Technische Universität Ilmenau
Institut für Mathematik

Preprint No. M 06/01

Adaptive Verfahren zur numerischen Quadratur und Kubatur

Werner Vogt

Januar 2006

Impressum:
Hrsg.: Leiter des Instituts für Mathematik
Weimarer Straße 25
98693 Ilmenau
Tel.: +49 3677 69 3621
Fax: +49 3677 69 3270
http://www.tu-ilmenau.de/ifm/

ISSN xxxx-xxxx

ilmedia

Adaptive Verfahren zur numerischen Quadratur und Kubatur

Werner Vogt
Technische Universität Ilmenau
Institut für Mathematik
Postfach 100565
98684 Ilmenau

Ilmenau, den 27. Januar 2006

Zusammenfassung

Der Beitrag stellt wesentliche Verfahren zur numerischen Bestimmung des d-dimensionalen Riemannschen Integrals (numerisches Integrationsproblem) vor. Grundlegend sind dabei die Quadraturformeln von Newton-Cotes und Gauß-Legendre im 1-dimensionalen Fall. Nach deren Herleitung und Analyse werden darauf basierend adaptive Quadraturverfahren entwickelt, mit denen auch schwierige Integranden behandelt werden können. Um auch Flächen- und Volumenintegrale bestimmen zu können, konstruieren und untersuchen wir numerische Kubaturformeln auf Rechteck- und Dreieckbereichen. Analog zum Quadraturfall werden die Eigenschaften entsprechender adaptiver Kubaturverfahren an schwierigen 2-dimensionalen Beispielen getestet.

MSC 2000: 65D30, 65D32, 65-01

Keywords: Numerical integration, quadrature and cubature formulas, multiple integrals

1 Numerische Quadratur

Im Unterschied zur Differenziation bildet die Integration einer reellen Funktion $f : \mathbb{R} \to \mathbb{R}$ eine schwierige, mit analytischen Mitteln häufig unlösbare Aufgabe, so dass numerische Verfahren von besonderer Bedeutung sind. Betrachtet wird als Standardaufgabe die *Quadraturaufgabe*, d.h. die Bestimmung des Riemannschen Integrals

$$I \;=\; \int_a^b f(t)dt\,, \qquad -\infty < a < b < \infty \tag{1}$$

einer gegebenen integrierbaren Funktion $f(t)$ auf einem endlichen Intervall $[a, b]$. Zur Behandlung von unbestimmten Integralen und uneigentlichen Integralen sei auf die angegebene Literatur, z.B. [9] verwiesen.

1.1 Newton-Cotes- und Gauß-Legendre-Formeln

Zerlegt man das Integrationsintervall in N Teilintervalle der Breite $h = (b-a)/N$ mit $h \ll 1$, so setzt sich der Gesamtwert (1) additiv aus den zugehörigen Teilintegralen zusammen:

$$I \;=\; \int_a^b f(t)\,dt \;=\; \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} f(t)\,dt \tag{2}$$

(vgl. Abb. 1). Um Näherungsformeln für ein beliebiges derartiges Teilintegral herzuleiten, empfiehlt es sich, den Koordinatenursprung in dessen Intervallmitte zu verschieben und das Standardintervall (den hinreichend schmalen „Streifen“ der Breite h) $[-\frac{h}{2}, \frac{h}{2}]$ zu betrachten. Die einfachsten Näherungen des exakten Integralwertes in Abb. 2

$$I \;=\; \int_{-\frac{h}{2}}^{\frac{h}{2}} f(t)\;dt \tag{3}$$

bekommen wir, indem wir den Integranden f auf $[-h/2, h/2]$ durch eine Gerade ersetzen. So ergeben sich folgende einfache Näherungsformeln (vgl. Abbildungen 3 bis 6):

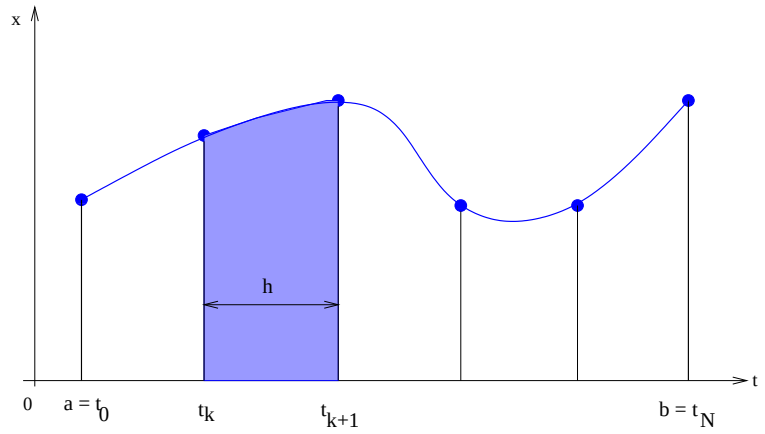


Abbildung 1: Zerlegung in N Teilintervalle

$$
\begin{array}{lllll}
(1) & \text{Rechteckregel/links} : & R_L & = & h \cdot f(-\frac{h}{2}) \\
(2) & \text{Rechteckregel/rechts:} & R_R & = & h \cdot f(\frac{h}{2}) \\
(3) & \text{Rechteckregel/Mitte:} & R_M & = & h \cdot f(0) \\
(4) & \text{Trapezregel} : & T & = & h \cdot [\frac{1}{2} f(-\frac{h}{2}) + \frac{1}{2} f(\frac{h}{2})] \, .
\end{array}
$$

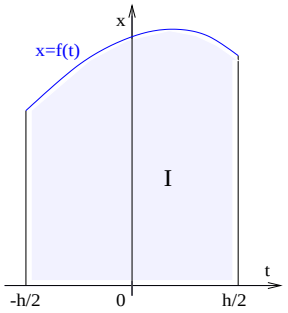


Abbildung 2: $\boldsymbol{I}$

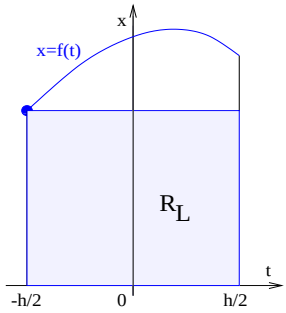


Abbildung 3: $\boldsymbol{R_L}$

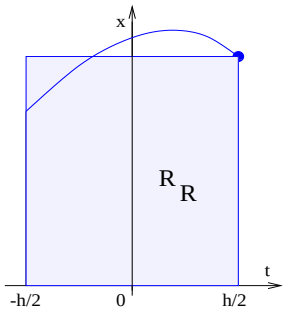


Abbildung 4: $\boldsymbol{R_R}$

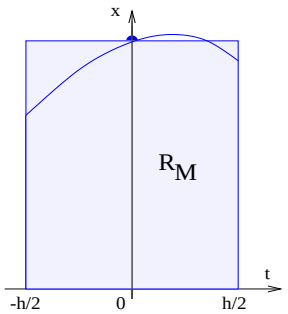


Abbildung 5: $\boldsymbol{R_M}$

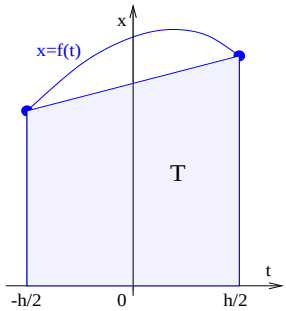


Abbildung 6: $\boldsymbol{T}$

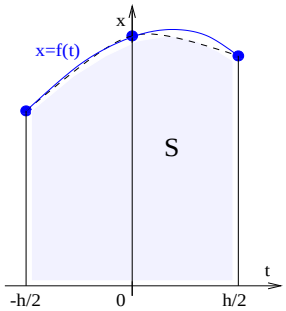


Abbildung 7: $\boldsymbol{S}$

Interpolationsquadraturen Um eine genauere Näherungsformel zu ermitteln, kann man das Intervall $[-h/2, h/2]$ in n Teilintervalle mit den äquidistanten Knoten $t_0, t_1, ..., t_n$ zerlegen, wobei

$$t_k = -\frac{h}{2} + k\frac{h}{n} \, , \quad f_k = f(t_k) \, , \quad k = 0(1)n \tag{4}$$

gilt (Im Unterschied zu diesen „abgeschlossenen“ Formeln mit Nutzung der Endpunkte liegen alle Knoten der „offenen“ Formeln im Intervallinnern).
Bestimmt man das zugehörige Interpolationspolynom $p_n(t)$ in Lagrangescher Form (vgl. [4], Kapitel 22)

$$p_n(t) \;=\; \sum_{k=0}^{n} f(t_k)\, L_k(t) \qquad \text{mit} \qquad L_k(t) \;=\; \prod_{\substack{j=0\\ j\neq k}}^{n} \frac{t-t_j}{t_k-t_j}$$

und integriert es über $[-\frac{h}{2}, \frac{h}{2}]$, so ergibt sich für das Integral (3) der Näherungswert

$$I_n \;=\; \int_{-\frac{h}{2}}^{\frac{h}{2}} \sum_{k=0}^{n} f(t_k) L_k(t)\, dt \;=\; \sum_{k=0}^{n} f(t_k) \int_{-\frac{h}{2}}^{\frac{h}{2}} \prod_{\substack{j=0\\ j\neq k}}^{n} \frac{t-t_j}{t_k-t_j}\, dt\,,$$

der sich mittels der Variablensubstitution $t = -\frac{h}{2} + sh$ in die leichter auswertbare Form

$$I_n \;=\; h \sum_{k=0}^{n} w_k f(t_k) \qquad \text{mit} \qquad w_k \;=\; \int_0^1 \prod_{\substack{j=0\\ j\neq k}}^{n} \frac{n\cdot s - j}{k-j}\, ds \tag{5}$$

überführen lässt. Derartige gewichtete Summen von Werten des Integranden an ausgewählten Stellen bezeichnet man als *Integrationsformeln*.

Definition 1 (Einfache Integrationsformel) $I_n = h\sum_{k=0}^{n} w_k f(t_k)$ *ist eine (einfache) Integrationsformel mit* $n+1$ *Integrationsknoten* $t_k \in [-h/2, h/2]$ *und Integrationsgewichten* w_k. *Sind die Knoten durch (4) festgelegt, so ist die Formel vom (abgeschlossenen) Newton-Cotes-Typ.*

Beispiel 2 1. Mit $n = 1$ liefert (5) zu den Knoten $t_0 = -h/2,\ t_1 = h/2$ die Gewichte $w_0 = w_1 = 1/2$, womit die *Trapezregel* folgt

$$I_1 = T = \frac{h}{2}(f_0 + f_1)\,. \tag{6}$$

2. Für $n = 2$ werden die Knoten $t_0 = -h/2,\ t_1 = 0,\ t_2 = h/2$ benutzt und ergeben die Gewichte $w_0 = w_2 = 1/6,\ w_1 = 4/6$ der *Simpson-Regel (Keplersche Regel*[1]*)*

$$I_2 = S = \frac{h}{6}(f_0 + 4f_1 + f_2) \qquad \text{(vgl. Abb 7)}\,. \tag{7}$$

3. Für $n = 3$ erhält man die so genannte *Newtonsche 3/8-Regel*

$$I_3 = N = \frac{h}{8}(f_0 + 3f_1 + 3f_2 + f_3)\,. \tag{8}$$

4. Der Wert $n = 4$ liefert die Fünfpunktregel von *Milne*

$$I_4 = M = \frac{h}{90}\cdot(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4) \tag{9}$$

als abgeschlossene Newton-Cotes-Formeln. In MAPLE lässt sich Formel (5) leicht durch

[1] Johannes Kepler (1571-1630), deutscher Mathematiker und Astronom, Entdeckung der nach ihm benannten 3 Bewegungsgesetze. 1615 entwickelte er die Fassregel $V = \pi h(r_a^2 + 4r_m^2 + r_b^2)/6$ für das Volumen von Fässern.

```
w := proc(n,k)
>   local u,v;
>   u := product((n*s-j)/(k-j), j=0.. k-1);
>   v := product((n*s-j)/(k-j), j=k+1 .. n);
>   int(u*v,s =0..1);
> end;
```

programmieren. Mit der MAPLE-Anweisung

```
for n to 10 do
>    for k from 0 to n do
>         printf("    n = %a, k = %a :   w(%a,%a) = %a\n",n,k,n,k,w(n,k));
>    od:    # k-Zyklus
>   printf(" \n");
> od:    # n-Zyklus
```

erhalten wir sofort die Gewichte für $n = 1, 2, \ldots, 10$:

```
    n = 1, k = 0 :   w(1,0) = 1/2
    n = 1, k = 1 :   w(1,1) = 1/2

    n = 2, k = 0 :   w(2,0) = 1/6
    n = 2, k = 1 :   w(2,1) = 2/3
    n = 2, k = 2 :   w(2,2) = 1/6

    n = 3, k = 0 :   w(3,0) = 1/8
    n = 3, k = 1 :   w(3,1) = 3/8
    n = 3, k = 2 :   w(3,2) = 3/8
    n = 3, k = 3 :   w(3,3) = 1/8

    n = 4, k = 0 :   w(4,0) = 7/90
    n = 4, k = 1 :   w(4,1) = 16/45
    n = 4, k = 2 :   w(4,2) = 2/15
    n = 4, k = 3 :   w(4,3) = 16/45
    n = 4, k = 4 :   w(4,4) = 7/90

    n = 5, k = 0 :   w(5,0) = 19/288
    n = 5, k = 1 :   w(5,1) = 25/96
    n = 5, k = 2 :   w(5,2) = 25/144
    n = 5, k = 3 :   w(5,3) = 25/144
    n = 5, k = 4 :   w(5,4) = 25/96
    n = 5, k = 5 :   w(5,5) = 19/288

    n = 6, k = 0 :   w(6,0) = 41/840
    n = 6, k = 1 :   w(6,1) = 9/35
    n = 6, k = 2 :   w(6,2) = 9/280
    n = 6, k = 3 :   w(6,3) = 34/105
    n = 6, k = 4 :   w(6,4) = 9/280
    n = 6, k = 5 :   w(6,5) = 9/35
```

```
n = 6, k = 6 :   w(6,6) = 41/840

n = 7, k = 0 :   w(7,0) = 751/17280
n = 7, k = 1 :   w(7,1) = 3577/17280
n = 7, k = 2 :   w(7,2) = 49/640
n = 7, k = 3 :   w(7,3) = 2989/17280
n = 7, k = 4 :   w(7,4) = 2989/17280
n = 7, k = 5 :   w(7,5) = 49/640
n = 7, k = 6 :   w(7,6) = 3577/17280
n = 7, k = 7 :   w(7,7) = 751/17280

n = 8, k = 0 :   w(8,0) = 989/28350
n = 8, k = 1 :   w(8,1) = 2944/14175
n = 8, k = 2 :   w(8,2) = -464/14175
n = 8, k = 3 :   w(8,3) = 5248/14175
n = 8, k = 4 :   w(8,4) = -454/2835
n = 8, k = 5 :   w(8,5) = 5248/14175
n = 8, k = 6 :   w(8,6) = -464/14175
n = 8, k = 7 :   w(8,7) = 2944/14175
n = 8, k = 8 :   w(8,8) = 989/28350

n = 9, k = 0 :   w(9,0) = 2857/89600
n = 9, k = 1 :   w(9,1) = 15741/89600
n = 9, k = 2 :   w(9,2) = 27/2240
n = 9, k = 3 :   w(9,3) = 1209/5600
n = 9, k = 4 :   w(9,4) = 2889/44800
n = 9, k = 5 :   w(9,5) = 2889/44800
n = 9, k = 6 :   w(9,6) = 1209/5600
n = 9, k = 7 :   w(9,7) = 27/2240
n = 9, k = 8 :   w(9,8) = 15741/89600
n = 9, k = 9 :   w(9,9) = 2857/89600

n = 10, k = 0 :   w(10,0) = 16067/598752
n = 10, k = 1 :   w(10,1) = 26575/149688
n = 10, k = 2 :   w(10,2) = -16175/199584
n = 10, k = 3 :   w(10,3) = 5675/12474
n = 10, k = 4 :   w(10,4) = -4825/11088
n = 10, k = 5 :   w(10,5) = 17807/24948
n = 10, k = 6 :   w(10,6) = -4825/11088
n = 10, k = 7 :   w(10,7) = 5675/12474
n = 10, k = 8 :   w(10,8) = -16175/199584
n = 10, k = 9 :   w(10,9) = 26575/149688
n = 10, k = 10:   w(10,10) = 16067/598752
```

Integrationsfehler Die Differenz zwischen dem exakten Integralwert I in (3) und dem Näherungswert I_n ergibt den *Integrationsfehler* (auch Quadraturfehler genannt)

$$F_n \;=\; I - I_n \,, \tag{10}$$

der eine Genauigkeitsaussage über I_n gestattet. Wir wollen eine Darstellung dieses Fehlers gewinnen, die für beliebige Lage der Knoten t_k gültig ist, um damit gegebenenfalls eine optimale Auswahl der Knoten zu ermöglichen. Zur Vereinfachung der Betrachtung setzen wir voraus, dass $f(t)$ im Intervall $[-h/2, h/2]$ in eine Taylorreihe

$$f(t) = \sum_{\nu=0}^{\infty} a_\nu t^\nu \quad \text{mit} \quad a_\nu = \frac{f^{(\nu)}(0)}{\nu!}$$

entwickelt werden kann. Einsetzen der Taylorreihe in I und I_n liefert für den Integrationsfehler

$$\begin{aligned} F_n &= \int_{-\frac{h}{2}}^{\frac{h}{2}} f(t)\,dt - h\sum_{k=0}^{n} w_k f(t_k) \\ &= \int_{-\frac{h}{2}}^{\frac{h}{2}} \sum_{\nu=0}^{\infty} a_\nu t^\nu dt - h\sum_{k=0}^{n} w_k \sum_{\nu=0}^{\infty} a_\nu t_k^\nu \\ &= h\sum_{\nu=0}^{\infty}\left\{\frac{1}{\nu+1}\left(\frac{h}{2}\right)^\nu \frac{1+(-1)^\nu}{2} - \sum_{k=0}^{n} w_k t_k^\nu\right\} a_\nu\,. \end{aligned} \tag{11}$$

Das Ziel, möglichst viele der Summanden von F_n für $\nu = 0, 1, 2, \ldots, m$ zu Null zu machen, führt auf die Bestimmungsgleichungen für t_k und w_k :

$$\boxed{\sum_{k=0}^{n} w_k\, t_k^\nu \;=\; \frac{1}{\nu+1}\left(\frac{h}{2}\right)^\nu \frac{1+(-1)^\nu}{2}\,, \quad \nu = 0(1)m\,.} \tag{12}$$

Während bei den (*abgeschlossenen*) *Newton-Cotes-Formeln* die Knoten t_k und bei den *Tschebyscheff-Formeln* die (identischen) Gewichte $w_k = w$ vorgegeben sind, sind bei den *Gauß-Legendre-Formeln* alle $2n + 2$ in (12) auftretenden Variablen frei bestimmbar. Damit sind auch die wesentlichsten 3 Formelgruppen charakterisiert (vgl. Tabelle 1).

Nr.	Formelgruppe	Knoten und Gewichte	m
1	Newton-Cotes-Formeln	t_k äquidistant gegeben, w_k gesucht	n
2	Tschebyscheff-Formeln	$w_k = w = const$ gegeben, t_k gesucht	n
3	Gauß-Legendre-Formeln	t_k und w_k gesucht	$2n+1$

Tabelle 1: Integrationsfehler einfacher Newton-Cotes-Formeln

Bemerkung 3 Man zeigt leicht, dass die nach dem Interpolationsansatz durch (5) bestimmten Gewichte w_k der abgeschlossenen Newton-Cotes-Formeln identisch mit den Lösungen des linearen Systems (12) sind.

Der erste nicht-verschwindende Summand der Entwicklung (11) mit einem Index $M > m$ ergibt im Allgemeinen eine gute Näherung des Integrationsfehlers F_n. Durch genauere Beachtung der Restglieder der Taylorentwicklungen erhalten wir die Darstellung

$$F_n \;=\; h\left\{\frac{1}{M+1}\left(\frac{h}{2}\right)^M \frac{1+(-1)^M}{2} \;-\; \sum_{k=0}^{n} w_k t_k^M\right\}\frac{f^{(M)}(\xi)}{M!} \quad \text{mit} \quad -\frac{h}{2} \le \xi \le \frac{h}{2}\,. \tag{13}$$

Für die bereits genannten 5 Newton-Cotes-Formeln sind die Fehlerentwicklungen in Tabelle 2 dargestellt, wobei hinreichende Glattheit von f vorausgesetzt wird. Die letzte Spalte der

n	Formel	$F_n = I - I_n$	p
0	Rechteckregel/links	$\frac{h^2}{2}\frac{f'(0)}{1!} + \mathcal{O}(h^4)$	2
1	Trapezregel	$-\frac{h^3}{6}\frac{f''(0)}{2!} + \mathcal{O}(h^5)$	3
2	Simpson-Regel	$-\frac{h^5}{120}\frac{f^{(4)}(0)}{4!} + \mathcal{O}(h^7)$	5
3	Newton's 3/8-Regel	$-\frac{h^5}{270}\frac{f^{(4)}(0)}{4!} + \mathcal{O}(h^7)$	5
4	Milne's Fünfpunktregel	$-\frac{h^7}{2688}\frac{f^{(6)}(0)}{6!} + \mathcal{O}(h^9)$	7

Tabelle 2: Integrationsfehler einfacher Newton-Cotes-Formeln

Tabelle gibt die Ordnung p der jeweiligen Formel an, die durch den Exponenten von h^p bestimmt ist. Wegen $h \ll 1$ sind im Allgemeinen Formeln hoher Ordnung am genauesten, wobei offenbar Formeln mit gerader Zahl n von Teilintervallen (Simpson-Regel, Milne-Regel) denen mit ungerader Zahl n überlegen sind.

Gauß-Legendre-Formeln Offene Gauß-Legendre-Formeln werden oft verwendet, wenn bei aufwändigen Funktionsberechnungen mit möglichst wenigen Knoten eine hohe Genauigkeit erreicht werden soll. Denn zu gegebener Zahl n von Teilintervallen lassen sich genauere Formeln konstruieren, wenn die Forderung nach Äquidistanz der Knoten t_k entfällt. Bei geeigneter Wahl der Knoten $t_0, t_1, ..., t_n$ besitzen die Integrationsformeln

$$G_n \;=\; h\sum_{k=0}^{n} w_k f(t_k) \tag{14}$$

dann einen Fehler maximaler Ordnung $p = 2n+3$ in h. Dazu benötigen wir die so genannten *Legendre-Polynome*, die für beliebige natürliche Zahl $n \ge 0$ durch

$$L_n(x) \;:=\; \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n \tag{15}$$

definiert sind. $L_n(x)$ besitzt in $[-1,1]$ genau n symmetrisch zu Null liegende reelle Nullstellen $x_0, x_1, \ldots, x_{n-1}$. Mit MAPLE können diese zu $n = 2, 3, 4, \ldots, 8$ leicht bestimmt werden. Anschließender Aufbau der Bestimmungsgleichungen (12) und deren Lösung mit den MAPLE-Funktionen und Anweisungen

```
> Digits := 25:
> nullstellen := proc(n)         # Nullstellen des Legendre-Polynoms
>   local leg,x;
>   leg := expand(LegendreP(n,x));
>   fsolve(leg,x);
> end:

> f := k -> 1/(k)*(h/2)^(k-1)*(1+(-1)^(k-1))/2:      # Hilfsfunktion
> g := (i,k) -> (h/2*x[k])^(i-1):                    # Hilfsfunktion
> for n from 2 to 8 do
>   x := nullstellen(n);                      # Bestimmung der Knoten
>   b := vector(n, f);  A := matrix(n,n,g);
>   w := linsolve(A,b);                    # Bestimmung der Gewichte
>   for k  to n do
>     printf(" t(%a,%a) = %a     w(%a,%a) = %a\n",n,k-1,x[k],n,k-1,w[k]);
>   od:
>   printf(" \n");
> od:
```

liefert die x_k und w_k auf 25 Dezimalstellen genau:

```
 t(2,0) = -.57735026918962576450914878     w(2,0) = .50000000000000000000000000
 t(2,1) =  .57735026918962576450914878     w(2,1) = .50000000000000000000000000

 t(3,0) = -.77459666924148337703585308     w(3,0) = .27777777777777777777777778
 t(3,1) = 0.                               w(3,1) = .44444444444444444444444445
 t(3,2) =  .77459666924148337703585308     w(3,2) = .27777777777777777777777778

 t(4,0) = -.86113631159405257522394649     w(4,0) = .17392742256872692868653196
 t(4,1) = -.33998104358485626480266575     w(4,1) = .32607257743127307131346804
 t(4,2) =  .33998104358485626480266575     w(4,2) = .32607257743127307131346804
 t(4,3) =  .86113631159405257522394649     w(4,3) = .17392742256872692868653196

 t(5,0) = -.90617984593866399279762690     w(5,0) = .11846344252809454375713202
 t(5,1) = -.53846931010568309103631442     w(5,1) = .23931433524968323402064560
 t(5,2) = 0.                               w(5,2) = .28444444444444444444444446
 t(5,3) =  .53846931010568309103631442     w(5,3) = .23931433524968323402064560
 t(5,4) =  .90617984593866399279762690     w(5,4) = .11846344252809454375713202

 t(6,0) = -.93246951420315202781230155     w(6,0) = .85662246189585172520148077e-1
 t(6,1) = -.66120938646626451366139959     w(6,1) = .18038078652406930378491676
 t(6,2) = -.23861918608319690863050172     w(6,2) = .23395696728634552369493534
 t(6,3) =  .23861918608319690863050172     w(6,3) = .23395696728634552369493534
 t(6,4) =  .66120938646626451366139959     w(6,4) = .18038078652406930378491676
 t(6,5) =  .93246951420315202781230155     w(6,5) = .85662246189585172520148077e-1

 t(7,0) = -.94910791234275852452618964     w(7,0) = .64742483084434846635305724e-1
 t(7,1) = -.74153118559939443986386482     w(7,1) = .13985269574463833395073393
 t(7,2) = -.40584515137739716690660641     w(7,2) = .19091502525255947247518454
```

```
t(7,3) = 0.                                  w(7,3) = .20897959183673469387755l6
t(7,4) =  .40584515137739716690660l64        w(7,4) = .19091502525255947247518l5
t(7,5) =  .74153118559939443986386l48        w(7,5) = .13985269574463833395073l9
t(7,6) =  .94910791234275852452618l97        w(7,6) = .64742483084434846635305l2e-1

t(8,0) = -.96028985649753623168356l09        w(8,0) = .50614268145188129576265l64e-1
t(8,1) = -.79666647741362673959155l39        w(8,1) = .11119051722668723527217l81
t(8,2) = -.52553240991632898581773l90        w(8,2) = .15685332293894364366898l11
t(8,3) = -.18343464249564980493947l61        w(8,3) = .18134189168918099148257l52
t(8,4) =  .18343464249564980493947l61        w(8,4) = .18134189168918099148257l52
t(8,5) =  .52553240991632898581773l90        w(8,5) = .15685332293894364366898l11
t(8,6) =  .79666647741362673959155l39        w(8,6) = .11119051722668723527217l81
t(8,7) =  .96028985649753623168356l09        w(8,7) = .50614268145188129576265l64e-1
```

Mit den auf unser Integrationsintervall $[-h/2, h/2]$ transformierten Nullstellen t_k lassen sich die Gauß-Legendre-Formeln gewinnen. Den umfangreichen Beweis des folgenden Satzes findet man z.B. in [3] und [5].

Satz 4 (Gauß-Legendre-Formeln) *$x_0, x_1, \ldots, x_n$ seien die Nullstellen des Legendre-Polynoms $L_{n+1}(x)$. Dann gilt für $f \in C^{(2n+2)}([-\frac{h}{2}, \frac{h}{2}])$:*

(*i*) *Mit den Knoten $t_k = \frac{h}{2}x_k$, $k = 0(1)n$, sind die Gewichte w_k durch das System (12) eindeutig bestimmt.*

(*ii*) *Die Integrationsformel (14) hat einen Fehler*

$$F_n \;=\; I - G_n \;=\; \frac{[(n+1)!]^4}{[(2n+2)!]^2(2n+3)} \cdot h^{2n+3} \cdot \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \;=\; \mathcal{O}(h^{2n+3}) \tag{16}$$

mit $-\frac{h}{2} \le \xi \le \frac{h}{2}$.

Beispiel 5 Die bekanntesten Gauß-Legendre-Formeln sind für $n = 1$ die 2-Punkt-Formel

$$G_1 = \frac{h}{2}\left[f\left(-\frac{1}{3}\sqrt{3}\frac{h}{2}\right) + f\left(+\frac{1}{3}\sqrt{3}\frac{h}{2}\right)\right] \tag{17}$$

mit Integrationsfehler

$$F_1 = \frac{h^5}{180}\,\frac{f^{(4)}(\xi)}{4!}\,, \qquad -\frac{h}{2} \le \xi \le \frac{h}{2} \tag{18}$$

und für $n = 2$ die 3-Punkt-Formel

$$G_2 = \frac{h}{18}\left[5f\left(-\sqrt{\frac{3}{5}}\,\frac{h}{2}\right) + 8f(0) + 5f\left(\sqrt{\frac{3}{5}}\,\frac{h}{2}\right)\right] \tag{19}$$

mit dem Fehler

$$F_2 = \frac{h^7}{2800}\,\frac{f^{(6)}(\xi)}{6!}\,, \qquad -\frac{h}{2} \le \xi \le \frac{h}{2}\,. \tag{20}$$

Offenbar integriert G_1 – wie auch die Simpson-Regel – alle Polynome bis zum Grad 3 exakt, d.h. sie besitzt den *Genauigkeitsgrad* 3. Dagegen hat die 3-Punkt-Formel G_2 bereits den Genauigkeitsgrad 5 der Milne-Regel. □

1.2 Zusammengesetzte Quadraturformeln

Während für analytische Funktionen f mit Integrationsformeln hoher Ordnung sehr genaue Approximationen I_n des Integralwertes möglich sind, führt die in der Praxis oft fehlende Glattheit meist zu unbefriedigenden Resultaten. So konvergieren die Newton-Cotes-Formeln nicht für jede stetige Funktion, d.h. $\lim_{n\to\infty} I_n = I$ gilt nicht für beliebiges stetiges $f(t)$. Um die einfachen Newton-Cotes-Formeln auf die Ausgangsaufgabe (1) anzuwenden, unterteilen wir deshalb das Gesamtintervall $[a, b]$ in N gleichlange Teilintervalle der Länge $h = (b - a)/N$, so dass $h \ll 1$ wird, und wenden in jedem Teilintervall eine Newton-Cotes-Formel mit $n + 1$ Knoten an. Für die Simpson-Regel z.B. ergibt sich die Unterteilung mit insgesamt $2N + 1$ Knoten $t_0, t_1, ..., t_{2N}$ in Abb. 8. Allgemein lauten nun die Knoten zu einer

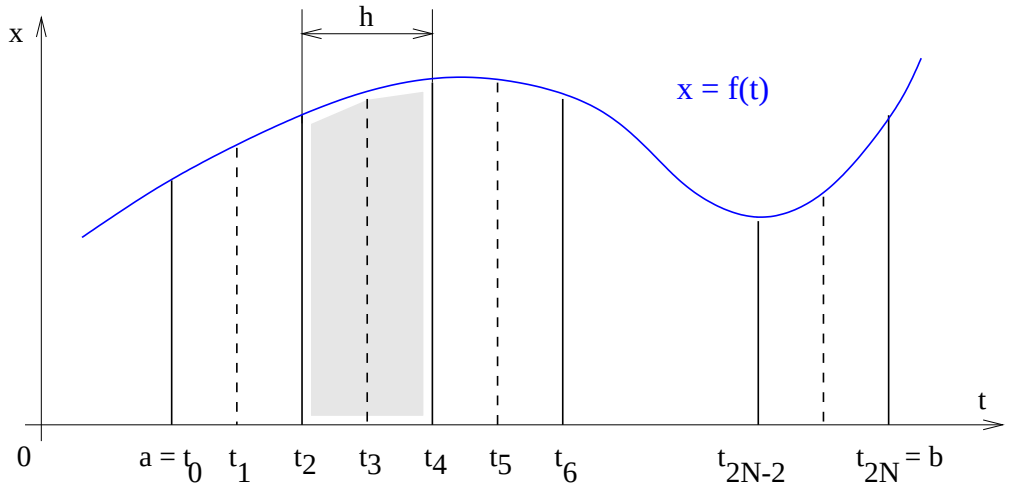


Abbildung 8: Zusammengesetzte Simpson-Regel

gegebenen Newton-Cotes-Formel I_n

$$t_k = a + k\frac{h}{n}, \quad k = 0(1)nN \qquad \text{mit} \qquad h = \frac{b-a}{N}\,. \tag{21}$$

Addiert man die N Teilintegrale I_n, so gelangt man zu den *zusammengesetzten* (*summierten*) *Newton-Cotes-Formeln* I_N . Die zusammengesetzte Rechteckregel/links lautet dann

$$R_L = h[f_0 + f_1 + \ldots + f_{N-1}]\,, \tag{22}$$

die Mittelpunktregel mit $f_{k+1/2} = f(t_k + h/2)$

$$R_M = h[f_{1/2} + f_{1+1/2} + \ldots + f_{N-1/2}]\,. \tag{23}$$

Durch Zusammenfassung der Teilintegrale (6) erhält man die Trapezregel

$$T = \frac{h}{2}[f_0 + 2f_1 + 2f_2 + \ldots + 2f_{N-1} + f_N]\,. \tag{24}$$

Auf analoge Weise folgt aus (7) die Simpson-Regel

$$S = \frac{h}{6}[f_0 + 4f_1 + 2f_2 + 4f_3 + \ldots + 4f_{2N-1} + f_{2N}] \tag{25}$$

und aus (8) die Newtonsche 3/8-Regel

$$N = \frac{h}{8}[f_0 + 3f_1 + 3f_2 + 2f_3 + 3f_4 + \ldots + 3f_{3N-2} + 3f_{3N-1} + f_{3N}]\,. \tag{26}$$

Schließlich liefert die einfache Fünfpunktregel von *Milne* (9) die folgende zusammengesetzte Milne-Regel:

$$M = \frac{h}{90}[7f_0 + 32f_1 + 12f_2 + 32f_3 + 14f_4 + 32f_5 + \ldots + 12f_{4N-2} + 32f_{4N-1} + 7f_{4N}]\,. \tag{27}$$

Ähnlich lassen sich aus (17) und (19) ebenfalls zusammengesetzte Gauß-Legendre-Formeln gewinnen.

Formel	N	I_n	K	$I - I_n$	Q
Trapez-Regel	20	1.995 885 973	21	4.11E-3	4.0058
	40	1.998 971 811	41	1.03E-3	4.0019
	80	1.999 742 972	81	2.57E-4	4.0000
	160	1.999 935 744	161	6.44E-5	3.9932
Simpson-Regel	10	2.000 006 784	21	-6.78E-6	16.143
	20	2.000 000 423	41	-4.23E-7	16.038
	40	2.000 000 026	81	-2.60E-8	16.269
	80	2.000 000 002	161	-2.00E-8	-

Tabelle 3: Genauigkeit von zusammengesetzter Trapez- und Simpson-Regel

Beispiel 6 Anwendung der zusammengesetzten Trapez- bzw. Simpson-Regel auf $I = \int_0^\pi sin\ t\ dt\ =\ 2$ liefert die Näherungswerte I_n in Tabelle 3. Vergleicht man die mit derselben Anzahl von Knoten K (also mit vergleichbarem numerischen Aufwand) erhaltenen Näherungswerte, so wird die wesentlich höhere Genauigkeit der Simpsonformel ersichtlich. □

Der Gesamtfehler einer zusammengesetzten Formel I_n ergibt sich naturgemäß aus der Summe der N Einzelfehler der Teilintervalle der Breite h. Für die Trapezregel erhält man mit der Mittelwerteigenschaft von $f''(t)$

$$\begin{aligned} F_T &= I - T = -\frac{h^3}{6}\frac{f''(\xi_1)}{2!} - \frac{h^3}{6}\frac{f''(\xi_2)}{2!} - \ldots - \frac{h^3}{6}\frac{f''(\xi_N)}{2!} \\ &= -\frac{h^3}{12}\left[f''(\xi_1) + f''(\xi_2) + \ldots + f''(\xi_N)\right]\,, \quad t_{i-1} \le \xi_i \le t_i\,, \\ &= -\frac{h^3}{12}\cdot N f''(\xi) = -\frac{b-a}{12}h^2 f''(\xi)\,, \quad a \le \xi \le b\,. \end{aligned}$$

Die Fehlerordnung p der zusammengesetzten Trapezregel beträgt also nur 2. Analog würde man für die Simpson-Regel und die 3/8-Regel $p = 4$ erhalten (vgl. Tabelle 4), während die Milne-Regel die Ordnung $p = 6$ besitzt. Denn für den Fehler der einfachen Milne-Regel gilt nach Tabelle 2

$$F_M \;=\; I - M \;=\; -\frac{h^7}{2688}\frac{f^{(6)}(0)}{6!} + \mathcal{O}(h^9) \;=\; -\frac{h^7}{2688}\frac{f^{(6)}(\xi)}{6!}$$

mit einem $\xi \in (-h/2, h/2)$. Summation über die N Teilintervalle liefert

$$\begin{aligned} F_M &= \sum_{k=1}^{N} -\frac{h^7}{2688}\frac{f^{(6)}(\xi)}{6!}, \quad t_{4k-4} < \xi_k < t_{4k} \\ &= -\frac{h^7}{2688}\frac{N}{6!} f^{(6)}(\xi) \end{aligned}$$

mit einem Wert $a < \xi < b$ nach der Mittelwerteigenschaft von $f^{(6)}$. Einsetzen von N liefert den gesuchten Gesamtfehler

$$F_M \;=\; -\frac{b-a}{1\;935\;360} \cdot h^6 f^{(6)}(\xi), \qquad a < \xi < b.$$

Die theoretische Fehlerordnung p einer Integrationsformel können wir übrigens leicht über-

n	Formel	$F_n = I - I_n$	p
0	Rechteckregel/links	$\frac{b-a}{2} h f'(\xi)$	1
1	Mittelpunktregel	$\frac{b-a}{24} h^2 f''(\xi)$	2
1	Trapezregel	$-\frac{b-a}{12} h^2 f''(\xi)$	2
2	Simpson-Regel	$-\frac{b-a}{2880} h^4 f^{(4)}(\xi)$	4
3	Newton's 3/8-Regel	$-\frac{b-a}{6480} h^4 f^{(4)}(\xi)$	4

Tabelle 4: Integrationsfehler zusammengesetzter Newton-Cotes-Formeln

prüfen, wenn wir die Integrationsfehler $F(h)$ und $F(2h)$ bei Rechnung mit Schrittweiten h und $2h$ (entsprechend $2N$ und N Teilintervallen) ermitteln und anschließend dividieren. Denn eine Taylorentwicklung liefert die genauere Darstellung

$$F(h) \;=\; I - I(h) \;=\; (b-a)\, c\, h^p \;+\; \mathcal{O}(h^{p+2}) \tag{28}$$

mit einer für die Formel typischen, aber von h unabhängigen Konstanten c. Bei hinreichend kleiner Schrittweite h folgt hieraus asymptotisch

$$Q \;:=\; \frac{F(2h)}{F(h)} \;\approx\; \frac{2^p h^p}{h^p} \;=\; 2^p\,.$$

In Tabelle 3 kommt diese Fehlerordnung $p = 2$ bzw. $p = 4$ durch die berechneten Q-Werte gut zum Ausdruck.

Beispiel 7 Versucht man, das Integral über dem glatten Integranden f

$$I = \int_{-100}^{100} \frac{dt}{10^{-4} + t^2} \;=\; 314.139\;265\;359\;045\;990\ldots \tag{29}$$

mit einer zusammengesetzten Newton-Cotes- oder Gaußformel zu bestimmen, so wird man

Formel	N	I_n	Formel	N	I_n
Trapez-	1000	2016.402	Simpson-	500	1352.469
Regel	2000	1032.664	Regel	1000	704.751
	4000	564.108		2000	407.923

Tabelle 5: Äquidistante Formeln für das Integral (29)

auch mit großer Zahl N von Teilintervallen falsche Ergebnisse erhalten (vgl. Tabelle 5). Ursache ist die schlechte Approximation des Impulses der Höhe 10 000, den der Integrand bei $t = 0$ besitzt. Für den flachen Verlauf des Integranden in den Bereichen $[-100, -1]$ und $[1, 100]$ dagegen wären wenige Integrationsknoten ausreichend. □

1.3 Romberg-Verfahren und adaptive Quadratur

Extrapolations-Prinzip und Extrapolation Um die Dichte der Knoten t_k dem Verlauf des Integranden $f(t)$ anzupassen, muss der Integrationsfehler $F(h) = I(h) - I$ eines Näherungswertes $I(h)$ ohne Kenntnis des exakten Wertes I bei Rechnung mit einer Schrittweite h abgeschätzt werden. Führt man wie in Beispiel 6 eine „Feinrechnung“ mit Schrittweite h und unabhängig davon eine „Grobrechnung“ mit doppelter Schrittweite $2h$ durch, so ergeben sich bei Anwendung einer Newton-Cotes-Formel der Ordnung p wegen (28) asymptotische Fehlerdarstellungen

$$\begin{aligned} F(h) &= I - I(h) &&= (b-a)\,c\,h^p \;+\; \mathcal{O}(h^{p+2}) \\ F(2h) &= I - I(2h) &&= (b-a)\,c\,(2h)^p \;+\; \mathcal{O}(h^{p+2})\,. \end{aligned}$$

Subtraktion beider Gleichungen und Division durch $2^p - 1$ liefert die Darstellung

$$\frac{I(h) - I(2h)}{2^p - 1} \;=\; (b-a)\,c\,h^p \;+\; \mathcal{O}(h^{p+2})\,,$$

die nach Einsetzen in (28) für $F(h)$

$$F(h) \;=\; I - I(h) \;=\; \frac{I(h) - I(2h)}{2^p - 1} \;+\; \mathcal{O}(h^{p+2}) \tag{30}$$

ergibt. Das so genannte *Extrapolations-Prinzip* besteht damit in der asymptotischen Schätzung des Fehlers der Feinrechnung durch den Term

$$\text{Error} \;:=\; \frac{I(h) - I(2h)}{2^p - 1}\,. \tag{31}$$

Stellt man Gleichung (30) nach dem exakten Wert I um

$$I \;=\; I(h) + \frac{I(h) - I(2h)}{2^p - 1} \;+\; \mathcal{O}(h^{p+2})\,,$$

so stellt der korrigierte (extrapolierte) Wert

$$I_1(h) \;:=\; I(h) + \frac{I(h) - I(2h)}{2^p - 1} \tag{32}$$

eine Integralnäherung der Fehlerordnung $p+2$ dar. Damit gilt folgender

Satz 8 (Extrapolations-Prinzip) *Sei $f \in C^{p+2}([a,b])$ mit der Fehlerordnung $p \in \mathbb{N}$ der Newton-Cotes-Formel. $I(h)$, $I(2h)$ seien die mit Fein- bzw. Grobrechnung erhaltenen Näherungen für den Integralwert I.*

(i) Für den Fehler der Feinrechnung $F(h) = I - I(h)$ stellt

$$\textit{Error} \;:=\; \frac{I(h) - I(2h)}{2^p - 1}$$

eine asymptotische Schätzung der Ordnung $p+2$ dar.

(ii) Der aus Fein- und Grobrechnung zusammengesetzte (extrapolierte) Wert

$$I_1(h) \;:=\; I(h) + \frac{I(h) - I(2h)}{2^p - 1}$$

ist eine Integralnäherung der Ordnung $p+2$.

Beide Beziehungen stellen *asymptotische Schätzungen* dar, d.h. je kleiner die Schrittweite h, desto genauer wird der Wert $F(h)$ bzw. I approximiert – falls die Rundungsfehler vernachlässigt werden können. Kombiniert man also z. B. zwei Trapezwerte $T(h)$ und $T(2h)$ der Fehlerordnung 2, so besitzt der verbesserte Wert

$$T_1(h) \;=\; T(h) + \frac{T(h) - T(2h)}{4 - 1} \tag{33}$$

die Fehlerordnung $p = 4$. Bestimmt man mit (33) auch einen Grobwert $T_1(2h)$, so kann man mittels der allgemeinen Formel (32) mit $p = 4$ den wiederum besseren Näherungswert

$$T_2(h) = T_1(h) + \frac{T_1(h) - T_1(2h)}{4^2 - 1} \tag{34}$$

der Fehlerordnung $p = 4{+}2 = 6$ ermitteln (besser: extrapolieren) etc. Das *Romberg-Verfahren* wiederholt diese Extrapolation mehrfach. Es benötigt deshalb eine Folge von Trapeznäherungen $T(h), T(2h), T(2^2h), ..., T(2^K h)$, die allgemein mit $T_0(h) \equiv T(h)$ bezeichnet werden. Mittels der aus (33) und (34) abgeleiteten Rekursionsformel werden diese Werte schrittweise verbessert zu Werten

$$\boxed{T_{k+1}(h) = T_k(h) + \frac{T_k(h) - T_k(2h)}{4^{k+1} - 1}, \quad k = 0(1)K-1,} \tag{35}$$

tol	$T_K(h)$	K	p	$T_K(h)$	K
1E-4	2.000 005 550 0	9	6	9520.703	129
1E-6	1.999 999 994 6	17	8	1210.207	1025
1E-8	2.000 000 000 0	33	10	295.046	8193

Tabelle 6: Romberg-Näherungen für Beispiele 6 (links) und 7 (rechts)

die eine Fehlerordnung $p = 2k+4$ besitzen. Der letzte berechnete Näherungswert $T_K(h)$ stellt i.A. den genauesten Wert mit einer Fehlerordnung $p = 2K + 2$ dar. Häufig gibt man die Anzahl K der Romberg-Verbesserungen (35) nicht a-priori vor, sondern testet in jedem Schritt, ob der Fehler von $T_k(h)$ – der nach (32) genau durch den Betrag des zweiten Summanden in (35) dargestellt wird – bereits kleiner als eine gegebene Toleranz *tol* ist.

Beispiel 9 1. Für Beispiel 6 mit $I = 2$ liefert das Romberg-Verfahren zu gegebener Genauigkeitsschranke *tol* die links stehenden Näherungen der Tabelle 6. Mit nur 33 Funktionsberechnungen wird so eine 11-stellige Genauigkeit erreicht!

2. Dagegen versagt auch das Romberg-Verfahren (wie bereits Trapezregel und Simpson-Regel) bei Beispiel 7 trotz Glattheit des Integranden. Zu vorgegebener Toleranz *tol* wird vom Verfahren nicht eine einzige richtige Dezimalstelle geliefert! □

Adaptive Integration Aus der Sicht des Nutzers muss ein Quadraturverfahren – unabhängig vom konkreten Zugang – folgende Grundaufgabe lösen: „Zu vorgegebener Integrationsgenauigkeit *tol* ist ein Näherungswert $I(h)$ für den Integralwert I zu finden, so dass $|I(h) - I| < tol$ gilt." Dazu muss das Intervall $[a, b]$ durch das Verfahren selbst, also „automatisch" unterteilt werden, bis die geforderte Genauigkeit erreicht wird. Die automatische Anpassung der

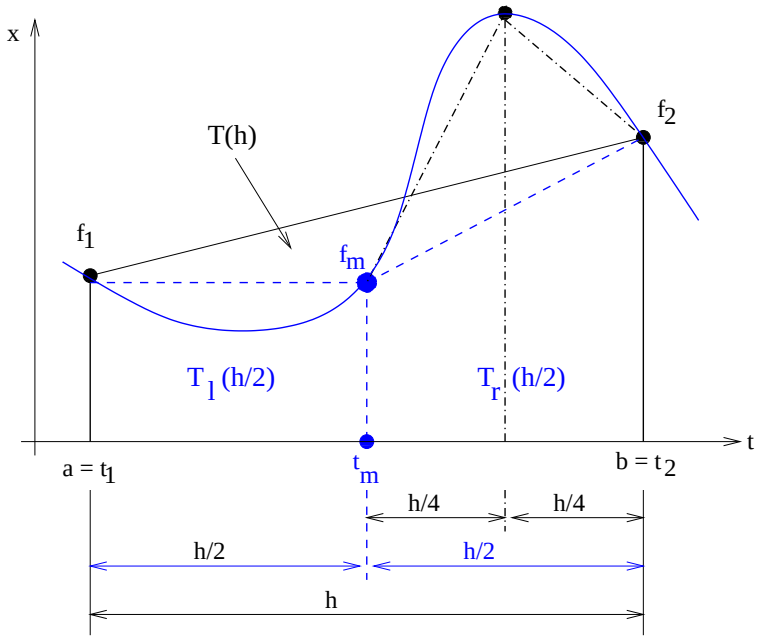


Abbildung 9: Adaptive Trapezregel

Gitterpunkte t_i an den Funktionsverlauf führt dann auf ein *adaptives Integrationsverfahren*. Um eine Information über den Integrationsfehler zu erhalten, bietet sich das Extrapolations-

Prinzip 8 an. Wenden wir z.B. die Trapezregel auf das Gesamtintervall $[a, b]$ an, so liegt mit $T(h)$, $h = b - a$, ein erster Grobwert vor. Nach Halbierung des Intervalls bestimmen wir die Trapezwerte $T_l(h/2)$ und $T_r(h/2)$ des linken und des rechten Teilintervalls und damit den Feinwert

$$T(\frac{h}{2}) = T_l(\frac{h}{2}) + T_r(\frac{h}{2}) \tag{36}$$

sowie den absoluten Fehler des Feinwertes nach (31) zu

$$Error \;=\; \frac{|T(\frac{h}{2}) - T(h)|}{3} \tag{37}$$

(vgl. Abb. 9). Gilt $Error < tol$, so ist $I(h/2) = T(h/2)$ ein geeigneter Näherungswert. Andernfalls führen wir für beide Teilintervalle dieselbe Rechnung durch usw. Dann stehen $T_l(h/2)$ und $T_r(h/2)$ als Grobwerte bereits zur Verfügung, so dass einmal berechnete Näherungswerte stets weiter verwendet werden können. Im Unterschied zum schrittweisen Vorgehen mit wachsenden t-Werten muss nach diesem „Teile-und-herrsche-Prinzip“ kein Funktionswert mehrfach bestimmt werden. Sämtliche nach (36) ermittelten Feinwerte $T(h/2), T(h/4)$ usw. der einzelnen Teilintervalle werden aufaddiert und liefern den gewünschten Näherungswert des Integrals I. Für den Algorithmus 10 der *adaptiven Trapezregel* bietet sich demzufolge eine rekursive Formulierung an, z.B. in der Sprache C/C++. Das Abbruch-

Algorithmus 10 (Adaptive Trapezregel)
Function integral $(t_1, t_2, f_1, f_2, T, h)$

1. Intervallhalbierung:
$$t_m = \frac{1}{2}(t_1 + t_2)\,,\; f_m = f(t_m),\; h = \frac{h}{2}$$
2. Feinwertbestimmung:
$$\begin{aligned} T_l &= \frac{h}{2}(f_1 + f_m)\,, \quad T_r \;=\; \frac{h}{2}(f_m + f_2) \\ T_F &= T_l + T_r \end{aligned}$$
3. Abbruchtest und Summation:
Falls $|T_F - T|/3 < tol(|T_F| + tol)$, so
$I \;=\; I + T_F$ und Return
4. Andernfalls rufe rekursiv auf:
   4.1. integral $(t_1, t_m, f_1, f_m, T_l, h)$ und
   4.2. integral $(t_m, t_2, f_m, f_2, T_r, h)$.

kriterium 3 führt einen kombinierten Relativ-Absolut-Fehlertest durch, der den kleiner werdenden Teilintegralen bei sukzessiver Halbierung Rechnung trägt. Zum Aufruf der Funktion sind Anfangswerte für das Gesamtintervall

$$t_1 = a,\; t_2 = b,\; f_1 = f(a),\; f_2 = f(b),\; h = b - a,\; T = h(f_1 + f_2)/2$$

bereitzustellen und der Integralwert ist auf $I = 0$ zu setzen.

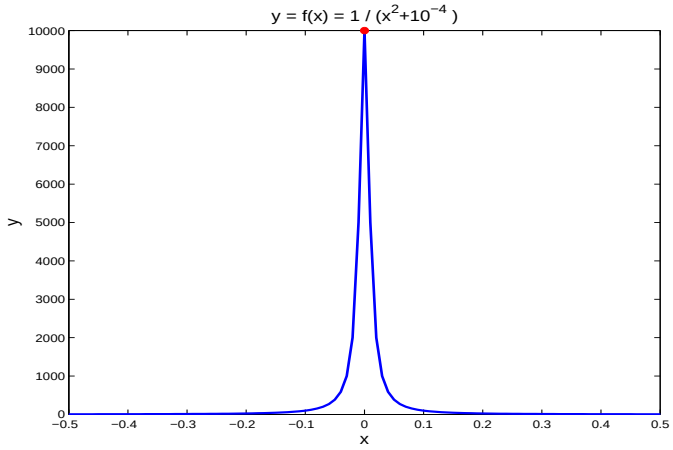


Abbildung 10: Integrand zu Beispiel 11

Selbstverständlich lässt sich derselbe Algorithmus auch auf anderen Newton-Cotes- bzw. Gauß-Legendre-Formeln in analoger Weise aufbauen.

Beispiel 11 Die adaptive Trapezregel 10 liefert für das Integral aus Beispiel 7

$$I = \int_{-100}^{100} \frac{dt}{10^{-4} + t^2} = 314.139\ 265\ 359\ 045\ 990 \ldots$$

die Näherungswerte in Tabelle 7. Abb. 10 zeigt den stark vergrösserten Ausschnitt des Graphen von $f(x) = 1/(x^2 + 10^{-4})$ in der Nähe des Koordinatenursprungs. Die Berechnung von

tol	I	K	Rek
1E-1	314.778 004	89	15
1E-2	314.376 313	297	18
1E-3	314.153 588	969	19
1E-4	314.142 156	3141	21

Tabelle 7: Näherungen der adaptiven Trapezregel für Bsp. 11

I mit der adaptiven Trapezregel stellt sich als sehr aufwendig dar, da $f(x) \approx 0$ ausserhalb einer Umgebung um 0 ist. Die maximale Rekursionstiefe Rek des Algorithmus wird stets bei $t = 0$ erreicht. □

Wir entwickeln nun nach dem Muster des Algorithmus 10 eine *adaptive Simpson-Regel*. Dazu modifizieren wir die adaptive Trapezregel, indem wir die Schrittweite $h = t_2 - t_1$, den Grobwert $S(h) = \frac{h}{6}(f_1 + 4f_m + f_2)$, die 3 Knoten t_1, t_m, t_2 und die zugehörigen Werte f_1, f_m, f_2 als Eingangsparameter übergeben. Mit dem Feinwert $S(h/2)$ liefert Satz 8 mit der Fehlerordnung $p = 4$ nun den asymptotischen Fehlerschätzer

$$Error := \frac{1}{15}[S(h/2) - S(h)]\,,$$

Algorithmus 12 (Adaptive Simpson-Regel)
Function integral $(t_1, t_m, t_2, f_1, f_m, f_2, S, h)$

1. Intervallhalbierung:
$$t_l = \frac{1}{2}(t_1 + t_m)\,,\ f_l = f(t_l),$$
$$t_r = \frac{1}{2}(t_2 + t_m)\,,\ f_r = f(t_r),\ h = \frac{h}{2}$$

2. Feinwertbestimmung:
$$\begin{aligned} S_l &= \frac{h}{6}(f_1 + 4f_l + f_m)\,, \quad S_r = \frac{h}{6}(f_m + 4f_r + f_2) \\ S_F &= S_l + S_r \end{aligned}$$

3. Abbruchtest und Summation:
Falls $|S_F - S|/15 < tol(|S_F| + tol)$, so
$I = I + S_F$ und Return

4. Andernfalls rufe rekursiv auf:

4.1. integral $(t_1, t_l, t_m, f_1, f_l, f_m, S_l, h)$ und

4.2. integral $(t_m, t_r, t_2, f_m, f_r, f_2, S_r, h)$.

womit sich die 4 Schritte des Algorithmus 12 ergeben.

Beispiel 13 Rechnung des Beispiels 11

$$I = \int_{-100}^{100} \frac{dt}{10^{-4} + t^2} = 3.141\ 392\ 653\ 590\ 459\ 90 \ldots e + 2$$

mit der adaptiven Simpson-Regel liefert die genaueren Näherungswerte in Tabelle 8. Mit

tol	I	K
1E-4	3.141 453 296 656 609e+002	107
1E-6	3.141 392 781 765 725e+002	375
1E-8	3.141 392 652 169 681e+002	1211
1E-10	3.141 392 653 634 381e+002	3743

Tabelle 8: Adaptive Simpson-Regel für Bsp. 11

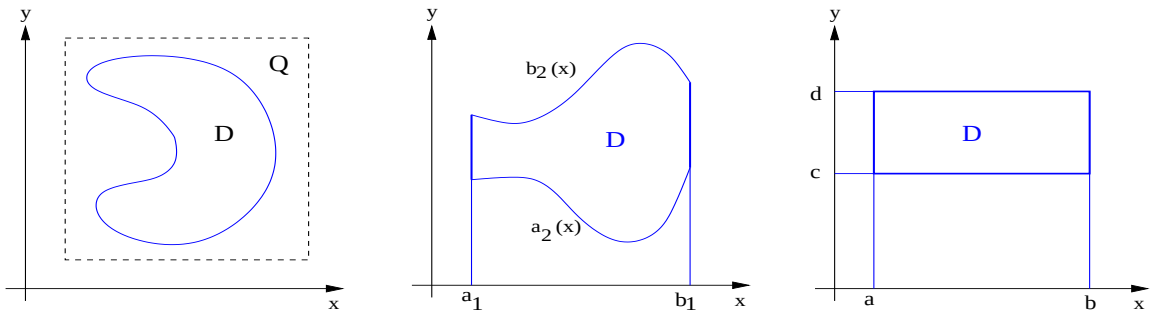
einer 16-Punkt-Gauß-Legendre-Formel konnte sogar der Wert $I = 314.139\ 265\ 36$ bis auf 11 Dezimalstellen mit nur 1569 Funktionswerten ermittelt werden. □

2 Numerische Kubatur

Wie im eindimensionalen Fall ist auch die multivariate Integration bei praktisch bedeutsamen Problemen meist nicht geschlossen ausführbar. Wir betrachten als Standardaufgabe die *Kubaturaufgabe*[2] zur Bestimmung des d-dimensionalen Riemann-Integrals

$$I = \int_D f(x)\,dx, \quad x = (x_1, x_2, \ldots, x_d), \quad d \geq 2 \tag{38}$$

einer gegebenen Riemann-integrierbaren Funktion $f : D \to \mathbb{R}$. Der Integrationsbereich $D \in \mathbb{R}^d$ wird als beschränkt mit stückweise glattem Rand vorausgesetzt. Numerische Verfahren sind in höheren Dimensionen d bei gleicher Genauigkeitsforderung wesentlich Rechenzeitaufwändiger als die behandelten Quadraturformeln. Hinzu kommt eine große Vielfalt möglicher Integrationsbereiche D. In Abb. 11 sind einige typische ebene Bereiche D dargestellt.



(a) (b) (c)

Abbildung 11: Typische Integrationsbereiche $D \in \mathbb{R}^2$

Wir wollen die Grundideen der numerischen Kubatur wegen der höheren Anschaulichkeit an *Flächenintegralen (2D-Integralen)*

$$I = \iint_D f(x,y)\,d(x,y), \quad f : D \subset \mathbb{R}^2 \to \mathbb{R} \tag{39}$$

erläutern. Die direkte Verallgemeinerung der Formeln auf *Volumenintegrale (3D-Integrale)*

$$I = \iiint_D f(x,y,z)\,d(x,y,z), \quad f : D \subset \mathbb{R}^3 \to \mathbb{R} \tag{40}$$

und auf noch höhere Dimensionen ist oft möglich und sollte selbstständig versucht werden.

2.1 Konstruktion von Kubaturformeln

Wie bei der Quadratur approximieren wir das Integral (38) durch eine gewichtete Summe von Werten des Integranden f an ausgewählten Punkten $x_k \in D$ und erhalten damit *numerische Kubaturformeln*. Nach geeigneten Annahmen ergeben sich mittels der Methode der

[2] Als Kubatur bezeichnet man die Volumenbestimmung räumlicher Gebiete.

unbestimmten Koeffizienten (oder der Anwendung auf Monome niedrigen Grades) spezielle Formelgruppen.

Definition 14 (Kubaturformeln)

(*i*) $C_n = \sum_{k=1}^{n} w_k f(x_k)$ *ist eine Kubaturformel mit n Integrationsknoten* $x_k \in D$ *und Integrationsgewichten* w_k.

(*ii*) *Sind die Komponenten der n Knoten* x_k *äquidistant festgelegt und die Gewichte* w_k *frei bestimmbar, so ist die Kubaturformel vom Newton-Cotes-Typ.*

(*iii*) *Werden die n Gewichte identisch vorgegeben und lediglich die Knoten* x_k *geeignet bestimmt, so erhält man Tschebyscheff-Kubaturformeln.*

(*iv*) *Gauß-Legendre-Kubaturformeln treffen keine Vorgaben an die* x_k *und* w_k *und lassen deshalb maximale Genauigkeit zu.*

Während die Verallgemeinerung der Quadraturformeln aus Abschnitt 1 im *Standardfall* kompakter d-dimensionaler Intervalle $Q := [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d]$ wie in Abb. 11 (c) leicht möglich wird, bieten komplizierte und krummlinig berandete Bereiche D meist größere Schwierigkeiten. Folgende Zugänge werden empfohlen, um die Aufgabe auf den Standardfall zu reduzieren.

1. Wenn wir den Bereich D wie in Abb. 11 (a) in ein d-dimensionales Intervall $Q \subset \mathbb{R}^d$ mit $D \subset Q$ einbetten, so lassen sich grobe Näherungswerte für I gewinnen. Dazu definieren wir die charakteristische Funktion von D

$$\chi_D(x) = \begin{cases} 1 & \text{für } x \in D \\ 0 & \text{für } x \notin D \end{cases}$$

und ersetzen (38) durch $\int_Q \chi_D(x) f(x) dx$. Die am Rand von D entstehenden Fehler lassen im Allgemeinen keine große Genauigkeit zu, weshalb dieser Ansatz nur bei hochdimensionalen Integralen empfohlen wird.

2. Oft ist eine Zerlegung des Integrationsbereichs D in paarweise disjunkte Teilbereiche $D_1, D_2, \ldots, D_N$ rechteckiger oder dreieckiger Form (im $\mathbb{R}^d$ in kompakte Intervalle oder Simplices) möglich. Damit kann I durch

$$I = \int_D f(x)\, dx = \sum_{k=1}^{N} \int_{D_k} f(x)\, dx \tag{41}$$

berechnet werden.

Beispiel 15 Ist das Volumen über dem Bereich D gemäß Abb. 13 durch

$$I = \iint_D \frac{\cos((x-y)/300)}{0.01 + \sin^2((x^2+y^2)/3000)}\, dx\, dy \tag{42}$$

zu bestimmen, so lässt sich die Zerlegung $D = D_1 \cup D_2 \cup D_3 \cup D_4$ in Abb. 12 angeben. □

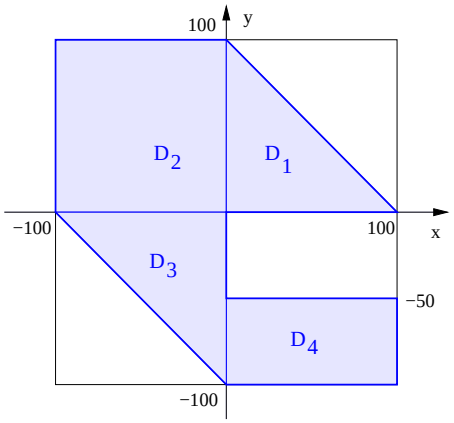


Abbildung 12: Bereich $\boldsymbol{D}$

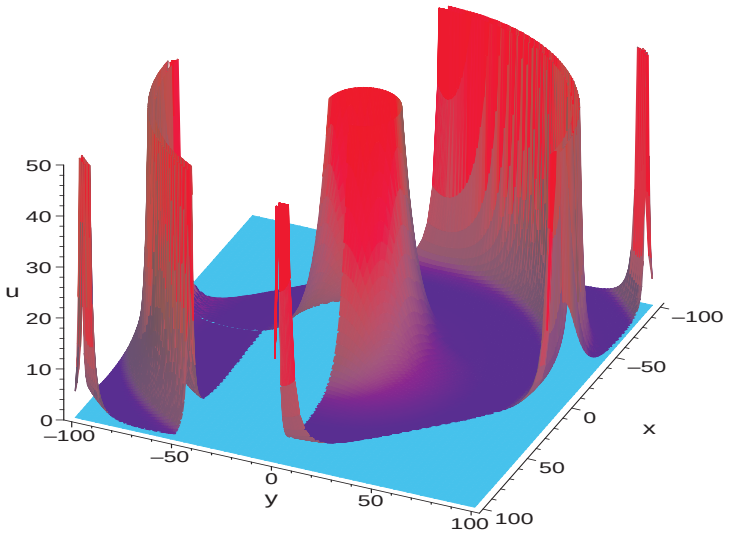


Abbildung 13: Integrand des Beispiels 15

3. Kartesische Normalbereiche wie in Abb. 11 (b) können leicht auf das *Einheitsquadrat* $\Omega = \{(s,t) \mid 0 \le s \le 1,\ 0 \le t \le 1\}$ transformiert werden. Sei z.B. der Normalbereich bezüglich der x-Achse $D_x = \{(x,y) \mid a_1 \le x \le b_1,\ a_2(x) \le y \le b_2(x)\}$ gegeben. Die Transformation $(s,t) \mapsto (x,y)$ mit

$$x(s,t) = a_1 + (b_1 - a_1)s, \qquad y(s,t) = a_2(x(s,t)) + \{b_2(x(s,t)) - a_2(x(s,t))\}\, t$$

überführt die Punkte (s,t) des Einheitsquadrates Ω in den Normalbereich D_x. Mit der Determinante

$$J(s,t) := \det \begin{pmatrix} \frac{\partial x}{\partial s} & \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial s} & \frac{\partial y}{\partial t} \end{pmatrix} = (b_1 - a_1)[b_2(a_1 + (b_1 - a_1)s) - a_2(a_1 + (b_1 - a_1)s)]$$

der Jacobi-Matrix transformieren wir das Flächenintegral (39) auf den Standardfall

$$I = \int_0^1 \int_0^1 f(x(s,t), y(s,t)) |J(s,t)|\, dt\, ds. \tag{43}$$

Andere Normalbereiche in Polar- Kugel- oder Zylinderkoordinaten lassen sich analog behandeln. Wir nehmen nun an, dass die Kubaturaufgabe in $\mathbb{R}^2$ nach Zerlegung (41) auf ein kleines Intervall, einen *Elementarbereich*

$$\Omega = \{(x,y) \mid 0 \le x \le h_x,\ 0 \le y \le h_y\} \quad \text{mit} \quad h := \max(h_x, h_y) \ll 1, \tag{44}$$

reduziert werden konnte, womit die iterierte Darstellung möglich ist:

$$I = \iint_\Omega f(x,y) d(x,y) = \int_0^{h_y} \left(\int_0^{h_x} f(x,y)\, dx \right) dy. \tag{45}$$

Mit den $n_x + 1$ paarweise verschiedenen x-Werten $0 \le x_0 < x_1 < \cdots < x_{n_x} \le h_x$ und den $n_y + 1$ paarweise verschiedenen y-Werten $0 \le y_0 < y_1 < \cdots < y_{n_y} \le h_y$ legen wir ein rechteckiges Gitter mit den Knoten $(x_k, y_l) \in \Omega$ fest. Die Idee der *Interpolationskubatur*

bedeutet die Ersetzung des Integranden $f(x,y)$ durch ein Interpolationspolynom $p(x,y)$, das an diesen Knoten mit f zusammenfällt. Wie im Quadraturfall setzen wir auch jetzt die Lagrange-Form

$$p(x,y) \;=\; \sum_{k=0}^{n_x}\sum_{l=0}^{n_y} L_k(x)L_l(y)f(x_k,y_l) \tag{46}$$

des Polynoms an. Aus der Interpolationsforderung $p(x_k,y_l) = f(x_k,y_l)$ für $k = 0(1)n_x$, $l = 0(1)n_y$, gewinnen wir leicht die zugehörigen Lagrange-Basispolynome (vgl. [4], Kap. 22)

$$L_k(x) \;=\; \prod_{\substack{i=0\\ i\neq k}}^{n_x} \frac{x-x_i}{x_k-x_i} \qquad \text{und} \qquad L_l(x) \;=\; \prod_{\substack{j=0\\ j\neq l}}^{n_y} \frac{y-y_j}{y_l-y_j}. \tag{47}$$

Einsetzen von $p(x,y)$ in (45) und Auswertung des Doppelintegrals liefert

$$\begin{aligned} C_n &= \int_0^{h_y}\int_0^{h_x} p(x,y)\,dx\,dy \\ &= \sum_{k=0}^{n_x}\sum_{l=0}^{n_y} \left(\int_0^{h_x} L_k(x)\,dx\right)\left(\int_0^{h_y} L_l(y)\,dy\right) f(x_k,y_l) \\ &= \sum_{k=0}^{n_x}\sum_{l=0}^{n_y} a_k b_l f(x_k,y_l) \end{aligned}$$

mit den $n = (n_x+1)(n_y+1)$ Konstanten

$$a_k \;=\; \int_0^{h_x} L_k(x)\,dx \qquad \text{und} \qquad b_l \;=\; \int_0^{h_y} L_l(y)\,dy. \tag{48}$$

Satz 16 (Interpolationskubatur) *Zu $n = (n_x+1)(n_y+1)$ ist die einfache Kubaturformel*

$$C_n \;=\; \sum_{k=0}^{n_x}\sum_{l=0}^{n_y} w_{kl} f(x_k,y_l), \quad w_{kl} = a_k b_l \tag{49}$$

mit n Knoten (x_k,y_l) und Gewichten w_{kl} definiert. Die $(n_x+1)\times(n_y+1)$-Matrix W der Gewichte ist durch das Tensorprodukt (dyadische Produkt)

$$W \;=\; ab^T \quad \textit{mit} \quad a = (a_0,a_1,\ldots,a_{n_x})^T \quad \textit{und} \quad b = (b_0,b_1,\ldots,b_{n_y})^T \tag{50}$$

bestimmt.

Bemerkung 17 Kubaturformeln für Volumenintegrale (40) auf 3-dimensionalen Referenzbereichen Ω mit $0 \le z \le h_z$ erhalten wir bei einer Unterteilung des z-Intervalls mit n_z+1 Punkten $0 \le z_0 < z_1 < \cdots < z_{n_z} \le h_z$ in der Form

$$C_n \;=\; \sum_{k=0}^{n_x}\sum_{l=0}^{n_y}\sum_{m=0}^{n_z} w_{klm} f(x_k,y_l,z_m), \quad w_{kl} = a_k b_l c_m \tag{51}$$

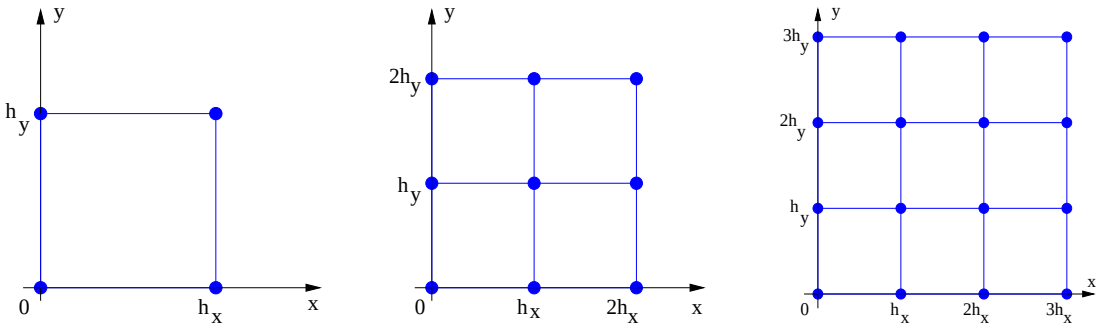
mit $n = (n_x+1)(n_y+1)(n_z+1)$ Gewichten w_{klm}. Die Gewichte c_m sind analog zu (48) definiert.

2.2 Kubaturformeln über Rechteckbereichen

Einfache Newton-Cotes-Formeln Wegen der Darstellung (50) können wir von eindimensionalen Newton-Cotes-Formeln ausgehen und damit Kubaturformeln auf Referenzbereichen Ω gewinnen. Zu fester natürlicher Zahl $r \in \mathbb{N}$ setzen wir den Referenzbereich nun vereinfachend mit

$$\Omega \;=\; \{(x,y) \mid 0 \le x \le rh_x,\; 0 \le y \le rh_y \} \quad \text{mit} \quad h \;:= \max(h_x, h_y) \tag{52}$$

an (vgl. Abb. 14). Wir wählen identische Anzahlen $r := n_x = n_y$ und Knoten (x_k, y_l) mit



(a) $r = 1$ (b) $r = 2$ (c) $r = 3$

Abbildung 14: Referenzbereiche Ω mit $r = 1, 2, 3$

$x_k = kh_x,\; k = 0(1)r$ und $y_l = lh_y,\; l = 0(1)r$, um abgeschlossene Formeln zu erhalten.

Beispiel 18 1. Für $r = 1$ nutzen wir die Gewichte der einfachen Trapezregel $T = \frac{h}{2}(f_0 + f_1)$, so dass (50) die Gewichtsmatrix

$$W \;=\; ab^T \;=\; \begin{pmatrix} h_x/2 \\ h_x/2 \end{pmatrix} (h_y/2,\; h_y/2) \;=\; \frac{h_x h_y}{4} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

ergibt. Die *2-dimensionale Trapezregel* über dem Rechteck hat damit die Form

$$T \;=\; C_4 \;=\; \frac{h_x h_y}{4} \left[f(0,0) + f(h_x,0) + f(0,h_y) + f(h_x,h_y) \right]. \tag{53}$$

Analog zur einfachen Trapezregel multipliziert sie das arithmetische Mittel der 4 Eckpunktwerte mit dem Inhalt $|\Omega| = h_x h_y$ des Bereiches. Da die einfache Trapezregel lediglich lineare Polynome exakt integriert, kann die 2-dimensionale Formel gemäß Konstruktion nicht für beliebige quadratische Polynome $p(x,y)$ exakte Werte liefern. Sie hat deshalb nur den *algebraischen Genauigkeitsgrad* $p = 1$.

2. Mit $r = 2$ gewinnen wir aus der einfachen Simpson-Regel die Gewichtsmatrix

$$W \;=\; ab^T \;=\; \frac{2h_x}{6} \begin{pmatrix} 1 \\ 4 \\ 1 \end{pmatrix} \cdot \frac{2h_y}{6} (1,4,1) \;=\; \frac{h_x h_y}{9} \begin{pmatrix} 1 & 4 & 1 \\ 4 & 16 & 4 \\ 1 & 4 & 1 \end{pmatrix},$$

die uns die *2-dimensionale Simpson-Regel (auch Kepler-Regel genannt)*

$$\begin{aligned} S \;=\; C_9 \;=\; \frac{h_x h_y}{9}\,\{&f(0,0)+f(2h_x,0)+f(0,2h_y)+f(2h_x,2h_y)+\\ &+4\,[f(h_x,0)+f(0,h_y)+f(2h_x,h_y)+f(h_x,2h_y)]+16f(h_x,h_y)\} \end{aligned} \tag{54}$$

über dem Rechteck in Abb. 14 (b) liefert. Mit dem algebraischen Genauigkeitsgrad $p = 3$ integriert sie auch bikubische Polynome exakt.

3. Der Wert $r = 3$ steht für die *2-dimensionale Newton'sche 3/8-Regel* $N = C_{16}$ über dem Rechteck in Abb. 14c mit der Gewichtsmatrix

$$W \;=\; ab^T \;=\; \frac{3h_x}{8}\begin{pmatrix}1\\3\\3\\1\end{pmatrix}\cdot\frac{3h_y}{8}(1,3,3,1) \;=\; \frac{9h_xh_y}{64}\begin{pmatrix}1&3&3&1\\3&9&9&3\\3&9&9&3\\1&3&3&1\end{pmatrix}$$

und dem algebraischen Genauigkeitsgrad $p = 3$, während für $r = 4$ mit den Gewichtsvektoren

$$a = \frac{4h_x}{90}(7,32,12,32,7)^T \qquad \text{und} \qquad b = \frac{4h_y}{90}(7,32,12,32,7)^T$$

leicht die *2-dimensionale Milne-Regel* $M = C_{25}$ mit Genauigkeitsgrad $p = 5$ hergeleitet werden kann. □

Für den Anwender einer Kubaturformel ist die Schätzung ihres *Integrationsfehlers (Kubaturfehlers)* allerdings oft bedeutsamer als der algebraische Genauigkeitsgrad.

Definition 19 (Integrationsfehler, Fehlerordnung und Genauigkeitsgrad)

(*i*) *Der lokale Integrationsfehler (Kubaturfehler) der Kubaturformel C_n ist durch*

$$F \;=\; I - C_n \;=\; \int_D f(x)\,dx \;-\; \sum_{k=1}^{n} w_k f(x_k) \tag{55}$$

definiert.

(*ii*) *Ist $F = \mathcal{O}(h^q)$ mit $h = diam(D)$ (in $\mathbb{R}^2$ sei $h = \max\{h_x, h_y\}$) und $q \in \mathbb{N}$, so hat die Kubaturformel (mindestens) die lokale Fehlerordnung q.*

(*iii*) *Integriert die Kubaturformel C_n alle Monome $p(x,y) = x^k y^l$ mit $k + l \le p$ (und damit alle Polynome maximal p-ten Grades) exakt, während dies für ein Monom $(p+1)$-ten Grades nicht gilt, so hat sie den algebraischen Genauigkeitsgrad[3] p.*

Nehmen wir an, der Integrand f sei analytisch, so lässt sich seine Taylor-Entwicklung

$$f(x,y) \;=\; f(0,0) + f_x(0,0)x + f_y(0,0)y + \frac{1}{2}f_{xx}(0,0)x^2 + f_{xy}(0,0)xy + \frac{1}{2}f_{yy}(0,0)y^2 + \ldots$$

in (55) einsetzen, integrieren und nach wachsenden Potenzen von h_x und h_y zusammenfassen. Diese Technik wird oft als *Taylor-Abgleich* bezeichnet. Die führenden Terme liefern dann mit $h = \max\{h_x, h_y\} \ll 1$ eine Ordnungsaussage der Kubaturformel.

[3] Detaillierte Ausführungen zum algebraischen Genauigkeitsgrad findet man in [2] und [5].

Beispiel 20 1. Die Trapezregel (53) besitzt mit $h = \max\{h_x, h_y\}$ die Fehlerdarstellung

$$F_T := I - T = -\frac{h_x h_y}{12}\left\{f_{xx}(0,0)h_x^2 + f_{yy}(0,0)h_y^2 + \mathcal{O}(h^3)\right\}, \tag{56}$$

womit $|F_T| \leq C_T h^4$, $C_T > 0$, und die Ordnung $q = 4$ folgt. Da alle in F_T auftretenden Ableitungen mindestens von 2. Ordnung sind, ergibt sich der Genauigkeitsgrad $p = 1$.
2. Die Fehlerdarstellung der 2-dimensionalen Simpson-Regel (54) lautet

$$F_S := I - S = -\frac{4h_x h_y}{180}\left\{f_{xxxx}(0,0)h_x^4 + f_{yyyy}(0,0)h_y^4 + \mathcal{O}(h^5)\right\}. \tag{57}$$

Wegen $|F_S| \leq C_S h^6$, $C_S > 0$, erhalten wir die Ordnung $q = 6$, während die auftretenden 4. Ableitungen den algebraischen Genauigkeitsgrad $p = 3$ liefern. □

Algorithmus 21 (Zusammengesetzte Trapezregel)

```
function I = trapez2d(fname,a,b,c,d,nx,ny);
% Eingangsparameter:
% fname ist ein Handle auf den Integranden f
% [a,b] ist das x-Intervall, [c,d] ist das y-Intervall
% nx (ny) - Zahl der Teilintervalle in x-Richtung
(y-Richtung)
%
I = 0;   hx = (b-a)/nx;   hy = (d-c)/ny;
for i=0:nx,
   xi = a + i*hx;
   for j=0:ny,
      yj = c + j*hy;
      fxy = feval(fname,xi,yj);
      if ((i==0)&(j==0))|((i==0)&(j==ny))|...
            ((i==nx)&(j==0))|((i==nx)&(j==ny))
         I = I + 0.25*fxy;
      elseif (i==0)|(i==nx)|(j==0)|(j==ny)
         I = I + 0.5*fxy;
      else
         I = I +fxy;
      end
   end
end
I = hx*hy*I;     % Integralnäherung I
```

Zusammengesetzte Newton-Cotes-Kubatur Der Integrationsbereich sei nun das kompakte Intervall $D = [a,b] \times [c,d]$ aus Abb. 11 (c). Wir unterteilen das Intervall $[a,b]$ in n_x gleich lange Teilintervalle der Länge $h_x = (b-a)/n_x$ und das Intervall $[c,d]$ in n_y Teilintervalle der Länge $h_y = (d-c)/n_y$. Wählen wir eine einfache Newton-Cotes-Formel C_n aus und wenden sie auf jedes entstehende Teilrechteck

$$\Omega_{ij} = \{(x,y) \mid x_{i-1} \leq x \leq x_i,\ y_{j-1} \leq y \leq y_j\}, \quad i = 1(1)n_x,\ j = 1(1)n_y$$

mit $x_i = a + ih_x$, $y_j = c + jh_y$, an, so erhalten wir eine *zusammengesetzte* (*summierte*) *Kubaturformel*.

Beispiel 22 Mit der einfachen Trapezregel (53) ergibt sich die *zusammengesetzte Trapezregel über Rechteckgebieten D*

$$\begin{aligned} T_D &= \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} \frac{h_x h_y}{4}\left[f(x_{i-1},y_{j-1}) + f(x_i,y_{j-1}) + f(x_{i-1},y_j) + f(x_i,y_j)\right] \\ &= \frac{h_x h_y}{4}\Bigg[f(a,c) + f(b,c) + f(a,d) + f(b,d) + 4\sum_{i=1}^{n_x-1}\sum_{j=1}^{n_y-1} f(x_i,y_j) + \\ &\quad +2\sum_{j=1}^{n_y-1}\left[f(a,y_j) + f(b,y_j)\right] + 2\sum_{i=1}^{n_x-1}\left[f(x_i,c) + f(x_i,d)\right]\Bigg]. \end{aligned} \tag{58}$$

Die 4 Eckpunkte (●) in Abb. 15 werden mit dem Faktor 1 gewichtet, die restlichen Randpunkte (■) mit 2 und alle inneren Punkte, die zu je 4 Teilintervallen gehören, mit dem Faktor 4. Den Gesamtfehler der Kubaturformel, also ihren *globalen Fehler*, liefert eine Summation aller $n_x n_y$ lokalen Fehler (56) zusammen mit der Mittelwerteigenschaft von f_{xx} und f_{yy}, ganz analog zum eindimensionalen Fall. Mit $h = \max\{h_x, h_y\}$ gewinnen wir daraus leicht die Abschätzung

$$|F_{T_D}| := |I - T_D| \le \frac{1}{12}(b-a)(d-c)\left\{\max_D |f_{xx}(x,y)|h_x^2 + \max_D |f_{yy}(x,y)|h_y^2\right\}, \tag{59}$$

womit $|F_{T_D}| \le C_{T_D} h^2$, $C_{T_D} > 0$, und die *globale Fehlerordnung* $q = 2$ folgt. □

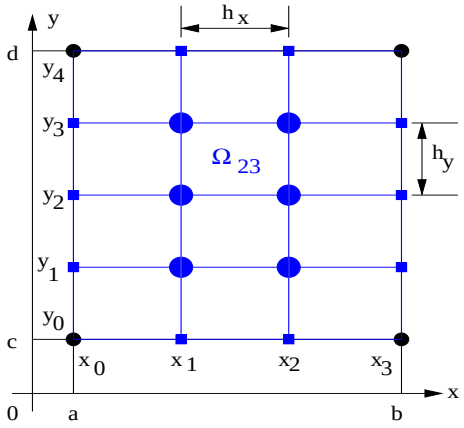


Abbildung 15: Gitter zu $\boldsymbol{T_D}$

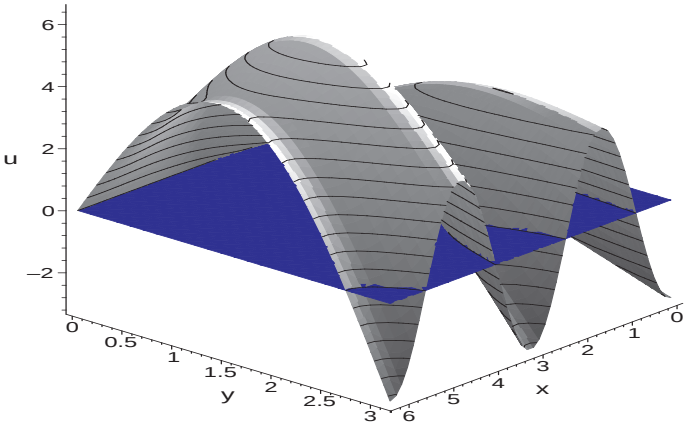


Abbildung 16: Integrand des Beispiels 23(1)

Ähnlich erhält man aus Formel (54) die *zusammengesetzte Simpson-Kubaturregel (Kepler-Regel)* S_D der Ordnung 4 auf Rechteckgebieten sowie mit Beispiel 18 die entsprechende *zusammengesetzte Newton'sche 3/8-Regel* N_D und die noch genauere *zusammengesetzte Milne-Regel* M_D (vgl. dazu [2]). Wir geben die zusammengesetzte Trapezregel als Algorithmus 21 in MATLAB-Notation an (vgl. [7]) und demonstrieren sie an folgendem

Beispiel 23 1. Anwendung der Trapezregel (58) auf das Integral (vgl. Abb. 16)

$$I = \int_0^{2\pi} \int_0^{\pi} (x \sin y - y \cos 2x)\, dy\, dx \;=\; 4\pi \;=\; 39.478417604357434...$$

liefert mit Schrittweiten $h = 2\pi/n_x$ die Näherungswerte $T_D(h)$ in Tabelle 9 mit den absoluten Fehlern $I - T_D(h)$. An deren Fehlerquotienten $Q(h) \to 2^2$ $(h \to 0)$ wird die Ordnung 2 sehr gut erkennbar. Damit ist wie im Quadraturfall eine *asymptotische Fehlerschätzung*

$$Error \;:=\; \frac{1}{3}\,[T_D(h) - T_D(2h)] \tag{60}$$

des globalen Kubaturfehlers von $T_D(h)$ leicht möglich.

$n_x = n_y$	$T_D(h)$	$I - T_D(h)$	$Q(h)$	$Error$
50	39.4654289	1.299E-2	—	—
100	39.4751706	3.247E-3	4.00062	3.2472E-3
200	39.4776059	8.117E-4	4.00025	8.1177E-4
400	39.4782147	2.029E-4	4.00049	2.0293E-4
800	39.4783668	5.073E-5	3.99961	5.07E-5
1600	39.4784049	1.268E-5	4.00079	1.27E-5

Tabelle 9: Genauigkeit der zusammengesetzten Trapezregel

n_x (n_y)	T_D (rel. Fehler)
100	8.00037E+6
	(-1.478E+1)
200	4.00108E+6
	(-6.890E+0)
400	2.00477E+6
	(-2.953E+0)
800	1.02969E+6
	(-1.030E+0)
1600	6.67053E+5
	(-3.153E-1)

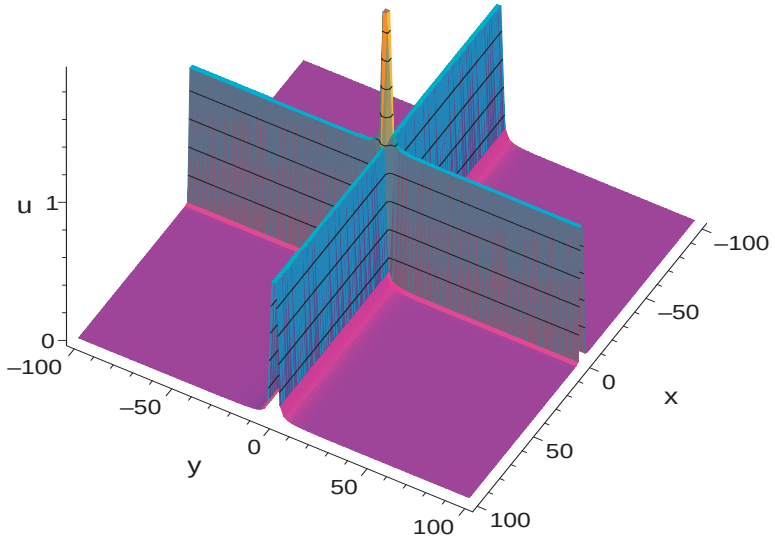


Abbildung 17: Näherungen T_D und Integrand des Beispiels 23(2)

2. Der Integrand der folgenden Aufgabe (vgl. Abb. 17)

$$I = \int_{-100}^{100} \int_{-100}^{100} \left(\frac{1}{x^4 + 10^{-4}} + \frac{1}{y^2 + 10^{-4}} \right) dx\, dy \;=\; 5.0711614675431... \cdot 10^5$$

besitzt auf der x- und y-Achse extrem große Werte, während $f(\pm 100, \pm 100) \approx 10^{-4}$ ist. Hierfür liefert T_D mit großem Aufwand die schlechten Approximationen in der Tabelle der Abb. 17. Dieses Beispiel erfordert offenbar eine adaptive Kubatur. □

Gauß-Legendre-Kubaturformeln Die skizzierten Zugänge sind nicht auf Newton-Cotes-Formeln beschränkt. Insbesondere Gauß-Legendre-Formeln hoher Ordnung können leicht mittels des Tensorprodukt-Ansatzes (49) auf Rechteckbereiche bzw. mittels Formel (51) auf Intervalle in $\mathbb{R}^3$ verallgemeinert werden.

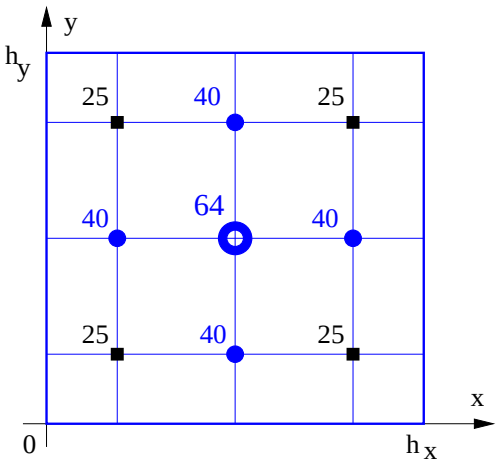


Abbildung 18: 9-Punkt-Formel

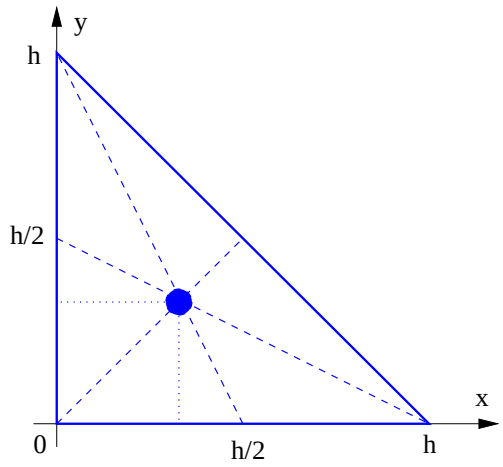


Abbildung 19: 1-Punkt-Formel

Beispiel 24 Die 3-Punkt-Gauß-Legendre-Quadraturformel (19)

$$G_3 = \frac{h}{18}\left[5f((1-c)h/2) + 8f(h/2) + 5f((1+c)h/2)\right] \quad \text{mit} \quad c = \sqrt{3/5} \tag{61}$$

sei auf dem Intervall $[0, h]$ gegeben. Mit Satz 16 erhalten wir auf dem in Abb. 18 dargestellten Referenzbereich $\Omega = \{(x, y) \mid 0 \le x \le h_x,\ 0 \le y \le h_y\}$ die Gewichtsmatrix

$$W = ab^T = \frac{h_x}{18}\begin{pmatrix} 5 \\ 8 \\ 5 \end{pmatrix} \cdot \frac{h_y}{18}(5, 8, 5) = \frac{h_x h_y}{324}\begin{pmatrix} 25 & 40 & 25 \\ 40 & 64 & 40 \\ 25 & 40 & 25 \end{pmatrix}.$$

Die *9-Punkt-Gauß-Legendre-Kubaturformel* über dem Rechteck in Abb. 18 lautet damit

$$G_9 = \frac{h_x h_y}{324}\left[64 f\left(\frac{h_x}{2}, \frac{h_y}{2}\right) + 25\sum_{(x_i,y_k)\in E} f(x_i, y_k) + 40\sum_{(x_i,y_k)\in S} f(x_i, y_k)\right] \tag{62}$$

mit der Menge der Eckpunkte E (■) und der Seitenpunkte S (●). Die Fehlerordnung $q = 6$ der zusammengesetzten Quadraturformel überträgt sich auf den Kubaturfall, so dass die zusammengesetzte Kubaturformel G_9 den *globalen Kubaturfehler* $F_{G_9} = G_9 - I = \mathcal{O}(h^6)$ der Ordnung 6 mit $h = \max\{h_x, h_y\}$ besitzt. □

2.3 Kubaturformeln über Dreieckbereichen

Einfache Kubaturformeln 2-dimensionale Newton-Cotes-Formeln, die stets durch äquidistante Vorgabe der Integrationsknoten (x_k, y_l) charakterisiert sind, lassen sich mit größerem Aufwand auch auf anderen *Elementarbereichen* konstruieren. Neben den behandelten Rechteckbereichen Ω sind Dreieckbereiche Δ von herausragender Bedeutung. Um eine 3-Punkt-Formel auf dem in Abb. 20 (a) dargestellten Elementarbereich zu gewinnen, transformieren wir Δ mittels $x = h_x s,\ y = h_y t$ linear auf das Referenz-Einheitsdreieck (vgl. Abb. 20 (b))

$$\Sigma \ = \ \{(s,t) \mid 0 \le s \le 1,\ 0 \le t \le 1 - s\,\}.$$

Beispiel 25 Bezeichnen $f_0 = f(0,0),\ f_1 = f(h_x,0),\ f_2 = f(0,h_y)$ die 3 Werte an den Integrationsknoten, so lässt sich ein linearer Interpolationsansatz

$$p(x,y) \ = \ f_0 B_0(x,y) + f_1 B_1(x,y) + f_2 B_2(x,y) \tag{63}$$

mit folgenden Basisfunktionen angeben:

$$\begin{array}{lcll} B_0(x,y) & = & 1-s-t & = & 1 - x/h_x - y/h_y \\ B_1(x,y) & = & s & = & x/h_x \\ B_2(x,y) & = & t & = & y/h_y. \end{array}$$

Integration von $p(x,y)$ über Δ liefert mittels Transformation in (s,t)-Koordinaten

$$\begin{aligned} C_3 \ &= \ \iint\limits_{\Delta} p(x,y)\, d(x,y) \ = \ \iint\limits_{\Sigma} p(x(s,t),y(s,t)) \left|\frac{\partial(x,y)}{\partial(s,t)}\right| d(s,t) \\ &= \ \int_0^1 \int_0^{1-s} (f_0 B_0(x,y) + f_1 B_1(x,y) + f_2 B_2(x,y)) \begin{vmatrix} h_x & 0 \\ 0 & h_y \end{vmatrix} dtds \\ &= \ h_x h_y \int_0^1 \int_0^{1-s} (f_0(1-s-t) + f_1 s + f_2 t)\ dtds, \end{aligned}$$

woraus wir die als *Prismenregel* über dem Dreieck bezeichnete Kubaturformel

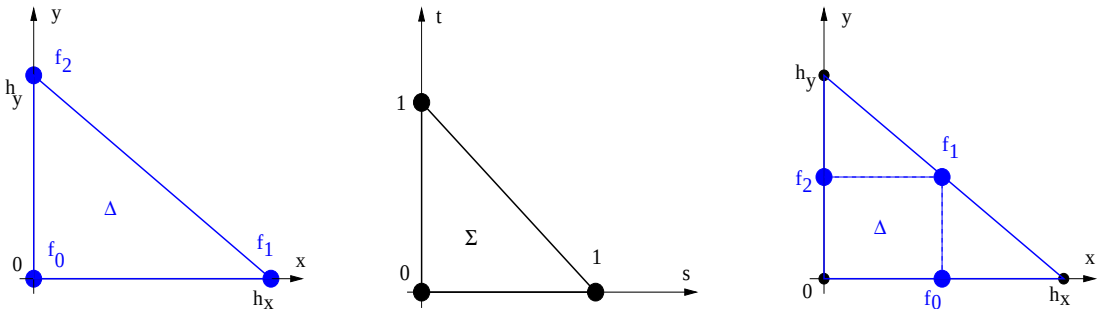
$$P_\Delta \ = \ C_3 \ = \ \frac{h_x h_y}{6}\,[f(0,0) + f(h_x,0) + f(0,h_y)] \tag{64}$$

erhalten. Der Inhalt $|\Delta| = \frac{1}{2} h_x h_y$ des Elementarbereiches wird mit dem arithmetischen Mittel der 3 Eckwerte multipliziert und beschreibt das Volumen des Prismas, das von der durch die 3 Eckpunkte verlaufenden Ebene begrenzt wird. □

Wegen des linearen Ansatzes ist der Genauigkeitsgrad dieser Regel $p = 1$. Durch Taylor-Abgleich gewinnen wir den lokalen Kubaturfehler

$$F_P \ := \ I - P_\Delta \ = \ -\frac{h_x h_y}{24}\left\{f_{xx}(0,0)h_x^2 + f_{yy}(0,0)h_y^2 - f_{xy}(0,0)h_x h_y + \mathcal{O}(h^3)\right\}, \tag{65}$$

womit sich wie bei Rechteckbereichen $|F_P| \le C_P h^4,\ C_P > 0$, und die lokale Fehlerordnung $q = 4$ ergibt.



(a) 3-Punkt-Formel 1 (b) Referenzdreieck (c) 3-Punkt-Formel 2

Abbildung 20: Dreieckige Elementarbereiche

Beispiel 26 Wählen wir als Knoten wie in Abb. 20 (c) die 3 Seitenmittelpunkte $(h_x, 0)$, $(0, h_y/2)$ und $(h_x/2, h_y/2)$, so ergibt sich mit einem quadratischen Interpolationsansatz die *Seiten-Mittelpunktregel* über dem Dreieck

$$M_\Delta \;=\; \frac{h_x h_y}{6}\left[f(h_x,0) + f(h_x/2, h_y/2) + f(0,h_y)\right], \tag{66}$$

die ebenfalls ein Prismenvolumen darstellt. Den Kubaturfehler erhalten wir durch

$$\begin{aligned} F_M \;:=\; I - M_\Delta \;=\; -\frac{h_x h_y}{1440}\big\{&3f_{xxy}(0,0)h_x^2 h_y + 3f_{xyy}(0,0)h_x h_y^2 \\ &-7f_{xxx}(0,0)h_x^3 - 7f_{yyy}(0,0)h_y^3 + \mathcal{O}(h^4)\big\}, \end{aligned} \tag{67}$$

womit diese 3-Punkt-Formel die höhere lokale Ordnung 5 und wegen des Fehlens von 2. Ableitungen den algebraischen Genauigkeitsgrad 2 besitzt! □

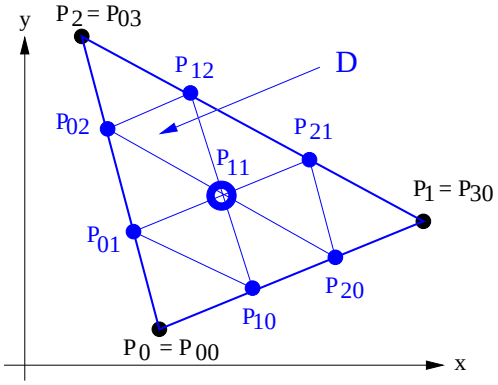


Abbildung 21: Bereich zu $\boldsymbol{P_D}$

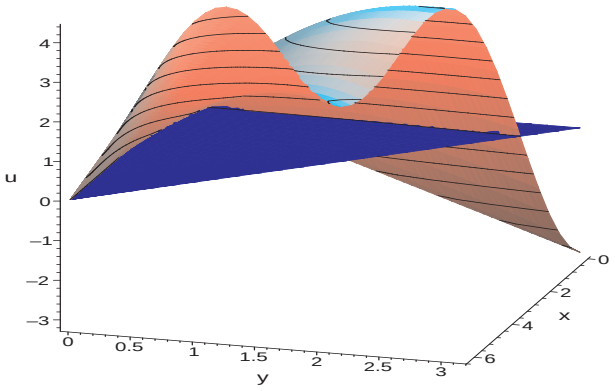


Abbildung 22: Beispiel 29(1)

Zusammengesetzte Kubatur über Dreieckbereichen Ist der Integrationsbereich D ein Dreieck mit den Eckpunkten P_0, P_1, P_2 wie in Abb. 21 dargestellt, so lässt sich eine Zerlegung in kongruente Teildreiecke Δ_i leicht bewerkstelligen, indem wir jede Seite äquidistant in m Intervalle zerlegen und die Teilpunkte seitenparallel verbinden. Da sich jeder Punkt von D als konvexe Linearkombination der 3 Eckpunkte darstellen lässt, können alle Teilpunkte P_{kl} leicht mittels

$$P_{kl} \;=\; \frac{k}{m}P_1 + \frac{l}{m}P_2 + \frac{m-l-k}{m}P_0, \quad k = 0(1)m,\; l = 0(1)m-k$$

beschrieben werden. In Abb. 21 ist der Fall $m = 3$ dargestellt.

Beispiel 27 Wenden wir auf die m^2 Teildreiecke Δ_i die Prismenregel (64) an, so ergibt sich die *zusammengesetzte Prismenregel über Dreieckgebieten D*

$$P_D \;=\; \frac{|D|}{3m^2}\left[f(P_{00}) + f(P_{0m}) + f(P_{m0}) \;+\; 3\sum_{P_{kl}\in edge(D)} f(P_{kl}) \;+\; 6\sum_{P_{kl}\in int(D)} f(P_{kl})\right], \tag{68}$$

deren Eckpunkte (schwarz) mit dem Faktor 1 gewichtet werde. Die restlichen Seitenpunkte $P_{kl} \in S$ (blau) erhalten die Wichtung 3 und alle inneren Punkte $P_{kl} \in int(D)$, die zu je 6 Teildreiecken gehören, den Faktor 6 (vgl. Abb. 21). □

Den Inhalt des Dreieckbereiches berechnen wir mit der Flächenformel (vgl. [4], Satz 9.17)

$$|D| \;=\; \frac{1}{2}\,|(x_0 - x_1)(y_0 + y_1) + (x_1 - x_2)(y_1 + y_2) + (x_2 - x_0)(y_2 + y_0)|\,, \tag{69}$$

wobei $P_i = (x_i, y_i)$, $i = 0, 1, 2,$ ist, und gewinnen so den Algorithmus 28.

Beispiel 29 Wir bestimmen das Integral (vgl. Abb. 22) über dem Dreieckbereich

$$I = \int_0^{2\pi}\int_0^{\pi - x/2} (x\sin y - y\cos 2x)\; dy\, dx \;=\; 11.346\;509\;720\;479\;993...$$

mit der Prismenregel gemäß Algorithmus 28. Die Eckpunkte $P_0 = (0, 0), P_1 = (2\pi, 0)$, $P_2 = (0, \pi)$ liefern bei den Unterteilungen m die Näherungswerte P_D in Tabelle 10 mit den absoluten Fehlern $I - T_D$. Da an den Fehlerquotienten Q auch bei dieser Formel die

m	P_D	$I - P_D$	Q
25	11.320429935	2.608E-2	—
50	11.340009328	6.500E-3	4.0123
100	11.344885833	1.623E-3	4.0049
200	11.346103824	4.059E-4	3.9985
400	11.346408251	1.015E-4	3.9990
800	11.346484353	2.537E-5	4.0008
1600	11.346503378	6.342E-6	4.0003
3200	11.346508135	1.585E-6	4.0013

Tabelle 10: Genauigkeit der zusammengesetzten Prismenregel

Ordnung 2 sehr gut erkennbar ist, sollte dies vom Leser allgemein verifiziert werden. □

Gauß-Legendre-Kubaturformeln Komplizierter wird die Herleitung von Gauß-Formeln, falls der Referenzbereich kein Intervall ist. Mehrpunktformeln hoher Ordnung für Dreieckbereiche findet man z.B. in [2].

Algorithmus 28 (Zusammengesetzte Prismenregel)

```
function I = prisma2d(fname,P0,P1,P2,m);
% Eingangsparameter:
% fname ist ein Handle auf den Integranden f,
% P0 = [x0,y0]' ist der Punkt P_0 (Spaltenvektor),
% P1 = [x1,y1]' ist der Punkt P_1 (Spaltenvektor),
% P2 = [x2,y2]' ist der Punkt P_2 (Spaltenvektor),
% m ist die Zahl der Teilintervalle jeder Dreieckseite
%
I = 0; n = m*m;     % Zahl der Teilintervalle
D = abs(0.5*((P0(1)-P1(1))*(P0(2)+P1(2))...
         +(P1(1)-P2(1))*(P1(2)+P2(2))...
         +(P2(1)-P0(1))*(P2(2)+P0(2))));
for k=0:m,
   for l=0:m-k,
      P = (k*P1 + l*P2 + (m-l-k)*P0)./m;
      fkl = feval(fname,P(1),P(2));
      if ((k==0)&(l==0))|((k==0)&(l==m))|((k==m)&(l==0))
         I = I + fkl./6;
      elseif (k==0)|(l==0)|(k+l==m)
         I = I + 0.5*fkl;
      else
         I = I +fkl;
      end
   end
end
I = 2*I*D/n;     % Integralnäherung I
```

Beispiel 30 Wir setzen auf dem Bereich Δ der Abb. 19 die 1-Punkt-Gauß-Kubaturformel mit $G_1 = \frac{h^2}{2} w_1 f(x_1, y_1)$ an und führen den Taylor-Abgleich mit der Entwicklung

$$f(x,y) = f(0,0) + f_x(0,0)x + f_y(0,0)y + \frac{1}{2} f_{xx}(0,0)x^2 + f_{xy}(0,0)xy + \frac{1}{2} f_{yy}(0,0)y^2 + \ldots$$

durch. Die Lösung $w_1 = 1$, $a = b = 1/3$ ergibt die *Schwerpunktregel (1-Punkt-Prismenregel)*

$$G_1 = \frac{h^2}{2} f\left(\frac{h}{3}, \frac{h}{3}\right) \quad \text{mit} \quad I - G_1 = \frac{h^4}{72} \left\{ f_{xx}(0,0) + f_{yy}(0,0) - f_{xy}(0,0) \right\} + \mathcal{O}(h^5). \tag{70}$$

Im Vergleich mit der Prismenregel (64) stimmen die lokalen Fehlerordnungen $q = 4$ und damit die globalen Ordnungen 2 beider zusammengesetzter Formeln überein. □

2.4 Adaptive Kubatur

Das Versagen zusammengesetzter Kubaturformeln an Beispiel 23 erfordert, wie bereits in Band 1 für die Quadratur dargestellt, eine *adaptive Kubatur*, d.h. eine automatische lokale Anpassung der Unterteilung des Grundbereiches D, bis eine geforderte Genauigkeit *tol* erreicht wird.

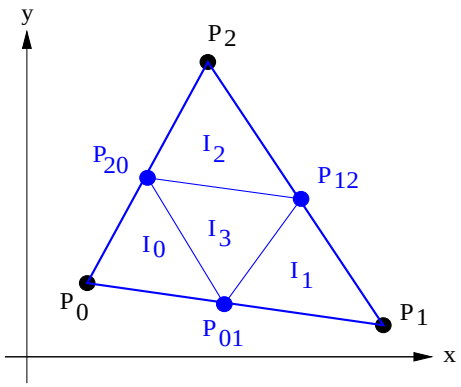


Abbildung 23: Zerlegung von Δ

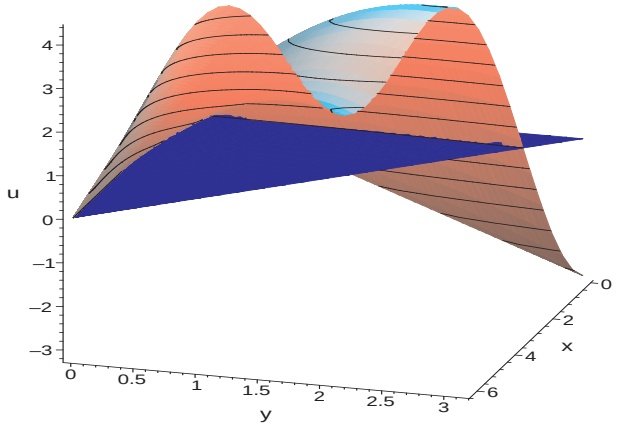


Abbildung 24: Beispiel 29(1)

Kubatur mit Dreieckbereichen Wir zerlegen D in elementare Dreiecke mittels einer *Triangulierung* und stellen uns die Aufgabe, das Integral I über einem Dreieckbereich Δ mit den Eckpunkten P_0, P_1, P_2 und der maximalen Seitenlänge h bis auf vorgegebene Genauigkeit *tol* mittels der *Prismenregel*

$$P_\Delta(h) \;=\; \frac{|\Delta|}{3}\,[f(P_0)+f(P_1)+f(P_2)] \tag{71}$$

zu berechnen. Um eine Fehlerschätzung zu gewinnen, verschaffen wir uns neben diesem „Grobwert“ einen „Feinwert“ $P_\Delta\left(\frac{h}{2}\right)$, indem wir die Seiten in Abb. 23 halbieren und die Prismenregel auf die vier entstehenden kongruenten Teildreiecke anwenden: $P_\Delta\left(\frac{h}{2}\right) = P_{\Delta_0}+P_{\Delta_1}+P_{\Delta_2}+P_{\Delta_3}$. Damit lässt sich der Fehler des Feinwertes mittels der asymptotischen Fehlerschätzung

$$Error \;:=\; \frac{1}{3}\,[P_\Delta(h/2)-P_\Delta(h)] \tag{72}$$

approximieren. Ist $|Error| < tol(|P_\Delta(h/2)|+1)$, so wird $P_\Delta(h/2)$ als ein geeigneter Näherungswert akzeptiert. Andernfalls führen wir für die 4 Teildreiecke dieselbe Rechnung durch usw. Die berechneten Prismenwerte $P_{\Delta_0}, P_{\Delta_1}, P_{\Delta_2}, P_{\Delta_3}$ stehen nun als Grobwerte zur Verfügung, so dass einmal berechnete Näherungswerte weiter verwendet werden können. Wegen der Kongruenzeigenschaft muss die Dreieckfläche nur einmal am Verfahrensstart berechnet werden. Zudem können die Dreiecke nicht entarten, denn sie behalten ihre Anfangsform. Sämtliche Feinwerte der einzelnen Teildreiecke werden aufaddiert und liefern den gewünschten Näherungswert des Integrals I. Eine rekursive Formulierung [4] der *adaptiven Prismenregel* findet man in Algorithmus 31.

[4] Die vorgestellte Implementierung in MATLAB ist ineffizient; eine C/C++-Funktion wird empfohlen.

Algorithmus 31 (Adaptive Prismenregel)

```
function [] = dreieck(fname,P0,P1,P2,f0,f1,f2,P,Delta);
% Eingangsparameter:
% fname ist ein Handle auf den Integranden
% P0 = [x0,y0]' ist der Punkt P0 (Spaltenvektor)
% P1 = [x1,y1]' ist der Punkt P1 (Spaltenvektor)
% P2 = [x2,y2]' ist der Punkt P2 (Spaltenvektor)
% f0,f1,f2 sind die Funktionswerte an P0,P1,P2
% P ist ein grober Prismenwert
% Delta ist die Fläche des Dreiecks P0-P1-P2
global I;     % Summe der Teilintegrale
global TOL;   % Toleranz
global K;     % Zahl der Funktionswert-Berechnungen
%
K = K + 3;
% Seiten-Halbierung
P01 = 0.5*(P0+P1); f01 = feval(fname,P01(1),P01(2));
P12 = 0.5*(P1+P2); f12 = feval(fname,P12(1),P12(2));
P20 = 0.5*(P2+P0); f20 = feval(fname,P20(1),P20(2));
Delta = 0.25*Delta; Delta3 = Delta/3;
% Feinwerte-Bestimmung
I0 = Delta3 * (f0 +f01+f20);
I1 = Delta3 * (f1 +f12+f01);
I2 = Delta3 * (f2 +f20+f12);
I3 = Delta3 * (f01+f12+f20);
Pfein = I0 + I1 + I2 + I3;
% Abbruchtest und Summation
if abs(Pfein-P)/3 < TOL*(abs(Pfein)+1),
   I = I + Pfein;
else
   dreieck(fname,P0, P01,P20,f0, f01,f20,I0,Delta);
   dreieck(fname,P1, P12,P01,f1, f12,f01,I1,Delta);
   dreieck(fname,P2, P20,P12,f2, f20,f12,I2,Delta);
   dreieck(fname,P01,P12,P20,f01,f12,f20,I3,Delta);
end
```

Das Abbruchkriterium führt einen kombinierten Relativ-Absolut-Fehlertest durch, der den kleiner werdenden Teilintegralen bei sukzessiver Unterteilung Rechnung trägt. Der Algorithmus kann in analoger Weise auch auf genaueren Newton-Cotes- oder Gauß-Legendre-Formeln aufbauen.

Beispiel 32 1. Für das Beispiel 29 mit dem Integranden in Abb. 24 sind 2 Triangulierungen (vgl. Abb. 25 und 26) dargestellt. Die mit mittlerem Aufwand gewonnenen Werte *int* findet man in Tabelle 11.

2. Das schwere Beispiel 23(2) bewältigt der Algorithmus – allerdings mit großem Aufwand.

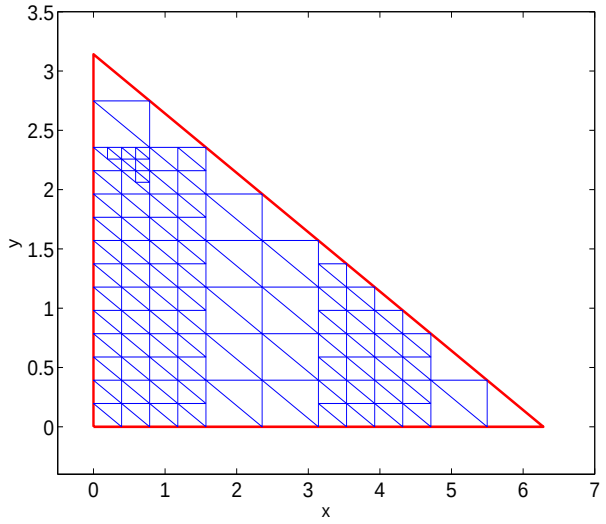


Abbildung 25: $tol = 1E-2$

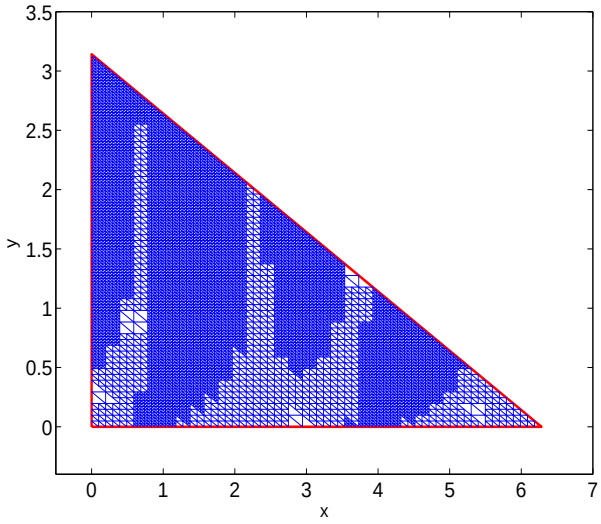


Abbildung 26: $tol = 1E-5$

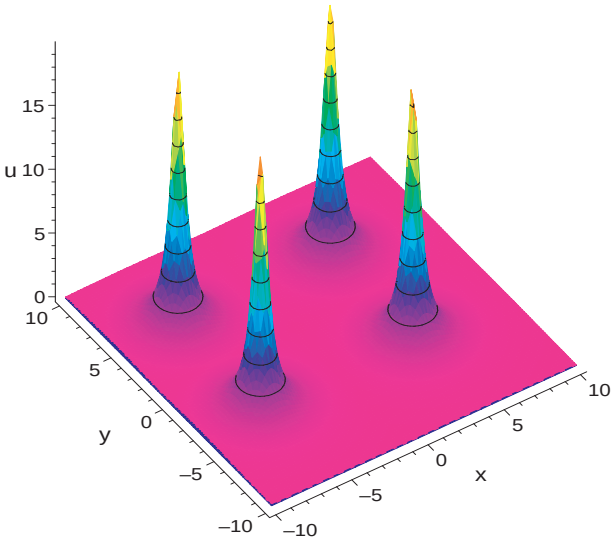


Abbildung 27: Beispiel 32(3)

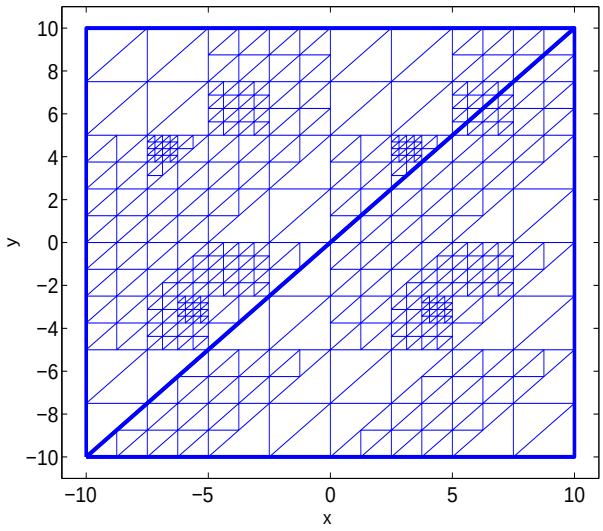


Abbildung 28: $tol = 1E-1$

Der Bereich D wird mittels der Winkelhalbierenden $y = x$ in die beiden Dreiecke D_1 und D_2 zerlegt. Mit 477 678 $< 700^2$ Funktionswerten wird so in Tabelle 11 ein wesentlich besserer Näherungswert ermittelt als die Werte in der Tabelle der Abb. 17.

3. Der Integrand der Aufgabe (vgl. Abb. 27)

$$I = \int_{-10}^{10}\int_{-10}^{10} \left(\frac{1}{20} + \frac{1}{4}(|x+1|-5)^2 + \frac{1}{4}(|y-1|-4)^2 \right)^{-1} dx\,dy = 251.0752677094...$$

hat 4 schmale „Peaks“, während er im Allgemeinen nahe bei Null liegt. Die adaptive Prismenregel liefert mit 351 666 Werten akzeptable Resultate (vgl. Tabelle 11). Entsprechende automatische Triangulierungen sind in den Abbildungen 28–30 dargestellt. □

Mehrdimensionale Normalbereiche Wir wollen nun das d-dimensionale Riemann-Integral

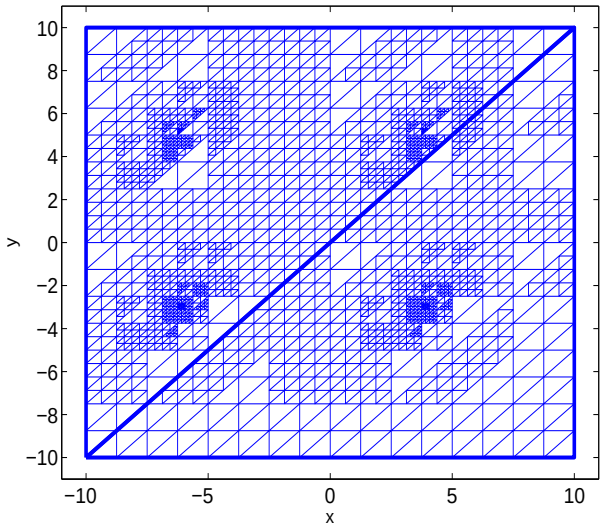


Abbildung 29: $tol = 1E-2$

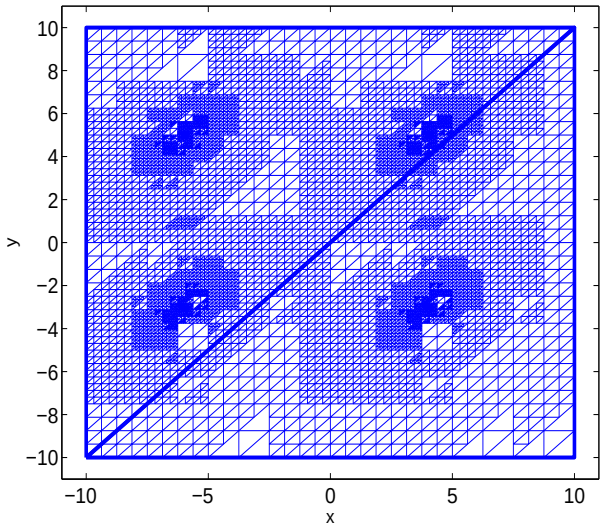


Abbildung 30: $tol = 1E-3$

Beispiel Nr.	Toleranz *tol*	Integral *int*	Rel. Fehler *relerror*	Anzahl *anz*
29	1E-2	11.279398	-5.91E-3	183
	1E-5	11.345506	-8.90E-5	12 315
	1E-8	11.346455	-5.26E-6	369 615
23(2)	1E-1	4.971088E5	1.97E-2	477 678
	1E-2	4.934915E5	2.68E-2	1 465 422
	1E-3	5.073616E5	-4.84E-4	10 835 382
32(3)	1E-1	238.800	4.89E-2	654
	1E-2	252.475	-5.57E-3	2 934
	1E-3	251.850	-3.08E-3	11 190
	1E-5	251.126	-2.02E-4	114 438
	1E-6	251.091	-6.26E-5	351 666

Tabelle 11: Adaptive Prismenregel gemäß Algorithmus 31

(38) einer Funktion $f : D \subset \mathbb{R}^d \to \mathbb{R}$ auf d-dimensionalen kartesischen Normalbereichen

$$I = \int_{a_1}^{b_1} \int_{a_2(x_1)}^{b_2(x_1)} \int_{a_3(x_1,x_2)}^{b_3(x_1,x_2)} \cdots \int_{a_d(x_1,\ldots,x_{d-1})}^{b_d(x_1,\ldots,x_{d-1})} f(x_1, x_2, \ldots, x_d)\, dx_d \ldots dx_3\, dx_2\, dx_1 \tag{73}$$

bestimmen. Die Variablen $x_1, x_2, \ldots, x_d$ müssen dabei so angeordnet werden, dass die Integrationsgrenzen a_k und b_k der Variablen x_k nur von den Variablen $x_1, x_2, \ldots, x_{k-1}$ abhängen und im Falle von x_1 konstant sind. Wir betrachten I als *iteriertes Integral* und führen die Funktionen $f_k : \mathbb{R}^k \to \mathbb{R}$ rekursiv ein:

$$
\begin{aligned}
I &= \int_{a_1}^{b_1} f_1(x_1)\,dx_1 \\
f_1(x_1) &= \int_{a_2(x_1)}^{b_2(x_1)} f_2(x_1,x_2)\,dx_2 \\
f_2(x_1,x_2) &= \int_{a_3(x_1,x_2)}^{b_3(x_1,x_2)} f_3(x_1,x_2,x_3)\,dx_3 \\
\ldots &\quad\ldots\qquad\qquad \ldots \\
f_{d-1}(x_1,\ldots,x_{d-1}) &= \int_{a_d(x_1,\ldots,x_{d-1})}^{b_d(x_1,\ldots,x_{d-1})} f(x_1,x_2,\ldots,x_d)\,dx_d.
\end{aligned}
\tag{74}
$$

Die so erhaltenen geschachtelten Integrale (engl.: nested integrals) können nun mit adaptiven Quadraturformeln approximiert werden. Zur Ermittlung von I mit einer geeigneten Quadraturformel benötigen wir z.B. den Wert $f_1(x_1)$ für ein festes Argument x_1. Deshalb setzen wir diesen Zahlenwert x_1 in die 2. Gleichung von (74) ein und erhalten wiederum ein Quadraturproblem, nun jedoch mit der Integrationsvariablen x_2 usw. Schließlich wird $f_{d-1}(x_1,\ldots,x_{d-1})$ für einen festen Argumentsatz $(x_1,\ldots,x_{d-1})$ aus der letzten Gleichung von (74) mittels adaptiver Quadratur bezüglich x_d bestimmt. Dieser rekursive Ansatz ist bei hohen Dimensionen d zeitaufwändig, weshalb die mit wenigen Knoten arbeitenden hoch genauen Gauß-Legendre-Formeln vorzuziehen sind.

MAPLE [6] gestattet die numerische Bestimmung von d-fachen Integralen (73) mit vorgebbarer Genauigkeit. Die Syntax des geschachtelten Aufrufes lautet

```
evalf(Int(Int(...(Int(f, xd=ad..bd),...),x2=a2..b2),x1=a1..b1,options))
```

mit dem Integranden `f=f(x1,x2,...,xd)` und den Integrationsgrenzen `ak` und `bk` entsprechend Darstellung (73). Gleichwertig hierzu ist die Notation mittels einer Liste

```
evalf(Int(f, [xd = ad..bd,...,x2 = a2..b2,x1 = a1..b1],options))
```

unter Beachtung der Reihenfolge der Variablen. Folgende Optionen sind zulässig:

1. `digits = <posint>` legt die Zahl der gewünschten Dezimalstellen fest. Setzt man `<posint>` $\leq$ `evalhf(Digits)`, so werden die compilierten Verfahren der NAG-Bibliothek [8] in Hardware-Float genutzt. Für höhere Genauigkeitsforderungen setzt MAPLE eigene Verfahren ein. Intern wird mit Schutzstellen gerechnet, um die geforderte Genauigkeit zu garantieren.
2. `epsilon = <numeric>` ist eine relative Fehlerschranke, die eine gröbere Genauigkeit zulässt. Standardmäßig wird `epsilon=0.5*10^(1-digits)` gesetzt.
3. `method = <name>` legt eine Kubatur-/Quadraturmethode fest. Fehlt diese Option, so wendet MAPLE eine als „Polyalgorithmus“ bezeichnete interne Strategie an, um das Integral numerisch zu approximieren.

Beispiel 33 1. Für das einfache Beispiel 29 über dem Dreieckbereich

$$I = \int_0^{2\pi} \int_0^{\pi - x/2} (x \sin y - y \cos 2x)\, dy\, dx \;=\; 11.346509720479993...$$

liefert der MAPLE-Aufruf

```
> evalf(Int(Int(f, y=0 .. Pi-x/2),x=0 .. 2*Pi,digits = 70));
```

den Wert

```
> 11.346509720479993082861151576842364410102752639559693025204830678401 61
```

mit relativem Fehler $0.2286 \cdot 10^{-69}$ in nicht messbar kurzer Zeit.
2. Für das „harte“ Beispiel 23(2) sollte `Digits:=14` (Hardware-Float) gesetzt werden.

$$I = \int_{-100}^{100} \int_{-100}^{100} \left(\frac{1}{x^4 + 10^{-4}} + \frac{1}{y^2 + 10^{-4}} \right) dx\, dy \;=\; 507\ 116.146\ 754\ 31...$$

Der Aufruf

```
> evalf(Int(Int(f,y=-100..100),x=-100..100,digits=10));
```

ergab nach 0.931 Sekunden Rechenzeit [5] den Wert

```
> 507116.1468
```

mit relativem Fehler $-0.9 \cdot 10^{-10}$. □

Folgende erprobte Methoden werden in MAPLE bereitgestellt (vgl. [2, 6, 8, 10]):

`method = _cuhre`	–	Mehrfachintegrale der Dimension $d \in [2, 15]$ über Intervallen mittels adaptiver Kubatur hoher Ordnung [1]
`method = _d01ajc`	–	adaptive 10-Punkt-Gauß- und 21-Punkt-Kronrod-Formel, geeignet für oszillierende Integranden
`method = _d01akc`	–	adaptive 30-Punkt-Gauß- und 61-Punkt-Kronrod-Formel, geeignet für oszillierende Integranden
`method = _d01amc`	–	Methoden für unendliche und halbunendliche Intervalle
`method = _NCrule`	–	adaptive Newton-Cotes-Formel konstanter Ordnung
`method = _Gquad`	–	adaptive Gauß-Legendre-Formeln
`method = _CCquad`	–	Clenshaw-Curtis-Quadraturformeln
`method = _Dexp`	–	adaptive Doppel-Exponentialmethode
`method = _Sinc`	–	adaptive sinc-Quadraturformeln
`method = _MonteCarlo`	–	Monte Carlo-Methode über einem Intervall in $\mathbb{R}^d$, geeignet nur bei geringer Genauigkeitsforderung.

Beispiel 34 1. Wir approximieren das Dreifachintegral mit `Digits:=10` (Software-Float) und `Digits:=14` (Hardware-Float). Nur bei `epsilon` $\geq 10^{-Digits/2}$ arbeiteten die

$$\int_0^1 \int_0^{\sqrt{1-x^2}} \int_0^{\sqrt{1-x^2-y^2}} \frac{1}{x^2 + y^2 + (z - 2.0)^2}\, dz\, dy\, dx \;=\; 0.18787404...$$

[5] CPU Intel Pentium 4, 1.6 GHz.

Integratoren und lieferten die Zahlenwerte[6] in Tabelle 12.

`Digits`	Option `method`	Genauigkeit	Integralwert `int`	Zeit in sec
10	k. A.	k. A.	0.187 874 048 8	2.363
14	k. A.	`digits = 14`	0.187 874 048 753 80	1.803
14	`_Gquad`	`epsilon=1E-7`	0.187 874 048 660 01	17.886
14	`_CCquad`	`epsilon=1E-7`	0.187 874 046 455 67	0.510
14	`_d01ajc`	`epsilon=1E-7`	0.187 874 046 455 67	0.479
14	`_d01akc`	`epsilon=1E-7`	0.187 874 046 455 67	0.319

Tabelle 12: Näherungswerte für Beispiel 34

2. Mit der Bezeichnung $u := \frac{3}{2}\,\pi\,x_1\,x_2\,x_3\,x_4$ wird das Vierfachintegral über dem Einheitswürfel mit `Digits:=14` berechnet:

$$\int_0^1\int_0^1\int_0^1\int_0^1 \tfrac{3}{2}\,\pi\,(\cos(u) - 7\,u\sin(u) - 6\,u^2\cos(u) + u^3\sin(u))\,dx_1\,dx_2\,dx_3\,dx_4 \;=\; -1.$$

Der MAPLE-Aufruf

```
> wert := evalf(Int(f8,[x1=0..1,x2=0..1,x3=0..1,x4=0..1],epsilon=1E-7));
```

liefert nach 10.9 Sekunden das Ergebnis

```
> wert := -1.000 000 000 331 7
```

mit dem Fehler $-0.33 \cdot 10^{-9}$. Mit der (Standard-)Option `method=_cuhre` erhält man dasselbe Resultat.[7] □

3 Zusammenfassung

Die *numerische Integration (Quadratur)* von Funktionen $f : \mathbb{R} \to \mathbb{R}$ wird durch Zerlegung des Integrationsintervalles in Teilintervalle auf eine Standardaufgabe über einem Streifen der Breite h reduziert. Eine *einfache Integrationsformel* ist eine gewichtete Summe von $n+1$ Funktionswerten an den Knoten t_k mit den Integrationsgewichten $h\omega_k$. Durch den sogenannten Taylor–Abgleich entstehen *Newton–Cotes–Formeln* mit äquidistanten Knoten, z.B. die bekannte Trapezregel, die Simpson–Regel und die Milne–Regel der Fehlerordnungen 3,5 bzw. 7. Bei freier Wahl der $n+1$ Integrationsknoten lassen sich so auch *Gauß–Legendre–Formeln* maximaler Fehlerordnung $2n+3$ gewinnen. Nach Summation erhält man aus diesen einfachen Formeln *zusammengesetzte Integrationsformeln* für das Gesamtintervall.

Eine *asymptotische Fehlerschätzung* lässt sich durch Parallelrechnung mit den Schrittweiten h und $2h$ gewinnen. Gleichzeitig kann der Näherungswert $I(h)$ „extrapoliert“ werden; sukzes-

[6] k.A. - keine Angabe einer Option

[7] Der originale FORTRAN-Code DCUHRE in [1] umfasst über 5 000 Zeilen.

sive Wiederholung führt auf das *Romberg–Verfahren* beliebig hoher Konvergenzordnung. Für Integranden mit lokal stark schwankenden Ableitungswerten ist allerdings eine automatische Intervallunterteilung und *adaptive Quadratur* erforderlich.

Numerische Kubaturformeln approximieren das d-dimensionale Riemann-Integral durch eine gewichtete Summe von n Funktionswerten $f(x_k)$ an den Knoten x_k. Die Güte der Formeln wird durch deren Fehlerordnung q und den algebraischen Genauigkeitsgrad p charakterisiert. *Einfache Newton-Cotes- oder Gauß-Legendre-Kubaturformeln* auf kartesischen Referenzbereichen leitet man aus den entsprechenden Quadraturformeln mittels des dyadischen Produktes der Gewichte her. Durch Zerlegung des kompakten Intervalles D in Referenzbereiche gewinnt man daraus zusammengesetzte Kubaturformeln.

Über den praktisch bedeutsamen Dreieckbereichen D lassen sich ebenfalls einfache und zusammengesetzte Kubaturformeln vom Newton-Cotes- oder Gauß-Legendre-Typ konstruieren. Damit können kompliziertere Geometrien von D mittels Triangulierung numerisch approximiert werden. Adaptive Kubatur bedeutet die automatische Anpassung von Zerlegung bzw. Kubaturformel an den jeweiligen Integranden f und den Bereich D. Leistungsfähige adaptive Kubaturverfahren findet man in der Spezialliteratur [1], [8], [10], während MAPLE [6] erprobte Software bereitstellt.

Literatur

[1] Berntsen, J.; Espelid, T. O.; Genz, A.: *Algorithm 698 - DCUHRE: an adaptive multidimensional integration routine*, ACM Transactions on Math. Software (TOMS), Vol. 17, 1991, S. 437-456

[2] Engeln-Müllges, G.; Reutter, F.: *Numerik-Algorithmen mit ANSI C-Programmen*, Wissenschaftsverlag Mannheim 1993

[3] Hanke-Bourgeois, M.: *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, B.G. Teubner Stuttgart 2002

[4] Hoffmann, A.; Marx, B.; Vogt, W.: *Mathematik für Ingenieure 1. Lineare Algebra, Analysis – Theorie und Numerik*, Pearson Studium München 2005

[5] Maess, G.: *Vorlesungen über numerische Mathematik. Band 1 und 2*, Akademie-Verlag Berlin 1984

[6] MAPLE 9.50, Maplesoft, Waterloo Maple Inc., 1981-2004
`http://www.maplesoft.com`

[7] MATLAB – The Language of Technical Computing. Using MATLAB, Version 6, The Math Works Inc., 2000
`http://www.mathworks.com`

[8] NAGNews: Electronic Newsletter. The Numerical Algorithms Group Ltd. 2005
`http://www.nag.co.uk/Local/NAGNews`

[9] Quarteroni, A.; Sacco, R.; Saleri, F.: *Numerische Mathematik. Band 1 und 2*, Springer Verlag Berlin 2002

[10] Überhuber, C.: *Computer-Numerik. Band 1 und 2*, Springer-Verlag Berlin 1995