gründen der Laufzeit wird darauf verzichtet.

der Rest $|r| \leq \ln(2)/64$ ist.

Um r zu berechnen wird also vom Argument x ein geeignetes ganzzahliges Vielfaches des Wertes $\ln(2)/32$ subtrahiert. Da es hierbei zu Auslöschung kommen kann, wird die Näherung des Wertes $\ln(2)/32$ im IEEE-Datenformat als Summe (die Addition wird nicht explizit ausgeführt) von 2 Maschinenzahlen $l_{\text{lead}}, l_{\text{trail}} \in S(2,53)$ mit $l_{\text{lead}} + l_{\text{trail}} \approx \ln(2)/32$ höhergenau dargestellt. l_{lead} wird so gewählt, daß die letzten 20 binären Mantissenziffern den Wert 0 haben. Dadurch kann l_{lead} mit einer ganzen Zahl $n, |n| < 2^{20}$ rundungsfehlerfrei multipliziert werden. Das reduzierte Argument r selbst wird mit einigen zusätzlichen Mantissenziffern berechnet, die Darstellung erfolgt ebenfalls als Summe zweier Maschinenzahlen $r_{\text{lead}}, r_{\text{trail}} \in S(2,53)$.

Mit Hilfe einer Polynomapproximation der Form

$$p(r) = r + a_0 r^2 + a_1 r^3 + \ldots + a_4 r^6 = r + r^2(a_0 + \ldots + a_4 r^4)$$

wird eine Näherung für $(e^r - 1)$ berechnet.

Die Koeffizienten $a_i, i = 0, 1, \ldots, 4$ wurden mit Hilfe eines Remez-Algorithmus gefunden (siehe Abschnitt 3). Die Berechnung des Polynoms $r^2(a_0 + a_1 r + \ldots + a_4 r^4)$ erfolgt in der Genauigkeit des IEEE-Datenformats, zur abschließenden Addition von r bei der Berechnung von $p(r)$ wird jedoch die höhergenaue Darstellung $r_{\text{lead}} + r_{\text{trail}}$ benutzt:

$$\begin{aligned} r &:= r_{\text{lead}} \boxplus r_{\text{trail}} \\ q &:= r \boxdot r \boxdot (a_0 \boxplus r \boxdot (a_2 \boxplus \ldots)\ldots) \\ p &:= r_{\text{lead}} \boxplus (r_{\text{trail}} \boxplus q) \qquad \text{(Klammerung beachten!)} \end{aligned}$$

Die Ergebnisanpassung erfolgt durch Anwendung der Gleichung

$$e^x - 1 = 2^m(2^{j/32} + 2^{j/32}(e^r - 1)) - 1, \tag{8}$$

wobei m und j die gemäß (7) bestimmten und bekannten Größen sind. Die benötigte Potenz $2^{j/32}$, $j \in \{0, 1, 2, \ldots, 31\}$ wird im voraus berechnet. Näherungen für $2^{j/32}$, $j = 0(1)31$ werden vorab in jeweils 2 Konstanten $\text{lead}(j)$, $\text{trail}(j)$ auf ungefähr 100 Mantissenbits in einer Tabelle abgespeichert. Die Summe $\text{lead}(j)+\text{trail}(j)$ dieser beiden Konstanten ergibt also eine Näherung für $2^{j/32}$ mit zusätzlichen Ziffern. Um den Fehler bei der Berechnung der rechten Seite von (8) auf dem Rechner möglichst klein zu halten, werden in Abhängigkeit von m die folgenden drei Ausdrücke verwendet:

$$\begin{aligned} m \geq 53: &\quad 2^m[\text{lead}(j) \boxplus (s \boxdot p \boxplus (\text{trail}(j) \boxminus 2^{-m}))] \\ m \leq -8: &\quad 2^m[\text{lead}(j) \boxplus (s \boxdot p \boxplus \text{trail}(j))] \boxminus 1 \\ \text{Sonst:} &\quad 2^m[(\text{lead}(j) \boxminus 2^{-m}) \boxplus (\text{lead}(j) \boxdot p \boxplus \text{trail}(j) \boxdot (1 \boxplus p))] \\ \text{jeweils mit} &\quad s := \text{lead}(j) \boxplus \text{trail}(j) \approx 2^{j/32} \end{aligned}$$

Bereich II $(\ln(1-\frac{1}{4}) < x < \ln(1+\frac{1}{4}))$

Ausgehend von der Reihenentwicklung

$$e^x - 1 = x + \frac{x^2}{2} + \sum_{j=3}^{\infty} \frac{x^j}{j} =: x + \frac{x^2}{2} + x \cdot x^2 \cdot g(x)$$

wird eine polynomiale Bestapproximation $p(x)$ für $g(x)$ verwendet. Mit dieser in Abschnitt 3 angegebenen Bestapproximation wird zunächst $q := x \boxdot x^2 \boxdot p(x)$ berechnet. Dies kann in normaler IEEE-Genauigkeit geschehen. Zur Addition des Terms $x + \frac{x^2}{2}$ muß eine höhere Genauigkeit simuliert werden. Dazu wird die Mantisse von x in zwei Teile u, v aufgespalten, u enthält die führenden 24 Bits der Mantisse, v ergibt sich durch $v := x - u$. Sowohl u als auch v sind auf dem Rechner exakt darstellbar.

Damit wird $y := u \cdot u \cdot \frac{1}{2}$ und $z := v \boxdot (x \boxplus u) \cdot \frac{1}{2}$ berechnet. Da von u nur die ersten 24 Bit ungleich Null sind, ist $u \cdot u$ mit maximal 48 Bit ebenfalls exakt darstellbar, woraus sich schließlich y durch eine einfache und ebenfalls rundungsfehlerfreie Exponentenanpassung ergibt. Die Summe $y + z$ ist dann eine höhergenaue Näherung für $\frac{x^2}{2}$. Dies wird durch die folgende Handrechnung (vgl. [13]) gezeigt:
Es sei o.B.d.A.: $x > 0$. Mit $x = 0.x_1x_2 \ldots x_{53} \cdot b^{\mathrm{ex}}$, $u = x|_{24} = 0.x_1x_2 \ldots x_{24} \cdot b^{\mathrm{ex}}$ und

$$x - x|_{24} = 0.\underbrace{00\ldots0}_{24\text{ Stück}} x_{25}x_{26}\ldots x_{53} \cdot b^{\mathrm{ex}} = v < 2^{-24}x$$

erhält man:

$$\begin{aligned}
y + z &= \frac{1}{2}u^2 + \frac{1}{2}v \boxdot (x \boxplus u) \\
&= \frac{1}{2}((x|_{24})^2 + (x - x|_{24}) \boxdot (x \boxplus x|_{24})) \\
&= \frac{1}{2}((x|_{24})^2 + (x - x|_{24})(x \boxplus x|_{24})(1+\varepsilon)) \\
&= \frac{1}{2}((x|_{24})^2 + (x - x|_{24})(x + x|_{24})(1+\varepsilon)^2) \\
&= \frac{1}{2}((x|_{24})^2 + (x^2 - (x|_{24})^2)(1+2\varepsilon)) \\
&= \frac{1}{2}(x^2 + 2\varepsilon((x|_{24} - v)^2 - (x|_{24})^2)) \\
&= \frac{1}{2}(x^2 + 2\varepsilon(2 \cdot v \cdot x|_{24} + v^2)) \\
&= \frac{1}{2}(x^2 + 2\varepsilon 2^{-22}x) \\
&= \frac{x^2}{2} + \varepsilon(2^{-22}x)
\end{aligned}$$

$$\Longrightarrow \left|\frac{x^2}{2} - (y+z)\right| \leq \varepsilon \cdot 2^{-22}x$$

Wegen $x \in [-0.29, 0.244]$ und somit $|x| \in [0, 0.29]$ ergibt sich die folgende Abschätzung:

$$|\frac{x^2}{2} - (y+z)| \leq \varepsilon \cdot 2^{-22}|x| < \varepsilon \cdot 2^{-23}$$

Die Summe $y+z$ ist damit eine Näherung für $\frac{x^2}{2}$ mit 22 zusätzlichen Mantissenstellen. Im Fehlerkalkül wird diese Handrechnung automatisch mit erfaßt (siehe vollständiges Programmlisting `ffexpm.p`).

Zum Schluß müssen die Werte x, y, z und q in Abhängigkeit ihrer Größe geschickt aufaddiert werden. Dazu werden die folgenden beiden Fälle unterschieden:

$$e^x - 1 \approx \begin{cases} (u \boxplus y) \boxplus (q \boxplus (v \boxplus z)), & y \geq 2^{-7}, \\ x \boxplus (y \boxplus (q \boxplus z)), & y < 2^{-7}\ . \end{cases}$$

4.3 Algorithmus

Der Gesamtalgorithmus `expm1` zur Berechnung von $e^x - 1$ stellt sich wie folgt dar:

I. Abprüfen von Spezialfälle und deren Behandlung

- x ist NaN (not a number) $\Longrightarrow$ Fehlermeldung: Invalid Argument
- $x >$`qExpT2a`$= 709.7827\ldots \Longrightarrow$ Fehlermeldung: Overflow
  Der Fall $x = +\infty$ wird mit abgedeckt!
- $x = 0 \Longrightarrow$ res $:= x = \pm 0$, d.h. der exakte Funktionswert $e^0 - 1 = 0$ wird zurückgegeben[4].
- $|x| <$`qExpT1`$= 2^{-54} = 5.5511\ldots \cdot 10^{-17} \Longrightarrow \mathrm{res}(x) := x$
- $x <$`qExpT3`$= -37.4299\ldots \Longrightarrow \mathrm{res}(x) := -1$
  $x = -\infty$ muß damit nicht gesondert behandelt werden.

II. Fallunterscheidung

1.) Bereich I: $x \leq \ln(1 - \frac{1}{4})$ oder $\ln(1 + \frac{1}{4}) \leq x$

(a) Reduktion (red. Argument r wird berechnet als $r_{\mathrm{lead}} + r_{\mathrm{trail}}$)

$$\begin{aligned} n &:= \mathrm{round}(x \boxdot \texttt{qExpInvL}), \quad \text{mit } \texttt{qExpInvL} \approx \frac{32}{\ln 2} \\ n_2 &:= n \bmod 32 \\ n_1 &:= n - n_2 \end{aligned}$$

$$\begin{aligned} r_{\mathrm{lead}} &:= (x \boxminus n \boxdot l_{\mathrm{lead}}) \qquad \text{mit } l_{\mathrm{lead}} + l_{\mathrm{trail}} \approx \frac{\ln(2)}{32} \\ r_{\mathrm{trail}} &:= -n \boxdot l_{\mathrm{trail}} \quad \{\Longrightarrow r_{\mathrm{lead}} + r_{\mathrm{trail}} = x - n \cdot (l_{\mathrm{lead}} + l_{\mathrm{trail}})\} \end{aligned}$$

$$\begin{aligned} m &:= n_1/32 \\ j &:= n_2 \end{aligned}$$

[4]Im IEEE-Standard wird ± 0 unterschieden.

(b) Approximation $p(r) = r + a_0 r^2 + a_1 r^3 + \ldots + a_4 r^6$,
$r \in [-0.01083\ldots, 0.01083\ldots]$

$$\begin{aligned} r &:= r_{\text{lead}} \boxplus r_{\text{trail}} \\ q &:= r \boxdot r \boxdot (a_0 \boxplus r \boxdot (a_1 \boxplus r \boxdot (a_2 \boxplus r \boxdot (a_3 \boxplus r \boxdot a_4)))) \\ p &:= r_{\text{lead}} \boxplus (r_{\text{trail}} \boxplus q) \end{aligned}$$

(c) Ergebnisanpassung
Falls $m \geq 53$:
$\text{res} := 2^m[\text{lead}(j) \boxplus (s \boxdot p \boxplus (\text{trail}(j) \boxminus 2^{-m}))]$
Falls $m \leq -8$:
$\text{res} := 2^m[\text{lead}(j) \boxplus (s \boxdot p \boxplus \text{trail}(j))] \boxminus 1$
Falls $-7 \leq m \leq 52$:
$\text{res} := 2^m[(\text{lead}(j) \boxminus 2^{-m}) \boxplus (\text{lead}(j) \boxdot p \boxplus \text{trail}(j) \boxdot (1 \boxplus p))]$
mit $s := \text{lead}(j) \boxplus \text{trail}(j) \approx 2^{j/32}$ { auf 100 Bit }

2.) Bereich II: $\ln(1 - \frac{1}{4}) < x < \ln(1 + \frac{1}{4})$

(a) Genaue Berechnung von $x^2/2$

$$\begin{aligned} u &:= \text{CUT24}(x), \qquad \{ \text{ die ersten 24 Bits von } x \} \\ v &:= x - u \\ y &:= u \cdot u \cdot 0.5 \\ z &:= v \boxdot (x \boxplus u) \cdot 0.5 \end{aligned}$$

$$\{ \Longrightarrow y + z \approx \frac{x^2}{2} \text{ mit zusätzlichen Bits } \}$$

(b) Approximation (Horner-Schema)
$q := x^3 p(x) = x^3(b_0 + b_1 x^1 + \ldots + b_8 x^8), x \in [-0.28768\ldots, 0.22314\ldots]$

$$q := x \boxdot x \boxdot x \boxdot (b_0 \boxplus x \boxdot (b_1 \boxplus x \boxdot (\ldots \boxplus x \boxdot b_8)\ldots))$$

(c) Genaue Addition von $x + y + z + q$ durch Fallunterscheidung:
Falls $y \geq 2^{-7}$:
$\text{res} := (u \boxplus y) \boxplus (q \boxplus (v \boxplus z))$
Falls $y < 2^{-7}$:
$\text{res} := x \boxplus (y \boxplus (q \boxplus z))$

Nun gilt `expm1` := res $\approx (e^x - 1)$.
Die Fehlerabschätzung für den eben beschriebene Algorithmus wird in Abschnitt 6 durchgeführt.

5 Implementierung von $e^x - 1$ für Punktargumente

Die Implementierung des in Abschnitt 4 vorgestellten Tabellenverfahrens erfolgte in ANSI-C [1, 9]. Dadurch wird die Übertragbarkeit auf die verschiedensten Rechner und Plattformen sichergestellt.

Es wurde Wert darauf gelegt, ein bezüglich der Laufzeit möglichst schnelles Verfahren zu erhalten. Dies hatte Auswirkungen auf den Programmierstil. So wurden z.B. Polynomauswertungen mit Hilfe des Hornerschemas immer explizit ausprogrammiert, anstatt der Verwendung einer Wiederholungsanweisung. Multiplikationen mit einer Potenz von 2 werden grundsätzlich über eine Änderung des Exponenten im Bitmuster der im Speicher abgelegten Zahl realisiert. Dadurch zu berücksichtigende Maschinenabhängigkeiten werden durch Preprozessoranweisungen berücksichtigt.

Um eine Übernahme in bestehende Compiler bzw. Programmpakete (z.B. PASCAL-XSC [10, 11], C-XSC [12]) zu ermöglichen, wurde ein modularer Aufbau gewählt. Es wurden die folgenden Dateien erstellt:

Dateiname	**Bedeutung, Verwendung**
`q_defs.h`	Headerfile, enthält globale Variablen- und Funktionsvereinbarungen, Konstanten zur Steuerung der bedingten Übersetzung, versch. Makros, z.B. zur Multiplikation mit einer Zweierpotenz
`q_glbl.c`	enthält alle globalen Variablen und Konstanten, sowie die Tabellenwerte
`q_errm.c`	Fehlerbehandlungsroutinen
`q_expm.c`	Funktion $e^x - 1$ als Tabellenverfahren
`q_test.c`	Kleines Testprogramm für $e^x - 1$ in ANSI-C

Tabelle 1: ANSI-C Quellprogramme

Die in diesen Dateien enthaltenen Funktionen und Programme sind im Anhang auszugsweise abgedruckt. Die verwendeten Funktions- und Variablennamen sind mit ihrer jeweiligen Bedeutung in Tabelle 4 (siehe Anhang B) zusammengestellt.

Die Tabellenwerte wurden mit dem Langzahlmodul `mp_ari` vom PASCAL-XSC berechnet. Jeder Tabellenwert wird als Summe zweier IEEE-Zahlen gespeichert. Wichtig ist dabei, daß mindestens die 6 letzten Mantissenbits der ersten IEEE-Zahl den Wert Null haben. Die Darstellung der Werte erfolgt als Bruch in der Form Zähler/Nenner, wobei der Nenner grundsätzlich eine Zweierpotenz ist. Dieser Bruch kann auf einem Rechner mit IEEE-Datenformat rundungsfehlerfrei berechnet werden. Dies wird bereits vom Compiler beim Übersetzen des Programmcodes erledigt. Der Vorteil dieser Darstellung liegt sowohl in der Maschinenunabhängigkeit als auch in der Verwendbarkeit derart dargestellter Konstanten in verschiedenen Programmiersprachen.

6 Fehlerabschätzung

Im folgenden wird eine relative Fehlerschranke für die Berechnung von $e^x - 1$ nach dem im Abschnitt 4 vorgestellten Verfahren hergeleitet. Diese Fehlerschranke soll für alle zulässigen Eingabeargumente aus dem Bereich der normalisierten IEEE-Zahlen gelten.

Die Fehlerabschätzung wird größtenteils auf dem Rechner mit PASCAL-XSC unter Verwendung des Moduls `eps_ari` (siehe Abschnitt 2) durchgeführt. Das zu-

gehörige Programm `ffexpm1.p` ist im Anhang C abgedruckt. Die Fehlerabschätzung wird unter der Annahme durchgeführt, daß das exakte reelle Ergebnis einer Grundoperation $(+, -, \bullet, /)$ auf dem Rechner zu einer benachbarten Gleitkommazahl gerundet wird. Der maximale relative Fehler einer Gleitkommaoperation ist demnach `eps52`$= 2^{-52}$. Der IEEE-Standard schreibt sogar einen Rundungsmodus zur nächstgelegenen Gleitkommazahl vor. Um nicht durch Umschalten des Rundungsmodus Laufzeiteinbußen zu verursachen, werden keine speziellen Annahmen über den eingestellten Rundungsmodus bei Programmaufruf vorausgesetzt. Dies bedeutet, daß die für alle Rundungsmodi gültige Fehlerschranke `eps52` verwendet werden muß.

Erzwingt man bei Programmausführung die Rundung zur nächstgelegenen Gleitkommazahl, so darf man die in dieser Arbeit angegebene Gesamtfehlerschranke (1) auf nahezu die Hälfte $(1.302 \cdot 10^{-16})$ reduzieren. Diese Aussage kann sehr einfach bewiesen werden, indem man in den Fehlerabschätzungsprogrammen überall die Größe `eps52` durch `eps53`$= 2^{-53}$ ersetzt!

Die im Algorithmus verwendeten Konstanten- und Tabellenwerte sind jeweils die zur nächstgelegenen Gleitkommazahl gerundeten exakten Werte, d.h. ihr maximaler relativer Fehler beträgt `eps53`.

Bei der automatische Fehlerabschätzung wird analog zum Algorithmus zwischen Bereich I und Bereich II unterschieden. Für beide Bereiche wird jeweils ein eigenes Unterprogramm erstellt:

Unterprogrammname	Abschätzung für Argument $x \in I$ mit
`fehler1`	$I = [-37.44077..., -0.27076] \cup [0.223143..., 709.79...]$
`fehler2`	$I = [-0.287682..., -5.55...e-17] \cup [5.55...e-17, 0.223144...]$

Tabelle 2: Teilbereiche mit unterschiedlichen Fehlerabschätzung

Die Funktion `fehler1` berechnet eine Oberschranke des maximalen relativen Gesamtfehlers bei der Auswertung von $e^x - 1$ für ein Argument x aus einem vorgegebenen Intervall $I^* = [a, b]$ mit $I^* \subset I$. In diesem Gesamtfehler sind alle Rundungsfehler, der Approximationsfehler und die Konstantenfehler berücksichtigt. Das Intervall I^* wird durch die Parameter `kk_lb`, `kk_ub`, `k_lb` und `k_ub` festgelegt. Diese Parameter werden beim Aufruf der Funktion `fehler1` übergeben. Es gilt:

$$\begin{aligned} I^* \quad := \quad & [((\ \texttt{kk\_lb} \cdot 32 - 0.5) + \texttt{k\_lb}\ ) \bigtriangledown \tfrac{\bigtriangledown(\ln 2)}{32}, \\ & ((\ \texttt{kk\_ub} \cdot 32 - 0.5) + \texttt{k\_ub} + 1) \bigtriangleup \tfrac{\bigtriangleup(\ln 2)}{32}] \end{aligned}$$

Da bei der Ergebnisanpassung 32 verschiedene Konstanten verwendet werden, müssen bei der Fehlerabschätzung 32 Teilfälle betrachtet werden. Dies bedeutet, daß im folgenden die Teilintervalle

$$\begin{aligned} I_{kk,k} \quad := \quad & [((kk \cdot 32 - 0.5) + k) \bigtriangledown \tfrac{\bigtriangledown(\ln 2)}{32}, \\ & ((kk \cdot 32 - 0.5) + k + 1) \bigtriangleup \tfrac{\bigtriangleup(\ln 2)}{32}] \end{aligned}$$

mit $k =$`k_lb`(1)`k_ub`, $kk =$`kk_lb`(1)`kk_ub` untersucht werden. Diese Intervalle $I_{kk,k}$ sind so gewählt, daß für alle $x \in I_{kk,k}$ im Algorithmus die gleiche Fallunterscheidung ausgewählt wird. Bei der Realisierung im Programm muß beachtet werden, das die

folgende Bedingung gilt:

$$I^* \subseteq \bigcup_{kk} \bigcup_{k} I_{kk,k}$$

für $k =$`k_lb(1)k_ub` und $kk =$`kk_lb(1)kk_ub`. Aufgrund der Überschätzung durch die Intervallarithmetik müssen die Teilintervalle $I_{kk,k}$ eventuell noch weiter unterteilt werden, um zu brauchbaren, scharfen Fehleraussagen zu kommen. Die Funktion `fehler1` wird vom Hauptprogramm aus mit 5 verschiedenen Parametersätzen aufgerufen:

`kk_lb`	`kk_ub`	`k_lb`	`k_ub`	Intervall $I^* = [a^*, b^*]$	Anzahl Unterteilung von $I_{kk,k}$
-54	-2	0	31	$[-37.44077..., -0.703977...]$	1
-1	-1	0	19	$[-0.703977..., -0.27076...]$ [5]	200
0	0	10	31	$[0.223143..., 0.6823...]$ [6]	500
1	1023	0	31	$[0.6823..., 709.77...]$	1
1024	1024	0	0	$[709.77..., 709.79...]$	1

In der Funktion `fehler1` werden für jedes Teilintervall alle Berechnungen des Algorithmus mit Hilfe der Intervallarithmetik von PASCAL-XSC nachvollzogen. Der bei normaler Gleitkommarechnung entstehende Rundungsfehler wird als absoluter Fehler mit Hilfe der Funktionen des Moduls `eps_ari` erfaßt. Die Konstantenfehler werden ebenfalls als absolute Fehler erfaßt, der absolute Approximationsfehler wird an geeigneter Stelle aufaddiert. Zum Schluß wird, sofern möglich, der maximale relative Gesamtfehler von allen betrachteten Teilintervallen bestimmt und an das Hauptprogramm übergeben.

Mit Hilfe der Funktion `fehler2` wird der Fehler für Argumente x mit $x \in [\ln(1-\frac{1}{4}), -2^{-54}] \cup [\ln(1+\frac{1}{4}), 2^{-54}]$ bestimmt. Um sowohl die Vorzeichen als auch die Fallunterscheidung bei der Summation des Ergebnisses von $e^x - 1$ richtig zu erfassen, müssen hier 4 Teilbereiche getrennt untersucht werden. Diese Teilbereiche können Tabelle 3 entnommen werden.

Unterprogrammname	Teilbereich	Abschätzung für Argument $x \in I$ mit
`fehler2`	a	$I = [-0.287682..., -0.125]$
	b	$I = [-0.125, -5.55...e-17]$
	c	$I = [5.55...e-17, 0.125]$
	d	$I = [0.125, 0.223144...]$

Tabelle 3: Teilbereiche des Bereichs II

Zur Bestimmung von scharfen Fehlerschranken ist es auch hier wieder nötig, die zu untersuchenden Teilbereiche in kleine Teilintervalle zu unterteilen. Eine Unterteilung in 50 bzw. 20 Teilintervalle hat sich als ausreichend erwiesen.

Analog zur Funktion `fehler1` wird auch hier wieder der Algorithmus nachvollzogen, sämtliche Fehler werden mit Hilfe der Funktionen des Moduls `eps_ari` erfaßt.

Die Ausführung des Hauptprogramms liefert das folgende Ergebnis:

[5] b^* wird explizit gesetzt.

[6] a^* wird explizit gesetzt.

```
Bereich Ia:  Max. rel. Fehler =  2.340427167524938E-016
Bereich Ib:  Max. rel. Fehler =  2.443828079287490E-016
Bereich IIa: Max. rel. Fehler =  2.489959685937339E-016
Bereich IIb: Max. rel. Fehler =  2.476426814206591E-016
Bereich IIc: Max. rel. Fehler =  2.458824919974750E-016
Bereich IId: Max. rel. Fehler =  2.359601267404139E-016
Bereich Ic:  Max. rel. Fehler =  2.592561649228397E-016
Bereich Id:  Max. rel. Fehler =  2.468390169928818E-016
Bereich Ie:  Max. rel. Fehler =  2.321294617584316E-016
```

Als Gesamtfehlerschranke aller bisher untersuchten Teilbereiche ergibt sich somit

```
Max. rel. Fehlerschranke der expm1-Funktion: 2.592561649228397E-016
```

Jetzt muß noch gezeigt werden, daß der relative Fehler in den bisher noch nicht untersuchten Teilintervallen (siehe Tabelle 2) kleiner als diese mit dem Programm berechnete Fehlerschranke ist. Die Schranke gilt dann für alle Eingabeargumente aus dem Bereich der normalisierten Gleitkommazahlen.

Einfache Handrechnung ergibt[7]:

- $x < -37.4298...$
  Es ist $e^{-37.4298...} = 5.5519\ldots \cdot 10^{-17}$ und aufgrund der Monotonie der Exponentialfunktion gilt $e^x \leq 5.5519\ldots \cdot 10^{-17}$ für alle $x < -37.4298\ldots$
  $\Longrightarrow (e^x - 1) \in [-1, -1 + \varepsilon^*]$ für $x < -37.4298\ldots$, $\varepsilon^* := 2^{-53}$
  Als Ergebnis für die Punktfunktion wird der Wert -1 zurückgeliefert, für den relativen Fehler gilt damit:

$$|\frac{(e^x - 1) - (-1)}{e^x - 1}| = |\frac{e^x}{e^x - 1}| \leq 5.6 \cdot 10^{-17}$$

- $x \in [-2^{-54}, 0)$
  Es gilt $x < 0$:

$$|\frac{(e^x - 1) - x}{e^x - 1}| \leq \frac{\frac{x^2}{2}(\frac{1}{1-x})}{-x} \leq -\frac{x}{2} \leq 2^{-54} = 5.5511... \cdot 10^{-17}$$

- $x = 0$
  Es gilt $(e^x - 1) = 0$, das Ergebnis wird direkt gesetzt und ist damit exakt.

- $x \in (0, 2^{-54}]$
  Es gilt $x > 0$:

$$|\frac{(e^x - 1) - x}{e^x - 1}| \leq \frac{\frac{x^2}{2}(\frac{1}{1-x})}{e^x - 1} \leq x^2 \frac{1}{x} = x \leq 2^{-54} = 5.5511... \cdot 10^{-17}$$

[7] Um zu verdeutlichen, daß hier die Funktion $e^x - 1$ untersucht wird, wird dieser Ausdruck geklammert.

Damit ist gezeigt, daß

$$|\frac{(e^x-1)-\texttt{expm1}(x)}{e^x-1}| \leq 2.592561649228397 \cdot 10^{16} =: \varepsilon(\texttt{expm1})$$

für alle zulässigen normalisierten Eingabeargumente zutrifft.

Die gefundene relative Fehlerschranke gilt nicht für Argumente aus dem Bereich der denormalisierten Zahlen. Es kann aber zumindest eine Aussage über den absoluten Fehler gemacht werden. Ausgehend von der Reihenentwicklung

$$(e^x-1) = \sum_{j+1}^{\infty} \frac{x^j}{j!} = x + \frac{x^2}{2} + \frac{x^3}{6} + \ldots$$

sind die folgenden beiden Aussagen für $x \in [-\texttt{q\_minr},\texttt{q\_minr}]$ (mit $\texttt{q\_minr} := 2.225073... \cdot 10^{-308}$, kleinste positive normalisierte Zahl) sofort einsichtig:

$$(e^x-1) \geq x \tag{9}$$

$$|(e^x-1)-x| \leq \frac{x^2}{2}(\frac{1}{1-x}) \leq 2.475... \cdot 10^{-616} \tag{10}$$

Der Abstand d zweier benachbarter denormalisierter Zahlen ist immer gleich und beträgt $d := 2^{-1022-52} = 4.9406... \cdot 10^{-324}$. Zusammen mit (9), (10) folgt, daß für die Einschließung des Funktionswertes $x \in [-\texttt{q\_minr},\texttt{q\_minr}]$ gilt[8]:

$$(e^x-1) \in [x, \mathrm{succ}(x)] \; .$$

Der absolute Fehler ist kleiner als der Abstand zweier benachbarter denormalisierter Zahlen.

7 Intervallfunktion $e^X - 1$

Ausgehend von der im Abschnitt 5 angegebenen Implementierung der Punktfunktion $e^x - 1$ wird unter Verwendung der in Abschnitt 6 hergeleiteten Gesamtfehlerschranke die Intervallfunktion $e^X - 1$ konstruiert.

Da die Funktion $f(x) := e^x - 1$ im gesamten Definitionsbereich monoton wachsend ist, kann die Intervallfunktion $f(X) := e^X - 1$ für $X = [a,b] \in I\!I\!R$, durch Funktionsauswertungen an den Intervallgrenzen a und b berechnet werden. Es gilt:

$$f(X) = e^X - 1 = e^{[a,b]} - 1 = \{e^x - 1 | a \leq x \leq b\} = [e^a - 1, e^b - 1].$$

Bei der Rechnung auf der Maschine muß der relative Gesamtfehler berücksichtigt werden. Der folgende Satz gibt darüber Auskunft:

Satz 5 (Einschließung des exakten Funktionswertes) Sei f der exakte Wert, $\tilde{f}$ der fehlerbehaftete Wert mit dem relativen Fehler ε_f, $|\varepsilon_f| \leq \varepsilon(f)$, ε^* sei die Maschinengenauigkeit. Dann gilt für

$$\tilde{f} > 0: \quad \tilde{f} \boxdot (1 \bigtriangledown \varepsilon(f)) \bigtriangledown (1 \bigtriangledown \varepsilon^*) \leq f \leq \tilde{f} \boxdot (1 \bigtriangleup \varepsilon(f)) \bigtriangleup (1 \bigtriangleup \varepsilon^*)$$

$$\tilde{f} < 0: \quad \tilde{f} \boxdot (1 \bigtriangleup \varepsilon(f)) \bigtriangleup (1 \bigtriangleup \varepsilon^*) \leq f \leq \tilde{f} \boxdot (1 \bigtriangledown \varepsilon(f)) \bigtriangledown (1 \bigtriangledown \varepsilon^*)$$

[8] Mit $\mathrm{succ}(x)$ wird der Gleitkommanachfolger der Gleitkommazahl x bezeichnet.

Beweis:
Fall $\tilde{f} > 0$:

$$\begin{aligned} & \tilde{f} \boxdot (1 \bigtriangledown \varepsilon(f)) \bigtriangledown (1 \bigtriangledown \varepsilon^*) \leq \tilde{f} \boxdot (1 - \varepsilon(f)) \cdot (1 - \varepsilon^*) \\ \leq\ & \tilde{f} \cdot (1 - \varepsilon(f)) \leq f \leq \tilde{f} \cdot (1 + \varepsilon(f)) \\ \leq\ & \tilde{f} \boxdot (1 + \varepsilon(f)) \cdot (1 + \varepsilon^*) \leq \tilde{f} \boxdot (1 \bigtriangleup \varepsilon(f)) \bigtriangleup (1 \bigtriangleup \varepsilon^*) \end{aligned}$$

Der Fall $\tilde{f} < 0$ wird analog gezeigt $\square$

Die Konstanten `q_exmm`:= $(1 \bigtriangledown \varepsilon(f)) \bigtriangledown (1 \bigtriangledown \varepsilon^*)$ und `q_exmp`:= $(1 \bigtriangleup \varepsilon(f)) \bigtriangleup (1 \bigtriangleup \varepsilon^*)$ müssen mit gerichtet gerundeten Operationen berechnet werden. Dies kann z.B. im voraus mit PASCAL-XSC geschehen. Bei der Multiplikation $\tilde{f} \boxdot$ `q_exmm` bzw. $\tilde{f} \boxdot$ `q_exmp` wird nur vorausgesetzt, daß die Rundung zu einer benachbarten Gleitkommazahl erfolgt. Die Ausführung dieser Multiplikation kann deshalb auch in ANSI-C ohne zeitaufwendige Umschaltung des Rundungsmodus implementiert werden.

Da der relative Gesamtfehler für die Funktion $e^x - 1$ nur im Bereich der normalisierten Zahlen gilt, wird im Bereich der denormalisierten Zahlen die Einschließung $(e^x - 1) \in [x, \mathrm{succ}(x)]$ (siehe Abschnitt 6) verwendet. Die benötigten Fallunterscheidungen können der Implementierung `j_expm.c` entnommen werden.

8 Zusammenfassung

Sichere und gleichzeitig genaue Fehlerabschätzungen im Zusammenhang mit der Entwicklung von Software für mathematische Funktionen sind in der Regel für eine Handrechnung zu aufwendig. Insbesondere benötigt man (nahezu) scharfe Schranken, so daß Vergröberungen bei den Abschätzungen vermieden werden müssen.

Der vorgestellte (noch erweiterbare) Fehlerkalkül kann im Zusammenwirken mit Intervallrechnung verwendet werden, verläßliche worst-case-Fehlerabschätzungen weitgehend automatisch durchzuführen. Es muß ein Programm entwickelt werden, welches den zu realisierenden Algorithmus genau nachbildet (mit Fallunterscheidungen, mit denselben Konstanten, ...). Dabei wird für alle Operationen der Maximalfehler durch die korrespondierenden Intervallroutinen des Fehlerkalküls erfaßt. Um genaue Fehlerschranken zu erhalten müssen die einzelnen Teilbereiche so lange weiter unterteilt werden, bis die durch Intervallrechnung auftretenden Wertebereichsüberschätzungen hinreichend klein bleiben.

Die hier besprochene Anwendung des Kalküls auf ein Tabellenverfahren für $e^x - 1$ ergibt eine sehr gute Fehlerschranke, die nahe an der Maschinengenauigkeit liegt und dies, obwohl aus Laufzeitüberlegungen keine interne höhergenaue Arithmetik benutzt wird. Für weitere Anwendungen, z.B. auf die Verfahren [6, 18, 19], trifft dies ebenfalls zu.

Insgesamt erlaubt der Kalkül auch Realisierungen von Funktionen zu behandeln, bei denen sehr viele Fallunterscheidungen auftreten. Bei Handrechnung müßte hier jeder Fall gesondert abgeschätzt werden.

Die Entwicklung eines geeigneten Algorithmus zur Funktionsberechnung bleibt nach wie vor eine eigenständige Aufgabe. Jedoch kann die hier vorgestellt Methode

angewendet werden, um heuristische Überlegungen abzusichern oder aber zu verwerfen.

Auch kann man kritische Stellen von Algorithmen offenlegen, bei denen die Simulation einer höhergenauen Arithmetik eine deutliche Verbesserung der Genauigkeit des Endergebnisses liefert.

A Modul eps_ari

Im folgenden ist das in Abschnitt 2 näher beschriebene PASCAL-XSC-Modul `eps_ari` auszugsweise abgedruckt.

```
MODULE eps_ari;
{------------------------------------------------------------------------}
{        F e h l e r s c h r a n k e n a r i t h m e t i k             }
{------------------------------------------------------------------------}

USE i_ari;  { Intervallarithmetik einbinden }

{------------------------------------------------------------------------}
{              Einige globale Groessen                                   }
{------------------------------------------------------------------------}

GLOBAL CONST Eps53 = 1.110224E-16;
{ Maschinenepsilon: IEEE RoundToNearest, 2**(-53)= 1.110223...E-16 }

GLOBAL CONST Eps52 = 2.220447E-16;
{ Maschinenepsilon: IEEE RoundToDown/Up, 2**(-52)= 2.220446...E-16 }

GLOBAL CONST MinReal = 2.2250738585072013E-308;
{ Kleinste positive normalisierte Gleitkommazahl }

VAR UnflowRange: interval;
{ Bereich des gradual Underflows und Underflows }

GLOBAL VAR EpsAriTest: boolean; { Unterlaufwarnungen ein/ausschalten }


...   { Im vollstaendigen Listing kommen hier Hilfsfunktionen! }


{------------------------------------------------------------------------}
{        F e h l e r s c h r a n k e n a r i t h m e t i k             }
{                      fuer +, -, *, /.                                  }
{ Absolute Fehlerschranken der beiden Operanden sind bekannt.            }
{ Die exakten Werte der Operanden liegen in den Intervallen              }
{ alpha und beta.                                                        }
{------------------------------------------------------------------------}


GLOBAL FUNCTION DeltaAdd(
               alpha, beta: interval; DeltaA, DeltaB: real ): real;
  {-------------------------------------------------------------}
  { Berechnet wird die absolute Fehlerschranke bei einer         }
  {   Addition von fehlerbehafteten Groessen.                    }
  {-------------------------------------------------------------}
  { Die exakten Werte des 1. Summanden liegen im Intervall alpha.}
  { Der absolute Fehler der fehlerbehafteten Werte ist durch     }
  { DeltaA beschraenkt.                                          }
  { Die exakten Werte des 2. Summanden liegen im Intervall beta. }
```

```
  { Der absolute Fehler der fehlerbehafteten Werte ist durch     }
  { DeltaB beschraenkt.                                          }
  {--------------------------------------------------------------}
VAR u, v: real;
    ResultSet: interval;
BEGIN
  IF (DeltaA=0) AND (DeltaB=0) AND ((alpha=0) OR (beta=0)) THEN
    DeltaAdd:= 0
  ELSE BEGIN
    IF EpsAriTest THEN BEGIN  { Unterlaufwarnung }
      ResultSet:= alpha + intval(-DeltaA, DeltaA)
                        + beta + intval(-DeltaB, DeltaB);
      IF NOT (ResultSet >< UnflowRange) THEN BEGIN
        write('  DeltaAdd: Ergebnis im Unterlauf!  Weiter mit <Return> ');
        readln;
      END;
    END; { Unterlaufwarnung }
    u:= Eps52 *> MaxAbs(alpha+beta);
    v:= (DeltaA +> DeltaB) *> (1 +> Eps52);
    DeltaAdd:= u +> v +> MinReal;
  END;
END;

GLOBAL FUNCTION DeltaMul(
                alpha, beta: interval; DeltaA, DeltaB: real ): real;
  {--------------------------------------------------------------}
  { Berechnet wird die absolute Fehlerschranke bei einer         }
  {   Multiplikation von fehlerbehafteten Groessen.              }
  {--------------------------------------------------------------}
  { Die exakten Werte des 1. Faktors liegen im Intervall alpha.  }
  { Der absolute Fehler der fehlerbehafteten Werte ist durch     }
  { DeltaA beschraenkt.                                          }
  { Die exakten Werte des 2. Faktors liegen im Intervall beta.   }
  { Der absolute Fehler der fehlerbehafteten Werte ist durch     }
  { DeltaB beschraenkt.                                          }
  {--------------------------------------------------------------}

VAR u, v: real;
VAR ResultSet: interval;
BEGIN
  IF (alpha=1) AND (DeltaA=0) THEN
    DeltaMul:= DeltaB
  ELSE IF (beta=1) AND (DeltaB=0) THEN
    DeltaMul:= DeltaA
  ELSE BEGIN
    u:= MaxAbs(alpha) *> DeltaB + MaxAbs(beta) *> DeltaA;
    v:= (1 +> Eps52) *> (u +> DeltaA *> DeltaB);
    DeltaMul:=  Eps52 *> MaxAbs(alpha*beta) +> v +> MinReal;
    IF EpsAriTest THEN BEGIN  { Unterlaufwarnung }
      ResultSet:= ( alpha + intval(-DeltaA,DeltaA) )
                    * ( beta + intval(-DeltaB,DeltaB) );
      IF NOT (ResultSet >< UnflowRange) THEN BEGIN
        write('  DeltaMul: Ergebnis im Unterlauf!  Weiter mit <Return> ');
        readln;
      END;
    END;  { Unterlaufwarnung }
  END;
END;


...   { Im vollstaendigen Listing kommen hier weitere Funktionen }
```

```
{-------------------------------------------------------------------}

BEGIN { Initialisierungen }
  UnflowRange:= intval(-MinReal, MinReal);
  EpsAriTest := false;
END.
{-------------------------------------------------------------------}
```

B Implementierung der Funktion expm1 als Tabellenverfahren

ANSI-C	PASCAL-XSC	Bedeutung
`q_p1ex`		Funktion für Argumentbereich 1 der Funktion $e^x - 1$
`q_p2ex`		Funktion für Argumentbereich 2 der Funktion $e^x - 1$
`q_expm`		Funktion $e^x - 1$ für Punktargumente
`j_expm`		Funktion $e^X - 1$ für Intervallargumente
`q_minr`		$2.225073...e - 308$, kleinste normalisierte Zahl
`q_p2h`		2^{100} zur Ergebnisberechnung in Sonderfällen
`q_p2mh`		2^{-100} zur Ergebnisberechnung in Sonderfällen
`q_ext1`	`qExpT1`	$5.5511...e - 017$, zur Fallunterscheidung
`q_ext2c`	`qExpT2c`	$7.0908...e + 002$, zur Fallunterscheidung
`q_ext3`	`qExpT3`	$-3.7429...e + 001$, zur Fallunterscheidung
`q_ext4`	`qExpT4`	$-2.8768...e - 001$, zur Fallunterscheidung
`q_ext5`	`qExpT5`	$2.2314...e - 001$, zur Fallunterscheidung
`q_exil`	`qExpInvL`	$4.6166...e + 001 \approx 32/\ln 2$
`q_exl1`	`qExpL1`	$2.1660...e - 002 = L1$
`q_exl2`	`qExpL2`	$2.3251...e - 012 = L2$, mit $L1 + L2 \approx \ln 2/32$
`q_exa`	`qExpA`	5 Polynomkoeffizienten für Approximation
`q_exb`	`qExpB`	9 Polynomkoeffizienten für Approximation
`q_exld`	`qExpLead`	32 Tabellenwerte lead(j)
`q_extl`	`qExpTrail`	32 Tabellenwerte trail(j) mit lead(j)+trail(j) $\approx 2^{j/32}$
`q_exme`		relative Fehlerkonstante für Funktion `q_expm`
`q_exmm`	`qExpEm1`	Fehlerkonstante für Intervallfunktion
`q_exmp`	`qExpEp1`	Fehlerkonstante für Intervallfunktion

Tabelle 4: Tabelle der Konstanten- und Funktionsnamen

In Tabelle 4 findet man zu wichtigen Funktions- und Konstantennamen einen Hinweis auf deren Bedeutung. Bei Konstanten sind die C-Namen und deren PASCAL-XSC Äquivalente zeilenweise aufgelistet. Aus Portabilitätsgründen wurden die C-Namen auf maximal 6 Zeichen beschränkt. Um mögliche Konflikte mit bereits bestehen Namen zu vermeiden, wird hier nicht der Name `expm1`, sondern `q_expm` verwendet.

Generell wird in [8] die folgende Namenskonvention verwendet: `q_` steht für Punktroutinen, `j_` für Intervallversionen.

Auszüge aus den wichtigsten ANSI-C Quellprogrammen (siehe Tabelle 1 in Abschnitt 5) sind hier aufgelistet. In ihrer Gesamtheit geben sie das realisierte genaue Tabellenverfahren wieder.

```
/*****************************************************************/
/*                                                               */
/*      Filename        : q_glbl.c                               */
/*                                                               */
/*      Description     : Constants for fast real functions      */
/*                        for double IEEE Floating-Point         */
/*                        Arithmetic                             */
/*                                                               */
/*****************************************************************/


/* ---- Globale Variablen fuer schnelle Standardfunktionen  ------------ */

   double q_minr = 2.2250738585072013e-308;   /* kl. normalisierte Zahl  */

/* ---- Globale Variablen fuer Tabellenwerte von q_expm1 und q_exp ----- */


/* qMinExp:= ln ( 2.22507...e-308 )  */
   double q_mine = -708.396418532264106261 7539;
/* qExpT1:= 5.551115123125783E-017 */
   double q_ext1= 4503599627370496.0 / 81129638414606681695789005144064.0;
/* qExpT2:= 1.809114141261457E+003 */
   double q_ext2= 7956568137163861.0 / 4398046511104.0;
/* qExpT2a:= 7.097827128933840E+002 */
   double q_ex2a= 6243314768165359.0 / 8796093022208.0;
/* qExpT2c:= 7.090895657128240E+002 */
   double q_ex2c= 6237217781087072.0 / 8796093022208.0;
/* qExpT3:=-3.742994775023704E+001 */
   double q_ext3=-5267796835639521.0 / 140737488355328.0;
/* qExpT4:=-2.876820724517810E-001 */
   double q_ext4=-5182419497180051.0 / 18014398509481984.0;
/* qExpT5:= 2.231435513142098E-001 */
   double q_ext5= 8039593716390434.0 / 36028797018963968.0;

/* qExpInvL:= 4.616624130844683E+001 */
   double q_exil=  6497320848556798.0 / 140737488355328.0;
/* qExpL1:= 2.166084939017310E-002 */
   double q_exl1=  6243314767495168.0 / 288230376151711744.0;
/* qExpL2:= 2.325192846878874E-012 */
   double q_exl2=  5756898648422637.0 / 2475880078570760549798248448.0;

   double q_exa[5]={      /* Polynomkoeffizienten Bereich I */
              4503599627370496.0 /           9007199254740992.0,
              6004799503129925.0 /          36028797018963968.0,
              6004799503120717.0 /         144115188075855872.0,
              4803856372824746.0 /         576460752303423488.0,
              6405142946487772.0 /        4611686018427387904.0 };

   double q_exb[9]={      /* Polynomkoeffizienten Bereich II */
```

```
              6004799503160660.0 /          36028797018963968.0,
              6004799503160579.0 /         144115188075855872.0,
              4803839602540513.0 /         576460752303423488.0,
              6405119470632951.0 /        4611686018427387904.0,
              7320136463514879.0 /       36893488147419103232.0,
              7320133978402734.0 /      295147905179352825856.0,
              6506931592885707.0 /     2361183241434822606848.0,
              5209762888694261.0 /    18889465931478580854784.0,
              7399039345524175.0 /   302231454903657293676544.0};

   double q_exld[32]={     /* Tabellenwerte 2^(j/32), j=0,1,...,31 */
/* qExpLead [ 0]:= 1.000000000000000E+000 */
                  4503599627370496.0 /                 4503599627370496.0,
/* qExpLead [ 1]:= 1.021897148654105E+000 */
                  4602215617889600.0 /                 4503599627370496.0,
/* qExpLead [ 2]:= 1.044273782427410E+000 */
                  4702991017412864.0 /                 4503599627370496.0,
/* qExpLead [ 3]:= 1.067140400676820E+000 */
                  4805973110840128.0 /                 4503599627370496.0,

...                       ...                                  ...

/* qExpLead [30]:= 1.915206561397142E+000 */
                  8625323556245696.0 /                 4503599627370496.0,
/* qExpLead [31]:= 1.957144124175400E+000 */
                  8814193548346688.0 /                 4503599627370496.0 };

   double q_extl[32]={     /* Tabellenwerte 2^(j/32), j=0,1,...,31 */
/* qExpTrail[ 0]:= 0.000000000000000E+000 */
                  0.0,
/* qExpTrail[ 1]:= 1.159741170639136E-014 */
                  7350732955350452.0 /   633825300114114700748351602688.0,
/* qExpTrail[ 2]:= 3.416187970930849E-015 */
                  8661065463685896.0 /  2535301200456458802993406410752.0,
/* qExpTrail[ 3]:= 3.695759744057116E-015 */
                  4684932057853331.0 /  1267650600228229401496703205376.0,

...                       ...                                  ...

/* qExpTrail[30]:= 5.666960267488855E-015 */
                  7183725584551774.0 /  1267650600228229401496703205376.0,
/* qExpTrail[31]:= 8.960767791036668E-017 */
                  7269838508040587.0 / 811296384146066816957890051440640
};


/* double q_exme=2.592561649228397E-016; */
/* qExpEm1:=(1-<qExpEps)*<(1-<eps52)     */
/* qExpEm1:= 9.999999999999993E-001      */
double q_exmm =  9007199254740986.0 /   9007199254740992.0;
/* qExpEp1:=(1+>qExpEps)*>(1+>eps52)     */
/* qExpEp1:= 1.000000000000001E+000      */
double q_exmp =  4503599627370501.0 /   4503599627370496.0;
```

```
/****************************************************************/
/*                                                              */
/*      Filename        : q_expm.c                              */
/*                                                              */
/*      Description     : Fast real (exponential function - 1)  */
/*                        for double IEEE Floating-Point        */
/*                        Arithmetic                            */
/*                                                              */
/****************************************************************/

#include "q_defs.h"
double q_p1ex(double x);
double q_p2ex(double x);

/* ------------------------------------------------------------------- */
/* - Berechnung der Funktion exp(x)-1, Implementierung als Tab.fahren  - */
/* ------------------------------------------------------------------- */

/* ------   Prozedur fuer Bereich I   ---------------------------------- */
double q_p1ex(double x)
  { int j;
    long int n,n1,m;
    double r,r1,r2,p,q,s;
    double res;

    /* Schritt 1 */
    if (x>0) n=CUTINT((x*q_exil)+0.5);
    else     n=CUTINT((x*q_exil)-0.5);     /* round (x)            */
    j=n % 32;                              /* n2=n mod 32          */
    if (j<0) j+=32;                        /* Es muss gelten n2>=0 */
    n1=n-j;
    r1=x-n*q_exl1;
    r2=-(n*q_exl2);
    m=n1/32;

    /* Schritt 2 */
    r=r1+r2;
    q=(((q_exa[4]*r+q_exa[3])*r+q_exa[2])*r+q_exa[1])*r+q_exa[0];
    q=r*r*q;
    p=r1+(r2+q);

    /* Schritt 3 */
    s=q_exld[j]+q_extl[j];
      if (m>=53) {
          res=1.0;
          POWER2(res,-m);
          res=(q_exld[j]+(s*p+(q_extl[j]-res)));
          POWER2(res,m);
      } else {
          if (m<=-8) {
             res=(q_exld[j]+(s*p+q_extl[j]));
             POWER2(res,m);
             res-=1;
          } else {
```

```
            res=1.0;
            POWER2(res,-m);
            res=((q_exld[j]-res)+(q_exld[j]*p+q_extl[j]*(1+p)));
            POWER2(res,m);
          }
      }
    return(res);
  }

/* ------   Prozedur fuer Bereich II   ---------------------------------- */
double q_p2ex(double x)
  { double  u,v,y,z,q;
    int i;

    /* Schritt 1 */
    u=(double) (CUT24(x));
    v=x-u;
    y=u*u*0.5;
    z=v*(x+u)*0.5;

    /* Schritt 2 */
    q=(((((((q_exb[8]*x+q_exb[7])*x+q_exb[6])*x+q_exb[5])
             *x+q_exb[4])*x+q_exb[3])*x+q_exb[2])*x+q_exb[1])*x+q_exb[0];
    q=x*x*x*q;

    /* Schritt 3 */
    if (y>=7.8125e-3)         /* = 2^-7 */
       return ((u+y)+(q+(v+z)) );
    else
       return (x+(y+(q+z)) );
  }

/* ------   Hauptprogramm mit Fallunterscheidungen   ------------------- */

double q_expm(double x)
{ double res;

  if NANTEST(x)                                        /* Test: x=NaN */
     res=q_abortnan(INV_ARG,&x);
  else {
     if ((-q_ext1<x) && (x<q_ext1)) res=x;
     else {
        if (q_ex2c<x)
           q_abortr1(OVER_FLOW,&x);  /* Ueberlauf */
        else {
           if (x<q_ext3) res=-1.0;
           else {
              if ((q_ext4<x) && (x<q_ext5)) res=q_p2ex(x);
              else res=q_p1ex(x); }
        }
     }
  }
  return(res);
}
```

C Programm zur Gesamtfehlerabschätzung

Die Fehlerabschätzung des in Anhang B angegebenen Tabellenverfahrens wurde mit dem folgenden PASCAL-XSC-Programm durchgeführt. Die Ergebnisse sind in Abschnitt 6 aufgelistet. Die Namenszuordnung kann Tabelle 4 entnommen werden.

```
{------------------------------------------------------------------------}
{- Fehlerabschaetzung fuer expm1-Funktion, Tabellenverfahren            -}
{- Version: getrennte Rechnung fuer alle 32 Tabellenwerte,              -}
{-          Rechnung mit absoluten Fehlern !!!                          -}
{------------------------------------------------------------------------}

PROGRAM ffexpm1(input,output);
USE eps_ari,                            { Fehlerschrankenarithmetik }
    i_ari;                              { Intervallarithmetik       }

{-globale Variablen-}
VAR qExpInvL,qExpL1,qExpL2:real;        { Tabellenwerte der exp-Funktion }
    qExpA:ARRAY [0..4] OF real;
    qExpB:ARRAY [0..8] OF real;
    qExpLead,
    qExpTrail:ARRAY[0..31] OF real;

{-Variablen Hauptprogramm -}
VAR relfehler,maxrelfehler:real;

{------------------------------------------------------------------------}
{-                      Initialisierungsteil                            -}
{------------------------------------------------------------------------}
PROCEDURE init;
BEGIN
{----- Tabellenwerte der exp-Funktion ---------------------------------}
  qExpInvL:=       6497320848556798.0 /           140737488355328.0;
  qExpL1:=       6243314767495168.0 /          288230376151711744.0;
  qExpL2:=       5756898648422637.0 / 24758800785707605497982484480.0;

  qExpA[0]:=  4503599627370496.0 /          9007199254740992.0;
  qExpA[1]:=  6004799503129925.0 /         36028797018963968.0;
  qExpA[2]:=  6004799503120717.0 /        144115188075855872.0;
  qExpA[3]:=  4803856372824746.0 /        576460752303423488.0;
  qExpA[4]:=  6405142946487772.0 /       4611686018427387904.0;

  qExpB[0]:=            6004799503160660.0 /         36028797018963968.0;
  qExpB[1]:=            6004799503160579.0 /        144115188075855872.0;
  qExpB[2]:=            4803839602540513.0 /        576460752303423488.0;
  qExpB[3]:=            6405119470632951.0 /       4611686018427387904.0;
  qExpB[4]:=            7320136463514879.0 /      36893488147419103232.0;
  qExpB[5]:=            7320133978402734.0 /     295147905179352825856.0;
  qExpB[6]:=            6506931592885707.0 /    2361183241434822606848.0;
  qExpB[7]:=            5209762888694261.0 /   18889465931478580854784.0;
  qExpB[8]:=            7399039345524175.0 /  302231454903657293676544.0;


   qExpLead [ 0]:=  4503599627370496.0 /                4503599627370496.0;
   qExpLead [ 1]:=  4602215617889600.0 /                4503599627370496.0;
   qExpLead [ 2]:=  4702991017412864.0 /                4503599627370496.0;
   qExpLead [ 3]:=  4805973110840128.0 /                4503599627370496.0;

   ...                      ...                                 ...

   qExpLead [30]:=  8625323556245696.0 /                4503599627370496.0;
   qExpLead [31]:=  8814193548346688.0 /                4503599627370496.0;
```

```
   qExpTrail[ 0]:=  0.0;
   qExpTrail[ 1]:=  7350732955350452.0 /   6338253001141147007483516026880.0;
   qExpTrail[ 2]:=  8661065463685896.0 /  25353012004564588029934064107520.0;
   qExpTrail[ 3]:=  4684932057853331.0 /  12676506002282294014967032053760.0;

   ...                          ...                                    ...

   qExpTrail[30]:=  7183725584551774.0 /  12676506002282294014967032053760.0;
   qExpTrail[31]:=  7269838508040587.0 / 81129638414606681695789005144064.0;
END;


{------------------------------------------------------------------------}
{-  Fehlerabschaetzung fuer Argument aus dem Bereich I                  -}
{------------------------------------------------------------------------}
FUNCTION fehler1(kk_lb,kk_ub,k_lb,k_ub,jmax:integer):real;
VAR d1,d2,d3,d4,d5,d6,d6a:real;{ abs. Fehler der Zwischenergebnisse    }
    dmax:real;                 { max. abs. Fehler der expm1-Funktion   }
    e1,                        { rel. Fehler einer Funktionsauswertung }
    emax:real;                 { max. rel. Fehler der expm1-Funktion   }

    n,n2,n1,m,j:integer;       { Var. zur Arg.red. der expm1-Funktion  }
    r1,r2:interval;

    arg:interval;              { Var. fuer die 32 Fallunterscheidungen }
    kkonst:interval;
    harg:real;

    maxjj,
    maxk,maxkk,maxn,maxm:integer;{ Fall mit max. rel. Fehler speichern }
    maxd,maxe:real;
    maxarg:interval;

    i,                          { Laufvariable Hornerschema            }
    k,                          { Laufvariable 32 Fallunterscheidungen }
    kk:integer;                 { Laufvariable gesamter Wertebereich   }
    jj:integer;                 { fuer Teilintervalle                  }
    jmin,jdiff:real;

    h,                          { Hilfsgroesse fuer Zwischenrechnungen }
    x,
    r,                          { Var. der exp-Fkt. fuer red. Argument }
    q,q1:interval;              { Var. der exp-Fkt. fuer Hornerschema  }


BEGIN
  {--- Initialisierung der Schleifenvariablen ---}
  dmax:=0; emax:=0;
  kkonst:=ln(intval(2.0))/32.0;
  maxk:=0; maxkk:=0; maxn:=0; maxm:=0; maxjj:=0;

  {--- Durchlaufen des gesamten Wertebereichs der Funktion ---}
  FOR kk:=kk_lb TO kk_ub DO BEGIN
                         { kk:=-54..-2, -1..-1, 0..0, 1..1023, 1024..1024 }
    harg:=kk*32-0.5;     { Hilfsargument fuer die 32 Fallunterscheidungen }

    {--- Durchlaufen der 32 Fallunterscheidungen ---}
    FOR k:=k_lb TO k_ub DO     {k:=0..31}
      FOR jj:=0 TO jmax-1 DO BEGIN

        { intval( (harg+k)  *<kkonst.inf , (harg+k+1)*>kkonst.sup ) }
        { wird unterteilt in jmax Intervalle                        }
```

```
jmin :=(harg+k)  *<kkonst.inf;
jdiff:=( (harg+k+1)*>kkonst.sup - (harg+k)*<kkonst.inf ) /jmax;

arg:=intval( jmin + jj*jdiff , jmin + (jj+1)*jdiff );

IF (jj=jmax-1) THEN arg.sup:= (harg+k+1)*>kkonst.sup;
IF ((kk=0) AND (k=10)) THEN BEGIN
  IF (arg.inf<0.223143) THEN arg.inf:=0.223143;
  IF arg.sup<arg.inf THEN arg.sup:=arg.inf;
END;
IF ((kk=-1) AND (k>=19)) THEN BEGIN
  IF (arg.sup>-0.287682072451781) THEN arg.sup:=-0.287682072451781;
  IF arg.inf>arg.sup THEN arg.inf:=arg.sup;
END;

{ Integer-Groessen n,n1,n2,m,j werden bestimmen                }
n:=round((mid(arg)*qExpInvL));          {  round ()            }
n2:=n MOD 32;                           {  n2=n mod 32         }
IF (n2<0) THEN n2:=n2+32;               {  Es muss gelten n2>=0 }
n1:=n-n2;
m:=n1 DIV 32;                           {  Ganzzahlige Div.    }
j:=n2;

{--- Argumentreduktion: r2:=-n*L2  ---}
h:=qExpL2*intval(1-<Eps53,1+>Eps53);
d1:=DeltaMul(intval(n),h,0.0,rel2abs(h,Eps53));
r2:=-n*h;

{--- alle anderen Berechnungen der Argumentreduktion sind exakt ---}
r1:=arg-n*qExpL1;

{--- Red. Argument als eine IEEE-Zahl:  r:=r1+r2 ---}
d2:=DeltaAdd(r1,r2,0.0,d1);
r:=r1+r2;

{--- Hornerschema: Polynomauswertung ---}
q:=qExpA[4]*intval(1-<Eps53,1+>Eps53);
d3:=rel2abs(q,Eps53);     { abs. Fehler von q }
FOR i:=3 DOWNTO 0 DO BEGIN
  d3:=DeltaMul(r,q,d2,d3);
  q:=r*q;
  h:=qExpA[i]*intval(1-<Eps53,1+>Eps53);  { h Hilfsvariable }
  d3:=DeltaAdd(h,q,rel2abs(h,Eps53),d3);
  q:=h+q;
END;

{--- absoluten Approximationsfehler aufaddieren ---}
d3:=d3+1.86e-15;

{--- Multiplikation mit r*r ---}
d4:=DeltaMul(r,r,d2,d2);
d4:=DeltaMul(r*r,q,d4,d3);
q:=r*r*q;

{--- Red. Argument hoehergenau aufaddieren:  p=r1+(r2+q) ---}
d5:=DeltaAdd(r2,q,d1,d4);
q:=r2+q;
d5:=DeltaAdd(r1,q,0.0,d5);
q:=r1+q;

{--- Ergebnisanpassung mit Fallunterscheidungen ---}
IF m>=53 THEN BEGIN
```

```
  h:=qExpTrail[j]*intval(1-<Eps53,1+>Eps53);
  d6:=DeltaAdd(intval(qExpLead[j]),h,0.0,rel2abs(h,Eps53));
  h:=qExpLead[j]+h;
  d6:=DeltaMul(h,q,d6,d5);
  q:=h*q;

  h:=qExpTrail[j]*intval(1-<Eps53,1+>Eps53);
  d6a:=DeltaAdd(h,intval(-power(2.0,-m)),rel2abs(h,Eps53),0);
  h:=h-power(2.0,-m);
  d6:=DeltaAdd(q,h,d6,d6a);
  q:=q+h;

  h:=qExpLead[j];
  d6:=DeltaAdd(h,q,0.0,d6);
  q:=q+qExpLead[j];

  { Multiplikation mit 2↑m, kein zusaetlicher Fehler }

END ELSE IF m<=-8 THEN BEGIN

  h:=qExpTrail[j]*intval(1-<Eps53,1+>Eps53);
  d6:=DeltaAdd(intval(qExpLead[j]),h,0.0,rel2abs(h,Eps53));
  h:=qExpLead[j]+h;
  d6:=DeltaMul(h,q,d6,d5);
  q:=h*q;

  h:=qExpTrail[j]*intval(1-<Eps53,1+>Eps53);
  d6:=DeltaAdd(h,q,rel2abs(h,Eps53),d6);
  q:=h+q;

  h:=qExpLead[j];
  d6:=DeltaAdd(h,q,0.0,d6);
  q:=q+qExpLead[j];

  { Multiplikation mit 2↑m, kein zusaetlicher Fehler }
  q:=q*power(2,m);
  d6:=d6*power(2,m);   { abs. Fehler wird mit 2↑m multipliziert }

  d6:=DeltaAdd(q,intval(-1.0),d6,0.0);
  q:=q-1.0;

END ELSE BEGIN

  d6:=DeltaAdd(intval(1.0),q,0.0,d5);
  q1:=1+q;

  h:=qExpTrail[j]*intval(1-<Eps53,1+>Eps53);
  d6:=DeltaMul(h,q1,rel2abs(h,Eps53),d6);
  q1:=h*q1;

  h:=qExpLead[j];
  d6a:=DeltaMul(h,q,0.0,d5);
  q:=h*q;

  d6:=DeltaAdd(q,q1,d6a,d6);
  q:=q+q1;

  d6a:=0;          { h+2↑-m ist exakt }
  q1:=h-power(2,-m);

  d6:=DeltaAdd(q1,q,d6a,d6);
  q:=q1+q;
```

```
        { Multiplikation mit 2↑m, kein zusaetzlicher Fehler }
       END;

      {--- Berechnung des relativen Fehlers ---}
      IF ((q.inf<=0) AND (q.sup>=0)) THEN
        writeln ('Relativer Fehler kann nicht berechnet werden !!!')
      ELSE BEGIN
        IF abs(q.inf)<abs(q.sup) THEN
          e1:=d6/>abs(q.inf)
        ELSE
          e1:=d6/>abs(q.sup);
        IF emax<e1 THEN BEGIN  { Fall mit max. rel. Fehler speichern }
          emax:=e1;
          maxk:=k;
          maxkk:=kk;
          maxjj:=jj;
          maxn:=n;
          maxm:=m;
          maxe:=e1;
          maxd:=d6;
          maxarg:=arg;
        END;
      END;

      IF dmax<d6 THEN dmax:=d6;
    END;
  END;

  fehler1:=maxe;
END;


{-------------------------------------------------------------------------}
{-  Fehlerabschaetzung fuer Argument aus dem Bereich II                  -}
{-------------------------------------------------------------------------}
FUNCTION fehler2(jmax,ubereich:integer):real;
...


{-  Im vollstaendigen Listing folgt hier das Hauptprogramms mit -}
{-  entsprechenden Funktionsaufrufen                            -}
...
```

Die Ausführung dieses Programms liefert die im Abschnitt 6 angegebene Gesamtfehlerschranke.

Literatur

[1] *American National Standard for Information Systems – Programming Language C.* X3.159-1989.

[2] American National Standards Institute / Institute of Electrical and Electronics Engineers: *A Standard for Binary Floating-Point Arithmetic.* ANSI/IEEE Std. 754-1985, New York, 1985 (reprinted in SIGPLAN **22**, 2, pp 9–25, 1987).

[3] Bohlender, D., Ullrich, C.: *Standards zur Computerarithmetik.* In *Wissenschaftliches Rechnen*, Herzberg, J. (Herausgeber), Akademie Verlag, Berlin, 1995.

[4] Braune, K.: *Hochgenaue Standardfunktionen für reelle und komplexe Punkte und Intervalle in beliebigen Gleitpunktrastern.* Dissertation, Universität Karlsruhe, 1987.

[5] Dekker, T.J.: *Floating-Point Technique for Extending the Availible Precision.* Numerische Mathematik 18, S. 224-242, 1971.

[6] Hart, J. F. et al.: *Computer Approximations.* Wiley, New York / London / Sydney, 1968.

[7] Heck, A.: *Introduction to Maple* Springer Verlag, 1993

[8] Hofschuster, W., Krämer, W.: *Eine Realisierung von schnellen und portablen Standardfunktionen für Punkt- und Intervallargumente in ANSI-C*. Bericht des Instituts für Angewandte Mathematik, erscheint Ende 1996.

[9] Kernighan, B. W., Ritchie, D. M.: *Programmieren in C — Mit dem C-Reference Manual in deutscher Sprache*. Zweite Ausgabe ANSI C, Hanser-Verlag, München/Wien, 1990.

[10] Klatte, R., Kulisch, U., Neaga, M., Ratz, D., Ullrich, Ch.: *PASCAL–XSC — Sprachbeschreibung mit Beispielen*. Springer-Verlag, Berlin/Heidelberg/New York, 1991.

[11] Klatte, R., Kulisch, U., Neaga, M., Ratz, D., Ullrich, Ch.: *PASCAL–XSC — Language Reference with Examples*. Springer-Verlag, Berlin/Heidelberg/New York, 1992.

[12] Klatte R., et. al.: *C-XSC, A C++ Class Library for Scientific Computing*, Springer 1993.

[13] Krämer, W.: *Inverse Standardfunktionen für reelle und komplexe Intervallargumente mit a priori Fehlerabschätzungen für beliebige Datenformate*. Dissertation, Universität Karlsruhe, 1987

[14] Krämer, W: *Multiple-Precision Computations with Result Verification*, in: *Scientific Computing with Automatic Result Verification*, Adams, E., Kulisch, U.(editors), Academic Press, pp. 311–343, 1992.

[15] Krämer, W.: *Sichere und genaue Abschätzung des Approximationsfehlers bei rationalen Approximationen*, Bericht des Instituts für Angewandte Mathematik, Universität Karlsruhe, 1996.

[16] Linnainmaa, F.: *Software for Double-Precision Floating-Point Computations.* ACM Trans. on Math. Software, Vol. **7**, No. 3, pp 272-282, 1981.

[17] Ratz, D.: *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen.* Dissertation, Universität Karlsruhe, 1992.

[18] Tang, P. T. P.: *Table-Driven Implementation of the Exponential Function in IEEE Floating-Point Arithmetic*. ACM Trans. on Math. Software, Vol. **15**, No. 2, pp 144–157, 1989.

[19] Tang, P. T. P.: *Table-Driven Implementation of the Logarithm Function in IEEE Floating-Point Arithmetic*. ACM Trans. on Math. Software, Vol. **16**, No. 4, pp 378–400, 1990.

[20] Tang, P. T. P.: *Table-Driven Implementation of the Expm1 Function in IEEE Floating-Point Arithmetic*. ACM Trans. on Math. Software, Vol. **18**, No. 2, pp 211–222, 1992.

In dieser Reihe sind bisher die folgenden Arbeiten erschienen:

Nr. 93/1: G. Aumann, K. Bentz: Geometrische Stetigkeit beliebiger Ordnung zwischen Tensor-Produkt-Bézier-Flächen

Nr. 93/2: G. Alefeld, G. Mayer: A Computer Aided Existence and Uniqueness Proof for an Inverse Matrix Eigenvalue Problem

Nr. 93/3: B. Weber: Symbolische Programmierung in der Mehrkörperdynamik

Nr. 93/4: R. Rihm: Über Einschließungsverfahren für gewöhnliche Anfangswertprobleme und ihre Anwendung auf Differentialgleichungen mit unstetiger rechter Seite

Nr. 93/5: J. Wittenburg: Explizite Lösungen für lineare Gleichungssysteme mit tridiagonalen Koeffizientenmatrizen. Anwendungen in der Mechanik

Nr. 93/6: N. Henze, B. Klar: Goodness–of–Fit Testing for a Space–Time Model for Daily Rainfall

Nr. 93/7: K. Schweizerhof, J. Riccius, M. Baumann: Verbesserung von Finite Element Berechnungen durch Adaptivität und Netzglättung am Beispiel ebener und gekrümmter Flächentragwerke

Nr. 93/8: G. Starke: Subspace Orthogonalization for Substructuring Preconditioners for Nonselfadjoint Elliptic Problems

Nr. 93/9: N. Henze, B. Klar: Empirical Distribution Function Tests for the Generalized Poisson Model

Nr. 94/1: G. Aumann: Geometric Continuity of Parametric Curves and Surfaces

Nr. 94/2: T. Dehn, M. Eiermann, K. Giebermann, V. Sperling: Structured Sparse Matrix-Vector Multiplication on Massively Parallel Architectures

Nr. 94/3: W. Krämer: Bericht über die Begutachtung des IWRMM im Dezember 1993

Nr. 95/1: L. Kobbelt: Interpolatory Refinement is Low Pass Filtering

Nr. 95/2: M. Paluszny, H. Prautzsch, M. Schäfer: Corner cutting and interpolatory refinement

Nr. 95/3: B. Klar: Analysis of and Goodness of Fit Testing for a Flexible Discrete Time Failure Model

Nr. 95/4: P. Vielsack: Regularisierung von Haftkräften bei Coulombscher Reibung

Nr. 95/5: P. Vielsack, M. Storz: Bifurcation of Motion in a Technical System with Stick-Slip and Impact

Nr. 95/6: M. Brühl: A Curve Tracing Algorithm for Computing the Pseudospectrum

Nr. 95/7: J. Riccius, K. Schweizerhof, M. Baumann: On the treatment of shell intersections in adaptive finite element analysis and combination with mesh smoothing

Nr. 96/1: M. Dormanns, H.-U. Heiß: Nutzung von Asynchronität bei iterativen Gleichungslösern auf Multirechnersystemen

Nr. 96/2: P. Vielsack, J. Kirillowa: Nichteindeutigkeit der Bewegungen eines Reibschwingers mit Selbsterregung

Nr. 96/3: L. Kobbelt, T. Hesse, H. Prautzsch, K. Schweizerhof: Diskrete Freiformflächenerzeugung für FEM-Anwendungen

Nr. 96/4: M. Brühl, M. Hanke, H. Wanzki: Ein Rekonstruktionsverfahren für die elektrische Impedanztomographie

Nr. 96/5 : W. Hofschuster, W. Krämer: Ein rechnergestützter Fehlerkalkül mit Anwendung auf ein genaues Tabellenverfahren

Weitere Arbeiten sind in Vorbereitung.