Zwei Adaptive Quadraturverfahren

Norris Löbnau

Wintersemester 2024

Ziel

Approximation $\hat{I}$ von

$$I(f) := I_a^b(f) := \int_a^b f(x)dx \tag{1}$$

mithilfe einer Quadraturformel

$$\sum_k w_k \cdot f(x_k)$$

mit Knoten x_k und Gewichten w_k. [1]

Inhalt

Grundlagen

Grundlage der heute hergeleiteten Verfahren werden die Trapez-Regel

$$T = \frac{b-a}{2}\Big(f(a) + f(b)\Big)$$

Simpson-Regel

$$S = \frac{b-a}{6}\Big(f(a) + 4f(m) + f(b)\Big),\ m = (a+b)/2$$

und Romberg-Quadratur sein.

Romberg-Quadratur

Prinzip: Die Trapezregel mit Schrittweite $h = (b-a)/n$

$$T(h) := T^n := h\left(\frac{1}{2}(f(a)+f(b)) + \sum_{i=1}^{n-1} f(a+ih)\right)$$

lässt sich für hinreichend glatte Funktionen in eine asymptotische Entwicklung in h^2 entwicklen, d.h.

$$T(h) = \int_a^b f(t)dt + O(h^2, h^4, h^6, ...)$$

Romberg-Quadratur

Durch Auswertung von $T(h_i)$ mit verschiedenen Schrittweiten $h_i = (b-a)/n_i$ und gezielte Rekombination der Terme wird versucht eine bessere Integralnäherung zu 'extrapolieren'.

Romberg-Quadratur

Algorithmus 9.24 (Extrapolationsverfahren).

1. Wähle für eine gegebene Grundschrittweite H eine Folge von Schrittweiten $h_1, h_2, \ldots$ mit $h_j = H/n_j$, $n_{j+1} > n_j$ und setze $i := 1$.
2. Bestimme $T_{i1} = T(h_i)$.
3. Berechne T_{ik} für $k = 2, \ldots, i$ mit dem Algorithmus von Aitken-Neville

$$T_{ik} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\left(\frac{n_i}{n_{i-k+1}}\right)^p - 1}.$$

4. Falls T_{ii} genau genug oder i zu groß, beende den Algorithmus. Ansonsten erhöhe i um 1 und gehe zurück zu 2.

Abbildung: Extrapolationsverfahren $(p = 2)$

Extrapolationstableau

$$
\begin{array}{lllll}
T_{11} & & & & \\
\searrow & & & & \\
T_{21} & \to T_{22} & & & \\
\vdots & & \ddots & & \\
T_{k-1,1} & \to \cdots & \to T_{k-1,k-1} & & \\
\searrow & \searrow & & \searrow & \\
T_{k1} & \to \cdots & \to T_{k,k-1} & \to T_{kk} &
\end{array}
\qquad
\begin{array}{lllll}
\varepsilon_{11} & & & & \\
\searrow & & & & \\
\varepsilon_{21} & \to \varepsilon_{22} & & & \\
\vdots & & \ddots & & \\
\varepsilon_{k-1,1} & \to \cdots & \to \varepsilon_{k-1,k-1} & & \\
\searrow & \searrow & & \searrow & \\
\varepsilon_{k1} & \to \cdots & \to \varepsilon_{k,k-1} & \to \varepsilon_{kk} &
\end{array}
$$

Abbildung: Extrapolationstableau und Fehlertableau

Mit $\varepsilon_{ik} := |T_{ik} - I(f)|$

Romberg-Quadratur

Sei A_i :=Anzahl zur Berechnung von T_{ii} benötigten Funktionsauswertungen. Zur Minimierung von A_i verwende für die n_i die

- Romberg-Folge: $\{1, 2, 4, 8, 16, ...\}$ oder
- Bulirsch-Folge: $\{1, 2, 3, 4, 6, 8, 12, 16, 24, ...\}$

Problem nicht-adaptiver Verfahren

Nicht-adaptive Quadraturformeln, d.h. Quadraturformeln mit einer im Vorhinein festgelegten Menge an Knoten (und Gewichten) können je nach Integrand ungeeignet sein. Z.B. kann das Integral ungleichmäßig auf das Integrationsintervall 'verteilt' sein $\rightsquigarrow$ Bsp. 'Nadelimpuls'

Nadelimpuls

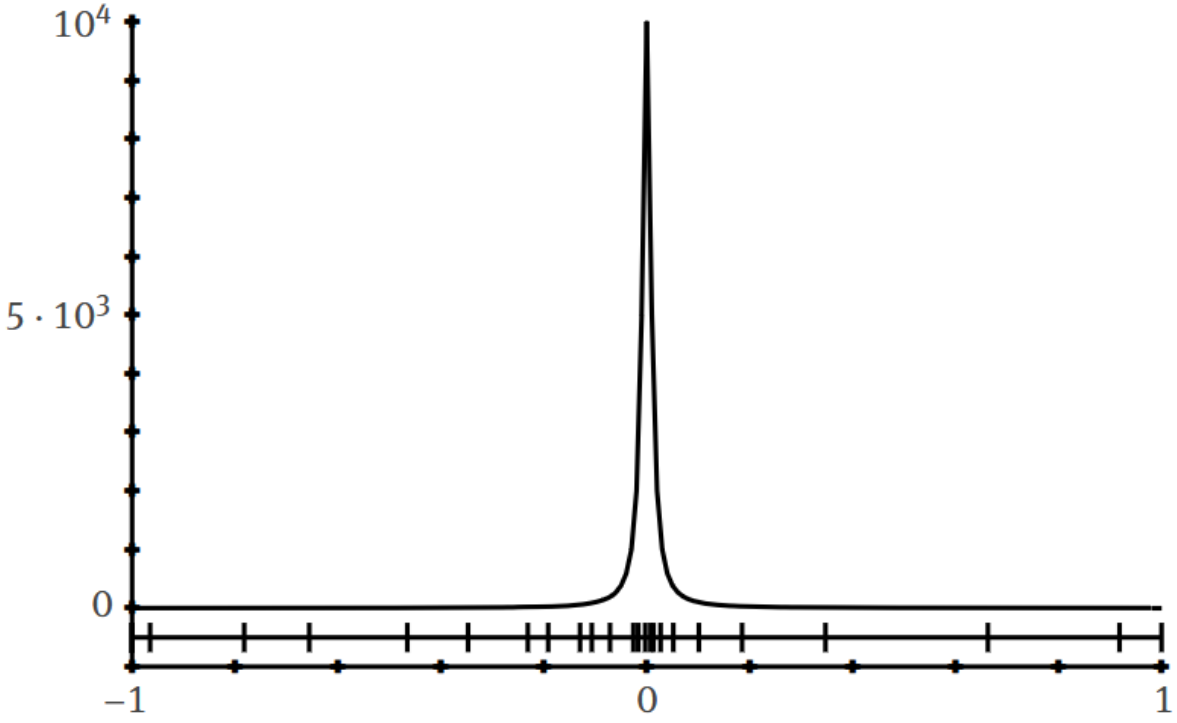


Abbildung: Nadelimpuls $\int_{-1}^{1} dt/(10^{-4} + t^2)$

Nadelimpuls

k	T_{kk}	ε_{kk}	A_k
1	1.999800	$9.9 \cdot 10^{-1}$	2
2	13333.999933	$4.2 \cdot 10^{1}$	3
3	2672.664361	$7.6 \cdot 10^{0}$	5
4	1551.888793	$4.0 \cdot 10^{0}$	9
5	792.293096	$1.5 \cdot 10^{0}$	17
6	441.756664	$4.2 \cdot 10^{-1}$	33
7	307.642217	$1.4 \cdot 10^{-2}$	65
8	293.006708	$6.1 \cdot 10^{-2}$	129
9	309.850398	$7.4 \cdot 10^{-3}$	257
10	312.382805	$7.2 \cdot 10^{-4}$	513
11	312.160140	$2.6 \cdot 10^{-6}$	1025
12	312.159253	$2.5 \cdot 10^{-7}$	2049
13	312.159332	$1.1 \cdot 10^{-9}$	4097

Abbildung: Relative Genauigkeit ε_{kk} der Romberg-Quadratur

Adaptive Verfahren

Alternativ kann das Integrationsintervall problemangepasst aufgeteilt werden, bis das Integral bis auf eine vorgegebene relative Genauigkeit bestimmt ist. Man spricht hierbei von 'Adaptiven Verfahren'. Die relative Genaugikeit muss dabei geschätzt werden.

Adaptives Prinzip

Berechne Approximation $\hat{I}$ von I sodass

$$|\hat{I} - I| \leq |I|\text{tol}$$

bzw. (da I unbekannt)

$$|\hat{I} - I| \leq I_{\text{skal}}\text{tol}$$

wobei I_{skal} in der Größenordnung von I liegen sollte und entweder vorgegeben oder geschätzt wird.

Schrittweitenanpassung

Die Schrittweite kann über 2 Methoden angepasst werden

- Anfangswertmethode
- Randwertmethode

Anfangswertmethode

Das Integral wird 'von links nach rechts' berechnet. Dabei wird das Grundintervall in Teilintervalle $[t_i, t_{i+1}]$ der Länge $H_i := t_{i+1} - t_i$ zerlegt und auf die Teilintegrale

$$I_i := \int_{t_i}^{t_{i+1}} f(t)dt$$

die Romberg-Quadratur bis zu einem Grad q_i angewandt.

Quadraturschritt

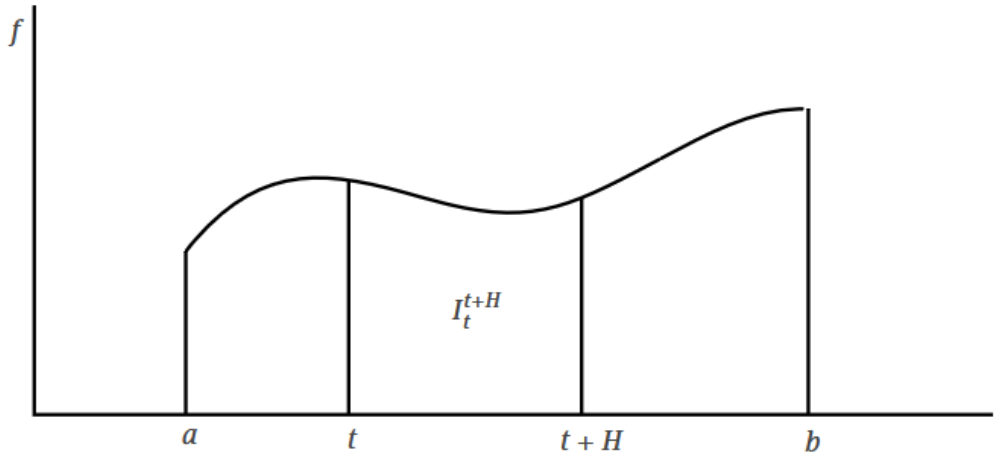


Abb. 9.7. Ein Schritt der adaptiven Romberg-Quadratur.

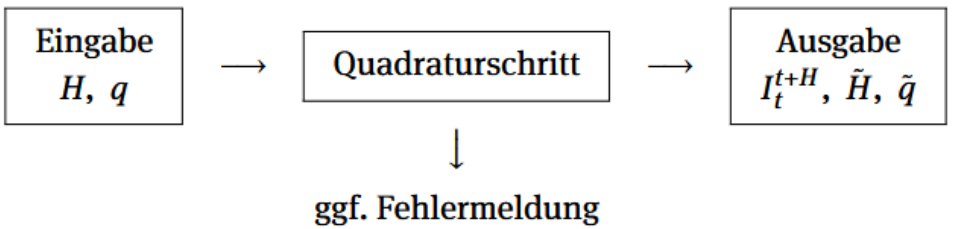


Abb. 9.8. Schematische Darstellung eines Quadraturschrittes.

Abbildung: Ein Quadraturschritt

Fragen

- Welche Schrittweite?
- Welche Quadratur-Ordnung im Teilintervall?
- Lokale Fehlerabschätzung?

Fehlerschätzung

$\hat{\varepsilon}$ heißt Schätzer für ε, falls

$$\exists\, \kappa_1, \kappa_2 : \kappa_1 \leq 1 \leq \kappa_2 \text{ mit } \kappa_1 \varepsilon \leq \hat{\varepsilon} \leq \kappa_2 \varepsilon$$

Konstruktion von Schätzern ist im Allgemeinen schwierig.
Hier: Konstruktion durch Vergleich Approximationen niedriger und höherer Ordnung.

Fehlerschätzung

Annahme: Höhere Approximationsordnung liefert kleineren Approximationsfehler, d.h.

$$\varepsilon_{i,k+1} \ll \varepsilon_{ik}$$

Im Extrapolationstableau sollten demnach zeilenweise die Diagonalelemente am genausten sein.

Fehlerschätzung

Problem: ε_{kk} schätzen durch

$$\hat{\varepsilon}_{kk} := |T_{k+1,k} - T_{k,k}|$$

ergibt keinen Sinn, da die bessere Approximation $T_{k+1,k+1}$ zur Verfügung steht, für welche wir aber keine Fehlerschätzung haben.

Fehlerschätzer

Lösung: Verwende subdiagonales Element $T_{k,k-1}$ und schätze

$$\hat{\varepsilon}_{k,k-1} := |T_{k,k-1} - T_{kk}|$$

Herleitung Algorithmus

Im Folgenden sei $I_{\text{skal}} = 1$. Im Algorithmus verwenden wir das 'subdiagonale Fehlerkriterium' als Abbruchkriterium

$$\hat{\varepsilon}_{k,k-1} \leq \rho \cdot \text{tol mit 'Sicherheitsfaktor'} \ \rho < 1$$

Herleitung Algorithmus

Der Algorithmus soll die Schrittweite und Ordnung des Verfahren anpassen. Sei H die Grundschrittweite und $p = 2k$ (k maximale Spalte Extrapolationstableau), zunächst fest, die Ordnung.
Für

$$\varepsilon = \varepsilon(t, H) = \left| \hat{I}_t^{t+H}(f) - \int_t^{t+H} f(\tau) d\tau \right| \triangleq \gamma(t) H^{p+1}$$

können wir $\gamma(t)$ durch $\varepsilon H^{-(p+1)}$ abschätzen.

Herleitung Algorithmus

Sei $\tilde{H}$ die 'optimale' Schrittweite, d.h. für welche die Genaugikeit

$$\text{tol} = \varepsilon(t, \tilde{H}) \triangleq \gamma(t)\tilde{H}^{p+1}$$

erreicht wäre. Es würde dann gelten

$$\tilde{H} \triangleq \left(\frac{\text{tol}}{\varepsilon}\right)^{1/(p+1)} H$$

$\tilde{H} \ll H$ wäre dann ein Anzeichen dafür, dass H 'zu groß' war, woraufhin der (bzw. ggf. auch nächste) Integrationsschritt mit $\tilde{H}$ wiederholt werden könnte (bzw. begonnen wird)

Herleitung Algorithmus

Zur Variation der Ordnung: Aus der letzten Gleichung folgern wir

$$\tilde{H}_k := \left(\frac{\rho \cdot \text{tol}}{\hat{\varepsilon}_{k,k-1}} \right)^{1/(2k-1)} H$$

die optimale Schrittweite für die Spalte k.

Herleitung Algorithmus

Um aus $(k, \tilde{H}_k, A_k) = (\text{Spalte, Schrittweitenvorschlag, Aufwand})$ für $k = 1, ..., k_{\max} = q/2$ in jedem Einzelschritt das beste $(k_j, \tilde{H}_{k_j}, A_j)$ zu wählen, d.h. den Gesamtaufwand zu minimieren, unter der Voraussetzung, dass das Integrationsintervall konstante Länge hat. Wir verwenden wir hierfür einen Greedy-Algorithmus, der im aktuellen Schritt j den Aufwand pro Schrittweite

$$W_k := \frac{A_k}{\tilde{H}_k}$$

minimiert.

Herleitung Algorithmus

D.h. wir wählen die Spalte $\tilde{k} = k_j$ mit

$$W_{\tilde{k}} = \min_{k=1,\ldots,k_{\max}} W_k$$

und $\tilde{q} = 2\tilde{k}$.

Algorithmus

Algorithmus 9.30 (Ein Schritt der adaptiven Romberg-Quadratur). Als Eingabe erhält die Prozedur *step* den Beginn des aktuellen Intervalls t, die vorgeschlagene Spalte k und die Schrittweite H. Zurückgegeben werden neben der etwaigen Erfolgsmeldung *done* die entsprechenden Werte $\tilde{t}$, $\tilde{k}$ und $\tilde{H}$ für den nächsten Schritt sowie die Approximation I für das Integral $I_t^{\tilde{t}}$ über dem Intervall $[t, \tilde{t}]$.

function $[done, I, \tilde{t}, \tilde{k}, \tilde{H}]$=*step*$(t, k, H)$
  $done$:= **false**;
  $i = 1$;
  while not $done$ **and** $i < i_{\max}$ **do**

Algorithmus

Berechne die Approximationen $T_{11}, \ldots, T_{kk}$ von I_t^{t+H};
while $k < k_{\max}$ **and** $\bar{\varepsilon}_{k,k-1} > \text{tol}$ **do**
    $k := k + 1$;
    Berechne T_{kk};
end
Berechne $\tilde{H}_1, \ldots, \tilde{H}_k$ und $W_1, \ldots, W_k$;
Wähle $\tilde{k} \leq k$ mit minimalem Aufwand $W_{\tilde{k}}$;
$\tilde{H} := H_{\tilde{k}}$;
if $k < k_{\max}$ **then**
    if $H > \tilde{H}$ **then**
        $H := \tilde{H}$; (wiederhole den Schritt sicherheitshalber)
    else
        $\tilde{t} := t + H$;
        $I = T_{kk}$;
        done := **true**; (fertig)
    end
end

Probleme Algorithmus

1. Stehenbleiben der Ordnung
2. Zu große Schrittweite erst spät erkannt
3. Annahmen nicht notwendig wahr; ggf. 'Pseudokonvergenz'

2,3 könnten durch 'Konvergenz-Monitor' remediert werden

Konvergen-Monitor

Konvergenz-Monitor soll überprüfen, ob sich Approximationen 'vernünftig', im Sinne der theoretischen Annahmen, verhalten.
Hier: Informationstheoretisches Modell

Konvergenz-Monitor

Quadraturalgorithmus = Codiermaschine

- Eingabe = Anzahl f-Auswertungen zur Berechnung von T_{ik}
- Ausgabe = Anzahl gültiger Binärziffern der Approximation
- Annahme: Ein- und Ausgabe proportional zueinander

$\Rightarrow$ geschätzten Fehler $\varepsilon_{i,i-1}$ mit geschätzten Fehler $\alpha_{i,i-1}^{(k)}$ des Konvergenzmodells vergleichen

Konvergenz-Monitor

- Eingabe: $\text{Ein} = \alpha(A_i - A_{i-k} + 1)$, $\alpha > 0$
- Ausgabe: $\text{Aus} = \log_2(\varepsilon_{ik}^{-1})$
- Annahme: $\text{Ein} = \beta \cdot \text{Aus}$, $\beta \in (0, 1]$

$$\Rightarrow \log_2(\alpha_{ij}^k) = \frac{A_i - A_{i-j} + 1}{A_k - A_1 + 1} \log_2(\text{tol})$$

Nadelimpuls

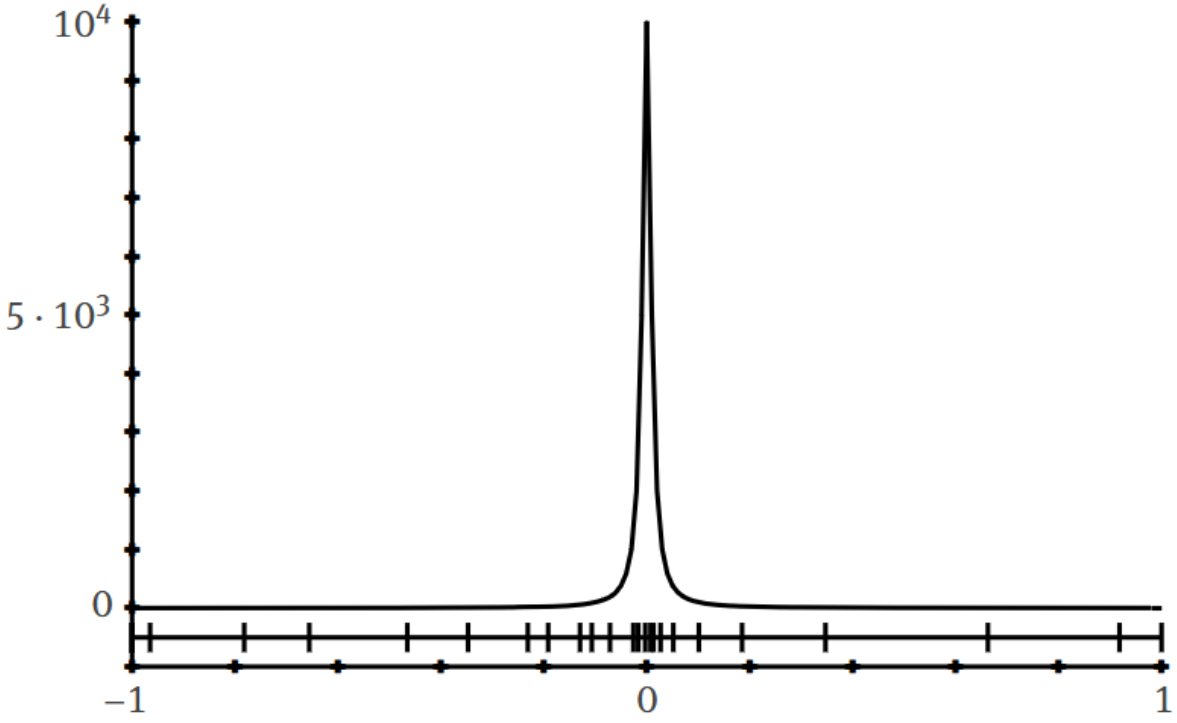


Abbildung: Nadelimpuls $\int_{-1}^{1} dt/(10^{-4} + t^2)$

Nadelimpuls

Die nicht-adaptive Romberg-Quadratur benötigte für einen relativen Fehler von ca. 10^{-9} insgesamt 4097 f-Auswertungen. Die adaptive Romberg-Quadratur benötigt für einen verlangten Fehler von 10^{-9} nur 321 f-Auswertungen bei einem genauen Fehler von $1.4 \cdot 10^{-9}$.

Schwierige Integranden

Viele Integranden verlangen jedoch zusätzlichen (manuellen Eingriff)

- unstetige Integranden
  - Integration zwischen Unstetigkeitstellen
- stark oszillierende Integranden
  - Integration zwischen Nullstellen/ Extremstellen
- schwach singuläre Integranden, d.h. $f \notin C^\infty[a,b]$
  - geeignete Substitution

Schwierige Integranden

- Parameterabhängige Integranden $I(\lambda) := \int_a^b f(t, \lambda) dt$
  - Quadratur-Parameterwerte speichern
  - Parameterabhängige Gitter verwenden
  - Festgelegte Gitter verwenden
- Diskrete Integranden
  - Trapezsumme verwenden, insbesonders falls Erwartungswert Messfehler $= 0$

Randwertmethode

Statt Integrationsintervall 'von links nach rechts' zu durchlaufen (Anfangswertmethode), verwende ausgehend von Ausgangsgitter $\Delta^0 = \{a, b\}$ Folge von Gittern $\Delta^1, \Delta^2 ...$ ($\Delta^0 \subseteq \Delta^1 \subseteq \Delta^2 \subseteq ...$) mit feineren Unterteilungen in Intervallen, wo es für die geforderte Genauigkeit notwendig ist. ('Mehrgitter-Verfahren')
Dabei benötigen wir

- lokale Fehlerschätzer
- lokale Verfeinerungsregeln

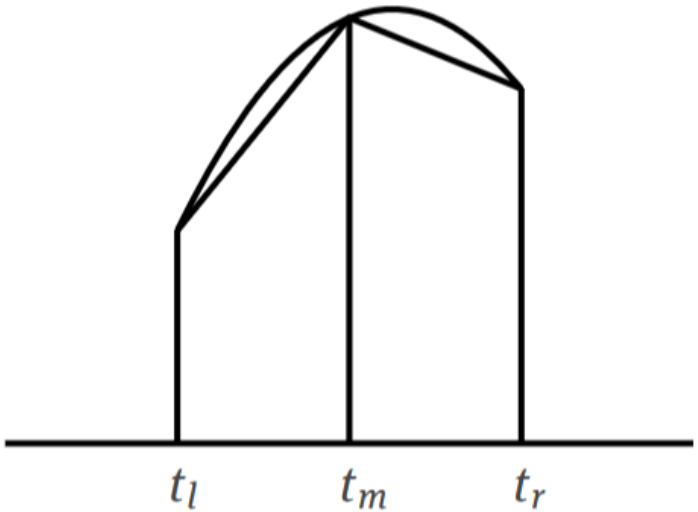
Fehlerschätzer

Fehler-Abschätzung wieder über Verfahren niedriger und höherer Ordnung: Trapez-Regel T und Simpson-Regel S.
Unter der Annahme dass Simpson-Regel lokal genauer als Trapez-Regel ist, schätzen wir

$$\hat{\varepsilon}(J) := |T(J) - S(J)|$$

wobei $J := (t_l, t_m, t_r)$, $[t_l, t_r] \subseteq [a, b]$, $t_m = (t_l + t_r)/2$

Fehlerschätzer



$$T(J) = \frac{h}{4}\left(f(t_l) + 2f(t_m) + f(t_r)\right)$$

$$S(J) = \frac{h}{6}\left(f(t_l) + 4f(t_m) + f(t_r)\right)$$

Abbildung: Trapez- und Simpson-Regel für ein J

Verfeinerungsregel

Verfeinerungsregel = Lokale Bisektion

$$J_l := \left(t_l, \frac{t_l + t_m}{2}, t_m \right)$$
$$J_r := \left(t_m, \frac{t_r + t_m}{2}, t_r \right)$$

Ist J durch Bisektion entstaden, bezeichnet J^- das Ausgangsintervall der letzten Stufe.

Verfeinerungsregel

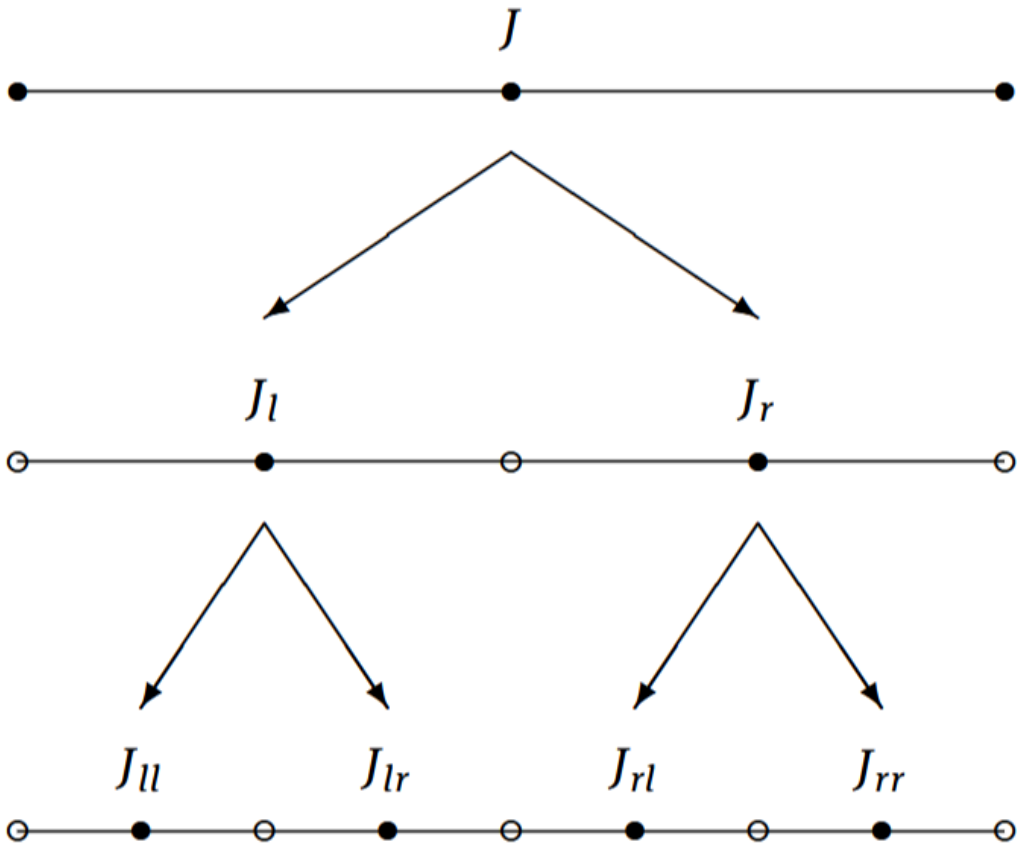


Abbildung: zweimalige Verfeinerung

Verfeinerungsregel

Wir versuchen hier so zu verfeinern, dass der lokale Diskretisierungsfehler gleichverteilt wird, d.h. dass Gitter Δ sollte so verfeinert werden, dass für das verfeinerte Gitter Δ^+ gilt

$$\hat{\varepsilon}(J) = \text{const für alle } J \in \Delta^+$$

Verfeinerungsregel

Über lokale Extrapolation kann vorausschauend ein Schätzer

$$\hat{\varepsilon}(J_l) = \varepsilon^+(J) := \frac{\hat{\varepsilon}(J)^2}{\hat{\varepsilon}(J^-)}$$

gewonnen werden. Übertritt dieser einen Schwellwert, so wird das Intervall unterteilt. Dazu verwenden wir den maximalen lokalen Fehler bei globaler Verfeinerung

$$\kappa(\Delta) := \max_{J \in \Delta} \varepsilon^+(J)$$

Verfeinerungsregel

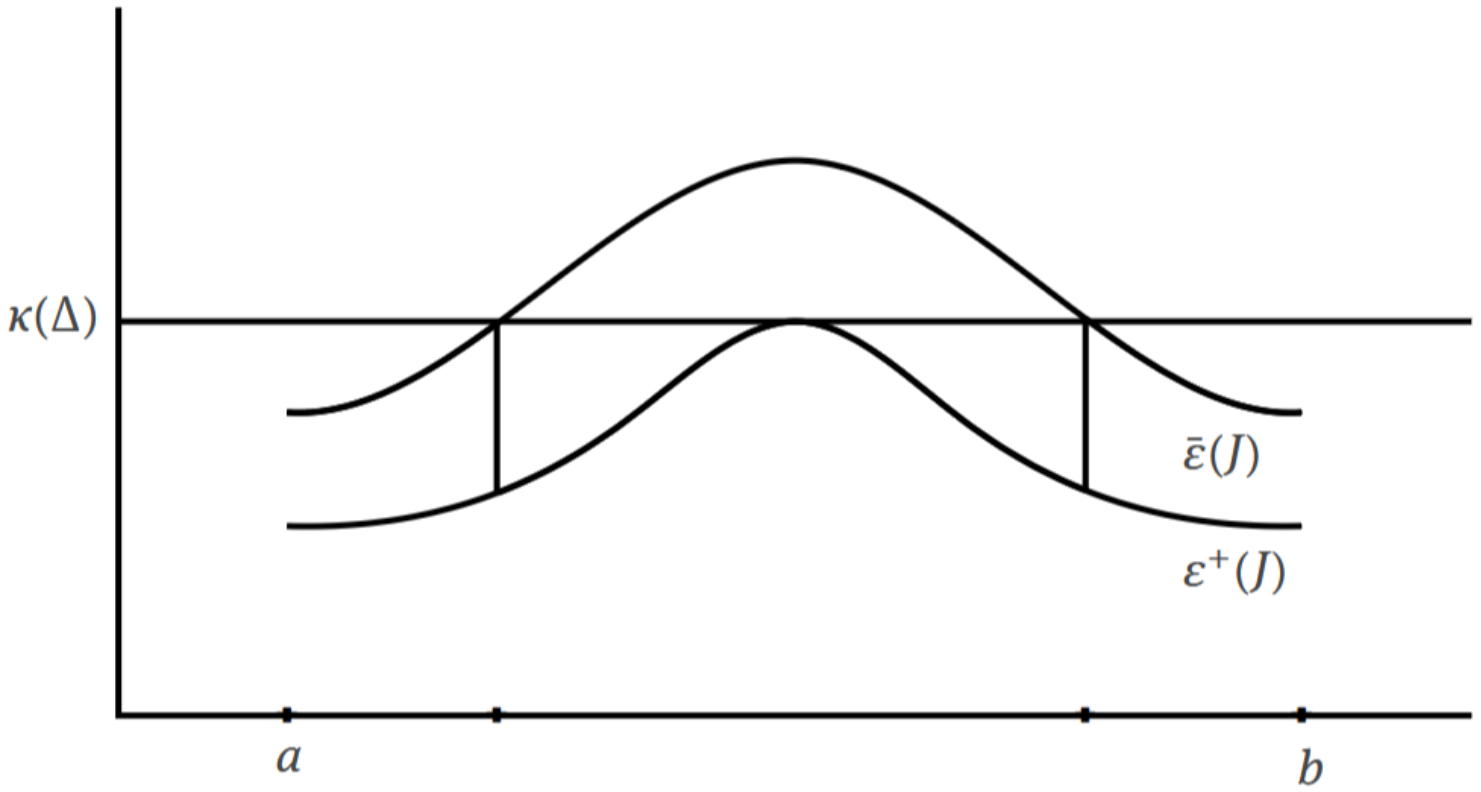


Abbildung: Geschätzte Fehlerverteilung vor und nach globaler und lokaler Verfeinerung

Verfeinerungsregel

Verfeinere also $J \in \Delta$ mit $\hat{\varepsilon}(J) \geq \kappa(\Delta)$

Fehlerschätzung

Die Schätzung des globalen Approximationsfehler

$$\varepsilon(\Delta) := \left| \int_a^b f(t)dt - S(\Delta) \right|$$

über die Summe der lokalen Fehlerschätzungen ist ungeeignet, da sich diese ausmitteln können. Stattdessen wird überprüft

$$\varepsilon(\Delta) \ll \varepsilon(\Delta^-)$$

wonach

$$\hat{\varepsilon}(\Delta) := |S(\Delta^-) - S(\Delta)|$$

Fehlerschätzung

Um $\varepsilon(\Delta) \ll \varepsilon(\Delta^-)$ garantieren zu können, sollten stufenweise hinreichend viele Intervalle verfeinert werden. Dafür wird $\kappa(\Delta)$ ersetzt durch

$$\hat{\kappa} := \min\left(\max_{J\in\Delta} \varepsilon^+(J), \frac{1}{2}\max_{J\in\Delta} \hat{\varepsilon}(J)\right)$$

Algorithmus

Algorithmus 9.36 (Einfache Mehrgitter-Quadratur).

Wähle ein Startgitter, z. B. $\Delta := \{(a, (a+b)/2, b)\}$;
for $i = 0$ **to** $i_{\max}$ **do**
  Berechne $T(J)$, $S(J)$ und $\bar{\varepsilon}(\Delta)$ für alle $J \in \Delta$;
  Berechne $\bar{\varepsilon}(\Delta)$;
  if $\bar{\varepsilon}(\Delta) \leq \text{tol } |S(J)|$ **then**
    break; (fertig, Lösung $S(\Delta)$)
  else
    Berechne $\varepsilon^+(J)$ und $\bar{\varepsilon}(J)$ für alle $J \in \Delta$;
    Berechne $\tilde{\kappa}(\Delta)$;
    Ersetze alle $J \in \Delta$ mit $\bar{\varepsilon}(J) \geq \tilde{\kappa}(\Delta)$ durch J_l und J_r;
  end
end

Abbildung: Algorithmus Mehrgitter-Quadratur

Stärken und Schwächen

- Stärken: auch relativ gut für z.B. unstetige und schwach singuläre Integranden geeignet
- Schwächen: z.B. Parameterabhängige Integranden

Nadelimpuls

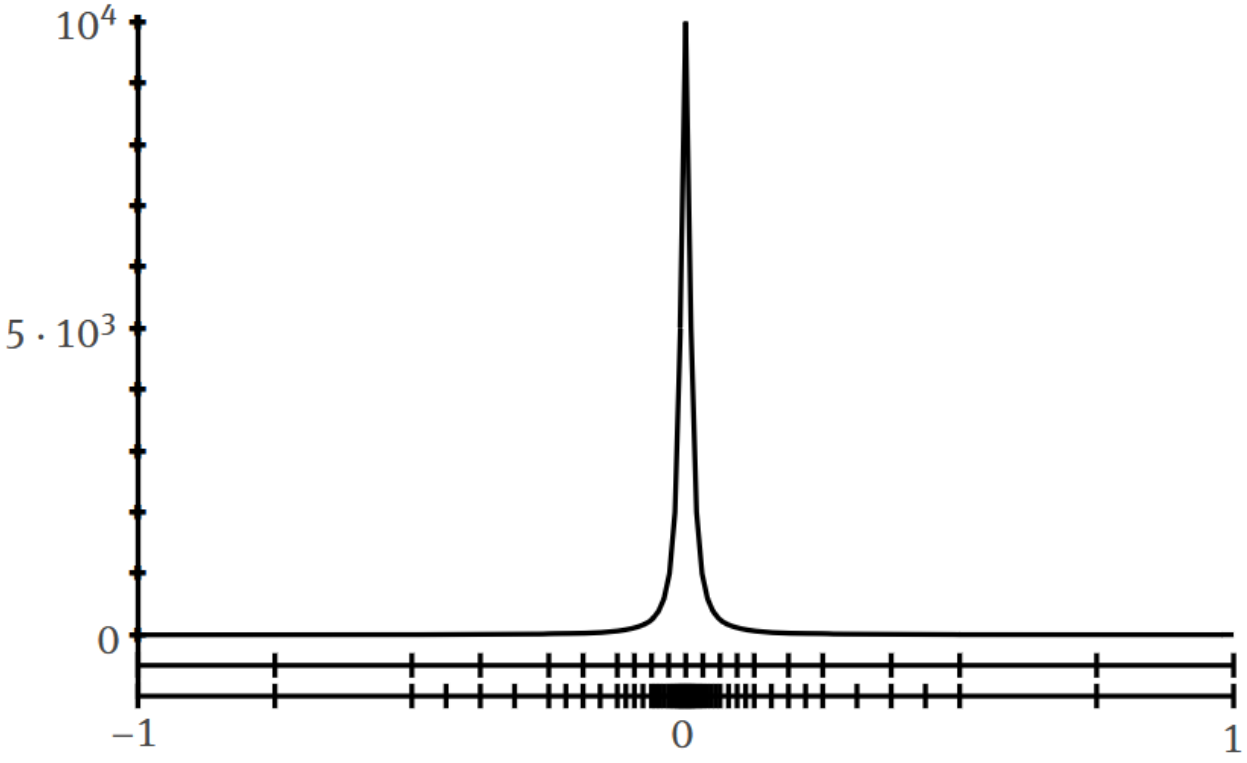


Abbildung: Gitter für Mehrgitter-Quadratur für tol$= 10^{-3}$ (Nadelimpuls)

Nadelimpuls

Für die Toleranz 10^{-3} benötigt der Algorithmus 121 f-Auswertungen und schätzt $\hat{\varepsilon} = 2.4 \cdot 10^{-4}$ bei einem tatsächlichen Fehler von $2.1 \cdot 10^{-4}$.

Beispiel 1

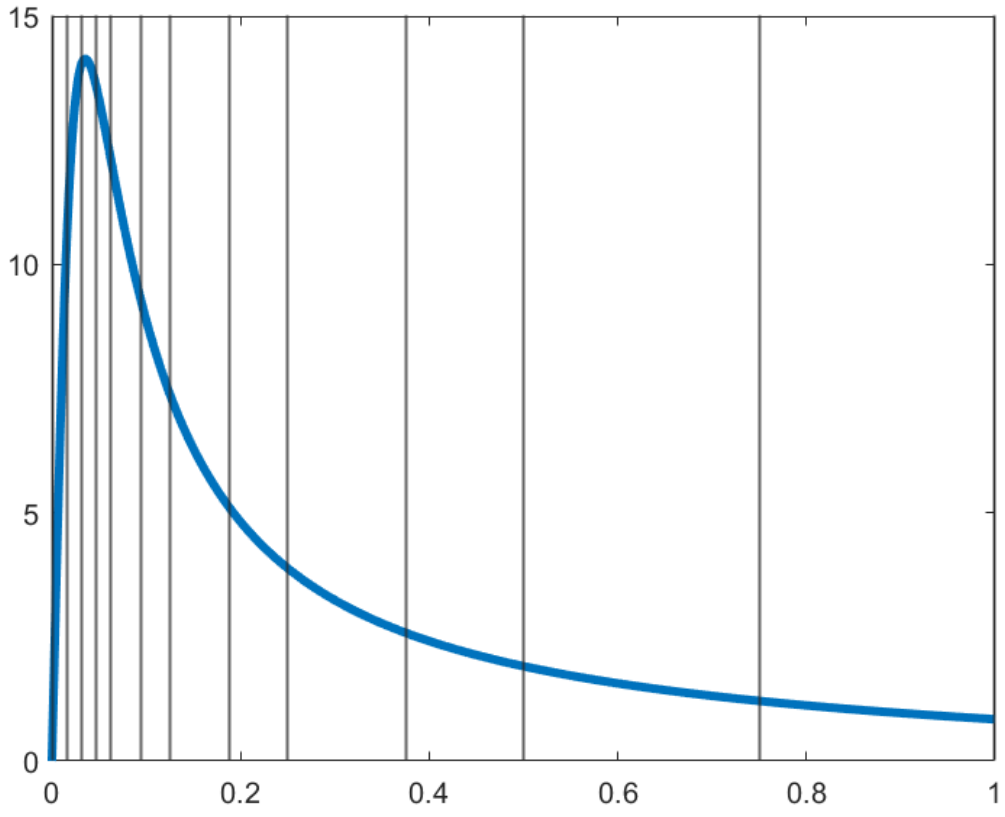


Abbildung: Funktion $\sin(x)/(0.00125 + x^2)$ auf $[0, 1]$

Beispiel 1

Klassiche Romberg-Quadratur benötigt insgesamt 33 Funktionsauswertungen für einen (relativen) Fehler von $4.283226e-03$.
Adaptive Romberg-Quadratur benötigt 27 Funktionsauswertungen für einen Fehler von $1.427033e-03$, bei einem geschätzten Fehler von $5.768818e-04$.

Beispiel 2

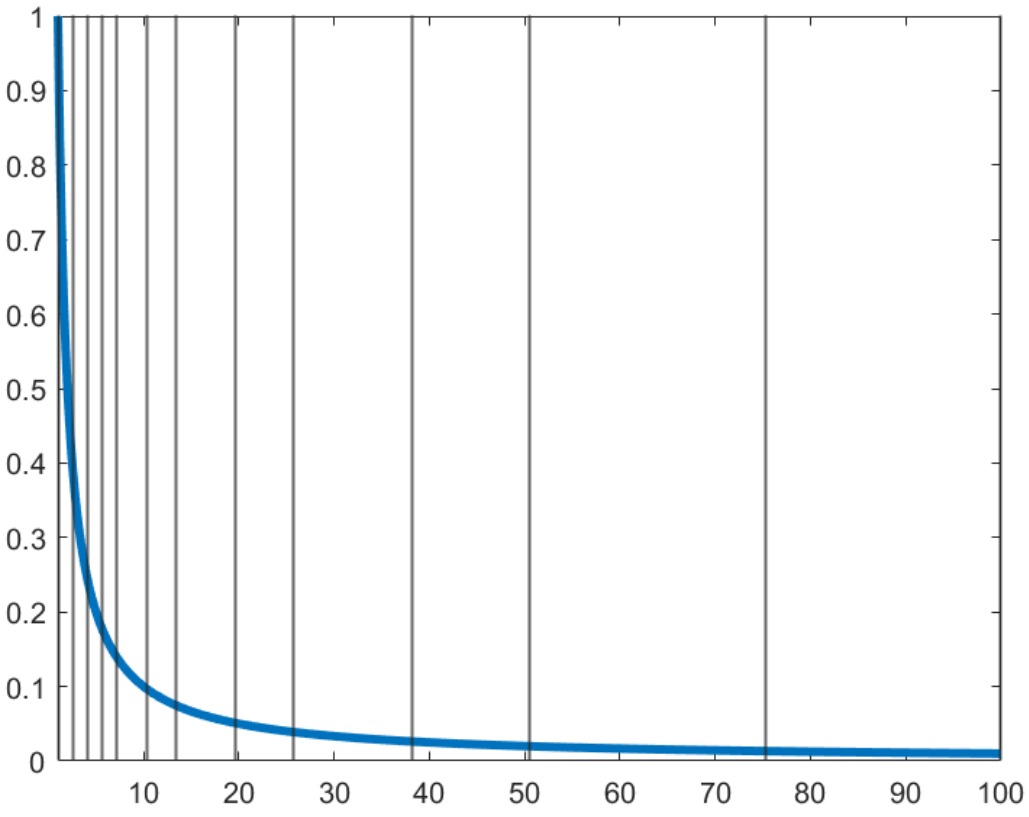


Abbildung: Funktion $1/x$ auf $[0, 100]$

Beispiel 2

Klassiche Romberg-Quadratur benötigt insgesamt 257 Funktionsauswertungen für einen (relativen) Fehler von $3.274580e-05$.
Adaptive Romberg-Quadratur benötigt nur 155 Funktionsauswertungen für einen Fehler von $2.698730e-06$, bei einem geschätzten Fehler von $2.236911e-05$.

Beispiel 2

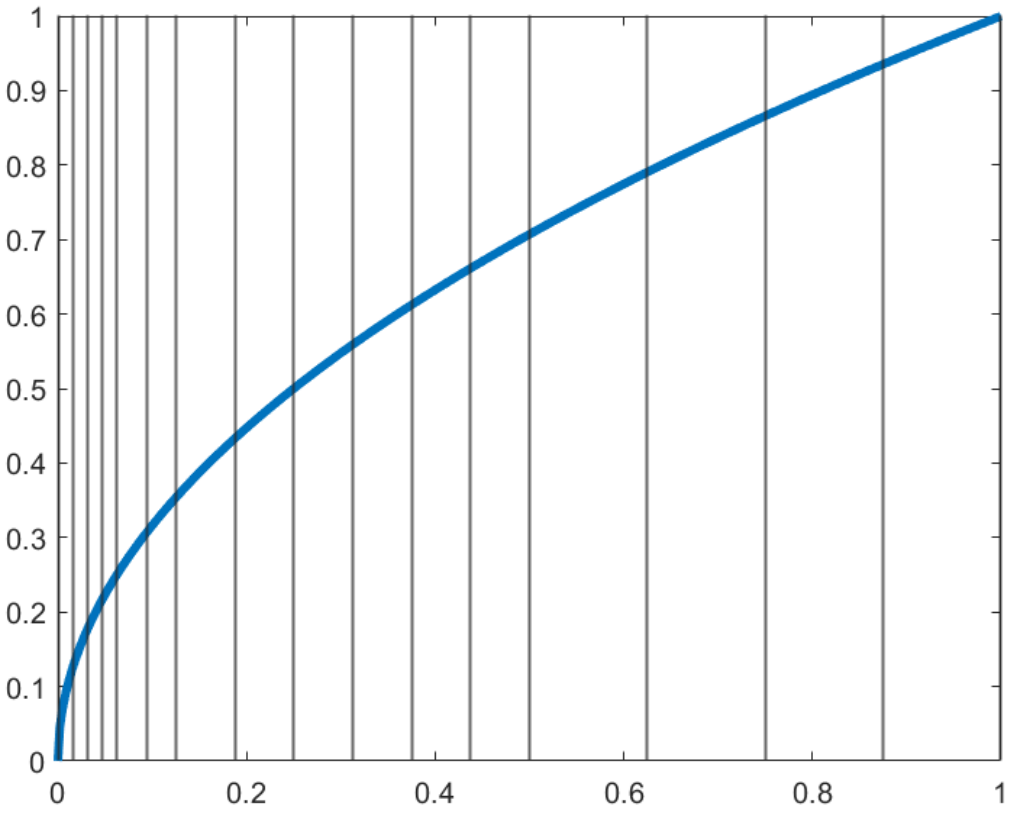


Abbildung: Funktion $\sqrt{x}$ auf $[0, 1]$

Beispiel 2

Klassiche Romberg-Quadratur benötigt insgesamt 4097 Funktionsauswertungen für einen (relativen) Fehler von $3.923013e-07$.
Adaptive Romberg-Quadratur benötigt nur 197 Funktionsauswertungen für einen Fehler von $1.752414e-07$, bei einem geschätzten Fehler von $9.572554e-09$.

Literatur I

[1] Peter Deuflhard und Andreas Hohmann. *Numerische Mathematik 1: Eine algorithmisch orientierte Einführung*. 5. Aufl. De Gruyter, 2019. ISBN: 978-3-11-0614329.