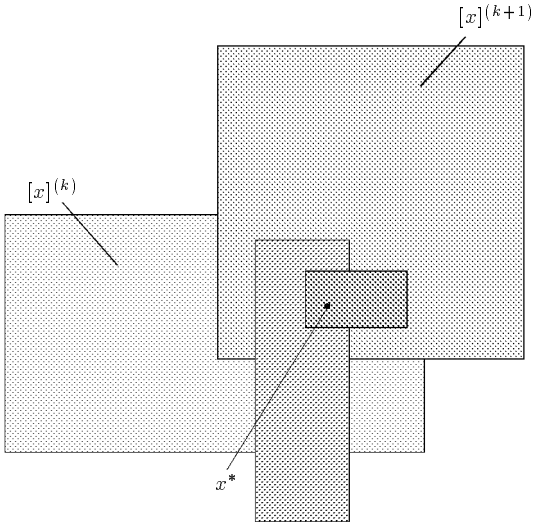
Institut für
Angewandte
Mathematik

Universität Karlsruhe (TH)
D-76128 Karlsruhe

# Sichere und genaue Abschätzung des Approximationsfehlers bei rationalen Approximationen

Walter Krämer

**Forschungsschwerpunkt Computerarithmetik, Intervallrechnung und Numerische Algorithmen mit Ergebnisverifikation**




**Bericht 3/1996**

# Impressum


| | |
|---|---|
| Herausgeber: | Institut für Angewandte Mathematik<br>Lehrstuhl Prof. Dr. Ulrich Kulisch<br>Universität Karlsruhe (TH)<br>D-76128 Karlsruhe |
| Redaktion: | Dr. Dietmar Ratz |


# Internet-Zugriff

Die Berichte sind in elektronischer Form erhältlich über

`ftp://iamk4515.mathematik.uni-karlsruhe.de`

im Verzeichnis: `/pub/documents/reports`

oder über die World Wide Web Seiten des Instituts

`http://www.uni-karlsruhe.de/~iam`

# Autoren-Kontaktadresse

Rückfragen zum Inhalt dieses Berichts bitte an


Walter Krämer
Institut für Angewandte Mathematik
Universität Karlsruhe (TH)
D-76128 Karlsruhe

E-Mail: Walter.Kraemer@math.uni-karlsruhe.de

# Sichere und genaue Abschätzung des Approximationsfehlers bei rationalen Approximationen

Walter Krämer

## Inhaltsverzeichnis

**Zusammenfassung**

**Sichere und genaue Approximationsfehlerabschätzung:** Die hier entwickelte Methode erlaubt es, für den relativen bzw. den absoluten Approximationsfehler einer rationalen Approximation eine sichere und gleichzeitig genaue Fehlerschranke im Sinne der Maximumnorm zu berechnen. Die gesuchte Oberschranke wird unter Einsatz der Intervallrechnung automatisch auf einer Rechenanlage gefunden. Es spielt keine Rolle, wie die Approximationskoeffizienten gewonnen wurden.

Das vorgestellte Verfahren kann auch dazu verwendet werden, den Verlauf der Fehlerkurve durch eine untere bzw. obere Treppenfunktion abzusichern.

Als Beispiele werden eine Approximation der Fehlerfunktion $\mathrm{erf}(x)$ sowie eine Approximation im Zusammenhang mit der Gammafunktion $\Gamma(x)$ untersucht.

**Abstract**

**Reliable and Almost Sharp Error Bounds for Rational Approximations:** The method discussed in this paper can be used to find reliable and (almost) sharp bounds in the uniform norm for the relative or absolute approximation errors of rational approximations.

The presented method can also be used to bound the error curve below by a lower and above by an upper piecewise constant function.

As examples an approximation for the error function $\mathrm{erf}(x)$ as well as an approximation connected with $\Gamma(x)$ are considered.

**MSC: 65D15, 65G05, 65G10, 68M15**

# 1 Einleitung

Koeffizienten von Approximationsfunktionen (z. B. polynomiale Bestapproximationen, rationale Approximationen usw. ) werden üblicherweise mittels Langzahlrechnug z. B. mit Hilfe eines Computeralgebrasystems ermittelt. Die dabei verwendeten Fehler*schätzer* beziehen sich naturgemäß auf die Langzahlkoeffizienten. Auch handelt es sich nur um Schätzer, so daß sichere, im mathematischen Sinn gültige Aussagen in der Regel nicht möglich sind.

Um Einschließungen von Funktionswerten berechnen zu können (z. B. bei der Realisierung von Intervallfunktionen), muß eine verläßliche Fehlerabschätzung mit den tatsächlich verwendeten, d. h. mit den in das Gleitkommadatenformat des Rechners gerundeten Koeffizienten durchgeführt werden. Die verwendete Approximationsfehleroberschranke $e_{\max}$ muß im mathematischen Sinn verläßlich und für *alle* Argumente des Approximationsintervalls $[z]$ gültig sein. Der für eine Intervallunterteilung (Zerlegung)

$$\bigcup_j [z_j] \;=\; [z]$$

naheliegende Ansatz

$$e_{\max} := \max_j \left| \mathrm{f}([z_j]) - \frac{\mathrm{p}([z_j])}{\mathrm{q}([z_j])} \right| \qquad (\Longrightarrow \; \| \mathrm{f} - \mathrm{p}/\mathrm{q} \|_\infty \;\leq e_{\max})$$

führt auf Grund des mehrfachen Auftretens des Intervallargumentes $[z_j]$ auch bei einer sehr feinen Zerlegung des Approximationsbereiches $[z]$ nicht zum Ziel. Je besser die

Approximation p/q die Funktion f approximiert, desto stärker macht sich Auslöschung bei der Differenzbildung bemerkbar. Man beachte in diesem Zusammenhang, daß bei der Intervallsubtraktion die Summe der Durchmesser der beteiligten Intervalloperanden den Durchmesser des Ergebnisintervalls ergibt, d. h. der Durchmesser des Ergebnisintervalls kann nicht kleiner werden wie der maximale Durchmesser der beiden Operanden. Eine auf obige Weise berechnete Schranke ist zwar verläßlich, sie wird jedoch in der Regel den tatsächlichen Maximalfehler der Approximation um Größenordnungen übertreffen und ist damit in der Praxis unbrauchbar.

Im folgenden wird nun ein Verfahren hergeleitet, das diese Schwierigkeiten beseitigt. Es wird sich zeigen, daß das gewonnene Verfahren nicht nur zur automatischen Bestimmung einer genauen Oberschranke verwendet werden kann. Vielmehr bietet es auch die Möglichkeit, den gesamten Verlauf der Fehlerkurve zu verifizieren.

# 2 Theorie

Die Funktion

$$\mathrm{f} : I\!R \subseteq D_f \longrightarrow I\!R$$

sei um $x_0$ entwickelbar, d. h.

$$\mathrm{f}(x) = \sum_{k=0}^{\infty} s_k \cdot (x - x_0)^k, \quad |x - x_0| \leq \eta\,. \tag{1}$$

Sie werde durch eine rationale Approximation der Form

$$\mathrm{f}(x) \approx \mathrm{g}(x) := \frac{\mathrm{p}_M(x - x_0)}{\mathrm{q}_N(x - x_0)}; \quad \mathrm{q}_N(x - x_0) \neq 0, \quad |x - x_0| \leq \eta\,, \tag{2}$$

angenähert. Der Grad des Zählerpolynoms sei $M$, der des Nennerpolynoms $N$. Weiter wird angenommen, daß die Reihenkoeffizienten $s_k$ bekannt sind, bzw. daß eine Darstellung der Form

$$\mathrm{f}(x) = \sum_{k=0}^{K} s_k \cdot (x - x_0)^k + \mathrm{R}_K(x) =: \mathrm{T}_K(x - x_0) + \mathrm{R}_K(x) \tag{3}$$

vorliegt.

Zur Abschätzung des dabei auftretenden relativen Approximationsfehlers

$$\begin{aligned} \mathrm{err}(x) &:= \frac{\mathrm{g}(x) - \mathrm{f}(x)}{\mathrm{f}(x)} = \frac{\dfrac{\mathrm{p}_M(x - x_0)}{\mathrm{q}_N(x - x_0)} - \mathrm{f}(x)}{\mathrm{f}(x)} \\ &= \frac{\mathrm{p}_M(x - x_0) - \mathrm{q}_N(x - x_0) \cdot \mathrm{f}(x)}{\mathrm{q}_N(x - x_0) \cdot \mathrm{f}(x)}, \quad |x - x_0| \leq \eta, \end{aligned} \tag{4}$$

schreibt man den Zähler unter Verwendung der Darstellung (3) für $f(x)$ um in

$$\left\{ \mathrm{p}_M(x - x_0) - \mathrm{q}_N(x - x_0) \cdot \sum_{k=0}^{K} s_k \cdot (x - x_0)^k \right\} - \mathrm{q}_N(x - x_0) \cdot \mathrm{R}_K(x) \tag{5}$$

Der erste Klammerausdruck stellt nun ein Polynom

$$\mathrm{h}_L(x) = \sum_{k=0}^{L} r_k \cdot (x - x_0)^k$$

vom Grad

$$L := \max\{M,\ N + K\}$$

dar. In der Praxis wird fast immer $L = N + K$ gelten. Die einzelnen Polynomkoeffizienten $r_k$ können dabei aus den bereits bekannten Polynomkoeffizienten $q_k$ des Polynoms $\mathrm{q}_N$, $p_k$ des Polynoms $\mathrm{p}_M$ sowie $s_k$ aus (3) wie folgt berechnet werden:

$$r_k := \sum_{j=0}^{k} (s_j \cdot q_{k-j}) - p_k \ , \quad k = 0, 1, \ldots L \ .$$

Taucht bei diesen Ausdrücken eine der Größen $s_j, q_{k-j}, p_k$ in den Ausgangspolynomen nicht auf (kommt also der entsprechende Index dort nicht vor), so ist diese Größe mit dem Wert 0 vorzubelegen.

Auf einer Rechenanlage wird sich beim Berechnen der Koeffizienten $r_k$ um so größere Auslöschung ergeben, je besser die rationale Approximationsfunktion $\mathrm{g}(x)$ mit der zu approximierenden Funktion f(x) übereinstimmt. Um mit diesem Problem fertig zu werden, muß eine Langzahlintervallarithmetik [19] verwendet werden. Genauer kann wie folgt vorgegangen werden:

- Berechne Intervalleinschließungen $[s_k]$ der Polynomkoeffizienten $s_k$, $k = 0, 1, \ldots, K$ der abgebrochenen Reihenentwicklung (3) mittels einer Langzahlintervallarithmetik [19].

- Einschließungen $[r_k]$ der Koeffizienten des Polynoms $\mathrm{h}_L(x)$ können dann mittels

$$[r_k] := \sum_{j=0}^{k} ([s_j] \cdot q_{k-j}) - p_k \ , \quad k = 0, 1, \ldots L \tag{6}$$

  berechnet werden. Dabei sind die Koeffizienten $q_{k-j}$ und $p_k$ als gegebene Punktintervalle (es handelt sich hierbei gerade um die im Zahlformat der Maschine dargestellten Approximationskoeffizienten) bekannt. Auch hierbei müssen wieder Langzahlintervalloperationen eingesetzt werden.

Unter Verwendung des eben berechneten Intervallpolynoms $[\mathrm{h}_L]$ kann nun eine sichere Oberschranke für den relativen Approximationsfehler der Approximation $\mathrm{g}(x)$ über dem Intervall $|x - x_0| < \eta$ angegeben werden. Man findet zunächst

$$\begin{aligned}
\mathrm{err}(x) \ &:= \ \frac{\mathrm{p}_M(x - x_0) - \mathrm{q}_N(x - x_0) \cdot \mathrm{f}(x)}{\mathrm{q}_N(x - x_0) \cdot \mathrm{f}(x)} \\
&= \ \frac{\mathrm{p}_M(x - x_0) - \mathrm{q}_N(x - x_0) \cdot \sum_{k=0}^{K} s_k (x - x_0)^k}{\mathrm{q}_N(x - x_0) \cdot \mathrm{f}(x)} - \frac{\mathrm{R}_K(x)}{\mathrm{f}(x)} \qquad (7) \\
&\in \ \frac{\sum_{k=0}^{L} [r_k] \cdot (x - x_0)^k}{\mathrm{q}_N(x - x_0) \cdot \mathrm{f}(x)} - \frac{\mathrm{R}_K(x)}{\mathrm{f}(x)} \ .
\end{aligned}$$

Diese Inklusionsbeziehung gilt für alle $x$ mit $|x - x_0| \leq \eta$ .

Kennt man eine obere Schranke $\alpha$ für den Betrag $|\mathrm{R}_K(x)|$ , gilt also

$$|\mathrm{R}_K(x)| \leq \alpha \text{ für alle } x \text{ mit } |x - x_0| \leq \eta\,, \tag{8}$$

so ergibt sich schließlich die Ungleichung

$$\begin{aligned} |\,\frac{\sum_{k=0}^{L}[r_k]\cdot(x-x_0)^k}{\mathrm{q}_N(x-x_0)\cdot\mathrm{f}(x)} \;-\; \frac{\mathrm{R}_K(x)}{\mathrm{f}(x)}\,| & \\ \leq\; \frac{1}{|\mathrm{f}(x)|}\cdot\{\;|\frac{\sum_{k=0}^{L}[r_k]\cdot(x-x_0)^k}{\mathrm{q}_N(x-x_0)}|+\alpha\;\}\,. & \end{aligned} \tag{9}$$

Eine Einschließung von $\mathrm{f}(x)$ mit bereits bekannten Größen ergibt sich gemäß

$$\mathrm{f}(x) \in \sum_{k=0}^{K}[s_k]\cdot(x-x_0)^k + [-\alpha,\,\alpha]\;,\quad |x-x_0| \leq \eta\,. \tag{10}$$

Dieser Ausdruck kann mittels „normaler“ Intervallarithmetik ohne große Auslöschung berechnet werden. Sollte der Wert 0 in diesem Intervall enthalten sein, so kann man auf diese Weise nur den absoluten Fehler der rationalen Approximation sicher nach oben abschätzten. Man beachte, daß man im Konvergenzbereich der Reihenentwicklung (3) die Fehlerschranke $\alpha$ in (8) durch Erhöhung des Approximationsgrades $K$ in der Darstellung (3) beliebig klein machen kann.

Bezeichnet man die rechte Seite der Ungleichung (9) mit $\gamma(x)$, d. h.

$$\gamma(x) := \frac{1}{|\mathrm{f}(x)|}\cdot\{\;|\frac{\sum_{k=0}^{L}[r_k]\cdot(x-x_0)^k}{\mathrm{q}_N(x-x_0)}|+\alpha\;\}\,, \tag{11}$$

so ist man am Maximum

$$\max\{\;\gamma(x)\;\mid\;|x-x_0|\leq\eta\;\}$$

interessiert. Um eine obere Schranke für $\gamma(x)$ zu bestimmen, kann man im einfachsten Fall überall im Ausdruck für $\gamma(x)$ die Punktgröße $x$ durch das gesamte Intervall $[z] := [x_0 - \eta,\, x_0 + \eta]$ ersetzen. Notationell soll dieser Vorgang (intervallmäßige Auswertung von $\gamma(x)$) durch die Schreibweise

$$\gamma([z]) := \frac{1}{<\mathrm{f}([z])>}\cdot\{\;|\frac{\sum_{k=0}^{L}[r_k]\cdot([z]-x_0)^k}{\mathrm{q}_N([z]-x_0)}|+\alpha\;\} \tag{12}$$

kenntlich gemacht werden. Dabei bezeichnet $<\mathrm{f}([z])>$ das Betragsminimum des Intervalls $f([z])$. Man findet dann wiederum aufgrund der Inklusionsmonotonie die Ungleichung

$$\|\;\mathrm{err}(x)\;\|_\infty = \|\;\frac{\mathrm{g}(x)-\mathrm{f}(x)}{\mathrm{f}(x)}\;\|_\infty \leq \sup(\gamma([z]))\,. \tag{13}$$

Der relative Fehler der Approximation ist bezüglich der Tschebyscheff-Norm kleiner als das Supremum des Intervalls $\gamma([z])$. Sollte sich bei der intervallmäßigen Auswertung $\gamma([z])$ eine zu große Überschätzung bemerkbar machen (dies kann man feststellen,

indem man mit Schranken bei Punktauswertungen vergleicht), so wird das Gesamtintervall [z] in hinreichend viele bis auf Randpunkte disjunkte Teilintervalle $[z_j]$ zerlegt:

$$[z] = \bigcup_j [z_j] \; .$$

Als Oberschranke erhält man dann die in der Regel bessere Schranke

$$\| \frac{\mathrm{g}(x) - \mathrm{f}(x)}{\mathrm{f}(x)} \|_\infty \leq \max_j \sup(\gamma([z_j])) \; . \tag{14}$$

# 3 Algorithmus

1. Bestimme eine Reihendarstellung mit Entwicklungspunkt $x_0$ (z. B. Taylorreihe) der zu approximierenden Funktion $\mathrm{f}(x)$.

2. Finde einen Ausdruck für den Abschneidefehler $\mathrm{R}_K(x)$ (siehe (3)) über dem Approximationsintervall $[z] := [x_0 - \eta,\, x_0 + \eta]$ , z. B. mittels einer geeigneten Restgliedformel.

3. Bestimme die Approximationskoeffizienten $p_k, k = 0, 1, \ldots M$ des Zählerpolynoms sowie die Koeffizienten $q_j, j = 0, 1, \ldots N$ des Nennerpolynoms, z. B. mittels eines Computeralgebrasystems.

   Der vorliegenden Algorithmus hat gerade zum Ziel, die Genauigkeit einer solchen rationalen Approximation verläßlich nach oben abzuschätzen. Es ist dabei unerheblich, wie die Approximationskoeffizienten gefunden wurden (z. B. mittels Tschebyscheff-Approximation oder mit einem Remez-Algorithmus, welcher mittels einer Langzahlrechnung durchgeführt wird; die ursprünglich als Langzahlen berechneten Koeffizienten werden dann in das Zieldatenformat gerundet und in dieser Form als $p_k$ und $q_j$ bezeichnet!).

4. Bestimme eine Oberschranke $\alpha$ des Abbruchfehlers $\mathrm{R}_K(x)$ bezüglich des Intervalls $[z]$. Diese Oberschranke findet man entweder durch Rechnung per Hand oder aber unter Verwendung der Technik der automatischen Differentiation.

5. Berechne mittels Langzahlintervallrechnung die Intervalleinschließungen der Koeffizienten des Polynoms $\mathrm{h}_L(x)$ gemäß (6).

6. Berechne gemäß (13) bzw. (14) unter Verwendung von (12) eine Oberschranke für den Absolutbetrag des relativen Approximationsfehlers. Das in (12) benötigte Intervall $\mathrm{f}([z])$ kann mittels der intervallmäßigen Auswertung

   $$\mathrm{f}([z]) := \sum_{k=0}^{K} [s_k] \cdot ([z] - x_0)^k + [-\alpha,\, \alpha] \tag{15}$$

   gewonnen werden (vergleiche mit (10)). Um eine möglichst gute Oberschranke zu erhalten, sollte man hier Algorithmen der globalen Optimierung (Branch-and-Bound-Verfahren [10, 25]) zur Wertebereicheinschließung der auftretenden Intervallausdrücke einsetzen.

# 4 Beispiele

Im folgenden sollen eine rationale Approximation der Fehlerfunktion $\mathrm{erf}(t)$ und eine Approximation, welche im Zusammenhang mit der Berechnung der Gammafunktion verwendet werden kann, untersucht werden.

## 4.1 Fehlerfunktion

Für die Fehlerfunktion

$$\mathrm{erf}(t) \;=\; \Phi(t) \;=\; \frac{2}{\pi}\int_0^t e^{-x^2}\mathrm{dx} \;=\; \frac{2}{\pi}\sum_{k=1}^{\infty}(-1)^k\frac{t^{2k-1}}{(2k-1)(k-1)!} \tag{16}$$

kann eine Fehlerschranke $\varepsilon$ für

$$||\frac{\mathrm{erf}(t) - t\frac{\mathrm{p}_M(t^2)}{\mathrm{q}_N(t^2)}}{\mathrm{erf}(t)}||_\infty < \varepsilon$$

folgendermaßen bestimmt werden. Zunächst gilt

$$||\frac{\mathrm{erf}(t) - t\frac{\mathrm{p}_M(t^2)}{\mathrm{q}_N(t^2)}}{\mathrm{erf}(t)}||_\infty = ||\frac{\frac{\mathrm{erf}(t)}{t} - \frac{\mathrm{p}_M(t^2)}{\mathrm{q}_N(t^2)}}{\frac{\mathrm{erf}(t)}{t}}||_\infty \;,$$

so daß mit $x := t^2$ die Funktion $\mathrm{f}(x)$ mit

$$\frac{\mathrm{erf}(t)}{t} =: \mathrm{f}(x) := \frac{2}{\pi}\sum_{k=0}^{\infty}(-1)^k\frac{x^2}{(2k+1)k!} \tag{17}$$

durch die rationale Approximation $\frac{\mathrm{p}_M(x)}{\mathrm{q}_N(x)}$ zu approximiern ist. Die Koeffizienten $s_k$ der Taylorapproximation um den Nullpunkt $x_0 = 0$ sind demnach wie folgt gegeben:

$$s_k := \frac{2}{\pi}\frac{(-1)^k}{(2k+1)k!} \;.$$

Da es sich um eine Leibnizreihe handelt, findet man für den Fehler $\mathrm{R}_K(x)$ in der Darstellung

$$\mathrm{f}(x) \;=\; \frac{2}{\pi}\sum_{k=0}^{K}(-1)^k\frac{x^k}{(2k+1)k!} + \mathrm{R}_K(x) \tag{18}$$

die Abschätzung

$$|\,\mathrm{R}_K(x)\,| \leq \frac{2}{\pi}\,|\,\frac{(-1)^{K+1}}{(2K+3)(K+1)!}\,|\,x^{K+1} \;.$$

Dabei hat der Fehlerterm das Vorzeichen des ersten vernachlässigten Gliedes.

Legt man das Approximationsintervall $[z]$ auf $[0, \eta]$ mit $\eta := (0.65)^2$ fest (dies entspricht dem Bereich $[-0.65, 0.65]$ für die ursprüngliche Variable $t$), so findet man als Oberschranke $\alpha$ für den Abbruchfehler

$$|\,\mathrm{R}_K(x)\,| \leq \frac{2}{\pi}\,\frac{1}{(2K+3)(K+1)!}\,\eta^{K+1}\ ,$$

welche simultan für alle $x$ im Approximationsintervall $[z]$ gültig ist. Wählt man z. B. für $K$ den Wert $K = 20$, so ergibt sich

$$|\,\mathrm{R}_K(x)\,| \leq \frac{2}{\pi \cdot 43 \cdot 21!}(0.65)^{42} \ < \ 4.1E-30 \ =: \ \alpha\ .$$

als maximaler Fehler der „nahezu perfekten“ Approximation (18). Die Werte $\eta$ und $\alpha$ werden auch im Programmlisting verwendet. Die (Langzahl-)Approximationskoeffizienten sind:

```
{ Initialize p: ( Polynomial of degree 4 ) }
p[0]:= 1.12837916709551257034408185509220539581728950 11;
p[1]:= 0.13586549296371048555126878891687297476339912625;
p[2]:= 0.04032318878172327983490986917883931 6150408859554;
p[3]:= 0.00120253791049528654993674531969282273781772453 00;
p[4]:= 0.00006492833394578692631881024335922070082650030 5394;

{ Initialize q: ( Polynomial of degree 4 ) }
q[0]:= 0.99999999999999999999999999999999999999999999999;
q[1]:= 0.45374099143768454697640240146082645596834144301;
q[2]:= 0.08698249276445501623449944980095413499031770931 0;
q[3]:= 0.00849531039365385507995509390404934972680468898 48;
q[4]:= 0.00036478942823305015667222171669015298476143498196;
```

Die tatsächlich verwendeten Approximationskoeffizienten sind also gerade die zu den angegebenen Dezimalwerten nächstgelegenen IEEE-double Zahlen.

Für die so gewonnene rationale Approximation wird nun automatisch eine gesicherte Oberschranke für den maximalen relativen Approximationsfehler ermittelt. Das entsprechende PASCAL-XSC Programm liefert als Oberschranke

$$\left|\left|\frac{\mathrm{erf}(t) - t\dfrac{\mathrm{p}_M(t^2)}{\mathrm{q}_N(t^2)}}{\mathrm{erf}(t)}\right|\right|_\infty < \varepsilon := \ 1.443E-017 \ \text{ für alle } x = t^2 \in [0,\, 0.65^2]\,.$$

Das Programmlisting wird hier nicht abgedruckt, da es dem Programmlisting zum nächsten Beispiel (dieses befindet sich im Anhang) sehr ähnlich ist.

## 4.2 Die Gammafunktion

In diesem Abschnitt soll die überall analytische Funktion $1/\Gamma(x)$ betrachtet werden. Für sie kennt man die Darstellung

$$\frac{1}{\Gamma(x)} = \exp\Big(\sum_{k=0}^{\infty} s_k \cdot (x-2)^k\Big) \tag{19}$$

mit

$$\begin{aligned} s_0 &:= 0\ , \\ s_1 &:= \gamma - 1\ , \\ s_k &:= \frac{(-1)^{k+1}(\zeta(k)-1)}{k}\ ,\ k = 2,3,\ldots \end{aligned}$$

Diese Koeffizienten sind ausgedrückt mittels der Eulerschen Konstanten $\gamma$ und Werten der Riemannschen Zetafunktion:

$$\begin{aligned} \gamma &:= \lim_{m\to\infty}\left(1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{m}-\ln(m)\right) = 0.5772156649\ldots\ , \\ \zeta(s) &:= \sum_{k=1}^{\infty} k^{-s}, \quad \mathrm{Re}(s) > 1 \end{aligned}$$

Die Reihe, welche als Argument der Exponentialfunktion dient, ist gerade die Taylorreihe der Funktion $-\ln(\Gamma(x))$ (siehe [1]). Um eine Approximation für $1/\Gamma(x)$ zu finden, wird diese Taylorreihe durch die rationale Approximation

$$\frac{\mathrm{p}_6(x-2)}{\mathrm{q}_5(x-2)} = \frac{\sum_{k=0}^{6} p_k \cdot (x-2)^k}{\sum_{j=0}^{5} q_j \cdot (x-2)^j} \quad \text{(Entwicklungspunkt } x_0 := 2) \tag{20}$$

mit den jeweils zur nächsten IEEE-double Zahl gerundeten Werten aus Tabelle 1 approximiert. Der tatsächlich verwendete Satz von Maschinenzahlen findet sich in der im Anhang (Seite 24) angegebenen Protokolldatei.

| $j$ | $p_j$ | $q_j$ |
|---|---|---|
| 0 | $+1.35327304816540868 \cdot 10^{-17}$ | $1.00000000000000000 \cdot 10^{+0}$ |
| 1 | $-4.22784335098468688 \cdot 10^{-1}$ | $1.23556370859698140 \cdot 10^{+0}$ |
| 2 | $-8.44844014435089770 \cdot 10^{-1}$ | $5.40473405509170129 \cdot 10^{-1}$ |
| 3 | $-5.59579952051023151 \cdot 10^{-1}$ | $9.91256197612588100 \cdot 10^{-2}$ |
| 4 | $-1.53556364516808078 \cdot 10^{-1}$ | $6.86477527603966571 \cdot 10^{-3}$ |
| 5 | $-1.65082854342856732 \cdot 10^{-2}$ | $1.09503363580423657 \cdot 10^{-4}$ |
| 6 | $-4.72192577137972617 \cdot 10^{-4}$ | |

Tabelle 1: Polynomkoeffizienten $p_j, q_j$

Es geht nun zunächst darum, den absoluten Fehler dieser rationalen Approximation für

$$f(x) := \sum_{k=0}^{\infty} s_k \cdot (x-2)^k\ ,\ |x-1| \le 0.5 \tag{21}$$

sicher nach oben abzuschätzen (er wird sich später bei der Approximation von $1/\Gamma(x)$ als relativer Fehler bemerkbar machen). Gesucht ist also eine Oberschranke $\Delta$ für

$$||\,\mathrm{f}(x) - \frac{\mathrm{p}_6(x-2)}{\mathrm{q}_5(x-2)}\,||_\infty \;<\; \Delta\;, \tag{22}$$

welche simultan für alle $x$ des Approximationsintervalles $[z] := [1.5,\, 2.5]$ gilt. Als Grad der „nahezu perfekten" Taylorapproximation wird $K = 30$ gewählt. Damit findet man als Oberschranke für den Abschneidefehler

$$\frac{[\zeta(31)-1]\cdot 2^{-31}}{31} \;<\; \frac{2^{-31}\cdot 4.7E-10}{31} \;<\; 8E-21 \;=:\alpha$$

Jetzt sind alle Größen bekannt, um das in dieser Arbeit beschriebene Verfahren anwenden zu können. Das im Anhang aufgelistete Programm berechnet die Oberschranke des absoluten Approximationsfehlers zu

$$\Delta \;<= 11.47E-17\;.$$

Zusammen mit (21) und (22) bedeutet dies

$$\frac{\mathrm{p}_6(x-2)}{\mathrm{q}_5(x-2)} \in \sum_{k=0}^{\infty} s_k\cdot(x-2)^k + [-\Delta,\,\Delta]\;,\; x\in[z]. \tag{23}$$

Zu jedem $x \in [z]$ existiert ein $\delta(x) \in [-\Delta,\,\Delta]$ mit

$$\frac{\mathrm{p}_6(x-2)}{\mathrm{q}_5(x-2)} = \sum_{k=0}^{\infty} s_k\cdot(x-2)^k + \delta(x)\,. \tag{24}$$

Diese Darstellung wird nun verwendet, um die folgende Gesamtfehlerabschätzung zu erhalten:

$$\begin{aligned}
\Big|\frac{\frac{1}{\Gamma(x)} - \exp(\frac{\mathrm{p}_6}{\mathrm{q}_5})}{\frac{1}{\Gamma(x)}}\Big| &= \Big|\frac{\exp_\infty - \exp_{\mathrm{p,q}}}{\exp_\infty}\Big| = \Big|\frac{\exp_\infty - \exp_\infty\cdot\exp(\delta(x))}{\exp_\infty}\Big| \\
&= |1-\exp(\delta(x))| \\
&< \Delta + \Delta^2 + \Delta^3 + \cdots \\
&< 1.0001\Delta\;,\; x\in[z] = [1.5,\,2.5].
\end{aligned}$$

Hier bezeichnet $\exp_\infty$ die Anwendung der Exponentialfunktion auf die unendliche Reihe $\mathrm{f}(x)$, wohingegen $\exp_{\mathrm{p,q}}$ deren Anwendung auf die rationale Approximation p/q bedeuten soll. Das zweite Gleichheitszeichen in der ersten Formelzeile ergibt sich aus (24).

Die eben gefundene Abschätzung ermöglicht es, für beliebiges $x \in [z]$ den Wert $\Gamma(x)$ durch

$$\Gamma(x) \in [\frac{1}{\exp(\frac{\mathrm{p}_6(x-2)}{\mathrm{q}_5(x-2)})}]\cdot[1-\gamma\,,1+\gamma]\,. \tag{25}$$

mit $\gamma := 11.5E-17 \; > \; 1.0001\Delta$ einzuschließen. Wendet man noch (eventuell mehrfach) die Funktionalgleichung $\Gamma(x+1) = x \cdot \Gamma(x)$ an, so ist man in der Lage, $\Gamma(x)$ für beliebige reelle Werte mit Ausnahme der nichtpositiven ganzen Zahlen sicher einzuschließen. Dies ist möglich, da die Approximation an $1/\Gamma(x)$ über ein Intrevall $[z]$ der Breite 1 durchgeführt wurde.

# 5 Absicherung des Verlaufes von Fehlerkurven

Es wird zunächst das generelle Vorgehen beschrieben. Anschließend werden nocheinmal die bereits in Abschnitt 4 untersuchten Beispiele herangezogen.

## 5.1 Generelles Vorgehen

Der Verlauf der Fehlerkurve $\mathrm{err}(x)$ gemäß (4) soll nun für das gesmate Approximationsintervall durch eine sichere Unter- bzw. Oberkurve eingeschlossen werden. Die Inklusionsbeziehung (7) sowie die Überlegungen, welche auf (9) führten sagen aus, daß für den exakten Wert des relativen Fehlers an der Stelle $x$ das folgende gilt:

$$\mathrm{err}(x) \in \frac{1}{\mathrm{f}([z])} \cdot \{ \; \frac{\sum_{k=0}^{L} [r_k] \cdot ([z]-x_0)^k}{\mathrm{q}_N([z]-x_0)} + [-\alpha, \, \alpha] \; \} \tag{26}$$

Diese Beziehung ist wieder für alle $x$ des Approximationsintervalls $[z]$ gültig. Um nun den exakten Fehlerverlauf graphisch durch eine Unter- bzw. eine Oberkurve sicher einzuschließen, wird das Gesamtintervall $[z]$ so in Teilintervalle $I_j$ unterteilt, daß jedes dieser Teilintervalle gerade der breite einer Bildschirmspalte entspricht. Für jedes dieser Intervalle $I_j$ wird dann die rechte Seite von (26) intervallmäßig ausgewertet. Dabei ergibt sich der $j$te Punkt der Unterkurve durch das Infimum des $j$ten Ergebnisintervalls, der entsprechende Punkt der Oberkurve durch dessen Supremum.

In der Regel wird bei dem eben beschriebenen Vorgehen die Intervallaufblähung bei der intervallmäßigen Berechnug des Ergebnisintervalls zu groß sein, so daß Unter- und Oberkurve weit auseinanderliegen. In solchen Fällen gilt es, die Wertebereichseinschließung der rechten Seite von (26) über dem aktuellen Teilbereich zu verbessern. Dies kann z. B. wie üblich dadurch geschehen, daß ein solches kritisches Teilintervall nocheinmal selbst in feinere Teilintervalle unterteilt wird.

Die folgenden Abbildungen zeigen Bereiche, in denen für die in Paragraph 4 untersuchten Funktionsapproximationen die Fehlerkurven sicher verlaufen. Im Anhang findet sich das PASCAL-XSC Programmlisting zur Untersuchung der rationalen Approximation, wie sie im Zusammenhang mit der Gammafunktion verwendet wird.

Die jeweils dritte Kurve in den Abbildungen wurde berechnet, indem die Fehlerkurve $\mathrm{err}(x)$ jeweils in einem zufällig gewählten Punkt eines jeden zu einer Bildspalte (ein Pixel breit) gehörenden Abszissenintervalls ausgewertet wurde. Bei allen Abbildungen ist zu beachten, daß die Ordinatenwerte skaliert sind. Man vergleiche den Wert der Variablen `PlotScale` des im Anhang aufgelisteten PASCAL-XSC Programms bzw. die ebenfalls angegebene Protokolldatei.

## 5.2 Beispiele von Fehlerkurven

Es werden hier die in Abschnit 4 bereits untersuchten Beispiele nochmals aufgegriffen.

### Fehlerkurve zur Approximation der Fehlerfunktion

Dieser Unterabschnitt zeigt einige Abbildungen zum Fehlerverlauf der in Pargraph 4.1 besprochenen Approximation an die Fehlerfunktion erf($x$). Es werden Zerlegungen des Approximationsintervalles mit 2048, 33333 und 333333 gewählt. Die oberen bzw. unteren Graphen sind dabei Treppenfunktionen, die den tatsächlichen Verlauf der Fehlerkurve sicher einschließen. Die Fehlerkurve zeigt den Verlauf des relativen Approximationsfehlers

$$\mathrm{err}(t) := \frac{\mathrm{erf}(t) - t\dfrac{\mathrm{p}_M(t^2)}{\mathrm{q}_N(t^2)}}{\mathrm{erf}(t)} = \frac{\dfrac{\mathrm{erf}(t)}{t} - \dfrac{\mathrm{p}_M(t^2)}{\mathrm{q}_N(t^2)}}{\dfrac{\mathrm{erf}(t)}{t}} \quad t^2 = x \in [0,\, 0.65^2] \;\; \text{(siehe (18))}\,.$$

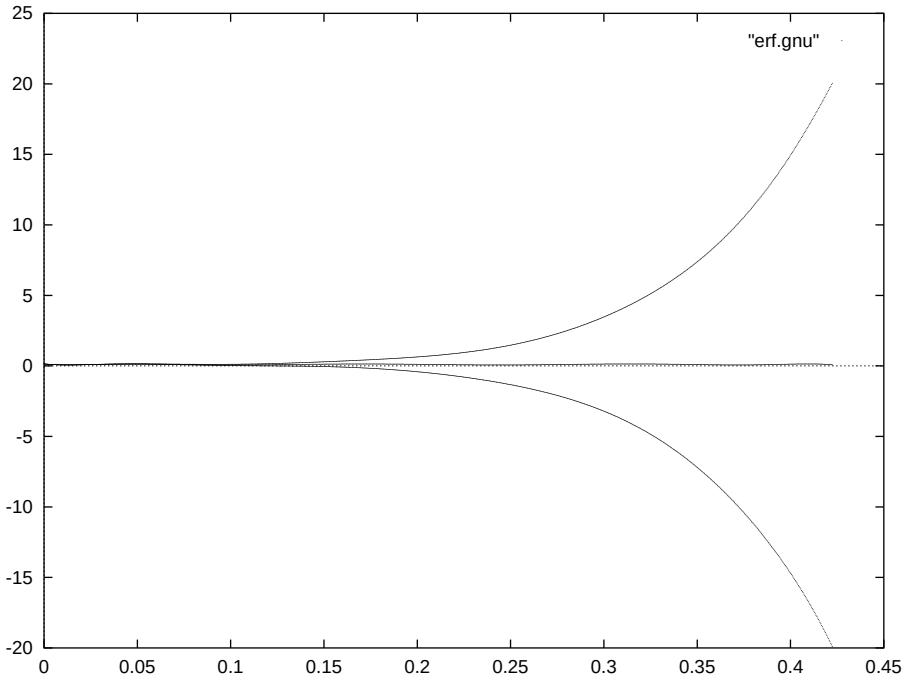



Abbildung 1: Fehlerkurve bei 2 048 Teilintervallen; erf

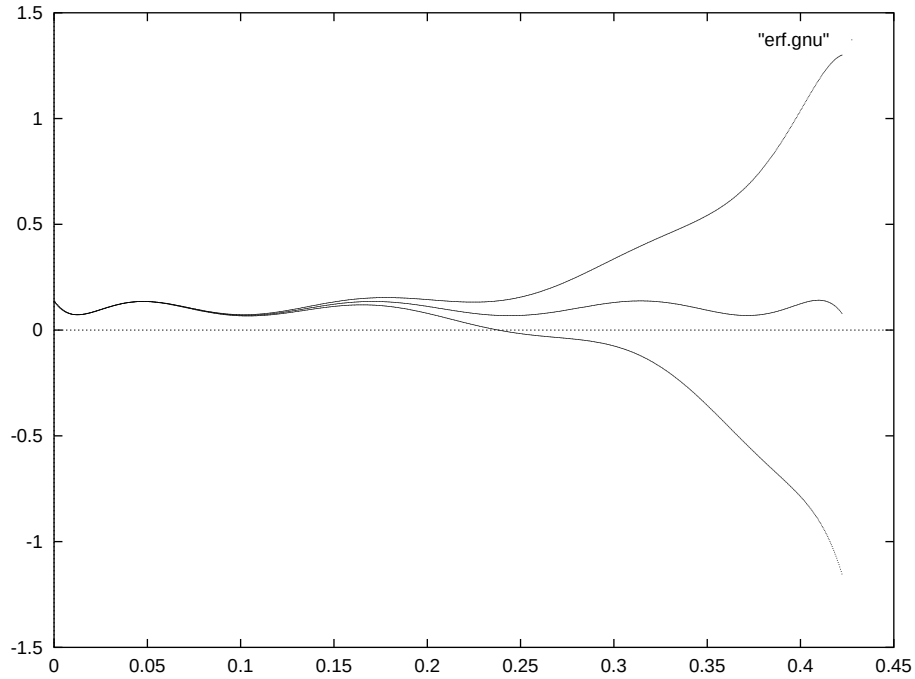



Abbildung 2: Fehlerkurve bei 33 333 Teilintervallen; erf

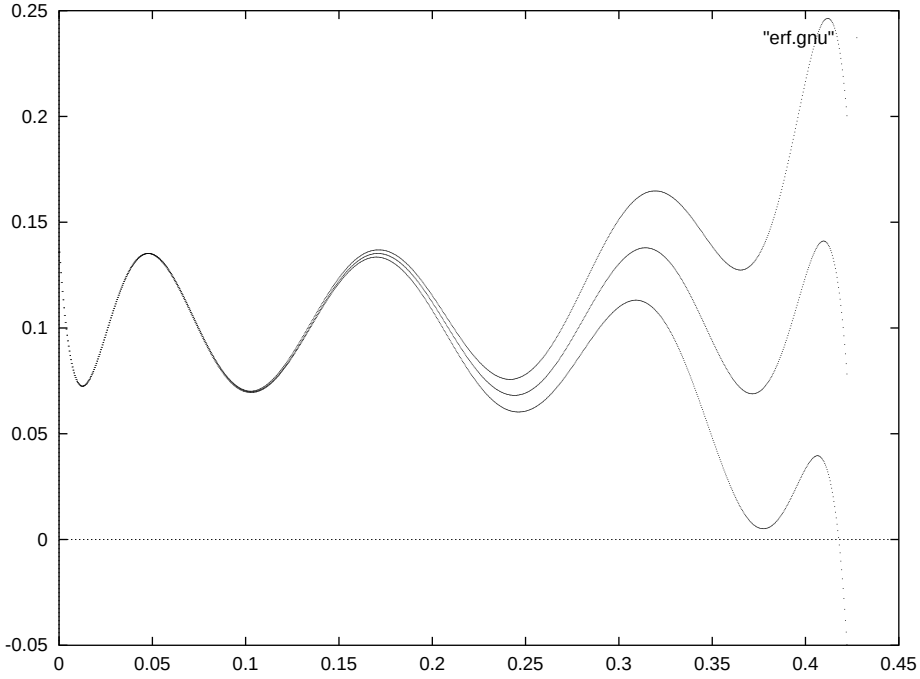



Abbildung 3: Fehlerkurve bei 333 333 Teilintervallen; erf

Verfeinert man die Zerlegung weiter, so fallen die drei Kurven optisch zusammen.

**Fehlerkurve im Zusammenhang mit der Approximation von $\Gamma(x)$**

Es wird hier die in Paragraph 4.2 gefundene Approximation näher untersucht. Die Approximationskoeffizienten sind in Tabelle 1 auf Seite 11 aufgeführt. Die Abbildungen zeigen die mit dem im Anhang angegebenen Programm berechneten Ergebnisse für 2048, 33333 und 333333 Teilintervalle. Die Fehlerkurve zeigt den Verlauf des absoluten Fehlers $\mathrm{err}(x) := -\ln(\Gamma(x)) - \mathrm{p}_4(x)/\mathrm{q}_4(x)$ im Approximationsintervall $[1.5,\ 2.5]$.

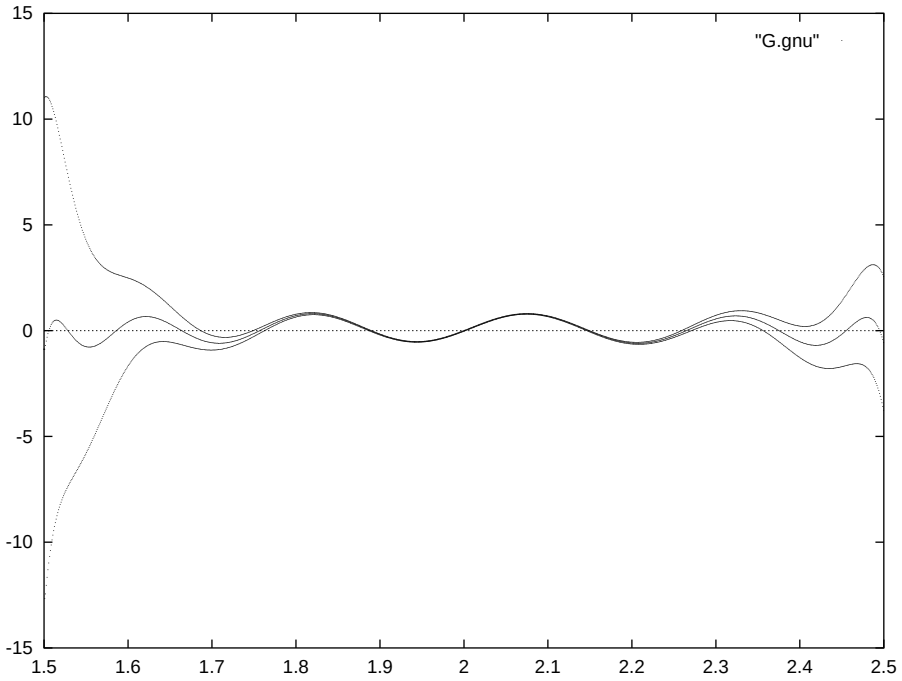



Abbildung 4: Fehlerkurve bei 2 048 Teilintervallen; Gamma

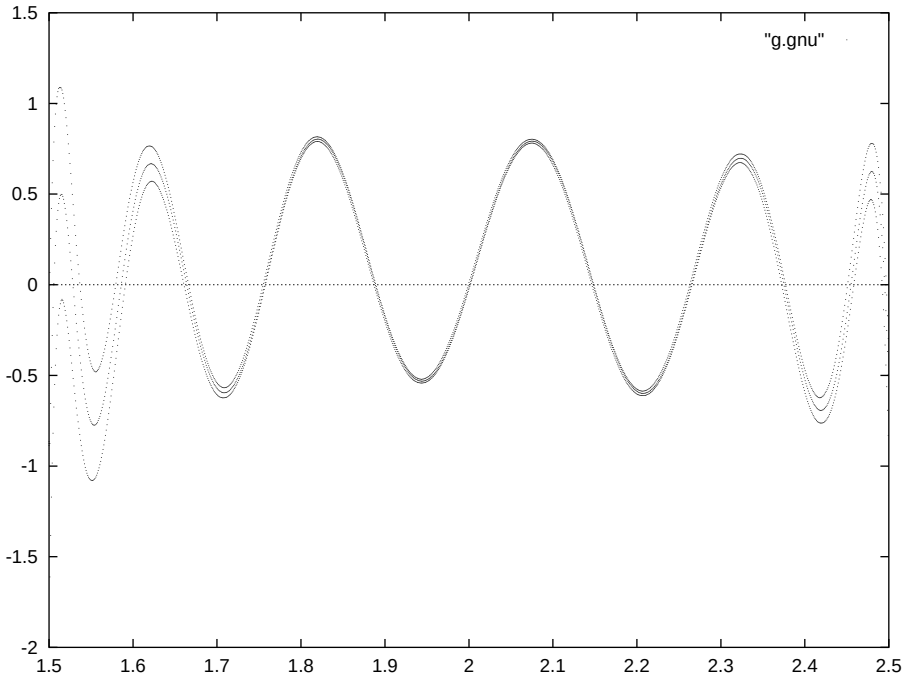



Abbildung 5: Fehlerkurve bei 33 333 Teilintervallen; Gamma

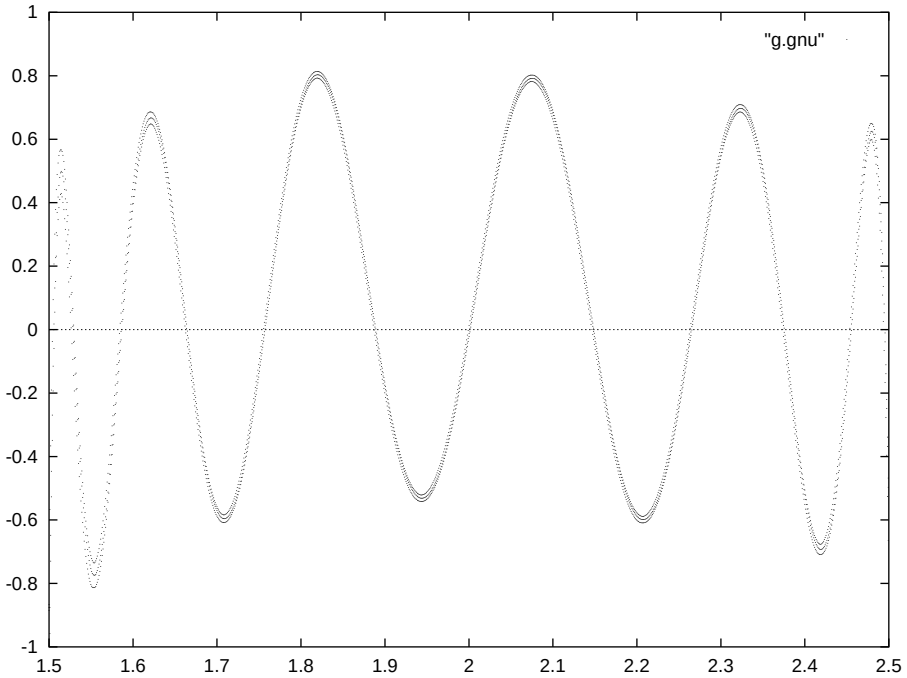



Abbildung 6: Fehlerkurve bei 333 333 Teilintervallen; Gamma

Man sieht sehr schön wie sich die Unter- und Oberfunktion bei fortwährender Verfeinerung der Intervallunterteilung annähern. Bei hinreichend feiner Zerlegung ergeben sich sehr gute Einschließungen.

**Modifizierte Approximation**

Die im letzten Unterabschnitt untersuchte Approximation wurde von Blomquist [5] mit Mathematica berechnet. Betrachtet man die Koeffizienten des Zählerpolynoms, so stellt man fest, daß $p_0$ nicht, wie man vielleicht erwarten würde, null ist. Es gilt ja für $x := x_0 = 2$, daß die zu approximierende Funktion (21)

$$f(x) = \sum_{k=0}^{\infty} s_k \cdot (x-2)^k \tag{27}$$

an der Entwicklungsstelle $x_0$ verschwindet (d. h. $f(x_0) = 0$). Setzt man dagegen den Entwicklungspunkt in die rationale Approximation (20) ein, so ergibt sich gerade der Koeffizient

$$p_0 = 1.35327\ldots \cdot 10^{-17} \tag{28}$$

des Zählerpolynoms. Dies bedeutet insbesondere, daß der maximale absolute Fehler der Approximation nicht kleiner als $p_0$ sein kann.

Es soll zum Schluß noch untersucht werden, wie sich die Fehlerkurve ändert, wenn man die Approximation (20) im Koeffizienten $p_0$ modifiziert. Für die Einschließung der Fehlerkurve wurde also $p_0 := 0$ verwendet. Alle anderen Koeffizienten stimmen mit denen der Tabelle 1 überein.

Auf den ersten Blick sehen die Fehlerkurven der Modifikation und der ursprünglichen Approximation sehr ähnlich aus. Tatsächlich stellt man fest, daß die modifizierte Approximation nicht nur eine kleinere Fehlerschranke aufweist, sondern daß durch die vorgenommene Modifikation auch die Laufzeit verringert wird. Im Zählerpolynom entfällt die sonst notwendige Addition des Konstanten Gliedes $p_0$.

Die sichere Fehlerschranke der ursprünglichen Approximation war

$$11.5E-17\ ,$$

wohingegen die verbesserte Schranke der modifizierten (und dadurch etwas effizienteren) Approximation bei

$$8.8E-17$$

liegt.

# 6 Diskussion

Insgesamt sieht man, daß mit den in dieser Arbeit beschriebenen Methoden sehr gute Ergebnisse erzielt werden. Möchte man Laufzeitoptimierte Funktionen implementieren, so muß man in der Regel von einfach abschätzbaren Taylorpolynomapproximationen abgehen und Bestapprioximationen verwenden. Eine sichere Schrankenberechnung (bei gleichzeitig hoher numerischer Güte) für den Fehler solcher Approximationen leisten die vorgestellten Verfahren. Die Methoden sind gerade bei der Entwicklung von Intervallfunktionen, also bei Funktionen, welche ohne eine verläßliche Fehlerabschätzung nicht realisierbar sind, unentbehrlich (vgl. z. B. [12]).

Das im Anhang angegebene Programm kann dadurch verbessert werden, daß die Breite des von links nach rechts über das Approximationsintervall wandernden 'Fensters' adaptiv dem Verlauf der Fehlerkurve angepaßt wird. Gleichzeitig können Methoden der Globalen Optimierung verwendet werden, um auf den Teilintervallen den Wertebereich einzuschließen. Durch die Zerlegung des Approximationsintervalles zerfällt das Gesamtproblem in natürlicher Weise in viele kleine Teilprobleme, die alle parallel bearbeitet werden können.

Die Umsetzung des Verfahrens in ein lauffähiges Programm benötigt allerdings auch mächtige Werkzeuge seitens der verwendeten Programmiersprache. Gleitkommaintervalloperationen sowie Langzahlintervalloperationen werden benötigt, ein Operatorkonzept sollte vorhanden sein. Diese Dinge werden z. B. von den sogenannten XSC-Sprachen dem Benutzer zur Verfügung gestellt.

# 7 Anhang

Das folgende Programmlisting zeigt eine Implementierung des vorgestellten Verfahrens in der Programmiersprache PASCAL-XSC. In dieser Sprache stehen mächtige Module für Intervalloperationen und Langzahlintervalloperationen zur Verfügung. All diese Operationen können über ihre in der Mathematik üblichen Operatorsymbole angesprochen werden. Diese klare Notation zusammen mit der Kommentierung des Quellcodes ergeben ein leicht zu verstehendes Programmlisting. Eine Einschließung der Eulerschen Konstanten sowie Einschließungen von Werten der Riemannschen Zetafunktion werden mittels der Eulerschen Summenformel [15] berechnet. Diese Berechnungen sind in das Modul `mpi_util` ausgelagert.

```
{------------------------------------------------------------------------}
{    Verifikation der Fehlerkurve einer rationalen Approximation         }
{------------------------------------------------------------------------}
PROGRAM GammaFnc;
{------------------------------------------------------------------------}
{    Approximation im Zusammenhang mit Gamma(x)                          }
{------------------------------------------------------------------------}

USE i_ari;       { Intervallarithmetik                    }
USE mp_ari;      { Reelle Langzahlrechnung                }
USE mpi_ari;     { Langzahlintervallrechnung              }
USE mpi_util;    { Eulersche Konstante, Zetafunktion, ... }
USE x_real;      { Hexadezimale Ausgabe, ...              }

CONST MaxDegree = 200;  { Maximaler Grad bei Produktpolynomen           }
CONST np = 6;           { Zaehlergrad der rationalen Approximation      }
CONST nq = 5;           { Nennergrad der rationalen Approximation       }
CONST ns = 26;          { Grad der nahezu perfekten Reihenapproximation }
CONST alpha= 1e-20;     { Abbruchfehlerschranke im betrachteten Bereich }
CONST x0 = 2.0;         { Entwicklungsstelle                            }
CONST Prec = 6;         { Genauigkeit der Langzahlintervallarithmetik   }

TYPE poly    = ARRAY[0..MaxDegree] OF real;
TYPE ipoly   = ARRAY[0..MaxDegree] OF interval;
TYPE mpipoly = ARRAY[0..MaxDegree] OF mpinterval;
VAR p, q:  poly; { Zaehler- und Nennerpolynom der rat. Approximation }
    s:  mpipoly; { Nahezu perfekte Approximation                     }
    Diff: ipoly; { Differenzpolynom s(x)*q(x)-p(x)                  }
```

```
    fPolEncl: ARRAY [0.. ns] OF interval; { Einschliessung von f     }
    RelErr: boolean;
    Prot: text;          { Datei zum Protokollieren der Berechnungen }

PROCEDURE ComputeDiff(s: mpipoly; q, p: poly; VAR Diff: ipoly);
{ Bestimme Intervallkoeffizienten des Differenzpolynoms }
{   Diff(x) = s(x)*q(x)-p(x)                             }
VAR h: mpinterval;  { Hilfsgroesse fuer mehrfach genaue Rechnung }
    k, j: integer;
    Koeff: interval;
BEGIN
  mpinit(h);  { Initialisierung der mp-Variablen }
  writeln(Prot, 'Koeffizienten des Differenzpolynoms: ');
  { Grad des Ergebnispolynoms: Grad s + Grad q, d.h. Grad Diff:= ns+nq   }
  FOR k:= 0 TO ns+nq DO BEGIN
    h:= 0;
    FOR j:= 0 TO k DO BEGIN
       h:= h + s[j]*q[k-j];
    END;
    h:= h - p[k];
    Koeff:= h;       { Mehrfachgenaues Intervall in 'normales' Intervall }
    Diff[k]:= Koeff;      { Einschluss des k-ten Polynomkoeffizienten    }
      writeln(Prot, '  Diff(', k:3, ')= ', Diff[k]);
    IF diam(Koeff)=0 THEN BEGIN
       writeln('Koeffizient Diff(', k:0, ') ist Punktintervall!');
       writeln(Prot, '  Koeffizient Diff(', k:0, ') ist Punktintervall!');
    END
    ELSE IF succ(inf(Koeff)) < sup(Koeff) THEN BEGIN
       writeln('  Koeffizient Diff(', k:0, ') ist nicht maximal genau!');
       writeln(Prot, '  Koeffizient Diff(', k:0,
                                      ') ist nicht maximal genau!');
    END;
  END;
  mpfree(h);  { Freigabe der lokalen mp-Variablen }
END;

{ Auswertung des Nennerpolynoms q(x) }
FUNCTION EvalQ(x: interval): interval;
VAR k: integer;
    s: interval;
BEGIN
  s:= intval(q[nq]);
  FOR k:= nq-1 DOWNTO 0 DO
    s:= s*(x-x0) + intval(q[k]);
  EvalQ:= s;
END;

{ Auswertung des Differenzpolynoms Diff(x) }
FUNCTION EvalDiff(x: interval): interval;
VAR k: integer;
    s: interval;
BEGIN
  s:= diff[nq+ns];
  FOR k:= nq+ns-1 DOWNTO 0 DO
    s:= s*(x-x0) + Diff[k];
  EvalDiff:= s;
END;

FUNCTION Err(x: interval): interval; { Fehlerkurve der Approximation }
{ Die Fehlerkurve wird durch ( s(x)*q(x) - p(x) ) / q(x) beschrieben }
BEGIN
  Err:= EvalDiff(x) / EvalQ(x) + intval(-alpha, alpha);
END;
```

```
{ Einschliessung des Wertebereiches von f ueber dem Intervall x }
{ Es wird die nahezu perfekte Approximation s verwendet.       }
FUNCTION fEncl(x: interval): interval;
VAR k: integer;
    s: interval;
BEGIN
  s:= fPolEncl[ns];
  FOR k:= ns-1 DOWNTO 0 DO
    s:= s*(x-x0) + fPolEncl[k];
  fEncl:= s + intval(-alpha, alpha);  { Fehler beruecksichtigen }
END;

{-------------------------------------------------------------------------}
VAR ProgName: string; { Prog-Name zur Kennzeichnung von Ausgabedateien  }
    k, n: integer;
    h: real;
    Anz, AnzTeilInt, TeilIntProPixel: integer;
    c: mpinterval;
    X: interval;      { Aktuelles Fenster                          }
    ErrX: interval;   { Err(X): Auswertung ueber aktuellem Fenster  }
    OutAll: text;     { Daten fuer GnuPlot erzeugen                }
    PlotScale: real;  { Skalierungsfaktor fuer Graphik in GnuPlot  }
    InfErrX, SupErrX: real;
    Left, Right: real;   { Grenzen des Approximationsintervalls    }
    Mini, MiniUb: real;  { Einschluss des minimalen Fehlers        }
    MiniX: interval;     { Hier tritt der minimale Fehler auf      }
    Maxi, MaxiLb: real;  { Einschluss des maximalen Fehlers        }
    MaxiX: interval;     { Hier tritt der maximale Fehler auf      }
    { max( |Mini|, |Maxi| ) ist die gesuchte sichere Oberschranke  }
    {   von |f - p/q|  bzw von |(f -p/g)/f| im betrachteten        }
    {   Approximationsintervall [Left, Right].                     }
BEGIN
  ProgName:='gammafnc';
  rewrite(Prot,   ProgName+ '.out');  { Protokoll der Berechnungen          }
  rewrite(OutAll, ProgName+ '.gnu');  { Zum Plotten aller Kurven mit Gnuplot }

  writeln(Prot, 'Ausgabe des Programms ' + ProgName);
  writeln(Prot);

  writeln(Prot, 'Approximationsfehler einer rat. Approximation fuer ');
  writeln(Prot, '   f(x) = -ln(Gamma(x))    ( err(x):= f(x)-p(x)/q(x) ) ');
  writeln(Prot);
  setprec(Prec);        { Genauigkeit der Langzahlintervalloperationen }
  writeln(Prot, 'Genauigkeitseinstellung: setprec(', Prec:0, ')');

  { Taylor Reihe fuer f(x) = -ln(Gamma(x))                              }
  { Berechne Intervallkoeffizienten der nahezu perfekten Approximation }
  {   an f(x).                                                          }
  FOR k:= 0 TO MaxDegree DO BEGIN
    mpinit(s[k]); { Initialisiere Langzahlintervallkoeffizienten }
    s[k]:=0;
  END;

  writeln(      'Koeffizienten der nahezu perfekten Approximation:');
  writeln(Prot, 'Koeffizienten der nahezu perfekten Approximation:');
  k:= 0;
  s[k]:= 0;
  fPolEncl[k]:= s[k];
  writeln(      k:3, ' ', fPolEncl[k]);
  writeln(Prot, k:3, ' ', fPolEncl[k]);
  k:= 1;
  s[k]:= EulerConst - 1;
```

```
fPolEncl[k]:= s[k];
writeln(      k:3, ' ', fPolEncl[k]);
writeln(Prot, k:3, ' ', fPolEncl[k]);
FOR k:= 2 TO ns DO BEGIN
  IF odd(k) THEN
     s[k]:=  ( Zeta(_mpinterval(k)) - 1 ) / k
  ELSE
     s[k]:= -( Zeta(_mpinterval(k)) - 1 ) / k;
  fPolEncl[k]:= s[k];  { Einschluss durch 'normales' Intervall }
  writeln(k:3, ' ', fPolEncl[k]);
  writeln(Prot, k:3, ' ', fPolEncl[k]);
END;

{ Koeffizienten des Zaehlerpolynoms aus Computeralgebrapaket       }
{ Es wird jeweils der angegebene Dezimalwert zur naechstgelegenen  }
{    Gleitkommazahl gerundet.                                      }
 p[0]:= +1.35327304816540868E-17;      { Originaler Wert           }
 { p[0]:= 0; }                         { Verbesserte Approximation }
 p[1]:=-4.22784335098468688E-1;
 p[2]:=-8.44844014435089770E-1;
 p[3]:=-5.59579952051023151E-1;
 p[4]:=-1.53556364516808078E-1;
 p[5]:=-1.65082854342856732E-2;
 p[6]:=-4.72192577137972617E-4;

 writeln(Prot, 'Polynomkoeffizienten des Zaehlerpolynoms: ');
 FOR k:= 0 TO np DO
    writeln(Prot, p[k]:'X', '   ', p[k]);

{ Koeffizienten des Nennerpolynoms aus Computeralgebrapaket        }
{ Es wird jeweils der angegebene Dezimalwert zur naechstgelegenen  }
{    Gleitkommazahl gerundet.                                      }
q[0]:= 1.00000000000000000E+0;
q[1]:= 1.23556370859698140E+0;
q[2]:= 5.40473405509170129E-1;
q[3]:= 9.91256197612588100E-2;
q[4]:= 6.86477527603966571E-3;
q[5]:= 1.09503636580423657E-4;

writeln(Prot, 'Polynomkoeffizienten des Nennerpolynoms: ');
FOR k:= 0 TO nq DO
   writeln(Prot, q[k]:'X', '   ', q[k]);

ComputeDiff(s, q, p, Diff);  { Polynom Diff(x)  =  s(x)*q(x) - p(x) }
                             { Der Grad von Diff(x) ist ns+nq.      }

RelErr:= false;  { Es ist der maximale absolute Fehler gesucht! }
writeln(Prot, 'RelErr = ', RelErr);
{ Betrachtetes Approximationsintervall = [1.5, 2.5]              }
Left:=  1.5;
Right:= 2.5;
writeln('Approximationsintervall = ', intval(Left, Right) );
writeln(Prot, 'Approximationsintervall = ', Left, '  ',  Right );
writeln('Funktionsauswertung ueber Gesamtintervall: ');
writeln('  ', fEncl(intval(Left, Right)) );
writeln(Prot, 'Funktionsauswertung ueber Gesamtintervall: ');
writeln(Prot, '  ', fEncl(intval(Left, Right)) );
writeln(Prot, 'Entwicklungsstelle x0 = ', x0);
X:= Left;          { Linken Randpunkt erfassen }
ErrX:= err(X);     { Einschliessung des Fehlers am linken Randpunkt }
IF RelErr THEN ErrX:= ErrX/fEncl(X);
InfErrX:= inf(ErrX);  { Initialisierung fuer repeat Schleife }
SupErrX:= sup(ErrX);  { Initialisierung fuer repeat Schleife }
```

```
MiniX:=X;
Mini:=   inf(ErrX);
MiniUb:= sup(ErrX);
MaxiX:= X;
Maxi:=   sup(ErrX);
MaxiLb:= inf(ErrX);
writeln(Prot, 'Am linken Randpunkt: ');
writeln(      '  Mini: ', Mini, '   Maxi: ', Maxi);
writeln(Prot, '  Mini: ', Mini, '   Maxi: ', Maxi);

X:= Right;         { Rechten Randpunkt erfassen }
ErrX:= err(X);     { Einschliessung des Fehlers am rechten Endpunkt }
IF RelErr THEN ErrX:= ErrX/fEncl(X);
IF Mini > inf(ErrX) THEN BEGIN
  MiniX:= X;
  Mini:=   inf(ErrX);
  MiniUb:= sup(ErrX);
END;
IF Maxi < sup(ErrX) THEN BEGIN
  MaxiX:= X;
  Maxi:=   sup(ErrX);
  MaxiLb:= inf(ErrX);
END;
writeln(Prot, 'Am rechten Randpunkt: ');
writeln(      '  Mini: ', Mini, '   Maxi: ', Maxi);
writeln(Prot, '  Mini: ', Mini, '   Maxi: ', Maxi);

{ write('Anzahl von Teilintervallen : '); read(AnzTeilInt); writeln; }
AnzTeilInt:= 7777777;
IF AnzTeilInt < 1024 THEN AnzTeilInt:= 1024;
writeln('Gewaehlte Anzahl von Teilintervallen: ', AnzTeilInt);
writeln(Prot, 'Gewaehlte Anzahl von Teilintervallen: ', AnzTeilInt);

h:= (Right-Left)/AnzTeilInt; { Breite eines Teilintervalls (Fenster) }
writeln(      'Breite eines Teilintervalls: ', h);
writeln(Prot, 'Breite eines Teilintervalls: ', h);

X:= Left; { Fenster ueber gesamtes Intervall bewegen }
Anz:= 0;
TeilIntProPixel:= AnzTeilInt DIV 1024;
PlotScale:= 1e15; { Skalierungsfaktor fuer Graphik in GnuPlot }
writeln('Skalierungsfaktor PlotScale fuer Graphik: ', PlotScale);
  REPEAT
  IF sup(X) > Right THEN X:= intval(inf(X), Right);
  ErrX:= err(X);   { Einschliessung des Fehlers im aktuellen  }
                   {    Teilintervall X.                      }
  IF RelErr THEN ErrX:= ErrX/fEncl(X);
  { Achtung!  Skalierung der Werte der Fehlerkurve! }
  IF Anz  MOD TeilIntProPixel = 0 THEN BEGIN
     writeln( OutAll, mid(x), '  ', PlotScale*InfErrX );
     writeln( OutAll, mid(x), '  ', PlotScale*SupErrX );
     InfErrX:= inf(ErrX);
     SupErrX:= sup(ErrX);
  END ELSE BEGIN
     IF InfErrX > inf(ErrX) THEN InfErrX:= inf(ErrX);
     IF SupErrX < sup(ErrX) THEN SupErrX:= sup(ErrX);
  END;

  IF Mini > inf(ErrX) THEN BEGIN { Aktualisiere Minimalwerte }
    MiniX:= X;           { Abszisse des Minimums }
    Mini:=   inf(ErrX); { Sichere Unterschranke fuer Minimum }
    MiniUb:= sup(ErrX); { Sichere Oberschranke  fuer Minimum }
  END;
```

```
    IF Maxi < sup(ErrX) THEN BEGIN { Aktualisiere Maximalwerte }
      MaxiX:= X;          { Abszisse des Maximums              }
      Maxi:=   sup(ErrX); { Sichere Oberschranke  fuer Maximum }
      MaxiLb:= inf(ErrX); { Sichere Unterschranke fuer Maximum }
    END;
    Anz:= Anz+1;
    IF Anz MOD 1000 = 0 THEN  { Bildschirmanzeige }
      writeln(Anz:9, ' Mini= ', Mini, '  Maxi= ', Maxi);
    X:= intval(sup(X), sup(X)+h); { Fenster um Schrittweite nach rechts }
    IF (inf(X) >= Right) THEN X:= Right;
  UNTIL X = Right;   { Rechter Rand des Approx.-Intervalls ist erreicht }

  writeln('Minimalstelle: ', MiniX);
  writeln('Einschliessung des Minimalwertes: ');
  writeln('  Mini: ', Mini,   '      MiniUb: ', MiniUb);
  writeln('Maximalstelle: ', MaxiX);
  writeln('Einschliessung des Maximalwertes: ');
  writeln('  MaxiLb: ', MaxiLb,   '      Maxi: ', Maxi);

  writeln(Prot, 'Minimalstelle: ', MiniX);
  writeln(Prot, 'Einschliessung des Minimalwertes: ');
  writeln(Prot, '  Mini: ', Mini,   '      MiniUb: ', MiniUb);
  writeln(Prot, 'Maximalstelle: ', MaxiX);
  writeln(Prot, 'Einschliessung des Maximalwertes: ');
  writeln(Prot, '  MaxiLb: ', MaxiLb,   '      Maxi: ', Maxi);
END.
{------------------------------------------------------------------------}
```

Im folgenden ist die Protokolldatei dieses Programms wiedergegeben. Dabei wurde das Approximationsintervall $[1.5, 2.5]$ in 7 777 777 gleichbreite Teilintervalle zerlegt.

```
Ausgabe des Programms gammafnc

Approximationsfehler einer rat. Approximation fuer
   f(x) = -ln(Gamma(x))    ( err(x):= f(x)-p(x)/q(x) )

Genauigkeitseinstellung: setprec(6)
Koeffizienten der nahezu perfekten Approximation:
  0 [  0.000000000000000E+000,  0.000000000000000E+000 ]
  1 [ -4.227843350984672E-001, -4.227843350984671E-001 ]
  2 [ -3.224670334241133E-001, -3.224670334241132E-001 ]
  3 [  6.735230105319808E-002,  6.735230105319811E-002 ]
  4 [ -2.058080842778455E-002, -2.058080842778454E-002 ]
  5 [  7.385551028673984E-003,  7.385551028673986E-003 ]
  6 [ -2.890510330741524E-003, -2.890510330741522E-003 ]
  7 [  1.192753911703260E-003,  1.192753911703262E-003 ]
  8 [ -5.096695247430425E-004, -5.096695247430423E-004 ]
  9 [  2.231547584535793E-004,  2.231547584535794E-004 ]
 10 [ -9.945751278180855E-005, -9.945751278180853E-005 ]
 11 [  4.492623673813313E-005,  4.492623673813315E-005 ]
 12 [ -2.050721277567070E-005, -2.050721277567069E-005 ]
 13 [  9.439488275268395E-006,  9.439488275268397E-006 ]
 14 [ -4.374866789907489E-006, -4.374866789907487E-006 ]
 15 [  2.039215753801366E-006,  2.039215753801367E-006 ]
 16 [ -9.551412130407422E-007, -9.551412130407419E-007 ]
 17 [  4.492469198764565E-007,  4.492469198764567E-007 ]
 18 [ -2.120718480555467E-007, -2.120718480555466E-007 ]
 19 [  1.004322482396809E-007,  1.004322482396811E-007 ]
 20 [ -4.769810169363982E-008, -4.769810169363980E-008 ]
 21 [  2.271109460894316E-008,  2.271109460894317E-008 ]
 22 [ -1.083865921489696E-008, -1.083865921489695E-008 ]
 23 [  5.183475041970046E-009,  5.183475041970048E-009 ]
 24 [ -2.483674543802479E-009, -2.483674543802478E-009 ]
```

```
 25 [  1.192140140586091E-009,  1.192140140586092E-009 ]
 26 [ -5.731367241678863E-010, -5.731367241678861E-010 ]
Polynomkoeffizienten des Zaehlerpolynoms:
0000000000000000    0.000000000000000E+000
BFDB0EE6072093EA   -4.227843350984687E-001
BFEB08F650870AC0   -8.448440144350897E-001
BFE1E8143731CDF8   -5.595799520510232E-001
BFC3A7BC25D89EC7   -1.535563645168081E-001
BF90E78C483691B0   -1.650828543428567E-002
BF3EF213AD1CDB01   -4.721925771379726E-004
Polynomkoeffizienten des Nennerpolynoms:
3FF0000000000000    1.000000000000000E+000
3FF3C4DE7388C6B2    1.235563708596982E+000
3FE14B8EE220A0D3    5.404734055091701E-001
3FB9604BEF1201AA    9.912561976125880E-002
3F7C1E3D14E1326F    6.864775276039666E-003
3F1CB4A57626272A    1.095033635804237E-004
Koeffizienten des Differenzpolynoms:
  Diff(  0)= [  0.000000000000000E+000,  0.000000000000000E+000 ]
  Koeffizient Diff(0) ist Punktintervall!
  Diff(  1)= [  1.549369319322788E-015,  1.549369319322789E-015 ]
  Diff(  2)= [  5.468793739146201E-015,  5.468793739146203E-015 ]
  Diff(  3)= [ -1.415389847242486E-013, -1.415389847242484E-013 ]
  Diff(  4)= [ -3.037914487431096E-013, -3.037914487431094E-013 ]
  Diff(  5)= [  3.682403947031797E-012,  3.682403947031799E-012 ]
  Diff(  6)= [  5.415627637868169E-012,  5.415627637868171E-012 ]
  Diff(  7)= [ -4.059768819892463E-011, -4.059768819892461E-011 ]
  Diff(  8)= [ -3.862075977654709E-011, -3.862075977654708E-011 ]
  Diff(  9)= [  2.112882994073416E-010,  2.112882994073417E-010 ]
  Diff( 10)= [  9.398822175389109E-011,  9.398822175389111E-011 ]
  Diff( 11)= [ -4.848992079090931E-010, -4.848992079090929E-010 ]
  Diff( 12)= [  7.181238053068402E-011,  7.181238053068405E-011 ]
  Diff( 13)= [  2.657186221730132E-010,  2.657186221730134E-010 ]
  Diff( 14)= [ -3.569935298761706E-010, -3.569935298761704E-010 ]
  Diff( 15)= [  3.089037018202058E-010,  3.089037018202060E-010 ]
  Diff( 16)= [ -2.220939267649864E-010, -2.220939267649863E-010 ]
  Diff( 17)= [  1.439585143965412E-010,  1.439585143965413E-010 ]
  Diff( 18)= [ -8.737804773161939E-011, -8.737804773161937E-011 ]
  Diff( 19)= [  5.071242559896112E-011,  5.071242559896114E-011 ]
  Diff( 20)= [ -2.850396880231035E-011, -2.850396880231033E-011 ]
  Diff( 21)= [  1.564501681739989E-011,  1.564501681739991E-011 ]
  Diff( 22)= [ -8.433071954612380E-012, -8.433071954612378E-012 ]
  Diff( 23)= [  4.482054462032078E-012,  4.482054462032080E-012 ]
  Diff( 24)= [ -2.355706213038190E-012, -2.355706213038189E-012 ]
  Diff( 25)= [  1.227064736601288E-012,  1.227064736601289E-012 ]
  Diff( 26)= [ -6.345122646421333E-013, -6.345122646421331E-013 ]
  Diff( 27)= [ -2.756261515875841E-010, -2.756261515875840E-010 ]
  Diff( 28)= [ -2.080757865054104E-010, -2.080757865054102E-010 ]
  Diff( 29)= [ -4.890072954499496E-011, -4.890072954499494E-011 ]
  Diff( 30)= [ -3.803911458604656E-012, -3.803911458604654E-012 ]
  Diff( 31)= [ -6.276039908784904E-014, -6.276039908784902E-014 ]
RelErr = FALSE
Approximationsintervall =  1.500000000000000E+000   2.500000000000000E+000
Funktionsauswertung ueber Gesamtintervall:
  [               -3.1E-001,                3.1E-001 ]
Entwicklungsstelle x0 =  2.000000000000000E+000
Am linken Randpunkt:
  Mini: -8.759083409168867E-017   Maxi: -8.757083409097624E-017
Am rechten Randpunkt:
  Mini: -8.759083409168867E-017   Maxi: -7.314886791180949E-017
Gewaehlte Anzahl von Teilintervallen: 7777777
Breite eines Teilintervalls:  1.285714414285727E-007
Minimalstelle: [           1.500000E+000,           1.500001E+000 ]
```

```
Einschliessung des Minimalwertes:
  Mini: -8.790193861745202E-017      MiniUb: -8.725653880839613E-017
Maximalstelle: [          1.818893E+000,           1.818894E+000 ]
Einschliessung des Maximalwertes:
  MaxiLb:  8.030679304085832E-017      Maxi:  8.032884350061394E-017
```

# Literatur


[1] Abramowitz, M., Stegun, I. A.: *Handbook of Mathematical Functions*. Nat. Bur. Standards, Appl. Math. Series, No. **55**, U.S. Government Printing Office, Washington, D.C. 1964.

[2] Agarwal, R. C. et al.: *New Scalar and Vector Elementary Functions for the IBM System/370*. IBM J. Res. Develop., Vol. **30**, No. 2, 1986.

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

[4] Black, Ch. M., Burton, R. B., Miller, T. H.: *The Need for an Industry Standard of Accuracy for Elementary-Function Programs*. ACM Trans. on Math. Software, Vol. **10**, No. 4, pp 361–366, 1984.

[5] Blomquist, F.: *Abschätzung des Approximationsfehlers*. Manuskript, 1995.

[6] Braune, K., Krämer, W.: *High-Accuracy Standard Functions for Intervals*. In M. Ruschitzka (Ed.): *Computer Systems: Performance and Simulation*. Elsevier Science Publishers, 1985.

[7] 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*, pp81–114, Teubner, Stuttgart, 1987.

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

[9] Gal, S.: *Computing Elementary Functions: A New Approach for Achieving High Accuracy and Good Performance.* IBM Technical Report 88.153, 1985.

[10] Hammer, R., Hocks, M., Kulisch, U., Ratz, D.: *Numerical Toolbox for Verified Computing I: Basic Numerical Problems.*

[11] Hart, J. F. et al.: *Computer Approximations.* Wiley, New York / London / Sydney, 1968. Springer–Verlag, Berlin / Heidelberg / New York, 1993.

[12] Hofschuster, W., Krämer, W.: *Ein rechnergestützter Fehlerkalkül mit Anwendungen auf ein genaues Tabellenverfahren*, Preprint 96/5 des IWRMM, Universität Karlsruhe, 1996.

[13] 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).

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

[15] Knopp, K.: *Theorie und Anwendung der unendlichen Reihen*, Springer,1964, 5. Auflage.

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

[17] Krämer, W.: *Fehlerschranken für häufig auftretende Approximationsausdrücke*. ZAMM **69**, pp T44–T47, 1989.

[18] Krämer, W.: *Die Berechnung von Standardfunktionen in Rechenanlagen*. In Chatterji, S. D., Kulisch, U., Laugwitz, D., Liedl, R., Purkert, W. (Eds.): *Jahrbuch Überblicke Mathematik 1992*, Vieweg, Braunschweig, 1992.

[19] 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.

[20] Krämer, W.: *Semimorphic Function Evaluation*. Talk given at the IMACS/GAMMM International Symposium SCAN93, Wien, 1993.

[21] Krämer, W.: *Mathematische Funktionen in Rechenanlagen – Entwurf, Realisierung und Fehlerabschätzungen*. Vorlesungsmanuskrip, Universität Karlsruhe, 1995.

[22] Markstein, P. W.: *Computation of Elementary Functions on the IBM RISC System/6000 Processor*. IBM J. Res. Develop., Vol. **34**, No. 1, 1990.

[23] Muller, J.-M.: *Towards Exact Rounding of the Elementary Functions*. Talk given at the IMACS/GAMMM International Symposium SCAN95, Wuppertal (Germany), 1995.

[24] Rall, L. B.: *Automatic Differentiation: Techniques and Applications*. Lecture Notes in Computer Science, No. **120**, Springer-Verlag, Berlin, 1981.

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

[26] 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.

[27] 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.

[28] Ziv, A.: *Fast Evaluation of Elementary Mathematical Functions with Correctly Rounded Last Bit*. ACM Trans. on Math. Software, Vol. **17**, No. 3, pp 410–423, 1991.

**In dieser Reihe sind bisher die folgenden Arbeiten erschienen:**

**1/1996** Ulrich Kulisch: *Memorandum über Computer, Arithmetik und Numerik.*

**2/1996** Andreas Wiethoff: *C–XSC — A C++ Class Library for Extended Scientific Computing.*

**3/1996** Walter Krämer: *Sichere und genaue Abschätzung des Approximationsfehlers bei rationalen Approximationen.*

**4/1996** Dietmar Ratz: *An Optimized Interval Slope Arithmetic and its Application.*

**5/1996** Dietmar Ratz: *Inclusion Isotone Extended Interval Arithmetic.*

**1/1997** Astrid Goos, Dietmar Ratz: *Praktische Realisierung und Test eines Verifikationsverfahrens zur Lösung globaler Optimierungsprobleme mit Ungleichungsnebenbedingungen.*

**2/1997** Stefan Herbort, Dietmar Ratz: *Improving the Efficiency of a Nonlinear-System-Solver Using a Componentwise Newton Method.*

**3/1997** Ulrich Kulisch: *Die fünfte Gleitkommaoperation für top-performance Computer — oder — Akkumulation von Gleitkommazahlen und -produkten in Festkommaarithmetik.*

**4/1997** Ulrich Kulisch: *The Fifth Floating-Point Operation for Top-Performance Computers — or — Accumulation of Floating-Point Numbers and Products in Fixed-Point Arithmetic.*

**5/1997** Walter Krämer: *Eine Fehlerfaktorarithmetik für zuverlässige a priori Fehlerabschätzungen.*