Bergische Universität
Wuppertal

Sichere a priori Fehlerabschätzung und Implementierung der Funktion zweier Variabler $\log(\sqrt{x^2+y^2})$

Frithjof Blomquist, Werner Hofschuster, Walter Krämer

Preprint 2004/1

Wissenschaftliches Rechnen/
Softwaretechnologie

Diese Arbeit entstand im Rahmen des gemeinsamen Projektes
Erweiterung des Funktionsumfanges der C-XSC Bibliothek

Impressum

Herausgeber:	Prof. Dr. W. Krämer, Dr. W. Hofschuster Wissenschaftliches Rechnen/Softwaretechnologie Fachbereich C (Mathematik und Naturwissenschaften) Bergische Universität Wuppertal Gaußstr. 20 D-42097 Wuppertal

Internet-Zugriff

Die Berichte sind in elektronischer Form erhältlich über die World Wide Web Seiten

`http://www.math.uni-wuppertal.de/wrswt/literatur.html`

Autoren-Kontaktadressen

Frithjof Blomquist
Adlerweg 6
D-66436 Püttlingen

E-mail: `blomquist@math.uni-wuppertal.de`

Werner Hofschuster
Bergische Universität Wuppertal
Gaußstr. 20
D-42097 Wuppertal

E-mail: `hofschuster@math.uni-wuppertal.de`

Walter Krämer
Bergische Universität Wuppertal
Gaußstr. 20
D-42097 Wuppertal

E-mail: `kraemer@math.uni-wuppertal.de`

Sichere a priori Fehlerabschätzung und Implementierung der Funktion zweier Variabler $\log(\sqrt{x^2+y^2})$

Frithjof Blomquist, Werner Hofschuster, Walter Krämer

Februar, 2004

Inhaltsverzeichnis

1 Zusammenfassung

In dieser Arbeit werden typische Vorgehensweisen zum Berechnen von Fehlerschranken für Funktionen zweier Variabler aufgezeigt. Die Herleitung solcher garantierter Schranken ist aufwendig und erfordert in verschiedenen Teilbereichen des Definitionsbereiches im Hinblick auf numerische Auswertefehler geeignet angepasste Funktionsterme, um die jeweiligen Fehlerschranken klein halten zu können. Durch geeignete Hilfsprogramme kann die Berechnung der a-priori Fehlerschranken wenigstens zum Teil automatisiert werden. Für alle ca. 50 verschiedenen Funktionsterme, die in dieser Arbeit zur Approximation der Funktion $\log(\sqrt{x^2+y^2})$ herangezogen werden, sind numerische Ergebnisse solcher worst-case a-priori Fehlerabschätzungen angegeben. Unter Verwendung der gefundenen Fehlerschranken wird die Funktion zur Ergänzung der Verifikationsbibliothek C-XSC für Punkt- und Intervall-Argumente implementiert.

Die in dieser Arbeit dargestellten Methoden sowie die zur automatischen Fehlerabschätzung verwendeten Hilfsprogramme werden in [2] ausführlich besprochen. Die zugehörigen Programme stehen im Netz unter

`http://www.math.uni-wuppertal.de/wrswt/literatur/a_priori.tgz`

zur Verfügung.

Keywords: C-XSC, a priori Fehlerschranken, genaue Funktionsapproximationen, Einschließungsmethoden.

MSC (2000): 65G20, 65Y15

2 Einleitung

Für Algorithmen aus dem Bereich der Numerik mit Ergebnisverifikation werden so genannte Intervallfunktionen benötigt, deren Ergebnisintervall den exakten Wertebereich der betrachteten Funktion über Intervallargumenten mit Sicherheit einschließt. Die berechnete Einschließung soll dabei möglichst eng sein.

Zur Implementierung solcher Funktionen sind sichere a-priori Abschätzungen der Approximationsfehler in den verschiedenen Teilbereichen sowie a-priori worst-case Fehlerabschätzungen der durch Rundungsfehler verursachten Ungenauigkeiten notwendig. Für die Standardfunktionen wurden dazu spezielle Algorithmen mit Tabellenverfahren entwickelt, mit denen die Rundungsfehler bei den Polynomauswertungen auf ein Minimum reduziert werden konnten, [6], [7], [8], [15], [16], [17].

Die genannten Tabellenverfahren und die damit oft gekoppelte Simulation einer doppelten Genauigkeit lassen sich in vielen Fällen auch dann anwenden, wenn Standardfunktionen für spezielle Terme auszuwerten sind. In dieser Arbeit wird gezeigt, wie diese Verfahren bei der Implementierung von $f(x,y) := \log(\sqrt{x^2+y^2})$ angewendet werden können. Dabei ist zusätzlich zu beachten, dass x, y als unabhängige Maschinenzahlen zu betrachten sind, wobei man sich auf $0 \leq y \leq x$ beschränken kann. Man beachte in diesem Zusammenhang insbesondere das Vorgehen in den Bereichen A_1 und A_7

(siehe Abschnitt 5), in denen durch geeignete weitere Unterteilungen und eine dem entsprechenden Unterteilungsbereich angepasste Wahl des zusätzlichen Parameters N Tabellenverfahren realisiert sind. Insgesamt kommen für die numerische Approximation der Funktion $f(x,y) = \log(\sqrt{x^2+y^2})$ ca. 50 unterschiedliche Funktionsausdrücke zum Einsatz, deren Güte jeweils durch eine a-priori worst-case Fehlerabschätzung sichergestellt werden muss.

Die sichere numerische Berechnung der Rundungsfehlerschranken erfolgt mit C-XSC Programmen, die in [2] vollständig angegeben und ausführlich beschrieben werden. Zur Implementierung von $f(x,y) := \log(\sqrt{x^2+y^2})$ findet man in [2] die Berechnung aller Fehlerschranken und alle dazu notwendigen Werkzeuge mit entsprechenden Anwendungsbeispielen.

3 Zur Notation

$\mathbb{R}$	Menge der reellen Zahlen
$I\mathbb{R}$	Menge der reellen Intervalle
IR	Menge der Maschinenintervalle
$S(b,l,\underline{e},\overline{e}) = S(b,l)$	Gleitkommaraster mit Basis b, Mantissenlänge l und Exponent e, mit: $\underline{e} \le e \le \overline{e}, \;\; e \in \mathbb{Z}$
$S(2,53,-1022,+1023) = S(2,53)$	IEEE-Datenformat *double*
x, y	reelle Größen; $x, y \in \mathbb{R}$
$\widetilde{x}, \widetilde{y}$	Maschinenzahlen $\widetilde{x}, \widetilde{y} \in S(2,53)$, die auch fehlerbehaftet sein können[1]
$\circ \in \{+,-,\bullet,/\}$	exakte reelle Operatoren
$\{\oplus, \ominus, \odot, \oslash\}$	Gleitkomma-Operatoren, deren Rundung vom jeweils eingestellten Rundungsmodus abhängt
$\{\, \hat{+}\,,\, \hat{-}\,,\, \hat{\cdot}\,,\, \hat{/}\,\}$	Gleitkomma-Intervalloperatoren; z.B. $A, B \in IR \quad \Longrightarrow \quad A \bullet B \subseteq A \diamond B$
$\texttt{Minreal} = 2^{-1022} = 2.225.. \cdot 10^{-308}$	kleinste positive, normalisierte *double* Zahl
$\texttt{minreal} = 2^{-1074} = 4.940.. \cdot 10^{-324}$	kleinste positive, denormalisierte *double* Zahl
$\texttt{Maxreal} < 2^{+1024} = 1.797.. \cdot 10^{+308}$	größte, normalisierte *double* Zahl
$U := (-\texttt{Minreal}, +\texttt{Minreal})$	denormalisierter Zahlenbereich
$\texttt{Minreal} \le \lvert\widetilde{x}\rvert \le \texttt{Maxreal}$	normalisierter Bereich im *double*-Format
$\log(x),\ x > 0$	reelle Logarithmusfunktion zur Basis e
$f(x,y) = \log(\sqrt{x^2+y^2})$	die in diesem Preprint behandelte Funktion
$f_i(x,y) = f(x,y), \quad i = 1,2,\ldots,7$	Funktionen zur Darstellung von $f(x,y)$ in den einzelnen Teilbereichen A_i
$\widetilde{f}(\widetilde{x},\widetilde{y}) \approx f(\widetilde{x},\widetilde{y})$	$\widetilde{f}(\widetilde{x},\widetilde{y})$: auf der Maschine ausgewerteter i.a. fehlerbehafteter Funktionswert
$\texttt{ln}(\widetilde{x}) \approx \log(\widetilde{x})$	$\texttt{ln}(\widetilde{x})$ ist eine C-XSC Standardfunktion
$\texttt{lnp1}(\widetilde{x}) \approx \log(1+\widetilde{x})$	$\texttt{lnp1}(\widetilde{x})$ ist eine C-XSC Standardfunktion
$\varepsilon(f, A_i)$	relative Fehlerschranke bei Auswertung von $f(\widetilde{x},\widetilde{y})$ über dem Teilbereich A_i
$\varepsilon(app)$	relativer Approximationsfehler
$\varepsilon(r) = 2^{-53}$	relative Fehlerschranke der Grundoperationen bei maximal genauer Arithmetik.
$W_f = \{f(x,y) \,\vert\, x \in X \wedge y \in Y\}$	Wertebereich von $f(x,y)$ über X und Y

[1] Berechnete Fehlerschranken von Funktionen $f(x)$ beziehen sich immer nur auf die Maschinenargumente $\widetilde{x}$ eines vorgegebenen Maschinenintervalls, wobei die $\widetilde{x}$ stets als rundungsfehlerfrei angenommen werden.

4 Aufgabenstellung

Wir betrachten die reelle Funktion

$$f : D_f = I\!R^2 \backslash \{(0,0)\} \to I\!R, \quad \text{mit} \quad f(x,y) = \log(\sqrt{x^2+y^2}),$$

die für alle Maschinenzahlen $\widetilde{x}, \widetilde{y} \in S(2,53)$[1], mit $|\widetilde{x}| + |\widetilde{y}| > 0$ auszuwerten ist. Ohne Einschränkung der Allgemeinheit kann dabei $0 \leq y \leq x$ vorausgesetzt werden, womit $f(x,y)$ als Realteil der komplexwertigen Logarithmusfunktion nur im 1. Quadranten der x,y-Ebene unterhalb der 1. Winkelhalbierenden betrachtet wird.

Bezeichnet man mit $\widetilde{f}(\widetilde{x},\widetilde{y})$ den auf der Maschine berechneten Näherungswert für $f(\widetilde{x},\widetilde{y}) \approx \widetilde{f}(\widetilde{x},\widetilde{y})$, so soll eine Schranke $\Delta(f)$ des absoluten Fehlers berechnet werden, falls $f(\widetilde{x},\widetilde{y})$ im Unterlaufbereich $U = (-\texttt{MinReal}, +\texttt{Minreal})$ liegt[2]:

$$|\widetilde{f}(\widetilde{x},\widetilde{y}) - f(\widetilde{x},\widetilde{y})| \leq \Delta(f) \quad \forall\, f(\widetilde{x},\widetilde{y}) \in U$$

Für alle Funktionswerte $f(\widetilde{x},\widetilde{y}) \notin U$ aus dem normalisierten Bereich ist eine Schranke $\varepsilon(f)$ des relativen Fehlers zu berechnen:

$$\left| \frac{\widetilde{f}(\widetilde{x},\widetilde{y}) - f(\widetilde{x},\widetilde{y})}{f(\widetilde{x},\widetilde{y})} \right| \leq \varepsilon(f) \quad \forall\, f(\widetilde{x},\widetilde{y}) \notin U.$$

Die Schranken $\Delta(f)$ und $\varepsilon(f)$ werden zur Implementierung einer Intervallfunktion benötigt, mit der zu vorgegebenen Intervallargumenten $X = [\widetilde{x}_1, \widetilde{x}_2]$ und $Y = [\widetilde{y}_1, \widetilde{y}_2]$ der entsprechende Wertebereich $W_f = \{r \in I\!R \,|\, r = f(x,y) \wedge x \in X \wedge y \in Y\}$ von $f(x,y)$ garantiert eingeschlossen wird. Dazu benötigt man eine Näherungsfunktion $\widetilde{f}(\widetilde{x},\widetilde{y}) \approx f(\widetilde{x},\widetilde{y})$, die wegen der Monotonie von $\log(\sqrt{x^2+y^2})$ bez. $s = x^2+y^2$ mit den entsprechenden Punktargumenten $\widetilde{x}_1, \widetilde{y}_1$ bzw. $\widetilde{x}_2, \widetilde{y}_2$ aufzurufen ist und deren Implementierung zusammen mit einer Fehlerabschätzung in den nächsten Abschnitt behandelt wird.

5 Algorithmus

Zur Implementierung von $f(x,y)$ wird der Bereich der Quadratsummen $s := x^2 + y^2$ in sieben Teilintervalle A_i unterteilt, und in jedem A_i wird $f(x,y)$ durch eine geeignete Funktion $f_i(x,y) \equiv f(x,y) = \log(\sqrt{x^2+y^2})$ realisiert:

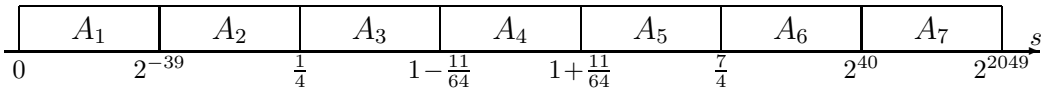

Abbildung 1: Teilintervalle A_i der Quadratsummen $s := \widetilde{x}^2 + \widetilde{y}^2$

Die Funktionen $f_i(x,y)$ werden für Punktargumente $\widetilde{x}, \widetilde{y} \in S(2,53)$ auf der Maschine ausgewertet und liefern die Näherungen $\widetilde{f}_i(\widetilde{x},\widetilde{y}) \approx f(\widetilde{x},\widetilde{y}) = \log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})$. In der folgenden Tabelle sind die Funktionen $f_i(x,y)$ zusammengestellt:

[1] $S(2,53)$ ist die Menge aller Rasterzahlen des *double*-Formats.

[2] $\texttt{MinReal} = 2^{-1022}$ ist die kleinste positive, normalisierte Zahl im *double*-Format.

A_i	$f_i(x,y) = f(x,y) = \log(\sqrt{x^2+y^2}), \quad 0 \le y \le x, \quad y + x > 0$
A_1	$f_1(x,y) = f_{1;N}(x,y) = -N \cdot \log(2) + \left[\log(2^N \cdot x) + \frac{1}{2}\log\left(1 + \left(\frac{y}{x}\right)^2\right)\right],$ $N \in \{20, 40, 61, 122, 160, 229, 259, 320, 366, 427, 488, 518, 549, 610, 671, 732, 763, 825, 885, 945, 976, 1067\}$
A_2	$f_2(x,y) = \frac{1}{2}\log(x^2+y^2), \quad x^2+y^2$ exakt im Akku berechnen
A_3	$f_3(x,y) = \frac{1}{2} \cdot \left[\log(r) + \log(1 + \frac{r_1}{r})\right], \quad x^2+y^2 = r + r_1$
A_4	$f_4(x,y) = \log(1 + [x^2+y^2-1]), \quad x^2+y^2-1$ exakt im Akku berechnen
A_5	$f_5(x,y) = \frac{1}{2} \cdot \left[\log(r) + \log(1 + \frac{r_1}{r})\right], \quad x^2+y^2 = r + r_1$
A_6	$f_6(x,y) = \frac{1}{2}\log(x^2+y^2), \quad x^2+y^2$ exakt im Akku berechnen
A_7	$f_7(x,y) = f_{7;N}(x,y) = N \cdot \log(2) + \left[\log(2^{-N} \cdot x) + \frac{1}{2}\log\left(1 + \left(\frac{y}{x}\right)^2\right)\right],$ $N \in \{20, 40, 61, 122, 160, 229, 259, 320, 366, 427, 488, 518, 549, 610, 671, 732, 763, 825, 885, 945, 976\}$

Tabelle 1: $f_i(x,y)$ zur Auswertung von $f(x,y)$ in den Bereichen A_i

Hinweise:

1. Die Bereiche A_1 und A_7 werden in weitere Teilintervalle unterteilt, in denen das $N \in \mathbb{N}$ jeweils so gewählt wird, dass die Auswertefehler von $f_1(x,y)$ bzw. $f_7(x,y)$ dabei möglichst klein ausfallen. Im Falle $y \ll x$ kann der Summand $\frac{1}{2}\log(1+\left(\frac{y}{x}\right)^2)$ in $f_1(x,y)$ bzw. $f_7(x,y)$ vernachlässigt werden.

2. Bei der Auswertung von $f_2(x,y)$ und $f_6(x,y)$ wird die Quadratsumme $\widetilde{x}^2 + \widetilde{y}^2$ rundungsfehlerfrei im Akkumulator berechnet, der danach mit der relativen Fehlerschranke $\varepsilon_0 = 2^{-53}$ maximal genau ins *double*-Format ausgelesen wird.

3. Bei der Auswertung von $f_3(x,y)$ und $f_5(x,y)$ wird $\widetilde{x}^2 + \widetilde{y}^2$ ebenfalls rundungsfehlerfrei im Akkumulator berechnet, der danach zweimal ins *double*-Format ausgelesen wird. Man erhält: $\widetilde{x}^2 + \widetilde{y}^2 = r + r_1 \approx r + r1$, mit $r, r1 \in S(2, 53)$. Die Maschinenzahl r wird als exakt angesehen, und der absolute Fehler von $r1$ wird abgeschätzt.

4. Die Auswertung von $f_4(x,y)$ erfolgt mit Hilfe der in C-XSC implementierten Funktion `lnp1(d)` $\approx \log(1 + \mathtt{d})$, wobei $\widetilde{x}^2 + \widetilde{y}^2 - 1$ exakt im Akku berechnet wird, der anschließend in die *double*-Variable `d` $\approx \widetilde{x}^2 + \widetilde{y}^2 - 1$ maximal genau

ausgelesen wird. Im Spezialfall $\widetilde{x} = 1$ und $\widetilde{y} < 2^{-28}$ gilt die Approximation $f_4(\widetilde{x}, \widetilde{y}) \approx \widetilde{y}^2/2$, wobei die Funktionswerte jetzt sogar in den Unterlaufbereich $U = (-\texttt{MinReal}, +\texttt{MinReal})$ fallen können. In diesem Fall wird nur der absolute Fehler abgeschätzt.

5.1 Fehlerabschätzung in $A_1 = (0, 2^{-39}]$

Verlangt man $\widetilde{x} < 2^{-20}$, so erhält man $\widetilde{x}^2 + \widetilde{y}^2 < 2 \cdot 2^{-40} = 2^{-39}$, und nach Tabelle 1 gilt

$$\text{(1)} \qquad f_1(x,y) = -N \cdot \log(2) + \left[\log(2^N \cdot x) + \frac{1}{2} \log\left(1 + \left(\frac{y}{x}\right)^2\right)\right],\ N \in I\!N,$$

wobei im Falle $\widetilde{y}/\widetilde{x} \leq 2^{-24}$ der letzte Summand vernachlässigt werden kann:

$$\text{(2)} \qquad f_1(x,y) \approx -N \cdot \log(2) + \log(2^N \cdot x),\ N \in I\!N;$$

Die Idee bei der Anwendung von $f_1(x,y)$ besteht nun darin, den ersten Summanden $-N \cdot \log(2)$ rechts in (1) so zu wählen, dass die Bedingung $f_1(x,y) \approx -N \cdot \log(2)$ möglichst gut erfüllt ist und dass die Konstante $-N \cdot \log(2)$ im *double*-Format möglichst genau gespeichert werden kann. Sind beide Bedingungen erfüllt, so liefert die Auswertung der rechten Seite von (1) einen nur kleinen Fehler, da zu einem betragsmäßig großen und fast fehlerfreien Summanden ein zwar fehlerbehafteter aber betragsmäßig kleiner Summand [...] zu addieren ist. Die Grundidee ist also, die unvermeidbaren Rundungsfehler möglichst ganz in den betragsmäßig kleineren Summanden [...] zu verlagern! Natürlich lässt sich diese Grundidee nur realisieren, wenn man den ganzen Bereich $2^{-1074} \leq \widetilde{x} < 2^{-20}$ in geeignete Teilintervalle B_j zerlegt und in jedem B_j die Zahl $N \in I\!N$ so wählt, dass die beiden genannten Bedingungen möglichst gut erfüllt werden. In der Tabelle auf Seite 10 sind für die einzelnen Teilintervalle B_j folgende Informationen zusammengestellt: Der relative Auswertefehler bei Anwendung von (1), der geeignete N-Wert und in der letzten Spalte die relative Fehlerschranke, die beim Speichern von $N \cdot \log(2)$ im *double*-Format entsteht.

5.1.1 Berechnung der N-Werte

Am Beispiel des Intervalls $B_{-1} = [2^{-1074}, 2^{-1022})$ wird jetzt gezeigt, wie der günstigste Wert $N = 1067$ zustande kommt. Beachten Sie bitte zunächst, dass in (1) der Summand $\log(2^N \cdot x)$ in B_{-1} nur für $52 \leq N \leq 2045$ ohne Fehlermeldung ausgewertet werden kann, da in C-XSC die Logarithmusfunktion `real ln(const real&)` für Argumente aus dem denormalisierten Teilbereich $(0, 2^{-1022})$ nicht definiert ist. Wir suchen daher in $52 \leq N \leq 2045$ zunächst solche N-Werte, für die $N \cdot \log(2)$ im *double*-Format mit einem möglichst kleinen relativen Fehler gespeichert werden kann. Dazu unterteilt man $52 \leq N \leq 2045$ in neun Teilintervalle T_i und berechnet in jedem T_i den Wert N_i, für den $N_i \log(2)$ optimal, d.h. also mit einem minimalen relativen Fehler gespeichert werden kann. Dieser relative Fehler wird für ein vorgegebenes N wie folgt berechnet:

Zunächst wird mit der in C_XSC vorhandenen staggered Arithmetik für $N \cdot \log(2)$ eine Näherung $\texttt{Nln\_2} \approx N \cdot \log(2)$ realisiert, mit $\texttt{Nln\_2} \in S(2, 53)$. Anschließend wird dann

j	B_j	Rel. Fehlerschranke	N	Rel. Schranke bez. $N \cdot \log(2)$
-1	$[2^{-1074}, 2^{-1022})$	$2.459639 \cdot 10^{-16}$	1067	$5.81871582840441 \cdot 10^{-19}$
0	$[2^{-1022}, 2^{-950})$	$2.464005 \cdot 10^{-16}$	976	$6.78099202380294 \cdot 10^{-19}$
1	$[2^{-950}, 2^{-900})$	$2.493001 \cdot 10^{-16}$	945	$7.44534477863647 \cdot 10^{-19}$
2	$[2^{-900}, 2^{-850})$	$2.462906 \cdot 10^{-16}$	885	$2.19718262365773 \cdot 10^{-18}$
3	$[2^{-850}, 2^{-800})$	$2.446647 \cdot 10^{-16}$	825	$5.56678584903675 \cdot 10^{-18}$
4	$[2^{-800}, 2^{-750})$	$2.488554 \cdot 10^{-16}$	763	$2.44007669626041 \cdot 10^{-18}$
5	$[2^{-750}, 2^{-700})$	$2.472065 \cdot 10^{-16}$	732	$6.78099202380294 \cdot 10^{-19}$
6	$[2^{-700}, 2^{-650})$	$2.447755 \cdot 10^{-16}$	671	$6.78099202380294 \cdot 10^{-19}$
7	$[2^{-650}, 2^{-600})$	$2.552012 \cdot 10^{-16}$	610	$6.78099202380294 \cdot 10^{-19}$
8	$[2^{-600}, 2^{-550})$	$2.673659 \cdot 10^{-16}$	549	$6.78099202380294 \cdot 10^{-19}$
9	$[2^{-550}, 2^{-500})$	$2.547771 \cdot 10^{-16}$	518	$1.91724602509176 \cdot 10^{-18}$
10	$[2^{-500}, 2^{-450})$	$2.677207 \cdot 10^{-16}$	488	$6.78099202380294 \cdot 10^{-19}$
11	$[2^{-450}, 2^{-400})$	$2.591254 \cdot 10^{-16}$	427	$6.78099202380294 \cdot 10^{-19}$
12	$[2^{-400}, 2^{-350})$	$2.677857 \cdot 10^{-16}$	366	$6.78099202380294 \cdot 10^{-19}$
13	$[2^{-350}, 2^{-300})$	$2.690160 \cdot 10^{-16}$	320	$1.42250834110490 \cdot 10^{-18}$
14	$[2^{-300}, 2^{-250})$	$2.959482 \cdot 10^{-16}$	259	$1.91724602509176 \cdot 10^{-18}$
15	$[2^{-250}, 2^{-200})$	$3.041687 \cdot 10^{-16}$	229	$3.61344598803646 \cdot 10^{-18}$
16	$[2^{-200}, 2^{-150})$	$3.290578 \cdot 10^{-16}$	160	$1.42250834110490 \cdot 10^{-18}$
17	$[2^{-150}, 2^{-100})$	$3.429172 \cdot 10^{-16}$	122	$6.78099202380294 \cdot 10^{-19}$
18	$[2^{-100}, 2^{-50})$	$4.295508 \cdot 10^{-16}$	61	$6.78099202380294 \cdot 10^{-19}$
19	$[2^{-50}, 2^{-30})$	$4.189496 \cdot 10^{-16}$	40	$1.42250834110490 \cdot 10^{-18}$
20	$[2^{-30}, 2^{-20})$	$4.142222 \cdot 10^{-16}$	20	$1.42250834110490 \cdot 10^{-18}$

Tabelle 2: Tabelle mit den relativen Fehlern bei Auswertung von (1) mit N für alle B_j

ebenfalls in staggered Arithmetik eine garantierte Einschließung von $N \cdot \log(2) -$ `Nln_2` berechnet und diese Einschließung in das *double*-Intervall `zz` gerundet. Damit gilt dann $N \cdot \log(2) -$ `Nln_2` $\in$ `zz`, und mit `zz` und `Nln_2` kann dann der absolute bzw. relative Fehler für jedes $N \in T_i$ direkt abgeschätzt werden. $N = N_i$ ist dann im Teilintervall T_i der Wert mit dem kleinsten relativen Fehler. Im Programm `Nln2.cpp` aus [2] findet man weitere Einzelheiten.

In der nachfolgenden Tabelle 3 sind für die neun Teilintervalle T_i neben der Approximation $N_i \cdot \log(2) \approx \frac{z}{n} \in S(2, 53)$ die Werte N_i zusammen mit den absoluten und relativen Fehlerschranken angegeben. Beachten Sie bitte, dass die für z, n angegebenen Werte und

N_i	$N_i \cdot \log(2) \approx \frac{z}{n}$	Relativer Fehler	Absoluter Fehler
61	$\frac{5950659388407608.0}{140737488355328.0}$	$6.78099202380294 \cdot 10^{-19}$	$2.86713755664608 \cdot 10^{-17}$
183	$\frac{8925989082611412.0}{70368744177664.0}$	$6.78099202380294 \cdot 10^{-19}$	$8.60141266993823 \cdot 10^{-17}$
1067	$\frac{6505485212531678.0}{8796093022208.0}$	$5.81871582840441 \cdot 10^{-19}$	$4.30345264449089 \cdot 10^{-16}$
1189	$\frac{7249317636082629.0}{8796093022208.0}$	$4.52589466947313 \cdot 10^{-19}$	$3.73002513316168 \cdot 10^{-16}$
1220	$\frac{7438324235509510.0}{8796093022208.0}$	$6.78099202380294 \cdot 10^{-19}$	$5.73427511329215 \cdot 10^{-16}$
1311	$\frac{7993150059633580.0}{8796093022208.0}$	$3.47369011067856 \cdot 10^{-19}$	$3.15659762183246 \cdot 10^{-16}$
1433	$\frac{8736982483184531.0}{8796093022208.0}$	$2.60064669099486 \cdot 10^{-19}$	$2.58317011050325 \cdot 10^{-16}$
1799	$\frac{5484239876918692.0}{4398046511104.0}$	$6.91986452186632 \cdot 10^{-20}$	$8.62887576515599 \cdot 10^{-17}$
2043	$\frac{6228072300469643.0}{4398046511104.0}$	$2.00527864084272 \cdot 10^{-20}$	$2.83967446142832 \cdot 10^{-17}$

Tabelle 3: Optimale $N_i \in [52, 2045]$ zur Speicherung von $N_i \cdot \log(2) \approx z/n$.

deren Quotienten in $S(2, 53)$ exakt gespeichert werden können, womit die Portabilität der Programme wesentlich erleichtert wird. Die Berechnung der Tabellenwerte erfolgte mit dem Programm `Nln2.cpp`, wobei man weitere Einzelheiten in [2] findet.
Mit $N = 2043$ kann damit die Konstante $2043 \cdot \log(2)$ durch die Maschinenzahl

$$\frac{z}{n} = \frac{6228072300469643.0}{4398046511104.0} \approx 2043 \cdot \log(2)$$

im *double*-Format mit dem kleinsten relativen Fehler $\varepsilon_0 = 2.00527864084272 \cdot 10^{-20}$ approximiert werden. Dies bedeutet jedoch nicht, dass wir mit $N = 2043$ auch bei Anwendung von (1) auf Seite 9 den kleinsten relativen Auswertefehler erhalten. Nach den Bemerkungen von Seite 9 oben muss für einen möglichst kleinen relativen Auswertefehler zusätzlich die Bedingung

$$|-N \cdot \log(2)| \gg \left| \log(2^N \cdot x) + \frac{1}{2} \log\left(1 + \left(\frac{y}{x}\right)^2\right) \right| \tag{3}$$

für alle Maschinenzahlen $\widetilde{x} \in B_{-1} = [2^{-1074}, 2^{-1022})$ möglichst gut erfüllt sein. Zur Überprüfung von (3) sind in der folgenden Tabelle 4 die Beträge von $-N \log(2)$ und $\log(2^N x)$ zusammengestellt. Der Summand $\frac{1}{2}\log(1 + (\frac{y}{x})^2) \leq \frac{1}{2}\log(2) = 0.346\ldots$ spielt bei diesen Vergleichen offensichtlich keine Rolle!
Aus Tabelle 4 erkennt man, dass die Bedingung (3) für $N = 1067$ am besten erfüllt ist, so dass wir mit $N = 1067$ bei der Maschinenauswertung von (1) für $\widetilde{x} \in B_{-1} = [2^{-1074}, 2^{-1022})$ den kleinsten Auswertefehler erwarten können.

5.1.2 Berechnung des Auswertefehlers

Wir betrachten zunächst wieder nur das Intervall $B_{-1} = [2^{-1074}, 2^{-1022}) \ni \widetilde{x}$ und daraus ein sehr enges Teilintervall[3] `Xi` $\subset B_{-1}$, für das der relative Auswertefehler bez. (1) zu berechnen ist. Das Maschinenergebnis $\widetilde{f}(\widetilde{x}, \widetilde{y})$ ist für $\widetilde{x} \in$ `Xi` und $0 \leq \widetilde{y} \leq \widetilde{x}$, zusammen mit der C-XSC Funktion `lnp1`$(x) \approx \log(1+x)$ gegeben durch:

$$\widetilde{f}(\widetilde{x}, \widetilde{y}) := -N\widetilde{\log(2)} \oplus \left[\widetilde{\log}(2^N \cdot \widetilde{x}) \oplus \frac{1}{2} \cdot \texttt{lnp1}\{(\widetilde{y} \oslash \widetilde{x}) \odot (\widetilde{y} \oslash \widetilde{x})\}\right]$$

Die Abschätzung des relativen Fehlers $\varepsilon_f := (f(\widetilde{x}, \widetilde{y}) - \widetilde{f}(\widetilde{x}, \widetilde{y}))/f(\widetilde{x}, \widetilde{y})$ erfolgt mit dem Programm `lnx2y2_Bd.cpp` und der dort implementierten Funktion

```
real T_x(const interval& Xi, const real& delx)
{
    interval Yi=interval(0,Sup(Xi)),A,B;
    real dela,delb,r,eps;
    abs_divh1(Yi,delx,Xi,delx,A,dela); // y/x
    abs_mulh1(A,dela,A,dela,B,delb);   // (y/x)*(y/x)
    abs_lnp1_abs(B,delb,A,dela);
    times2pown(A,-1);    // Division by 2
    times2pown(dela,-1); // Division by 2
    Yi = Xi;  times2pown(Yi,1067);
    abs_ln_abs(Yi,delx,B,delb);
    abs_addh1(A,dela,B,delb,Yi,r);
    rel_addh1(Yi,r,ln2_1067,abs_1067,A,eps);
    return eps;     // eps: rel. error bound
}
```

Das Intervall `Yi` schließt alle Maschinenzahlen $\widetilde{y}$ ein, mit $0 \leq \widetilde{y} \leq$ `Sup(Xi)`, und der Aufruf `abs_divh1(Yi,delx,Xi,delx,A,dela)` liefert mit `dela` eine garantierte Oberschranke des absoluten Fehlers δ_1 für alle $\widetilde{x} \in$ `Xi` und für alle $\widetilde{y} \in$ `Yi`; das ergibt:

$$\delta_1 := \widetilde{y}/\widetilde{x} - \widetilde{y} \oslash \widetilde{x}, \quad |\delta_1| \leq \texttt{dela} \ \forall \widetilde{x} \in \texttt{Xi}, \text{ mit } 0 \leq \widetilde{y} \leq \widetilde{x} \leq \texttt{Sup(Xi)};$$

[3] Die Fehlerabschätzung erfolgt durch Intervallrechnung, deren unvermeidbaren Überschätzungen durch möglichst enge Teilintervalle `Xi` begrenzt werden können.

N_i	$N_i \cdot \log(2)$	$\lvert\log(2^{N_i}x)\rvert \leq$	N_i	$N_i \cdot \log(2)$	$\lvert\log(2^{N_i}x)\rvert \leq$
61	42.281...	702.158...	1220	845.63...	137.24...
183	126.85...	617.59...	1311	908.71...	200.31...
1067	739.58...	31.191...	1799	1246.9...	538.57...
1189	824.15...	115.75...	2043	1416.0...	707.70...

Tabelle 4: Vergleich von $N_i \cdot \log(2)$ mit $\lvert\log(2^{N_i}x)\rvert$ für $x \in [2^{-1074}, 2^{-1022})$

Da die Funktion `abs_divh1(..)` mit `delx` $= 0$ aufgerufen wird, setzen wir voraus, dass alle Operanden aus den Intervallen `Yi` und `Xi` als exakt angenommen werden, d.h. bei der Fehlerabschätzung wird angenommen, dass die Funktion $f(x, y,)$ nur für exakte Maschinenargumente auszuwerten ist. Das Intervall `A` $:=$ `Yi` $\diamondsuit$`Xi` liefert eine Einschließung aller exakten Quotienten $\widetilde{y}/\widetilde{x}$, mit $\widetilde{x} \in$ `Xi` und $0 \leq \widetilde{y} \leq \widetilde{x}$.
Der nächste Aufruf `abs_mulh1(A,dela,A,dela,B,delb)` liefert mit `delb` eine Oberschranke des absoluten Fehlers δ_2 :

$$\delta_2 := (\widetilde{y}/\widetilde{x}) \cdot (\widetilde{y}/\widetilde{x}) - (\widetilde{y} \oslash \widetilde{x}) \odot (\widetilde{y} \oslash \widetilde{x}), \quad |\delta_2| \leq \texttt{delb} \ \forall (\widetilde{y}/\widetilde{x}) \in \texttt{A}$$

und `B` $:=$ `A` $\diamond$ `A` ist eine garantierte Einschließung aller exakten Produkte $(\widetilde{y}/\widetilde{x}) \cdot (\widetilde{y}/\widetilde{x})$.

Der nachfolgenden Aufruf `abs_lnp1_abs(B,delb,A,dela)` liefert mit `dela` eine Oberschranke des absoluten Fehlers δ_3 für alle $\widetilde{x} \in$ `Xi` und $0 \leq \widetilde{y} \leq \widetilde{x}$:

$$\delta_3 := \log(1 + (\widetilde{y}/\widetilde{x})^2) - \texttt{lnp1}\{(\widetilde{y} \oslash \widetilde{x}) \odot (\widetilde{y} \oslash \widetilde{x})\}, \qquad |\delta_3| \leq \texttt{dela}$$

Ganz entsprechend findet man mit dem Rückgabewert `eps` eine garantierte Oberschranke des relativen Fehlers $\varepsilon_f := (f(\widetilde{x}, \widetilde{y}) - \widetilde{f}(\widetilde{x}, \widetilde{y}))/f(\widetilde{x}, \widetilde{y})$ für alle $\widetilde{x} \in$ `Xi` und $0 \leq \widetilde{y} \leq \widetilde{x}$.
Im Programm `lnx2y2_Bd.cpp` wird dann das Maximum der Oberschranken `eps` aller Teilintervalle `Xi` $\subset B_{-1}$ berechnet. Weitere Einzelheiten und ausführliche Beispiele zur Abschätzung von Auswertefehlern findet man in [2].

In der Tabelle 5 sind in Abhängigkeit von N die berechneten relativen Fehlerschranken bei Auswertung von (1) auf Seite 9 für $\widetilde{x} \in B_{-1} = [2^{-1074}, 2^{-1022})$ zusammengestellt:

N	Rel. Fehler	N	Rel. Fehler	N	Rel. Fehler	N	Rel. Fehler
61	$7.12 \cdot 10^{-16}$	183	$6.53 \cdot 10^{-16}$	1067	$2.46 \cdot 10^{-16}$	1189	$3.08 \cdot 10^{-16}$
1220	$3.24 \cdot 10^{-16}$	1311	$3.70 \cdot 10^{-16}$	1799	$5.83 \cdot 10^{-16}$	2043	$5.17 \cdot 10^{-16}$

Tabelle 5: Relative Fehlerschranken in Abhängigkeit von N

Wie erwartet erhalten wir für $N = 1067$ die kleinste Schranke des relativen Fehlers bei Auswertung von (1) im Bereich $\widetilde{x} \in B_{-1} = [2^{-1074}, 2^{-1022})$

$$\left| \frac{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2}) - \widetilde{f}_1(\widetilde{x}, \widetilde{y})}{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2})} \right| \leq \varepsilon(f, N = 1067) = 2.459639 \cdot 10^{-16}$$

Für die restlichen Teilintervalle B_j, $j = 0, 1, \ldots, 20$ werden die relativen Fehlerschranken ganz entsprechend berechnet. Die Ergebnisse sind in der Tabelle 2 auf Seite 10 zusammengestellt. Danach gilt für alle $\widetilde{x} \in [2^{-1074}, 2^{-20})$ bei Auswertung von (1) die Abschätzung:

$$\left| \frac{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2}) - \widetilde{f}_1(\widetilde{x}, \widetilde{y})}{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2})} \right| \leq \varepsilon(f, N = 61) = 4.295508 \cdot 10^{-16}$$

Für $\widetilde{x} \in [2^{-1022}, 2^{-20})$ müssen wir jetzt noch den relativen Fehler abschätzen, wenn bei gegebenem $X = [x_1, x_2]$ im Falle $\widetilde{y} \leq x_1 \cdot 2^{-24}$ in (1) auf Seite 9 der dritte Summand vernachlässigt wird und damit (2) zur Anwendung kommt. Bevor wir den relativen Fehler $\varepsilon(g)$ bei Auswertung der Näherungsfunktion $g(x) := -N \cdot \log(2) + \log(2^N \cdot x)$ berechnen, wollen wir zunächst den relativen Approximationsfehler abschätzen. Für das Teilintervall $X = [x_1, x_2]$ gilt wegen $y \leq x_1 \cdot 2^{-24}$

$$\begin{aligned}\left|\frac{\frac{1}{2}\log(1+(\frac{y}{x})^2)}{\frac{1}{2}\log(x^2+y^2)}\right| &= \frac{\log(1+(\frac{y}{x})^2)}{|\log(x^2+y^2)|} \leq \frac{2^{-48}}{|\log(x^2+y^2)|} \leq \frac{2^{-48}}{|\log(2x_2^2)|} \\ &= \frac{2^{-48}}{|\log(2)+2\log(x_2)|} = \frac{2^{-48}}{-\log(2)-2\log(x_2)} = \varepsilon(app)\end{aligned}$$

Für den relativen Gesamtfehler gilt dann nach [2, S.176]

$$\varepsilon(f) = \varepsilon(app) + \varepsilon(g) \cdot [1 + \varepsilon(app)]$$

Die Berechnung der Fehlerschranke $\varepsilon(f)$ in den einzelnen Teilintervallen B_j erfolgt in [2] mit dem Programm `lnx2y2_Bj2.cpp` und man erhält im Bereich $\widetilde{x} \in [2^{-1022}, 2^{-20})$:

$$\left|\frac{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) - \widetilde{f}_1(\widetilde{x},\widetilde{y})}{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})}\right| \leq \varepsilon(f, N=20) = 4.524090 \cdot 10^{-16}$$

Zusammenfassung:

Für alle $\widetilde{x} \in [2^{-1074}, 2^{-20})$ und damit für alle $s := \widetilde{x}^2 + \widetilde{y}^2 \in A_1 = (0, 2^{-39})$ gilt bei Auswertung von (1) bzw. (2) auf Seite 9 für die Schranke $\varepsilon(f, A_1)$ des relativen Fehlers:

(4)
$$\boxed{\left|\frac{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) - \widetilde{f}_1(\widetilde{x},\widetilde{y})}{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})}\right| \leq \varepsilon(f, A_1) = 4.524090 \cdot 10^{-16}}$$

5.2 Fehlerabschätzung in $A_7 = [2^{40}, 2^{2049})$

Bezüglich A_7 erfolgt im Algorithmus die Abfrage

(5)
$$\widetilde{x} \geq 2^{20} \quad \Longrightarrow \quad 2^{40} \leq \widetilde{x}^2 + \widetilde{y}^2 < 2^{2049}$$

Die Fehlerabschätzung muss daher so durchgeführt werden, dass die Ungleichung rechts in (5) erfüllt wird[4]. Für $\widetilde{x} \geq 2^{+20}$ benutzen wir nach Tabelle 1:

(6)
$$f_7(x,y) = N \cdot \log(2) + \left[\log(2^{-N} \cdot x) + \frac{1}{2}\log\left(1 + \left(\frac{y}{x}\right)^2\right)\right],\ N \in \mathbb{N},$$

wobei im Falle $\widetilde{y}/\widetilde{x} \leq 2^{-24}$ der letzte Summand vernachlässigt werden kann:

(7)
$$f_7(x,y) \approx g(x) := N \cdot \log(2) + \log(2^{-N} \cdot x),\ N \in \mathbb{N};$$

[4] Beachten Sie dabei: $\widetilde{x} \leq \texttt{MaxReal} < 2^{+1024} \rightsquigarrow \widetilde{x}^2 + \widetilde{y}^2 < 2 \cdot 2^{2 \cdot 1024} = 2^{2049}$.

Die Idee bei der Anwendung von $f_7(x,y)$ besteht jetzt wieder darin, den Summanden $N \cdot \log(2)$ rechts in (6) so zu wählen, dass die Bedingung $f_7(x,y) \approx N \cdot \log(2)$ möglichst gut erfüllt ist und dass die wertbestimmende Konstante $N \cdot \log(2)$ sich im *double*-Format möglichst genau approximieren lässt. Sind beide Bedingungen erfüllt, so liefert die Auswertung der rechten Seite von (6) einen nur kleinen Fehler, da zu einem betragsmäßig großen und fast fehlerfreien Summanden ein zwar fehlerbehafteter aber nur kleiner Summand $[...]$ zu addieren ist. Die Grundidee ist also, die unvermeidbaren Rundungsfehler möglichst ganz in den betragsmäßig kleineren Summanden $[...]$ zu verlagern! Natürlich lässt sich diese Grundidee nur realisieren, wenn man den ganzen Bereich $2^{20} \le \widetilde{x} < 2^{1024}$ in geeignete Teilintervalle C_j zerlegt und in jedem C_j die Zahl $N \in I\!N$ so wählt, dass die beiden genannten Bedingungen möglichst gut erfüllt werden. Die Berechnung der N-Werte in den einzelnen C_j erfolgt wie auf Seite 9 beschrieben, und für $j = 0, 1, \ldots, 20$ findet man genau die gleichen N-Werte wie in Tabelle 2. Die Ergebnisse sind in Tabelle 6 auf Seite 16 zusammengestellt. Es soll noch betont werden, dass bei der Auswertung von $f_7(x,y)$ an Stelle von $\log(\sqrt{x^2+y^2})$ für $0 \le N < 2^{1024}/\log(2)$ und $x \ge 2^{20}$ kein Overflow enstehen kann! Für C_{18} erhält man nach Tabelle 6 mit $N = 61$ die maximale Fehlerschranke:

$$\left| \frac{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) - f_7(\widetilde{x},\widetilde{y})}{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})} \right| \le \varepsilon(f, N=61) = 4.296234 \cdot 10^{-16}$$

Im Falle $\widetilde{y} \le \widetilde{x} \cdot 2^{-24}$ muss jetzt noch nach (7) von Seit 14 der relative Approximationsfehler berechnet werden. Für $x \in X = [x_1, x_2]$ gilt die folgende Abschätzung:

$$\begin{aligned} \left| \frac{\frac{1}{2}\log(1+(\frac{y}{x})^2)}{\frac{1}{2}\log(x^2+y^2)} \right| &= \frac{\log(1+(\frac{y}{x})^2)}{|\log(x^2+y^2)|} \le \frac{2^{-48}}{\log(x^2+y^2)} \le \frac{2^{-48}}{\log(x_1^2+0)} \\ &= \frac{2^{-49}}{\log(x_1)} = \varepsilon(app) \end{aligned}$$

Mit dieser Fehlerschranke berechnet sich dann der relative Gesamtfehler bei Auswertung von (7) nach

$$\varepsilon(f) = \varepsilon(app) + \varepsilon(g) \cdot [1 + \varepsilon(app)]$$

In [2] wird $\varepsilon(f)$ für alle C_j mit dem Programm `lnx2y2_Cj2.cpp` ausgewertet. Der maximale Wert ergibt sich für das Intervall C_{20} mit $N = 20$:

$$\left| \frac{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) - \widetilde{f}_7(\widetilde{x},\widetilde{y})}{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})} \right| \le \varepsilon(f, N=20) = 4.491234 \cdot 10^{-16}$$

Zusammenfassung:

Für alle $\widetilde{x} \in [2^{20}, 2^{1024})$ und damit für alle $s := \widetilde{x}^2 + \widetilde{y}^2 \in A_7 = [2^{40}, 2^{2049})$ gilt bei Auswertung von (6) bzw. (7) auf Seite 14 für die Schranke $\varepsilon(f, A_7)$ des relativen Fehlers:

(8)
$$\boxed{\left| \frac{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) - \widetilde{f}_7(\widetilde{x},\widetilde{y})}{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})} \right| \le \varepsilon(f, A_7) = 4.491234 \cdot 10^{-16}}$$

j	C_j	Rel. Fehlerschranke	N	Abs. Schranke bez. $N \cdot \log(2)$
0	$[2^{950}, 2^{1024})$	$2.474572 \cdot 10^{-16}$	976	$4.58742009063372 \cdot 10^{-16}$
1	$[2^{900}, 2^{950})$	$2.491616 \cdot 10^{-16}$	945	$4.87688015582011 \cdot 10^{-16}$
2	$[2^{850}, 2^{900})$	$2.461457 \cdot 10^{-16}$	885	$1.34782928257584 \cdot 10^{-15}$
3	$[2^{800}, 2^{850})$	$2.445118 \cdot 10^{-16}$	825	$3.18334658073368 \cdot 10^{-15}$
4	$[2^{750}, 2^{800})$	$2.489775 \cdot 10^{-16}$	763	$1.29048653144292 \cdot 10^{-15}$
5	$[2^{700}, 2^{750})$	$2.478382 \cdot 10^{-16}$	732	$3.44056506797529 \cdot 10^{-16}$
6	$[2^{650}, 2^{700})$	$2.449179 \cdot 10^{-16}$	671	$3.15385131231069 \cdot 10^{-16}$
7	$[2^{600}, 2^{650})$	$2.553465 \cdot 10^{-16}$	610	$2.86713755664608 \cdot 10^{-16}$
8	$[2^{550}, 2^{600})$	$2.675131 \cdot 10^{-16}$	549	$2.58042380098147 \cdot 10^{-16}$
9	$[2^{500}, 2^{550})$	$2.549491 \cdot 10^{-16}$	518	$6.88387644547236 \cdot 10^{-16}$
10	$[2^{450}, 2^{500})$	$2.674233 \cdot 10^{-16}$	488	$2.29371004531686 \cdot 10^{-16}$
11	$[2^{400}, 2^{450})$	$2.588015 \cdot 10^{-16}$	427	$2.00699628965226 \cdot 10^{-16}$
12	$[2^{350}, 2^{400})$	$2.680061 \cdot 10^{-16}$	366	$1.72028253398765 \cdot 10^{-16}$
13	$[2^{300}, 2^{350})$	$2.692661 \cdot 10^{-16}$	320	$3.15522446707157 \cdot 10^{-16}$
14	$[2^{250}, 2^{300})$	$2.961951 \cdot 10^{-16}$	259	$3.44193822273618 \cdot 10^{-16}$
15	$[2^{200}, 2^{250})$	$3.034083 \cdot 10^{-16}$	229	$5.73564826805304 \cdot 10^{-16}$
16	$[2^{150}, 2^{200})$	$3.293453 \cdot 10^{-16}$	160	$1.57761223353579 \cdot 10^{-16}$
17	$[2^{100}, 2^{150})$	$3.412027 \cdot 10^{-16}$	122	$5.73427511329215 \cdot 10^{-17}$
18	$[2^{50}, 2^{100})$	$4.296234 \cdot 10^{-16}$	61	$2.86713755664608 \cdot 10^{-17}$
19	$[2^{30}, 2^{50})$	$4.119673 \cdot 10^{-16}$	40	$3.94403058383947 \cdot 10^{-17}$
20	$[2^{20}, 2^{30})$	$4.147195 \cdot 10^{-16}$	20	$1.97201529191974 \cdot 10^{-17}$

Tabelle 6: Tabelle mit den relativen Fehlern bei Auswertung von (6) mit N für alle C_j

Die Fehlerabschätzung muss jetzt noch für die Quadratsummen $s := \widetilde{x}^2 + \widetilde{y}^2$ in folgendem Intervall durchgeführt werden:

$$2^{-39} \leq s < 2^{40}, \quad s := \widetilde{x}^2 + \widetilde{y}^2, \quad 0 \leq \widetilde{y} \leq \widetilde{x}; \tag{9}$$

Beachten Sie bitte, dass die Berechnung von s jetzt weder zum Underflow noch zum Overflow führen kann, wenn s im Akkumulator `dot` rundungsfehlerfrei berechnet und mit der Anweisung `s=rnd(dot)` zur nächsten *double*-Zahl $\mathtt{s} \in S(2, 53)$ gerundet wird. Die Schranke $\varepsilon(s)$ des dabei auftretenden relativen Fehlers ε_s ist dann gegeben durch:

$$\frac{s - \mathtt{s}}{s} = \varepsilon_s, \quad |\varepsilon_s| \leq \varepsilon(s) := 2^{-53} = 1.1102\ldots \cdot 10^{-16}; \tag{10}$$

5.3 Fehlerabschätzung in $A_4 = [1 - \frac{11}{64}, 1 + \frac{11}{64}]$

Man könnte jetzt annehmen, dass die Auswertung unserer Funktion

$$f(x,y) := \log(\sqrt{x^2+y^2}) \equiv \frac{1}{2}\log(x^2+y^2)$$

mit dem rechten Term $\frac{1}{2}\log(x^2+y^2)$ wegen der vergleichsweise kleinen Fehlerschranke $\varepsilon(s) = 2^{-53}$ jetzt problemlos möglich ist. Im Falle $s \approx 1$ ist dies jedoch ein Irrtum, da wir dann die Logarithmus-Funktion in der Näher ihrer Nullstelle mit fehlerbehafteten Argumenten aufrufen und damit erhebliche relative Fehler erzeugen. Diese Schwierigkeiten umgeht man bei Anwendung der Identität

$$f(x,y) := \log(\sqrt{x^2+y^2}) \equiv \frac{1}{2}\cdot\log(1+[x^2+y^2-1]) =: f_4(x,y), \tag{11}$$

wobei man $r := x^2+y^2-1$ im Akkumulator rundungsfehlerfrei berechnet und die rechte Seite von (11) mit Hilfe der C-XSC Funktion `lnp1(r)` $\approx \log(1+\mathtt{r})$ auswertet. Mit $\mathtt{r} = \widetilde{r}$ bezeichnen wir den aus dem Akku ausgelesenen Maschinenwert, wobei gilt:

$$\widetilde{r} = r\cdot(1+\varepsilon_r), \quad \text{mit: } |\varepsilon_r| \le 2^{-53} = \varepsilon(r)$$

Mit $\varepsilon(r) = 2^{-53}$ setzen wir voraus, dass $\widetilde{r}$ und r beide entweder verschwinden oder beide im normalisierten Bereich liegen. Für $\widetilde{x} = 1$ und $\widetilde{y} \to 0$ ist diese Voraussetzung jedoch nicht erfüllt, da $r = \widetilde{y}^2$ jetzt durchaus in den denormalisierten Bereich fallen kann. Wir betrachten daher zunächst den Fall:

$\widetilde{x} = 1,\ 0 \le \widetilde{y} < 2^{-28}$;

Mit $r = y^2$ und wegen $\widetilde{y} < 2^{-28}$ erhalten wir

$$f(x,y) := \log(\sqrt{x^2+y^2}) \equiv \frac{1}{2}\cdot\log(1+y^2) \approx \frac{1}{2}\cdot y^2 =: f_{4a}(y), \tag{12}$$

und $f_{4a}(\widetilde{y})$ kann jetzt für $\widetilde{y} \to 0$ in den Underflow-Bereich fallen. In diesem Fall ist dann nur eine Schranke des absoluten Fehlers $|f_4(1,\widetilde{y}) - \widetilde{f}_{4a}(\widetilde{y})|$ zu berechnen, wobei das Maschinenergebnis $\widetilde{f}_{4a}(\widetilde{y})$ wie folgt definiert wird:

$$\widetilde{f}_{4a}(\widetilde{y}) := (0.5 \odot \widetilde{y}) \odot \widetilde{y} \tag{13}$$

Um entscheiden zu können, für welche $\widetilde{y} \in S(2,53)$ die Maschinenwerte $\widetilde{f}_{4a}(\widetilde{y})$ in den denormalisierten Bereich fallen können, stellen wir folgende Frage:
Für welches $y_0 \in S(2,53)$ gilt:

1. $\widetilde{y} < y_0 \implies \widetilde{f}_{4a}(\widetilde{y}) = (0.5 \odot \widetilde{y}) \odot \widetilde{y} < \mathtt{MinReal} = 2^{-1022}$, falls bei beiden Multiplikationen abgerundet wird.
2. $\widetilde{y} \ge y_0 \implies \widetilde{f}_{4a}(\widetilde{y}) = (0.5 \odot \widetilde{y}) \odot \widetilde{y} \ge \mathtt{MinReal} = 2^{-1022}$, falls die beiden Multiplikationen im beliebigen Rundungsmodus erfolgen.

Mit dem in [2] angegebenen Programm `lnx2y2_y0.cpp` erhält man das Ergebnis:

$$\begin{aligned} y_0 \;=\; & 6369051672525773.0/ \\ & 30191699398572330817932436647906151127335369763 33 \\ & 15234270096504019649932991371908166890138014212 70 \\ & 14033174700024611075919816467703939834106049147 40 \\ & 11461568349195162615808.0 \approx \sqrt{2}\cdot 2^{-511} \end{aligned}$$

Für $\widetilde{x} = 1$ und $0 \le \widetilde{y} \le y_0$ berechnen wir jetzt zunächst eine Schranke des absoluten Auswertefehlers $|\widetilde{f}_{4a} - \frac{1}{2}\widetilde{y}^2|$. Mit dem in [2] angegebenen Programm `lnx2y2_abs.cpp` erhält man das Ergebnis:

$$|\widetilde{f}_{4a} - \frac{1}{2}\widetilde{y}^2| = |(0.5 \odot \widetilde{y}) \odot \widetilde{y} - \frac{1}{2}\widetilde{y}^2| \le 2.225074 \cdot 10^{-308} = \Delta(f_{4a})$$

Wir benötigen jetzt noch eine garantierte Oberschranke des absoluten Approximationsfehlers $|\frac{1}{2}\cdot\log(1+y^2) - \frac{1}{2}\cdot y^2|$, ebenfalls für $0 \le \widetilde{y} \le y_0$. Da $\log(1+y^2)$ eine alternierende Leibniz-Reihe besitzt, gilt:

$$\left|\frac{1}{2}\cdot\log(1+y^2) - \frac{1}{2}\cdot y^2\right| \le \frac{1}{2}\cdot\frac{1}{2}\cdot y^4 \le \frac{1}{4}\cdot y_0^4 < 2^{-2043} \sim 10^{-615}$$

Nach [2, Seite 177] folgt dann für den absoluten Gesamtfehler

$$\left|\frac{1}{2}\cdot\log(1+\widetilde{y}^2) - \widetilde{f}_{4a}(\widetilde{y})\right| \le \Delta(f_{4a}) + 2^{-2043} < 2.225075 \cdot 10^{-308} = \Delta(f_4)$$

Zusammenfassung: Für $\widetilde{x} = 1$ und $0 \le \widetilde{y} \le y_0$ gilt mit $f_{4a}(y) := \frac{1}{2}\cdot y^2$

(14)
$$\boxed{\left|\log(\sqrt{1+\widetilde{y}^2}) - \widetilde{f}_{4a}(\widetilde{y})\right| \le \Delta(f_4) = 2.225075 \cdot 10^{-308}}$$

Für $\widetilde{x} = 1$ bleibt jetzt noch der Bereich $y_0 \le \widetilde{y} < 2^{-28}$, in dem $f_{4a}(\widetilde{y}) := \frac{1}{2}\cdot\widetilde{y}^2$ und $\widetilde{f}_{4a}(\widetilde{y}) := (0.5 \odot \widetilde{y}) \odot \widetilde{y}$ nach Seite 17 bei beliebigem Rundungsmodus beide im normalisierten Zahlenbereich liegen, so dass jetzt wieder eine Schranke des relativen Fehlers berechnet werden kann. Wegen $0.5 \odot \widetilde{y} = 0.5 \cdot \widetilde{y}$ ist der relative Auswertefehler bez. $\widetilde{f}_{4a}(\widetilde{y}) := (0.5 \odot \widetilde{y}) \odot \widetilde{y} = (0.5 \cdot \widetilde{y}) \odot \widetilde{y}$ nur durch die letzte Multiplikation gegeben, und es gilt bei vorausgesetzter hochgenauer Arithmetik $\varepsilon(\odot) = 2^{-52} = 2.220446\ldots\cdot 10^{-16}$. Der relative Approximationsfehler lässt sich wie folgt abschätzen:

$$\begin{aligned} \left|\frac{\frac{1}{2}\cdot\log(1+y^2) - \frac{1}{2}\cdot y^2}{\frac{1}{2}\cdot\log(1+y^2)}\right| \;&=\; \frac{|\log(1+y^2) - y^2|}{\log(1+y^2)} \le \frac{\frac{1}{2}y^4}{\log(1+y^2)} \le \frac{2^{-113}}{\log(1+2^{-56})} \\ &<\; 6.938894 \cdot 10^{-18} = \varepsilon(app), \end{aligned}$$

wobei der letzte Bruch durch Intervallrechnung eingeschlossen wurde und $\varepsilon(app)$ die berechnete Intervallobergrenze ist. Mit $\varepsilon(f_{4a}) = \varepsilon(app) + \varepsilon(\odot)\cdot[1+\varepsilon(app)]$ folgt dann für die Schranke $\varepsilon(f_{4a})$ des relativen Gesamtfehlers

$$\varepsilon(f_{4a}) = 2.289836 \cdot 10^{-16}$$

Zusammenfassung: Für $\widetilde{x} = 1$ und $y_0 \le \widetilde{y} < 2^{-28}$ gilt mit $f_{4a}(y) := \frac{1}{2} \cdot y^2$

$$\left| \frac{\log(\sqrt{1+\widetilde{y}^2}) - \widetilde{f}_{4a}(\widetilde{y})}{\log(\sqrt{1+\widetilde{y}^2})} \right| \le \varepsilon(f_{4a}) = 2.289836 \cdot 10^{-16} \tag{15}$$

Es fehlt jetzt noch die Fehlerabschätzung im Bereich $s := \widetilde{x}^2 + \widetilde{y}^2 \in A_4 = [1 - \frac{11}{64}, 1 + \frac{11}{64}]$, wobei $(\widetilde{x} \neq 1 \vee \widetilde{y} \ge y_0)$ erfüllt sein muss. Nach (11) von Seite 17 erfolgt die Auswertung von $f(x,y) := \log(\sqrt{x^2+y^2})$ in diesem Bereich durch

$$f(x,y) \equiv \frac{1}{2} \cdot \log(1 + [x^2 + y^2 - 1]) =: f_4(x,y),$$

wobei $r := x^2 + y^2 - 1$ im Akkumulator exakt berechnet wird. Zur Fehlerabschätzung benötigen wir sowohl die relative Fehlerschranke, die beim Auslesen des Akkumulators ins *double*-Format entsteht, als auch eine Einschließung aller exakten r-Werte. Zur Berechnung dieser Einschließung ist zu beachten, dass $s := x^2 + y^2$ ebenfalls im Akku exakt berechnet und mit `s = rnd(dot)` ins *double*-Format gerundet wird. Es gilt daher

$$\widetilde{s} = \mathtt{s} = s \cdot (1 + \varepsilon_s), \quad \text{mit } |\varepsilon_s| \le \varepsilon(s) = 2^{-53}$$

Bevor $r = x^2 + y^2 - 1$ im Akku rundungsfehlerfrei berechnet wird, erfolgt die Abfrage: `if (s >= 0.828125 && s <= 1.171875) {...}`, daraus ergeben sich die folgenden Einschließungen:

$$\begin{aligned}
& 1 - \frac{11}{64} \le \widetilde{s} \le 1 + \frac{11}{64} \quad \Longleftrightarrow \quad 1 - \frac{11}{64} \le s \cdot (1+\varepsilon_s) \le 1 + \frac{11}{64} \\
\Longleftrightarrow \quad & \frac{1 - \frac{11}{64}}{1+\varepsilon_s} \le s \le \frac{1 + \frac{11}{64}}{1+\varepsilon_s} \quad \Longleftrightarrow \quad \frac{1 - \frac{11}{64}}{1+\varepsilon_s} - 1 \le r \le \frac{1 + \frac{11}{64}}{1+\varepsilon_s} - 1 \\
\Longleftrightarrow \quad & \frac{-\frac{11}{64} - \varepsilon_s}{1+\varepsilon_s} \le r \le \frac{\frac{11}{64} - \varepsilon_s}{1+\varepsilon_s} \\
\Longrightarrow \quad & \frac{-\frac{11}{64} - \varepsilon(s)}{1+\varepsilon(s)} \le r \le \frac{\frac{11}{64} + \varepsilon(s)}{1-\varepsilon(s)}, \quad \varepsilon(s) := 2^{-53}; \qquad (16)
\end{aligned}$$

Wertet man die Brüche in (16) intervallmäßig aus, so erhält man für r die Einschließung:

$$-0.17187501 \le r \le +0.17187501 \tag{17}$$

Im Programm `lnx2y2.cpp` [2], in dem der Algorithmus zur Berechnung von $f(\widetilde{x}, \widetilde{y})$ angegeben ist, gilt `dot` $= r$, und durch `r = rnd(dot)` wird der exakte r-Wert in die *double*-Variable `r` $= \widetilde{r}$ ausgelesen. Für den dabei auftretenden relativen Fehler ε_r gilt:

$$\widetilde{r} = r \cdot (1 + \varepsilon_r), \quad |\varepsilon_r| \le \varepsilon(r) = 2^{-53}; \tag{18}$$

Mit $\varepsilon(r) = 2^{-53}$ setzen wir dabei voraus, dass r und $\widetilde{r}$ stets im normalisierten Zahlenbereich liegen. Würde r in den denormalisierten Bereich fallen[5], so müsste man nach [2,

[5] Aus $r = 0$ folgt $\widetilde{r} = 0$, und der relative Fehler kann Null gesetzt werden.

Seite 29, oben] für $\varepsilon(r)$ sehr viel größere Schranken wählen! Von der Anschauung her ist es jedoch keinesfalls sicher, dass mit $r := \widetilde{x}^2 + \widetilde{y}^2 - 1$ und $|r| > 0$ stets gelten soll: $|r| > 2^{-1022}$. Die Frage kann auch so gestellt werden: Warum sollte es in der $(\widetilde{x}, \widetilde{y})$-Ebene nicht Gitterpunkte $P(\widetilde{x}|\widetilde{y})$, mit $\widetilde{x}, \widetilde{y} \in S(2, 53)$ geben, die so dicht am Einheitskreis liegen, dass gilt: $0 < |r| < 2^{-1022}$? Zur Beantwortung der Frage formulieren wir den

Satz 5.1 Unter den Voraussetzungen

1. $\frac{1}{2}\sqrt{2} < \widetilde{x}$ und $\widetilde{x} \neq 1$
2. $0 \leq \widetilde{y} \leq \widetilde{x}; \quad \widetilde{x}, \widetilde{y} \in S(2, 53)$
3. $r := \widetilde{x}^2 + \widetilde{y}^2 - 1 \neq 0$

gilt: $\quad |\widetilde{x}^2 + \widetilde{y}^2 - 1| \geq 2^{-158} = 2.73691106\ldots \cdot 10^{-48}$

Den Beweis dieses Satzes findet man in [2, Seite 238]. Danach gilt also $|r| \geq 2^{-158}$, so dass $|r| > 0$ stest im normalisierten Bereich liegt. Nach (17) und wegen $|r| \geq 2^{-158}$ ist die Fehlerabschätzung damit durchzuführen in den beiden Bereichen

$$-0.17187501 \leq r \leq -2^{-158} \qquad \text{und} \qquad 2^{-158} \leq r \leq +0.17187501$$

Da wegen $(\widetilde{x} \neq 1 \ \vee \ \widetilde{y} \geq y_0)$ alle Werte $r = \widetilde{x}^2 + \widetilde{y}^2 - 1$ im normalisierten Bereich liegen, gilt nach (18) von Seite 19 für den relativen Auslesefehler ε_r die Abschätzung: $|\varepsilon_r| \leq \varepsilon(r) = 2^{-53}$. Für den relativen Fehler

$$\varepsilon_{f_4} := \frac{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2}) - 0.5 \odot \texttt{lnp1(r)}}{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2})}$$

liefert in [2] das Programm

```
// Program lnx2y2_ser.cpp for calculating the relative
// error by evaluating the term ln(1+[x^2+y^2-1]);

#include "abs_relh.hpp" // For abs_addh1, ...
#include "bnd_util.hpp" // For Max_bnd_Xi()
#include <iostream>     // For cout

using namespace cxsc;
using namespace std;

real T_x(const interval& Xi, const real& delx)
{
    interval A;
    real eps;
    rel_lnp1_rel(Xi,delx,A,eps);
    return eps;
}
```

```
int main()
{
    interval X = interval(-0.17187502,comp(-0.5,-157));
    real bnd, diam = 1e-5, epsr=comp(0.5,-52);
    // epsr: Upper bound of the relative error by
    //       reading the accumulator.
    Max_bnd_Xi(T_x,X,epsr,diam,bnd);
    cout << RndUp << "Relative error bound = "
         << bnd << endl;
}
```

die relative Fehlerschranke

$$|\varepsilon_{f_4}| \le \varepsilon(f_4) = 3.730016 \cdot 10^{-16}, \text{ falls } -0.17187501 \le r \le -2^{-158};$$

Im Bereich $2^{-158} \le r \le +0.17187501$ liefert das Programm `lnx2y2_ser.cpp` mit der Zeile `interval X = interval(comp(0.5,-157),0.17187502);` das Ergebnis: $|\varepsilon_{f_4}| \le \varepsilon(f_4) = 3.618424 \cdot 10^{-16}$.

Zusammenfassung:

Mit $r := \widetilde{x}^2 + \widetilde{y}^2 - 1$ gilt im Bereich $-0.17187501 \le r \le +0.17187501$ unter der Bedingung $(\widetilde{x} \ne 1 \ \vee \ \widetilde{y} \ge y_0)$ die Abschätzung:

(19)
$$\boxed{\left| \frac{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2}) - 0.5 \odot \texttt{lnp1(r)}}{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2})} \right| \le \varepsilon(f, A_4) = 3.730016 \cdot 10^{-16}}$$

Im Programm `lnx2y2.cpp` aus [2, S. 251] wird in den Bereichen $\widetilde{s} \in A_3 = [\frac{1}{4}, 1 - \frac{11}{64}]$ und $\widetilde{s} \in A_5 = [1 + \frac{11}{64}, \frac{7}{4}]$ die Auswertung von $\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2})$ realisiert durch:

(20)
$$\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2}) = \frac{1}{2} \cdot \left[\log(r) + \log(1 + \frac{r}{r_1}) \right], \quad \widetilde{x}^2 + \widetilde{y}^2 = r + r_1,$$

dabei wird $r + r_1$ auf dem Rechner approximiert durch den folgenden Algorithmus:

1. Exakte Addition von $\widetilde{x}^2, \widetilde{y}^2$ im Akkumulator: `dot` $= \widetilde{x}^2 + \widetilde{y}^2$;
2. $r =$ `rnd(dot)`; Rundung zur nächsten Rasterzahl
3. `dot` $=$ `dot` $- r$; d.h. `dot` $= \widetilde{x}^2 + \widetilde{y}^2 - r =: r_1$, d.h. $\widetilde{x}^2 + \widetilde{y}^2 = r + r_1$;
4. `r1` $=$ `rnd(dot);` d.h. `r1` $\in S(2, 53)$;

Es gilt damit $\widetilde{x}^2 + \widetilde{y}^2 = r + r_1 \approx r +$ `r1` in hoher Genauigkeit, wobei $r \in S(2, 53)$ als exakt aufzufassen ist und `r1` $\approx r_1$ der fehlerbehaftete Summand ist.

5.4 Fehlerabschätzung in $A_3 = [\frac{1}{4}, 1 - \frac{11}{64}]$

Zunächst gilt wieder $\widetilde{s} \in A_3$, wobei die *double*-Zahl $\mathtt{s} = \widetilde{s}$ die exakte Summe $\widetilde{x}^2 + \widetilde{y}^2$ approximiert. Wegen $\widetilde{s} = \mathtt{s} = \mathtt{rnd(dot)}$, mit $\mathtt{dot} = s := \widetilde{x}^2 + \widetilde{y}^2$ ergibt sich[6]:

$$\mathtt{s} = (\widetilde{x}^2 + \widetilde{y}^2) \cdot (1 + \varepsilon_s), \quad |\varepsilon_s| \le \varepsilon(s) = 2^{-53}; \tag{21}$$

Für die Fehlerabschätzungen benötigen wir nach [2, Seite 53] stets Einschließungen der exakten Operanden, hier also eine Einschließung der s-Werte. Nach (21) gilt:

$$\mathtt{s} = s \cdot (1 + \varepsilon_s) \ge \frac{1}{4} \quad \rightsquigarrow \quad s \ge \frac{0.25}{1 + \varepsilon_s} \ge \frac{0.25}{1 + \varepsilon(s)} > 0.24999999$$

$$\mathtt{s} = s \cdot (1 + \varepsilon_s) \le 1 - \frac{11}{64} \quad \rightsquigarrow \quad s \le \frac{1 - \frac{11}{64}}{1 + \varepsilon_s} \le \frac{1 - \frac{11}{64}}{1 - \varepsilon(s)} < 0.82812501$$

Für $\widetilde{s} = \mathtt{s} \in A_3$ werden die exakten Summen $s = \widetilde{x}^2 + \widetilde{y}^2$ daher eingeschlossen durch:

$$s = \widetilde{x}^2 + \widetilde{y}^2 \in S_3 := [0.24999999, 0.82812501]$$

Für die Fehlerabschätzung betrachten wir ein Teilintervall $T := [\mathtt{t1}, \mathtt{t2}] \subset S_3$, d.h. $\widetilde{x}^2 + \widetilde{y}^2 \equiv r + r_1 \in T$ und benötigen nach (20) Einschließungen der exakten Werte r und r_1. Für alle $s \in T$ sei $\widetilde{x}^2 + \widetilde{y}^2 = \mathtt{t1}$ der kleinstmögliche Wert. Daraus folgt:

$$\begin{aligned} r &= \mathtt{rnd(t1)}, \quad \text{d.h. } r = \mathtt{t1} \cdot (1 + \varepsilon_s), \quad |\varepsilon_s| \le \varepsilon(s) = 2^{-53} \\ \mathtt{r1} &= \mathtt{rnd(t1} - r\mathtt{)} = \mathtt{rnd(} - \mathtt{t1} \cdot \varepsilon_s\mathtt{)}, \quad r_1 := \mathtt{t1} - r = -\mathtt{t1} \cdot \varepsilon_s \\ &\rightsquigarrow \; r \ge \mathtt{t1} \cdot (1 - \varepsilon(s)); \qquad r_1 \ge -\mathtt{t1} \cdot \varepsilon(s); \end{aligned}$$

Für alle $s \in T$ sei $\widetilde{x}^2 + \widetilde{y}^2 = \mathtt{t2}$ der größtmögliche Wert. Daraus ergibt sich:

$$\begin{aligned} r &= \mathtt{rnd(t2)}, \quad \text{d.h. } r = \mathtt{t2} \cdot (1 + \varepsilon_s), \quad |\varepsilon_s| \le \varepsilon(s) = 2^{-53} \\ \mathtt{r1} &= \mathtt{rnd(t2} - r\mathtt{)} = \mathtt{rnd(} - \mathtt{t2} \cdot \varepsilon_s\mathtt{)}, \quad r_1 := \mathtt{t2} - r = -\mathtt{t2} \cdot \varepsilon_s \\ &\rightsquigarrow \; r \le \mathtt{t2} \cdot (1 + \varepsilon(s)); \qquad r_1 \le \mathtt{t2} \cdot \varepsilon(s); \end{aligned}$$

Wir erhalten damit für r und r_1 die folgenden Einschließungen:

$$\mathtt{t1} \cdot (1 - 2^{-53}) \le r \le \mathtt{t2} \cdot (1 + 2^{-53}) \tag{22}$$

$$-\mathtt{t1} \cdot 2^{-53} \le r_1 \le \mathtt{t2} \cdot 2^{-53} \tag{23}$$

Wir betrachten r als rundungsfehlerfrei und benötigen für die folgende Fehlerabschätzung jetzt noch eine Oberschranke des absoluten Fehlers $|\mathtt{r1} - r_1|$. Nimmt man zunächst an, dass die $\mathtt{r1} \in S(2, 53)$ alle im normalisierten Bereich liegen, so gilt wegen $\mathtt{t2} \in [0.24999999, 0.82812501]$ nach Definition des relativen Fehlers:

$$|\mathtt{r1} - r_1| = |\varepsilon_s| \cdot |r_1| \le \varepsilon(s) \cdot |r_1| \le \varepsilon(s) \cdot (\mathtt{t2} \cdot 2^{-53}) = \mathtt{t2} \cdot 2^{-106}$$

Falls die $\mathtt{r1}$ jedoch in den denormalisierten Bereich fallen, so ist der absolute Fehler beschränkt durch $\mathtt{MinReal} = 2^{-1022} < \mathtt{t2} \cdot 2^{-106}$, so dass der obige Wert $\mathtt{t2} \cdot 2^{-106}$

[6] Die Summen $\widetilde{x}^2 + \widetilde{y}^2$ liegen jetzt alle im normalisierten Bereich, und daher gilt $\varepsilon(s) = 2^{-53}$.

auch dann eine gültige Oberschranke bleibt! Der bei Auswertung von (20) auftretende relative Fehler

$$\varepsilon_{f_3} := \frac{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) - 0.5 \odot [\texttt{ln}(r) \oplus \texttt{lnp1}(\texttt{r1} \oslash r)]}{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})}$$

wird in [2, Seite 243] abgeschätzt mit dem C-XSC Programm `lnx2y2_A3.cpp`, und man erhält mit $\widetilde{f_3} := 0.5 \odot [\texttt{ln}(r) \oplus \texttt{lnp1}(\texttt{r1} \oslash r)]$ für alle $s = \widetilde{x}^2 + \widetilde{y}^2 \in S_3 = [0.24999999, 0.82812501] \supset A_3$ für den relativen Auswertefehler die folgende Abschätzung:

(24)
$$\left| \frac{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) - \widetilde{f_3}}{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})} \right| \leq \varepsilon(f, A_3) = 5.160435 \cdot 10^{-16}$$

5.5 Fehlerabschätzung in $A_5 = [1 + \frac{11}{64}, \frac{7}{4},]$

Die Fehlerabschätzung in diesem Teilbereich erfolgt völlig analog zum Bereich A_3. Die exakten Summen $s := \widetilde{x}^2 + \widetilde{y}^2$ werden jedoch jetzt eingeschlossen durch:

$$s = \widetilde{x}^2 + \widetilde{y}^2 \in S_5 := [1.17187499999, 1.7500001] \supset A_5$$

Die Berechnung einer Oberschranke des relativen Auswertefehlers kann daher auch mit dem gleichen Programm `lnx2y2_A3.cpp` aus [2, Seite 243] erfolgen, in dem die Programmzeile `interval X; string("0.24999999,...` zu ersetzen ist durch: `interval X; string("1.17187499999,1.7500001") >> X;` Das Programm liefert dann für $s = \widetilde{x}^2 + \widetilde{y}^2 \in S_5 = [1.17187499999, 1.7500001] \supset A_5$ mit $\widetilde{f_5} := 0.5 \odot [\texttt{ln}(r) \oplus \texttt{lnp1}(\texttt{r1} \oslash r)]$ für den relativen Auswertefehler die folgende Abschätzung:

(25)
$$\left| \frac{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) - \widetilde{f_5}}{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})} \right| \leq \varepsilon(f, A_5) = 5.160563 \cdot 10^{-16}$$

Die Fehlerabschätzung ist jetzt noch für die Bereiche $2^{-39} \leq \widetilde{x}^2 + \widetilde{y}^2 \ \wedge \ \widetilde{s} \leq \frac{1}{4}$ und $\frac{7}{4} \leq \widetilde{s} \ \wedge \ \widetilde{x}^2 + \widetilde{y}^2 \leq 2^{40}$ durchzuführen, in denen $\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})$ wie folgt ausgewertet wird:

$$\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) = \frac{1}{2} \cdot \log(\widetilde{x}^2+\widetilde{y}^2) \approx 0.5 \odot \texttt{ln}(\texttt{s})$$

Die Summe $s := \widetilde{x}^2 + \widetilde{y}^2 = \texttt{dot}$ wird im Akkumulator zunächst rundungsfehlerfrei berechnet. Danach wird diese Summe mit $\widetilde{s} = \texttt{s} = \texttt{rnd}(\texttt{dot})$ zur nächsten *double*-Zahl $\widetilde{s} = \texttt{s}$ gerundet, und die C-XSC Funktion `ln(...)` wird dann mit diesem Argument `s` ausgewertet. Da alle s im normalisierten Bereich liegen, gilt $\texttt{s} = s \cdot (1 + \varepsilon_s)$, mit $|\varepsilon_s| \leq \varepsilon(s) = 2^{-53}$, und die exakten Summen $s := \widetilde{x}^2 + \widetilde{y}^2$ werden in den beiden Bereichen eingeschlossen durch:

(26) $s \in S_2 := [2^{-39}, 0.25000001] \supset A_2, \qquad s \in S_6 := [1.74999999, 2^{40}] \supset A_6;$

5.6 Fehlerabschätzung in $A_2 = [2^{-39}, \frac{1}{4}]$

Die exakten Summen $s := \widetilde{x}^2 + \widetilde{y}^2$ werden nach (26) in diesem Bereich eingeschlossen durch $s \in S_2 := [2^{-39}, 0.25000001] \supset A_2$, und der relative Fehler

$$\varepsilon_{f_2} := \frac{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2}) - 0.5 \odot \mathtt{ln(s)}}{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2})}, \qquad \mathtt{s} = \mathtt{rnd}(\widetilde{x}^2 + \widetilde{y}^2)$$

wird in [2, Seite 246] abgeschätzt mit dem C-XSC Programm `lnx2y2_A2.cpp`, das für alle exakten $s = \widetilde{x}^2 + \widetilde{y}^2 \in S_2 = [2^{-39}, 0.25000001] \supset A_2$ mit $\widetilde{f_2} := 0.5 \odot \mathtt{ln(s)}$ für den relativen Auswertefehler die folgende Oberschranke liefert:

(27)
$$\boxed{\left| \frac{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2}) - \widetilde{f_2}}{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2})} \right| \leq \varepsilon(f, A_2) = 3.740657 \cdot 10^{-16}}$$

5.7 Fehlerabschätzung in $A_6 = [\frac{7}{4}, 2^{+40}]$

Die exakten Summen $s := \widetilde{x}^2 + \widetilde{y}^2$ werden nach (26) in diesem Bereich eingeschlossen durch $s \in S_6 := [1.74999999, 2^{40}] \supset A_6$, und der relative Fehler

$$\varepsilon_{f_6} := \frac{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2}) - 0.5 \odot \mathtt{ln(s)}}{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2})}, \qquad \mathtt{s} = \mathtt{rnd}(\widetilde{x}^2 + \widetilde{y}^2)$$

wird in [2] mit dem C-XSC Programm `lnx2y2_A2.cpp` abgeschätzt, in dem lediglich die Zeile `interval X; string("[1.8189894E-12,...` zu ersetzen ist durch: `interval X; string("[1.74999999,1.09951163E+12]") >> X;`

Das Programm liefert dann für alle $s = \widetilde{x}^2 + \widetilde{y}^2 \in S_6 = [1.74999999, 2^{40}] \supset A_6$ mit $\widetilde{f_6} := 0.5 \odot \mathtt{ln(s)}$ für den relativen Auswertefehler die folgende Abschätzung:

(28)
$$\boxed{\left| \frac{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2}) - \widetilde{f_6}}{\log(\sqrt{\widetilde{x}^2 + \widetilde{y}^2})} \right| \leq \varepsilon(f, A_6) = 4.923703 \cdot 10^{-16}}$$

Damit haben wir in allen Teilbereichen A_i eine absolute oder relative Fehlerschranke bei Auswertung der jeweiligen Funktionen $f_i(\widetilde{x}, \widetilde{y})$ berechnet.

Zusammenfassung:

Im Falle $\widetilde{x} = 1$ und $0 \leq \widetilde{y} \leq y_0$ gilt mit $f_{4a}(y) := \frac{1}{2} \cdot y^2$

(29) $$\boxed{\left|\log(\sqrt{1+\widetilde{y}^2}) - \widetilde{f}_{4a}(\widetilde{y})\right| \leq \Delta(f_4) = 2.225075 \cdot 10^{-308}}$$

Sonst gilt für $|\widetilde{x}| + |\widetilde{y}| > 0$ in allen Teilbereichen A_i die Abschätzung:

(30) $$\boxed{\left|\frac{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2}) - \widetilde{f}_i(\widetilde{x},\widetilde{y})}{\log(\sqrt{\widetilde{x}^2+\widetilde{y}^2})}\right| \leq \varepsilon(f) = 5.160563 \cdot 10^{-16}}$$

$y_0 \in S(2,53)$ ist auf Seite 18 als Quotient zweier Maschinenzahlen angegeben, wobei diese Maschinenzahlen in dezimaler Form exakt dartellbar sind.

6 Intervallargumente

Durch die Vereinbarung `interval x,y;` werden Maschinenintervalle $\mathtt{x}, \mathtt{y} \in IR$ vorgegeben, und für alle reellen $x \in \mathtt{x}$ und $y \in \mathtt{y}$ ist eine Maschineneinschließung `W` des Wertebereichs

(31) $$W_f := \left\{ f(x,y) \in I\!R \,\middle|\, f(x,y) := \log(\sqrt{x^2+y^2}) \ \wedge\ x \in \mathtt{x} \ \wedge\ y \in \mathtt{y} \right\} \subset \mathtt{W}$$

gesucht. Wegen der Quadrate x^2, y^2 kann man sich durch[7] `x=abs(x)` und `y=abs(y)` auf die nicht negativen Argumente $x \in \mathtt{x}$ und $y \in \mathtt{y}$ beschränken. Da die Funktion $\log(\sqrt{s})$ in Abhängigkeit von $s = x^2 + y^2$ streng monoton wächst, ist die Implementierung einer Einschließung $\mathtt{W} \in I\,R$ relativ einfach. Zur Berechnung von `Sup(W)` ruft man die Funktion `real ln_sqrtx2y2(const real& x, const real& y)` mit den Argumenten `Sx=Sup(x)` und `Sy=Sup(y)` auf und erhält so mit der Anweisung `f = ln_sqrtx2y2(Sx,Sy)` den Maschinenwert $\mathtt{f} \in S(2,53)$ als Approximation des exakten Funktionswertes $\log(\sqrt{\mathtt{Sx}^2 + \mathtt{Sy}^2}) \approx \mathtt{f}$. Liegt diese Maschinennäherung `f` im normalisierten Zahlenbereich, so muss im Falle $\mathtt{f} \geq 0$ dieser Wert noch multipliziert werden mit `q_lnx2y2p` $\in S(2,53)$, wobei der Quotient

```
q_lnx2y2p = 4503599627370502.0 / 4503599627370496.0
```

vorher mit Hilfe der C-XSC Funktion `eps2fractions(str,Z1p,N1p,Z1m,N1m)` berechnet wird. Die Zeichenkette `str = "5.160563E-16"` ist dabei die relative Fehlerschranke aus (30), und die Rückgabewerte `Z1p=4503599627370502.0` und `N1p=4503599627370496.0` sind Zähler und Nenner von `q_lnx2y2p`. Wir erhalten so mit `u2 = f*q_lnx2y2p = Sup(W)` die gesuchte Oberschranke `Sup(W)` der

[7] Mit `x1=abs(x)` erhält man das Intervall `x1` der Beträge aller $x \in \mathtt{x}$.

Einschlieung `W` $\supset W_f$. Weitere Einzelheiten zur Funktion `eps2fractions()` findet man in [2, Seite 83]. Liegt `f` jedoch im denormalisierten Bereich, so muss zur Berechnung von `Sup(W)` noch die absolute Fehlerschranke `ln_x2y2_abs = 2.225076E-308` addiert werden. Im Vergleich zur Schranke $\Delta(f_4) = 2.225075 \cdot 10^{-308}$ aus (29) wurde dieser Wert so aufgerundet, dass bei der Addition `u2 = f + ln_x2y2_abs` keine zeitaufwendige gerichtete Rundung notwendig ist, d.h. auch wenn der Rechner gerade im Abrundungsmodus laufen sollte, wird durch die genannte Addition mit `u2` stets eine garantierte Oberschranke ermittelt.

Die Berechnung einer Unterschranke `Inf(W)` erfolgt ganz analog. Wenn jedoch im Falle `(Ix==1 && Iy<b0 || Iy==1 && Ix<b0)` die mit Hilfe der Anweisung `f = ln_sqrtx2y2(Ix,Iy)` berechneten Funktionswerte `f` im denormalisierten Bereich liegen und negativ sind, so kann die Unterschranke `u1 = f - ln_x2y2_abs` auf Null gesetzt werden, denn für die exakten Funktionswerte gilt dann z.B. $f(x,y) = f(1,y) = \frac{1}{2}\ln(1+y^2) \geq 0$. Weitere Einzelheiten dazu findet man im Quelltext ab Seite 28. Die Funktion zur Berechnung des einschließenden Intervalls `W` wird deklariert durch:

```
interval ln_sqrtx2y2(const interval& x, const interval& y);
```

7 Numerische Ergebnisse

In diesem Abschnitt werden zu gegebenen Intervallargumenten `x,y` Einschließungen `W` des Wertebereichs $W_f \subset$ `W` berechnet, wobei W_f in (31) definiert ist. Mit der obigen Funktion `ln_sqrtx2y2(...)` liefert der Aufruf `W = ln_sqrtx2y2(x,y);` die gesuchte Einschließung `W` $\in IR$, vergleichen Sie dazu den Quelltext ab Seite 28. Bitte beachten Sie, dass die auf der Maschine berechneten binären Intervalle `W` $\in IR$ durch die mit der Anweisung `cout << ln_sqrtx2y2(x,y) << endl;` ausgegebenen dezimalen Intervalle stets eingeschlossen werden.

1. `x` $= [-1,1]$ und `y` $= [-1,1]$ liefern die Fehlermeldung

   ```
   real ln_sqrtx2y2(const real&, const real&):
                    STD_FKT_OUT_OF_DEF
   ```

   da die Bedingung $|x| + |y| > 0$ nicht erfüllt ist.

2. `x` $= [-1,-1]$, `y` $= [0,0]$
   $\rightsquigarrow \quad W_f \subset$ `W` $\subset [0.000000000000000 \cdot 10^0, 0.000000000000000 \cdot 10^0]$

3. `x` $= [1,1]$, `y` $= [1,1]$
   $\rightsquigarrow \quad W_f \subset$ `W` $\subset [3.465735902799723 \cdot 10^{-1}, 3.465735902799731 \cdot 10^{-1}]$

4. `x` $= [2,2]$, `y` $= [2,2]$
   $\rightsquigarrow \quad W_f \subset$ `W` $\subset [1.039720770839916 \cdot 10^0, 1.039720770839920 \cdot 10^0]$

5. `x` $= [1,2]$, `y` $= [1,2]$
   $\rightsquigarrow \quad W_f \subset$ `W` $\subset [3.465735902799723 \cdot 10^{-1}, 1.039720770839920 \cdot 10^0]$

6. $\mathtt{x} = [0.8, 0.8],\ \mathtt{y} = [0.6, 0.6]$
   $\rightsquigarrow \quad W_f \subset \mathtt{W} \subset [-6.661338147750948 \cdot 10^{-17}, 8.881784197001264 \cdot 10^{-17}]$
   Beachten Sie bitte, dass jetzt x und y keine Punktintervalle sind und 0.8 bzw. 0.6 optimal einschließen. Wegen $0.8^2 + 0.6^2 = 1$ muss daher W die Zahl Null zwar einschließen, es darf aber nicht gelten $\mathtt{W} = [0, 0]$.

7. $\mathtt{x} = \mathtt{interval}(9007199254740988.0/9007199254740992.0);$
   $\mathtt{y} = \mathtt{interval}(9007199254740991.0/3022314549036572936765 44.0);$
   $\rightsquigarrow \quad W_f \subset \mathtt{W} \subset [5.473822126268811 \cdot 10^{-48}, 5.473822126268824 \cdot 10^{-48}]$
   Für die obigen Punktintervalle x, y liefert Maple das Ergebnis:

$$\log(\sqrt{x^2 + y^2}) = 5.473822126268816683295818684726234\ldots \cdot 10^{-48}$$

8. $\mathtt{x} = [10^{100}, 10^{100}],\ \mathtt{y} = [10^{100}, 10^{100}]$
   $\rightsquigarrow \quad W_f \subset \mathtt{W} \subset [2.306050828896843 \cdot 10^2, 2.306050828896849 \cdot 10^2]$
   Beachten Sie bitte, dass x,y keine Punktintervalle sind, die jedoch den Wert 10^{100} optimal einschließen, d.h. Inf(x) und Sup(x) aus $S(2, 53)$ sind die zu 10^{100} nächstgelegenen Rasterzahlen.

9. $\mathtt{x} = [10^{308}, 10^{308}],\ \mathtt{y} = [10^{308}, 10^{308}]$
   $\rightsquigarrow \quad W_f \subset \mathtt{W} \subset [7.095427822324453 \cdot 10^2, 7.095427822324470 \cdot 10^2]$
   Auch jetzt sind x,y keine Punktintervalle; beachten Sie, dass die Berechnung der Quadratsumme $s = x^2 + y^2$ hier nicht zum Overflow führt!

10. $\mathtt{x} = [10^{-100}, 10^{-100}],\ \mathtt{y} = [10^{-100}, 10^{-100}]$
    $\rightsquigarrow \quad W_f \subset \mathtt{W} \subset [-2.299119357091250 \cdot 10^2, -2.299119357091244 \cdot 10^2]$
    Beachten Sie bitte, dass x,y keine Punktintervalle sind, die jedoch den Wert 10^{-100} optimal einschließen.

11. $\mathtt{x} = [10^{-308}, 10^{-308}],\ \mathtt{y} = [10^{-308}, 10^{-308}]$
    $\rightsquigarrow \quad W_f \subset \mathtt{W} \subset [-7.088496350518871 \cdot 10^2, -7.088496350518854 \cdot 10^2]$
    Auch jetzt sind x,y keine Punktintervalle; beachten Sie, dass die Berechnung der Quadratsumme $s = x^2 + y^2$ hier nicht zum Underflow führt!

12. $\mathtt{x} = [2^{-1022}, 2^{-1022}],\ \mathtt{y} = [2^{-1022}, 2^{-1022}]$
    $\rightsquigarrow \quad W_f \subset \mathtt{W} \subset [-7.080498449419851 \cdot 10^2, -7.080498449419834 \cdot 10^2]$
    x,y sind Punktintervalle, die jeweils die kleinste positive, normalisierte Rasterzahl MinReal $:= 2^{-1022} \in S(2, 53)$ einschließen.

13. $\mathtt{x} = [2^{-1074}, 2^{-1074}],\ \mathtt{y} = [2^{-1074}, 2^{-1074}]$
    $\rightsquigarrow \quad W_f \subset \mathtt{W} \subset [-7.440934983311023 \cdot 10^2, -7.440934983311005 \cdot 10^2]$
    x,y sind jedoch wieder Punktintervalle, die jeweils die kleinste positive Rasterzahl minreal $:= 2^{-1074} \in S(2, 53)$ einschließen.

14. $\mathtt{x} = [1, 1],\ \mathtt{y} = [2^{-1022}, 2^{-1022}]$
    $\rightsquigarrow \quad W_f \subset \mathtt{W} \subset [0.000000000000000 \cdot 10^0, 2.225076000000001 \cdot 10^{-308}]$

8 Quelltexte für Punkt- und Intervallargumente

Der nachfolgende Quelltext enthält die vollständige Implementierung der Funktionen mit Punkt- und Intervall-Argumenten:

```
// Program:  lnx2y2_source.cpp  -----------------------------------
#include <iostream>
#include <rmath.hpp>
#include <interval.hpp>

using namespace cxsc;
using namespace std;

// With the following b0
real b0 = 6369051672525773.0 / 30191699398572330817932436647906151127
          33536976333152342700965040196499329913719081668901380142127
          01403317470002461107591981646770393983410604914740114615683
          49195162615808.0;
// it holds:
// 1. b < bo  ==> g(b) := (0.5*b)*b < MinReal with rounding downwards
// 2. b >= b0 ==> g(b) := (0.5*b)*b >=MinReal with arbitrary rounding
//                                    modus by the two multiplications.

dotprecision dot;
// Error bounds for the interval function:
real ln_x2y2_abs(2.225076E-308); // Absolute error bond
real q_lnx2y2p(4503599627370502.0 / 4503599627370496.0); // 1+e(f)
real q_lnx2y2m(9007199254740984.0 / 9007199254740992.0); // 1-e(f)

real ln2_1067(6505485212531678.0 / 8796093022208.0); // 1067*ln(2)
// Exponents of the interval bounds:
int B_lnx2y2_1[22] = {21, 31, 51, 101, 151, 201, 251, 301, 351, 401,
                      451, 501, 551, 601, 651, 701, 751, 801, 851,
                      901, 951, 1025};
int B_lnx2y2_2[22] = {-1021,-949,-899,-849,-799,-749,-699,-649,-599,
                      -549,-499,-449,-399,-349,-299,-249,-199,-149,
                      -99,-49,-29,-19};
// Optimal values N for N*ln(2):
int B_lnx2y2_N1[21] ={20, 40, 61, 122, 160, 229, 259, 320, 366, 427,
                      488, 518, 549, 610, 671, 732, 763, 825, 885,
                      945, 976};
real B_lnx2y2_c1[21] =
{
    7804143460206699.0 / 562949953421312.0, // N*ln(2) with N = 20;
    7804143460206699.0 / 281474976710656.0, // N*ln(2) with N = 40;
    5950659388407608.0 / 140737488355328.0, // N*ln(2) with N = 61;
    5950659388407608.0 / 70368744177664.0,  // N*ln(2) with N = 122;
    7804143460206699.0 / 70368744177664.0,  // N*ln(2) with N = 160;
    5584840163710419.0 / 35184372088832.0,  // N*ln(2) with N = 229;
    6316478613104797.0 / 35184372088832.0,  // N*ln(2) with N = 259;
    7804143460206699.0 / 35184372088832.0,  // N*ln(2) with N = 320;
    8925989082611412.0 / 35184372088832.0,  // N*ln(2) with N = 366;
    5206826964856657.0 / 17592186044416.0,  // N*ln(2) with N = 427;
    5950659388407608.0 / 17592186044416.0,  // N*ln(2) with N = 488;
    6316478613104797.0 / 17592186044416.0,  // N*ln(2) with N = 518;
```

```
    6694491811958559.0 / 17592186044416.0,  // N*ln(2) with N = 549;
    7438324235509510.0 / 17592186044416.0,  // N*ln(2) with N = 610;
    8182156659060461.0 / 17592186044416.0,  // N*ln(2) with N = 671;
    8925989082611412.0 / 17592186044416.0,  // N*ln(2) with N = 732;
    4652001140732587.0 / 8796093022208.0,   // N*ln(2) with N = 763;
    5030014339586349.0 / 8796093022208.0,   // N*ln(2) with N = 825;
    5395833564283538.0 / 8796093022208.0,   // N*ln(2) with N = 885;
    5761652788980727.0 / 8796093022208.0,   // N*ln(2) with N = 945;
    5950659388407608.0 / 8796093022208.0    // N*ln(2) with N = 976;
};

int Interval_Nr(int* v, const int& n, const int& ex)
// n>0 subintervals:   |...\|...\|...\ ..... |...|
// subinterval Nr.:      0    1    2   .....  n-1
{
    int i=0,j=n,k;  // n>0:  Number of subintervals
    do {
        k = (i+j)/2;
        if (ex < v[k]) j = k-1;
        else i = k+1;
    } while(i<=j);
    return j;  // x with ex=expo(x) lies in the subinterval number j
}

real ln_sqrtx2y2(const real& x, const real& y)
                                  throw(STD_FKT_OUT_OF_DEF)
// ln( sqrt(x^2+y^2) ) == 0.5*ln(x^2+y^2); Blomquist, 21.11.03;
{
    int j,N;
    real a,b,r,r1;
    a = sign(x)<0 ? -x : x;  // a = |x| >= 0;
    b = sign(y)<0 ? -y : y;  // b = |y| >= 0;
    int exa=expo(a), exb=expo(b), ex;
    if (b > a)
    {
        r = a;   a = b;   b = r;
        ex = exa;   exa = exb;   exb = ex;
    }
    // It holds now:   0 <= b <= a
    if (sign(a)==0)
        cxscthrow(STD_FKT_OUT_OF_DEF
                    ("real ln_sqrtx2y2(const real&, const real&)"));
    if (exa>20) // to avoid overflow by calculating a^2 + b^2
    {  // a>=2^(20):
        j = Interval_Nr(B_lnx2y2_1,21,exa); // j: No. of subinterval
        N = B_lnx2y2_N1[j];    // N: Optimal int value
        if (exb-exa > -25)
        {   // For (exb-exa>-25) we use the complete term:
            // N*ln(2) + [ln(2^(-N)*a)+0.5*ln(1+(b/a)^2)]
            b = b/a;  // a > 0
            b = lnp1(b*b);
            times2pown(b,-1);  // exact division by 2
            times2pown(a,-N);
            r = b + ln(a); // [ ... ] calculated!
```

```
            r += B_lnx2y2_c1[j];
        }
        else { // For (exb-exa<=-25) only two summands!:
            times2pown(a,-N);
            r = ln(a) + B_lnx2y2_c1[j];
        }
    }
    else  // exa<=20 or a<2^(20):
    {     // Now calculation of a^2+b^2 without overflow:
        if (exa<=-20) // to avoid underflow by calculating a^2+b^2
            if (exa<=-1022) // a in the denormalized range
            {
                r = b/a;
                r = lnp1(r*r);  times2pown(r,-1); // r: 0.5*ln(1+..)
                times2pown(a,1067);
                r += ln(a);     // [ .... ] ready
                r -= ln2_1067;  // rel. error = 2.459639e-16;
            }
            else // MinReal=2^(-1022) <= a < 2^(-20)
            {    // Calculating the number j of the subinterval:
                j = 20 - Interval_Nr(B_lnx2y2_2,21,exa);
                r = a;  times2pown(r,B_lnx2y2_N1[j]);
                r = ln(r);  // r: ln(2^N*a);
                if (exb-exa > -25) { // calculating the complete term
                    b = b/a;
                    a = lnp1(b*b);
                    times2pown(a,-1);
                    r += a;  // [ ... ] ready now
                }
                // We now have: exb-exa<=-25,  ==>  b/a <= 2^(-24);
                r -= B_lnx2y2_c1[j]; // 0.5*ln(1+(b/a)^2) neglected!
                // relative error = 4.524090e-16 in both cases;
            }
        else // calculation of a^2+b^2 without overflow or underflow:
        {   // exa>-20  respective  a>=2^(-20):
            dot = 0;
            accumulate(dot,a,a);
            accumulate(dot,b,b);  // dot = a^2+b^2, exact!
            real s = rnd(dot);    // s = a^2 + b^2, rounded!
            if (s>=0.25 && s<=1.75)
                if (s>=0.828125 && s<=1.171875)
                { // Series:
                    if (a==1 && exb<=-28)
                    {
                        r = b;  times2pown(r,-1);
                        r *= b;
                    }
                    else {
                        dot -= 1;
                        r = rnd(dot); // r = a^2+b^2-1 rounded!
                        r = lnp1(r);
                        times2pown(r,-1);
                    }
                }
```

```
                else { // Reading dot = a^2+b^2 twice:
                    r = rnd(dot);
                    dot -= r;
                    r1 = rnd(dot); // a^2+b^2 = r+r1, rounded!
                    r1 = lnp1(r1/r);
                    r = ln(r) + r1;
                    times2pown(r,-1); // exact division by 2
                }
            else { // calculating straight from: 0.5*ln(x^2+y^2)
                r = ln(s);
                times2pown(r,-1);
            }
        }
    }
    return r;
} // ln_sqrtx2y2

interval ln_sqrtx2y2(const interval& x, const interval& y) throw()
// ln( sqrt(x^2+y^2) ) == 0.5*ln(x^2+y^2);   Blomquist, 22.11.03;
{
    interval ax=abs(x), ay=abs(y);
    real Ix=Inf(ax), Sx=Sup(ax), Iy=Inf(ay), Sy=Sup(ay),f,u1,u2;
    // Calculating the lower bound u1:
    f = ln_sqrtx2y2(Ix,Iy);
    if (Ix==1 && Iy<b0 || Iy==1 && Ix<b0) {
    // f in the denormalized range!
        u1 = f - ln_x2y2_abs;   // directed rounding not necessary!
        if (sign(u1)<0) u1 = 0;
    } else  u1 = (sign(f)<0) ? f*q_lnx2y2p : f*q_lnx2y2m;
    // Calculating the upper bound u2:
    if (Ix==Sx && Iy==Sy) // x and y are point-intervals
        if (Sx==1 && Sy<b0 || Sy==1 && Sx<b0) {
        // f in the denormalized range!
            u2 = (Sy==0 || Sx==0) ? f : f+ln_x2y2_abs;
        } else  u2 = (sign(f)<0) ? f*q_lnx2y2m : f*q_lnx2y2p;
    else // x or y is no point-interval:
    {
        f = ln_sqrtx2y2(Sx,Sy);
        if (Sx==1 && Sy<b0 || Sy==1 && Sx<b0)
        // f in the denormalized range!
            u2 = (sign(Sy)==0 || sign(Sx)==0) ? f : f+ln_x2y2_abs;
        else  u2 = (sign(f)<0) ? f*q_lnx2y2m : f*q_lnx2y2p;
    }
    return interval(u1,u2);
}
int main() {
/*  real x,y; // Function call with point arguments
    while (1) {
        cout << "real x = ? ";  cin >> x;
        cout << "real y = ? ";  cin >> y;
        cout << SetPrecision(18,18) << Scientific
             << "ln_sqrtx2y2(x,y) = " << ln_sqrtx2y2(x,y)
             << endl << endl;
    }
```

```
*/
    interval x,y; // Function call with interval arguments
    while (1) {
        cout << "interval x = ? ";  cin >> x;
        cout << "interval y = ? ";  cin >> y;

        cout << SetPrecision(16,15) << Scientific
             << "ln_sqrtx2y2(x,y) = " << ln_sqrtx2y2(x,y)
             << endl << endl;
    }
} // main
//--------------------------------------------------------------------
```

Anmerkungen:

1. Im obigen Quelltext ist der Nenner von `bo` $= y_0$ aus drucktechnischen Gründen in vier getrennten Zeilen angegeben, vgl. Seite 18.

2. Mit der Funktion

   ```
   int Interval_Nr(int* v,const int& n,const int& ex)
   ```

   kann man zu einem Feld `v` aus `n` $+ 1$ *integer* Zahlen, mit denen `n` halboffene Teilintervalle definiert werden und zu einem vorgegebenen Exponenten `ex` die Nummer des Intervalls `[v[j],v[j+1])` bestimmt werden, in dem `ex` liegt. Das erste Teilintervall erhält die Nummer 0 und das letzte die Nummer `n` $- 1$.

Literatur

[1] American National Standards Institute/Institute of Electrical and Electronics Engineers: “IEEE Standard for Binary Floating-Point Arithmetic”; ANSI/IEEE Std 754–1985, New York, 1985.

[2] Blomquist, F.: A priori Fehlerabschätzungen in C-XSC für Grundoperationen, Horner-Schema und Funktionen; Wissenschaftliches Rechnen / Softwaretechnologie, Universität Wuppertal, 2004. URL:
`http://www.math.uni-wuppertal.de/wrswt/literatur/a_priori.ps`

[3] Braune, K.; Krämer, W.: High Accuracy Standard Functions for Real and Complex Intervals, in Kaucher, E., Kulisch, U., Ullrich, Ch: *Computerarithmetic: Scientific Computation and Programming Languages*, B. G. Teubner, Stuttgart, pp. 81–114, 1987.

[4] Hammer, R.; Hocks, M.; Kulisch, U.; Ratz, D., C++ Toolbox for Verified Computing: Basic Numerical Problems. Springer-Verlag, Berlin / Heidelberg / New York, 1995.

[5] Herzberger, J. (Ed), Topics in Validated Computations. Proceedings of IMACS-GAMM International Workshop on Validated Numerics, Oldenburg, 1993. North Holland, 1994.

[6] Hofschuster, W.; Krämer, W.: Ein rechnergestützter Fehlerkalkül mit Anwendung auf ein genaues Tabellenverfahren. Preprint 96/5 des Instituts für Wissenschaftliches Rechnen und Mathematische Modellbildung, Universität Karlsruhe, 1996.

[7] Hofschuster, W.; Krämer, W.: FI_LIB, eine schnelle und portable Funktionsbibliothek für reelle Argumente und reelle Intervalle im IEEE-*double*-Format. Preprint 98/7 des IWRMM, Universität Karlsruhe, 227 Seiten, 1998.

[8] Hofschuster, W.: Zur Berechnung von Funktionswerteinschließungen bei speziellen Funktionen der mathematischen Physik. Dissertation, Universität Karlsruhe, 2000.

[9] Hofschuster, W.; Krämer, W.; Wedner, S.; Wiethoff, A., C-XSC 2.0 – A C++ Class Library for Extended Scientific Computing. Preprint 2001/1, Wissenschaftliches Rechnen / Softwaretechnologie, Universität Wuppertal, 2001.

[10] Hofschuster, W., Krämer, W.: C-XSC – A C++ Class Library for Extended Scientific Computing. To appear in *Numerical Software with Result Verification*. R. Alt, A. Frommer, B. Kearfott, W. Luther (eds), Springer Lecture Notes in Computer Science, 2004.

[11] Klatte, R.; Kulisch, U.; Lawo, C.; Rauch, M.; Wiethoff, A.: C-XSC, A C++ Class Library for Extended Scientific Computing. Springer - Verlag, Berlin / Heidelberg / New York, 1993.

[12] Krämer, W.: A priori Worst Case Error Bounds for Floating-Point Computations, IEEE Transactions on Computers, Vol. 47, No. 7, July 1998.

[13] Krämer, W., Wolff von Gudenberg, J. (eds): *Scientific Computing, Validated Numerics, Interval Methods*, Kluwer Academic Publishers Boston/Dordrecht/London, 398 pages, 2001.

[14] Krämer, W.; Blomquist, F.: Algorithms with Guaranteed Error Bounds for the Error Function and the Complementary Error Function. Eingereicht für Theoretical Computer Science (TCS), Special issue: Real Numbers and Computers, 2004.

[15] 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.

[16] 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.

[17] 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.