Numerik Projekt 1 – Aufgabe 1

Lukas Moser & Bernhard Kepka

WS 2017

1. Gauss-Quadratur auf $[-1,1]$

Die zur Gauß-Quadratur auf dem Intervall $[-1,1]$ mit der Gewichtsfunktion $w \equiv 1$ gehörenden (normierten) Orthogonalpolynome sind durch die Legendre-Polynome L_j gegeben. Letztere erfüllen die Rekursion

$$L_0(x) = 1, \quad L_1(x) = x \quad L_{n+1}(x) = xL_n(x) - \frac{n^2}{4n^2-1} L_{n-1}(x),\ n \in \mathbb{N}. \tag{1}$$

Da das gegeben Intervall und die Gewichtsfunktion symmetrisch sind, folgt für die $n+1$ Knoten (x_j) und Gewichte (α_j) (nach einem Übungsbeispiel) $x_j = -x_{n-j}$ beziehungsweise $\alpha_j = \alpha_{n-j}$ für $j = 0, \ldots, n$.
Zur konkreten Implementierung wurden zwei Weg verfolgt:

(i) Die Quadraturknoten wurden gemäß Satz 4.23 des Numerik-Skriptums über eine Eigenwertaufgabe und die Quadraturgewichte über entsprechende Eigenvektoren berechnet.

(ii) Über die rekursive Darstellung der Legendre-Polynome lassen sich lokale Beziehungen zwischen Nullstellen zweier Polynome L_n und L_{n+1} herleiten. Mittels bekannter Nullstellen des n-ten Polynomes und eines Bisektions- bzw. Newton-Verfahrens wurden die Nullstellen von L_{n+1} ermittelt.

In beiden Fällen können die Gewichte mit Hilfe der Nullstellen von L_{n+1} und der Legendre-Polynome $L_0, \ldots, L_n$ explizit angegeben werden.

1.1. Via Eigenwertaufgabe.

Mit Satz 4.23 und obiger 3-Term-Rekursion folgt (Bezeichnungen wie im Satz) $\gamma_n^2 = \frac{n^2}{4n^2-1}$ und $\beta_n = 0$, denn $L_{n+1}(X) = \det(IX - T)$. Damit hat die entsprechende Matrix T die Form

$$T = \begin{pmatrix} 0 & \gamma_1 & & & \\ \gamma_1 & 0 & \gamma_2 & & \\ & \gamma_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \gamma_n \\ & & & \gamma_n & 0 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}.$$

Die Nullstellen von L_{n+1} sind die Eigenwerte von T. Für die Gewichte gilt

$$\alpha_j = \left(\frac{(v_j)_1}{\|v_j\|_2} \right)^2 \int_{-1}^{1} w(x)dx = 2 \left(\frac{(v_j)_1}{\|v_j\|_2} \right)^2, \tag{2}$$

wobei $(v_j)_1$ die 1. Komponente eines Eigenvektors v_j von T ist.
Mit der internen Funktion `eig` von Matlab wurden nun die Eigenwerte bzw. Eigenvektoren berechnet und gemäß (2) die Quadraturen aufgestellt.

1.2. Direkte Knotenberechnung

Wegen der Symmetrie um den Ursprung genügt es die nichtnegativen Nullstellen zu betrachten. Für ungerades n ist 0 stets ein Quadraturknoten, also eine Nullstelle von L_n.
Wir nutzen nun folgende Eigenschaften zur besseren Bestimmung der Knoten.

- Für die Nullstellen der Legendre-Polynome $L_1, \ldots, L_n$ gilt: Zwischen zwei nichtnegativen Nullstellen von L_{n-1} befindet sich genau eine von L_n.

  Dies sieht man induktiv ein: für $n = 1, 2, 3$ gilt die Aussage, wenn man explizit die Nullstellen berechnet. Seien nun $x^{(n)}, y^{(n)}$ zwei Nullstellen von L_n. Nach Induktionsannahme befindet sich eine Nullstelle $z^{(n-1)}$ von L_{n-1} in $(x^{(n)}, y^{(n)})$. Da $z^{(n-1)}$ einfach ist, hat L_{n-1} einen Vorzeichenwechsel in diesem Intervall, $xL_n(x)$ hingegen nicht. Mit der 3-Term-Rekursion (1) sieht man, dass das Vorzeichen von L_{n+1} in einer Umgebung eines Randpunktes ($xL_n(x)$ verschwindet dort) sich ändert. Aus dem Zwischenwertsatz folgt, dass sich eine Nullstelle in $(x^{(n)}, y^{(n)})$ befindet. Da L_{n-1} genau einmal sein Signum wechselt, kann es nur genau eine sein (und eine mehrfache liegt auch nicht vor).

- Nun folgt: Zwischen der größten Nullstelle von L_{n-1} und dem Punkt 1 befindet sich (genau) eine weitere von L_n.

  Für $n = 1, 2, 3$ gilt dies. Seien $x_1^{(n-1)}, \ldots, x_k^{(n-1)}$ die positiven Nullstellen von L_{n-1} mit $k = \lfloor \frac{n}{2} \rfloor$. In jedem der Intervalle $(x_1^{(n-1)}, x_2^{(n-1)}), \ldots, (x_{k-1}^{(n-1)}, x_k^{(n-1)})$ befindet sich genau eine Nullstelle von L_n, insgesamt also $\lfloor \frac{n}{2} \rfloor - 1$. Ist n ungerade, so ist 0 eine weiter und wir haben $n-2$ Nullstellen in $(-1, 1)$ gefunden. Ebenso viele haben wir im Falle n gerade. Die letzten zwei müssen sich ebenfalls in diesem Intervall befinden. Da sie nicht mit $\pm x_k^{(n-1)}$ übereinstimmen (dies sieht man mit der Rekursion), befinden sie sich in den Intervallen $(-1, -x_k^{(n-1)})$ und $(x_k^{(n-1)}, 1)$. Wegen der Symmetrie folgt die Behauptung.

Darauf aufbauend wendet man bei bekannten nichtnegativen Nullstellen von L_n ein Bisektions- bzw. Newton-Verfahren auf zwei benachbarte Nullstellen an. Zusätzlich macht man selbiges mit dem Intervall $(x_k^{(n)}, 1)$, wobei $x_k^{(n)}$ die größte Nullstelle von L_n sei. Je nachdem, ob n gerade oder ungerade ist, muss man noch gleiches zwischen Null und der kleinsten Nullstelle durchführen oder Null selbst als Knoten hinzufügen.

Bei der durchgeführten Implementierung wurde folgendes Vorgehen realisiert:
Input: Funktion f inklusive erster sowie zweiter Ableitung, Intervallpunkte x_1, x_2, zwischen denen sich die (eindeutige) Nullstelle befindet und eine Toleranz TOL.

(1) **while** $((|f(x_1)| > TOL$ or $|x_1 - x_2| > TOL)$ and $f''(x_1)f''(x_2) > 0)$ **do:**

(2) ein Durchlauf des Bisektions-Verfahrens mit x_1 und x_2 (Skriptum: Algorithmus 7.1);

(3) **end do;**

(4) **while** $(|f(x_t)| < TOL)$ **do:**

(5) ein Durchlauf der Newton-Iteration mit Startwert x_1 (Skriptum: Algorithmus 7.12, allerdings mit anderem Abbruchkriterium);

(6) **end do;**

Die Bedingung an die zweiten Ableitungen in den Eckpunkten ist heuristisch und wurde gefordert, in der Hoffnung die Nullstelle befinde sich in einem Intervall, indem die erste Ableitung nicht verschwindet. Da die Nullstelle einfach ist, existiert so ein Intervall. Wenn die Krümmung des Graphen in diesem Bereich überall gleichartig ist und dadurch (möglicherweise) Extrema

nicht vorhanden sind, konvergiert das Newton-Verfahren, falls man sich bereits nahe genug an der Nullstelle befindet. In allen Tests mit einem Polynom-Grad $n \leq 50$ hat diese Heuristik tatsächlich zu einer Konvergenz geführt. Für große Wert handelte es sich bei den Grenzwerten in der Regel nicht mehr um die gewünschten Nullstellen. Ein Vergleich beider Implementierungen folgt im vierten Unterabschnitt.

1.3. Explizite Darstellung der Gewichte

Ein Eigenvektor v_k zum Eigenwert x_k von $T \in \mathbb{R}^{(n+1)\times(n+1)}$ hat die Form

$$(c_0 L_0(x_k), \ldots, c_n L_n(x_k))^T,$$

wobei c_j noch zu spezifizierende Konstanten sind. Letztere ergeben sich aus $Tv_k \stackrel{!}{=} x_k v_k$ durch komponentenweises Vergleichen. Das Wissen um diese Darstellung und die darauffolgende Berechnung der Gewichte wurden aus Schwarz, Köckler: *Numerische Mathematik*, Vieweg 2011, (S. 328f) entnommen. Für die erste Zeile gilt

$$\gamma_1 c_1 L_1(x_k) = \gamma_1 c_1 x_k \stackrel{!}{=} x_k c_0 L_0(x_k) = x_k c_0,$$

also $c_1 = c_0/\gamma_1$. Wir setzen $c_0 := 1$, damit dann $(v_k)_1 = 1$ erfüllt ist. In der i-te Zeile ist mit der Rekursion (1)

$$\begin{aligned} & \gamma_{i-1} c_{i-2} L_{i-2}(x_k) + \gamma_i c_i L_i(x_k) \stackrel{!}{=} x_k c_{i-1} L_{i-1}(x_k) \\ \Longrightarrow\ & \gamma_{i-1} c_{i-2} L_{i-2}(x_k) + \gamma_i c_i \left(x_k L_{i-1}(x_k) - \gamma_{i-1}^2 L_{i-2}(x_k) \right) \stackrel{!}{=} x_k c_{i-1} L_{i-1}(x_k) \\ \Longrightarrow\ & \left(\gamma_{i-1} c_{i-2} - \gamma_i c_i \gamma_{i-1}^2 \right) L_{i-2}(x_k) + \gamma_i c_i x_k L_{i-1}(x_k) \stackrel{!}{=} x_k c_{i-1} L_{i-1}(x_k). \end{aligned}$$

Wir haben folglich die Forderungen

$$\begin{aligned} \gamma_{i-1} c_{i-2} - \gamma_i c_i \gamma_{i-1}^2 &= 0 \\ \gamma_i c_i &= c_{i-1}. \end{aligned}$$

Wenn wir die letzte Gleichung als rekursive Definition von c_j nutzen, $c_i := c_{i-1}/\gamma_i$, gilt auch die erste:

$$\gamma_i \gamma_{i-1}^2 c_i = \gamma_{i-1}^2 c_{i-1} = \gamma_{i-1} c_{i-2}.$$

Da T symmetrisch ist, sind (v_k) zueinander orthogonal. Aus der Definition der Quadratur-Formel folgt dann

$$\sum_{i=0}^{n} \alpha_i c_k L_k(x_i) = Q^{(n)}(c_k L_k) = (c_0 L_0, c_k L_k)_w = 2\delta_{k0} \quad \text{für } k = 0, \ldots n.$$

Schreibt man diese Identitäten in einen Vektor und berücksichtigt $(v_k)_1 = 1$, so erhält man

$$\begin{aligned} 2 = v_k^T 2e_1 = v_k \left(\sum_{j=0}^{n} \alpha_j c_0 L_0(x_j), \ldots, \sum_{j=0}^{n} \alpha_j c_n L_n(x_j) \right)^T = \\ = v_k^T \sum_{j=0}^{n} \alpha_j v_j = \alpha_k v_k^T v_k = \alpha_k \sum_{j=0}^{n} c_j^2 L_j(x_k)^2. \end{aligned}$$

Für die Gewichte (α_j) gilt damit:

$$\alpha_j = \frac{2}{\sum_{j=0}^{n} c_j^2 L_j(x_k)^2} \qquad j = 0, \ldots, n. \tag{3}$$

1.4. Vergleich der zwei Implementierungen

Für die Gegenüberstellung wurden die Knoten auf $[-1,1]$ auf beide Arten berechnet. Mehr Aufschluss darüber gibt die folgende Tabelle 1, die einige Knoten mit 16 Stellen in den Fällen $n = 10, 30, 45$ für den Polynom-Grad wiedergibt.
In der zweiten Spalte befinden sich die Ergebnisse über das Eigenwertproblem, in der dritten über die direkte Berechnung und in der letzten die absoluten Differenzen davon. Dabei wurden je zwei mit dem kleinsten bzw. größten Unterschied ausgewählt. Da die Symmetrie ausgenutzt wurde, sind nur positive Nullstellen angegeben.

Grad	Via Eigenwertproblem	Direkte Methode	Absolute Differenz
$n = 10$	0.148874338981631	0.148874338981631	0.000000000000000
	0.433395394129247	0.433395394129247	0.000000000000000
	0.865063366688984	0.865063366688989	0.000000000000004
	0.973906528517172	0.973906528517170	0.000000000000002
$n = 30$	0.051471842555318	0.051471842555440	0.000000000000122
	0.153869913608583	0.153869913609074	0.000000000000491
	0.926200047429274	0.926200043360629	0.000000004068645
	0.983668123279747	0.983668127580494	0.000000004300747
$n = 45$	0.205647489783264	0.205647490371551	0.000000000588287
	0.338392654250603	0.338392668499781	0.000000014249178
	0.922163936719000	0.920917699952659	0.001246236766342
	0.981968715034541	0.963996589317694	0.017972125716846

Tabelle 1: Vergleich der Methoden: Knoten in den Fällen $n = 10, 30, 45$ und Differenzen.

Die Tabelle suggeriert, dass für niedrigen Grad die Knoten und damit auch die Quadraturen das selbe leisten. Für höhere Ordnung unterscheiden sie sich schon merkbar. Tatsächlich sind für $n \geq 45$ die Ergebnisse über die direkte Berechnung nicht mehr sinnvoll (sie befinden sich nicht in den betrachteten Intervallen). Die erste Implementierung liefert aber noch für größere Werte von n die Nullstellen der Legendre-Polynome.

Die Gegenüberstellung unterstreicht die schlechte Kondition, die das Berechnen von Nullstellen eines Polynoms birgt. Eine konkrete Fehlerquelle ist nach einer Beobachtung das Newton-Verfahren. Das Abbruchkriterium des Bisektions-Verfahrens, das Forderungen an die zweiten Ableitungen in den betrachteten Näherungen macht, führt nicht mehr zur Konvergenz gegen die gewünschte Nullstelle. Vielmehr verlässt der Algorithmus den betrachteten Bereich und konvergiert gegen eine (beliebige) andere Nullstelle. Dies passiert gerade in der Nähe der Eins, wo Knoten nahe beieinander liegen (also bis zur zweiten Nachkommastelle übereinstimmen). Dieses Verhalten lässt sich durch das starke Variieren der Ableitung zwischen solchen Nullstellen begründen. Um dies zu umgehen, könnte man nun nur das Bisektions-Verfahren verwenden. Allerdings steigt dabei die Laufzeit spürbar an, wenn nur der Grad n größer als 20 ist.

Zusammenfassend zeigt sich die erste Implementierung über das Eigenwertproblem unempfindlicher gegenüber der Höhe der Ordnung.

2. Quadratur auf $[a,b]$ und $[a,b]\times[c,d]$

Mit Hilfe der Transformation

$$\psi : [-1,1] \to [a,b] : \xi \mapsto a + \frac{b-a}{2} + \xi\frac{b-a}{2} = \frac{b+a}{2} + \xi\frac{b-a}{2}$$

wird die Quadratur-Formel aus dem ersten Kapitel auf $[a,b]$ übertragen (f stetig):

$$\int_a^b f(x)dx = \int_{-1}^1 f(\psi(\xi))\left(\frac{b-a}{2}\right)d\xi \approx \sum_j \left(\frac{b-a}{2}\right)\alpha_j f(\psi(x_j)).$$

Die Quadratur-Knoten $(\tilde{x}_j)$ bzw. Gewichte $(\tilde{\alpha}_j)$ sind also gegeben durch

$$\tilde{x}_j = \psi(x_j) = \frac{b+a}{2} + x_j\frac{b-a}{2} \qquad \tilde{\alpha}_j = \left(\frac{b-a}{2}\right)\alpha_j. \tag{4}$$

Seien nun zwei Quadraturen $Q^{(x)}$, $Q^{(y)}$ auf $[a,b]$ respektive auf $[c,d]$ mit Knoten (x_i), (y_j) und Gewichten (α_i), (β_j) gegeben. Auf $R := [a,b]\times[c,d]$ folgt mit Fubini

$$\int_c^d\int_a^b f(x,y)dxdy \approx \int_c^d \sum_{i=1}^{N_x}\alpha_j f(x_i,y)dy = \sum_{i=1}^{N_x}\alpha_j\int_c^d f(x_i,y)dy \approx \sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\alpha_i\beta_j f(x_i,y_j).$$

Jede einzelne Quadratur-Formel ist für Polynome vom Grad $2N_x+1$ bzw. $2N_y+1$ exakt. Wegen der Linearität ist die Quadratur auf R für den Funktionenraum

$$\left\{(x,y)\mapsto \sum_{j=0}^{2N_x+1}\sum_{k=0}^{2N_y+1} a_{jk}x^j y^k \mid a_{jk}\in\mathbb{R}\right\}$$

es ebenso.

2.1. Fehlerschranke

Nach Satz 4.18 des Numerik-Skriptums gilt die folgende (a priori) Darstellung für den Fehler einer Quadratur angewendet auf eine hinreichend of stetig differenzierbare Funktion f:

$$Q(f) - Q_n(f) = \frac{f^{(n+1)}(\xi)}{(2n+2)!}\int_a^b w(x)\prod_{j=0}^n (x-x_j)^2 dx.$$

Sie lässt sich durch Abschätzen vereinfachen zu

$$|Q(f) - Q_n(f)| \le \frac{\left\|f^{(n+1)}\right\|_\infty}{(2n+2)!}(b-a)^{2n+3}. \tag{5}$$

Genauso kann man das Integral abschätzen durch die Rechnung

$$\int_a^b \prod_{j=0}^n (x-x_j)^2 dx \le (b-a)^2\int_a^b \prod_{j=0,\ j\neq k}^n (x-x_j)^2 dx = (b-a)^2\alpha_k \prod_{j=0,\ j\neq k}^n (x_k-x_j)^2, \tag{6}$$

denn es liegt ein Polynom vom Grad $2n$ vor. Man wähle k so, dass die Differenzen möglichst klein sind.

2.2. Testen der Implementierung

Um die Ordnung der Quadraturen zu verifizieren, wurden Polynome unterschiedlichen Grades auf dem Intervall $[0,100]$ betrachtet. Die Anzahl der Knoten wurde dabei zwischen 1 und 22 variiert. Ein Graph mit den relativen Fehlern findet sich in Abbildung 1. Es ist ersichtlich, dass die von der Theorie her vorgegebene Genauigkeit erfüllt wird. (Denn die Ordnung beträgt $2n+2$ für $n+1$ Knoten.)

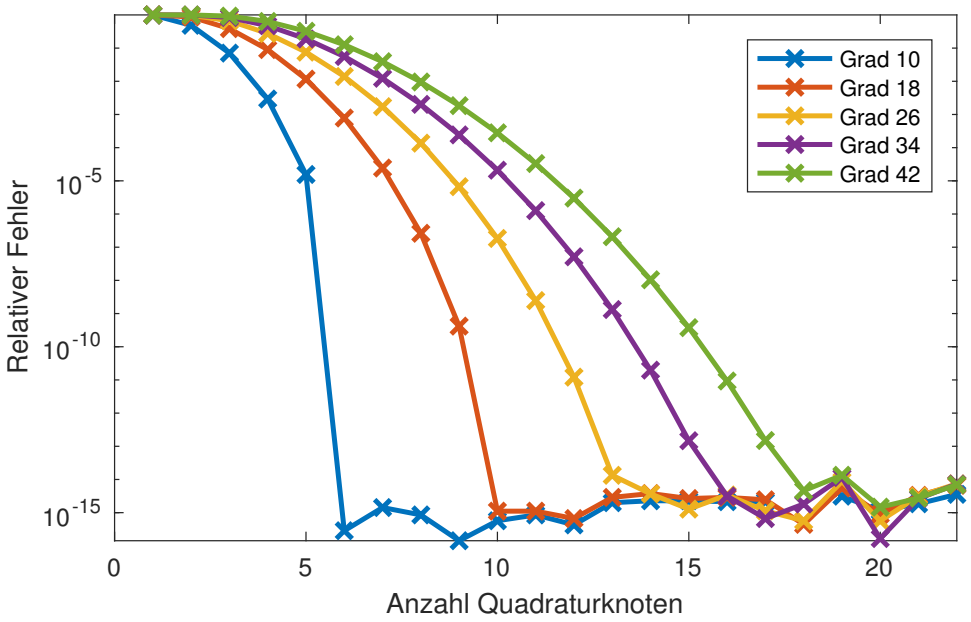

Abbildung 1: Relativer Fehler der Quadratur auf $[0,100]$ angewendet auf Polynome mit Grad zwischen 10 und 42 bei einer Knotenanzahl von 1 bis 22.

Um die Konvergenzrate zu betrachten, wurde die Exponentialfunktion auf dem Intervall $[-10,20]$ mit einer Knotenanzahl zwischen 1 und 25 integriert. Die Abbildung 2 zeigt den relativen Fehler sowie die a priori Fehlerschranke zu (5) und (6). Man erkennt, dass der Unterschied zu den tatsächlichen Ergebnissen in der Größenordnung 10^5 für die zweite Fehlerabschätzung liegt. Die erste Abschätzung weist dagegen einen viel größeren Abstand auf.

Zuletzt wurde ein Beispiel von Runge dazu verwendet, um die Genauigkeit zu untersuchen. Da die Quadraturen auf der Polynom-Interpolation basieren, ist zu erwarten, dass bei einer schlechten Approximation der Funktion auch große Fehler auftreten. Die hier betrachtete Runge-Funktion lautet $(1+x^2)^{-1}$ auf $[-20,20]$. Einen Graphen der Funktion und der Interpolations-Polynome mit den Quadratur-Knoten zur Knotenanzahl 11, 16, 31 (für das Intervall $[-20,20]$) als Stützstellen ist in Abbildung 3 zu finden. Man erkennt, dass die ungeraden Interpolations-Polynome stark um die Stützwerte schwingen. Für das Polynom mit ungeradem Grad, passiert dies (vor allem für höheren Grad) nur am Rand. Dafür ist im Nullpunkt der Fehler größer. Die Quadratur liefert für eine Knotenanzahl zwischen 1 und 50 Ergebnisse, die auf zwei Nachkommastellen genau sind, siehe Abbildung 4. (Man erkennt darüber hinaus, dass eine gerade Anzahl an Stützstellen bessere Werte liefert. Der Grund dafür ist das weniger starke Variieren ungerader Interpolations-Polynome bei der Approximation.) Für einen Fehler in der Größenordnung der Maschinengenauigkeit muss die Knotenanzahl größer als 350 sein. Dieses Resultat wird in der Theorie durch Satz 4.20 des Numerik-Skriptums sichergestellt. Obwohl der Fehler der Interpolation nicht gegen Null konvergiert (Ableitungen höherer Ordnung wachsen in der Nähe des Nullpunktes stark an), passiert das bei der Gauß-Quadratur sehr wohl (siehe Abbildung 5).

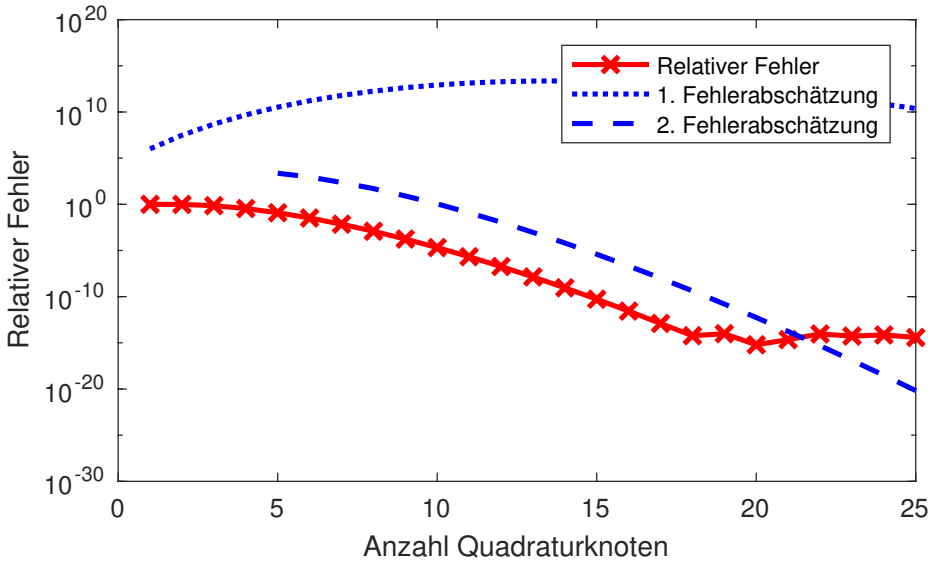

Abbildung 2: Relativer Fehler zur num. Integration von e^x über Anzahl der Knoten auf $[-10, 20]$. Die Fehlerabschätzungen wurden durch (5) bzw. (6) berechnet.

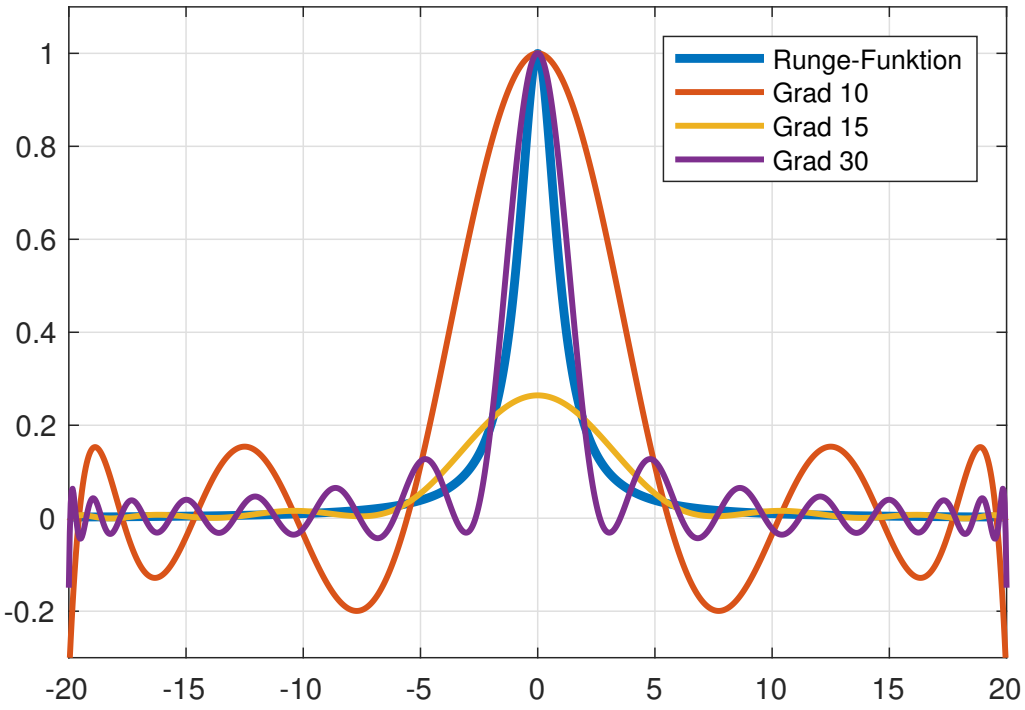

Abbildung 3: Runge-Funktion und zugehörige Interpolations-Polynome verschiedenen Grades mit entsprechenden Quadratur-Knoten als Stützstellen.

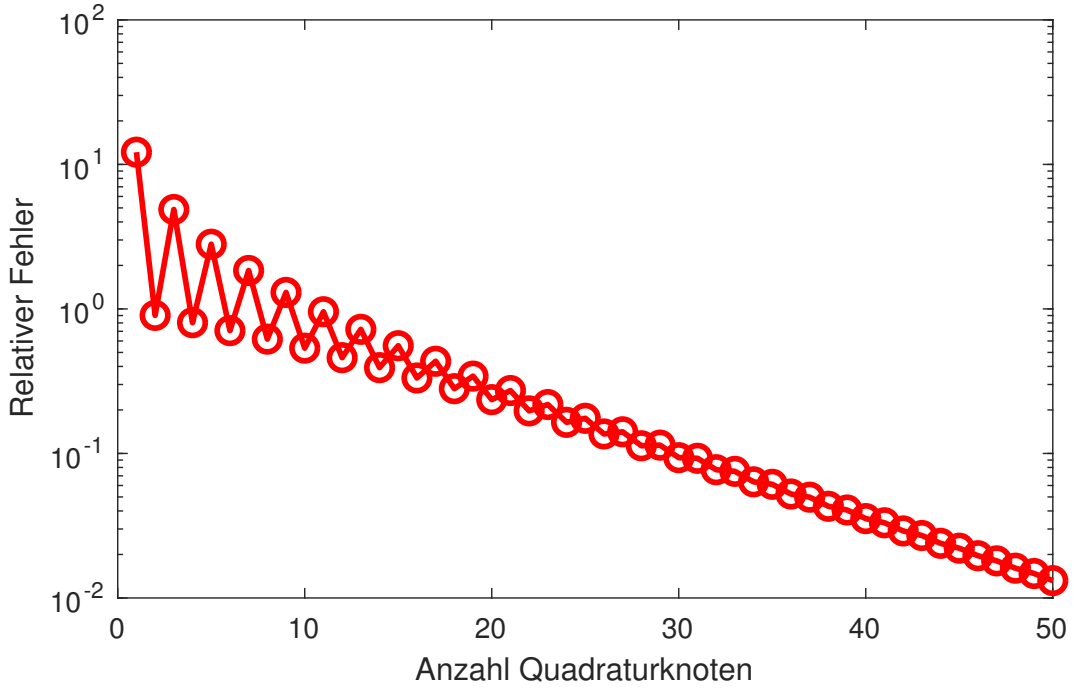

Abbildung 4: Relativer Fehler zur num. Integration der Runge-Funktion auf $[-20, 20]$ über Anzahl der Stützstellen.

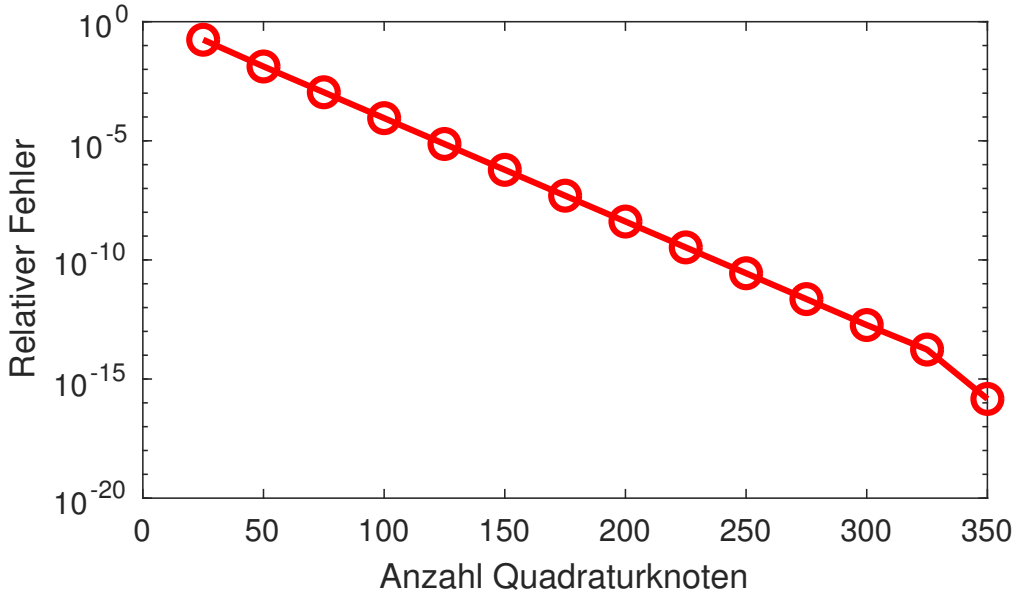

Abbildung 5: Relativer Fehler der Quadratur auf $[-20, 20]$ angewendet auf die Runge-Funktion über Anzahl der Stützstellen von 25 bis 350 (mit Schrittweite 25).

Auf dem Rechteck $[0,1]\times[0,5]$ wurden Polynome mit unterschiedlichem Grad in x bzw. y numerisch integriert. (Genauer wurde das Produkt von zwei Polynomen in x respektive in y betrachtet. In diesem Sinne ist auch der angegebene Grad zu verstehen.) Die Knotenanzahl wurde in beiden Koordinaten zwischen 1 und 22 gewählt. Die Ergebnisse sind in Abbildung 6 festgehalten. Man erkennt, dass der Fehler vom höchsten Grad abhängt und über der zu erwartenden Knotenanzahl in der Maschinengenauigkeit liegt. Zum Beispiel liegt der Fehler des ersten Polynoms (Grad 8×34, blau) in der Größenordnung 10^{-15} bei 17 Stützstellen (also Ordnung 34).

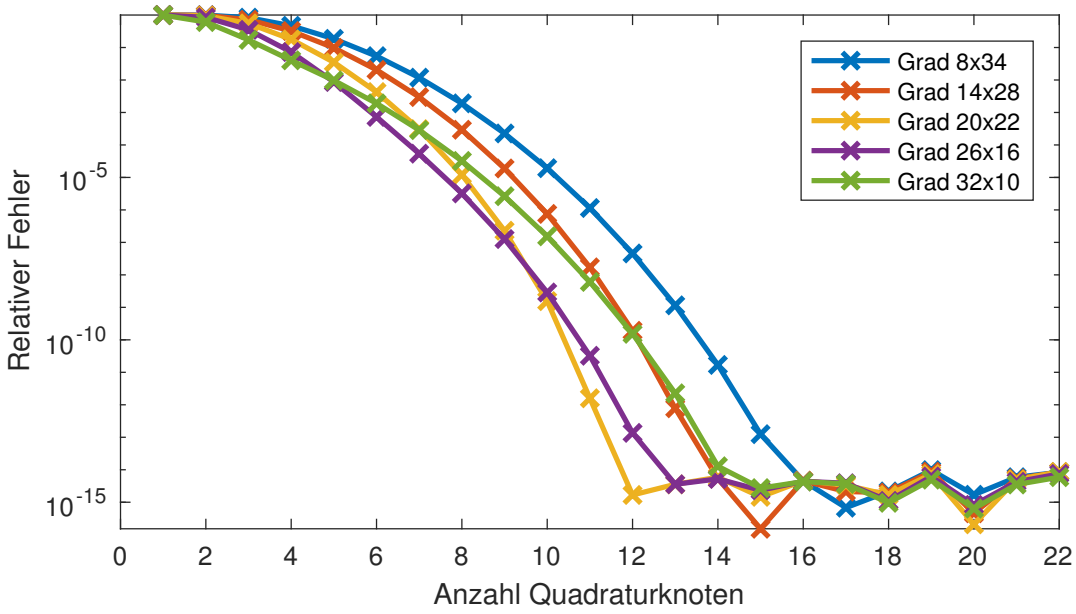

Abbildung 6: Relativer Fehler zur num. Integration von Polynomen mit zwei Variablen auf $[0,1]\times[0,5]$ über Anzahl von Quadraturknoten in beiden Koordinaten zwischen 1 und 22.

3. Quadratur auf $\hat{T}$

3.1. Interpolations-Quadratur

Sei das zweidimensionale Dreieck $\hat{T}=\mathrm{conv}\{e_0,e_1,e_2\}$ mit $e_0=(0,0)^T$, $e_1=(1,0)^T$, $e_2=(0,1)^T$ gegeben. Die Kantenmittelpunkte seien $k_1=(1/2,0)^T$, $k_2=(0,1/2)^T$, $k_2=(1/2,1/2)^T$ und

$$P_n := \left\{ \sum_{j=0}^{n} \sum_{k=0}^{n-j} a_{jk} x^j y^k \mid a_{jk} \in \mathbb{R} \right\}$$

der Funktionenraum der Polynome in x,y mit maximalem Grad n.
Um Quadraturen $Q^{(1)}$, $Q^{(2)}$ auf $\hat{T}$ zu definieren betrachten wir die Interpolationsaufgaben:

(a) Gesucht $p_1\in P_1$ mit $p_1(e_j)=f(e_j)$ für $j=0,1,2$.

(b) Gesucht $p_2\in P_2$ mit $p_2(e_j)=f(e_j)$ für $j=0,1,2$ und $p_2(k_j)=f(k_j)$ für $j=1,2,3$.

Beide Probleme lassen sich stets und eindeutig durch Basispolynome lösen.

Ad(a): Man wähle:

$$E_0(x,y) := 1-x-y, \quad E_1(x,y) := x, \quad E_2(x,y) := y.$$

Für diese Polynome gilt $E_j(e_k) = \delta_{jk}$. Die Lösung der Interpolation ist gegeben durch $p_1(x,y) = \sum_{j=0}^{2} E_j(x,y) f(e_j)$ und die Gewichte (α_j) der Quadratur-Formel $Q^{(1)}$

$$\alpha_j = \int_{\hat{T}} E_j(x,y) d(x,y) \quad \text{also} \quad \alpha_0 = \alpha_1 = \alpha_2 = \frac{1}{6}.$$

Ad(b) : Im selben Sinne ist mit

$$E_0(x,y) := 2x^2 + 2y^2 + 4xy - 3x - 3y + 1, \quad E_1(x,y) := 2x^2 - x, \quad E_2(x,y) := 2y^2 - y,$$
$$K_1(x,y) := -4x^2 - 4xy + 4x, \quad K_2(x,y) := -4y^2 - 4xy + 4y, \quad K_3(x,y) := 4xy$$

stets $E_j(e_k) = \delta_{jk}$ und $K_j(k_i) = \delta_{ji}$ erfüllt und damit das Interpolationsproblem stets unzweideutig lösbar. Die Gewichte (α_j) sind gegeben durch

$$\alpha_j = \int_{\hat{T}} E_j(x,y) d(x,y) \quad \text{bzw.} \quad = \int_{\hat{T}} K_j(x,y) d(x,y).$$

Es folgt $\alpha_0 = \alpha_1 = \alpha_2 = 0$ und $\alpha_3 = \alpha_4 = \alpha_5 = \frac{1}{6}$.

Die Quadratur-Formeln sind also

$$Q^{(1)}(f) = \frac{1}{6}\left(f(0,0) + f(1,0) + f(0,1)\right) \tag{7}$$
$$Q^{(2)}(f) = \frac{1}{6}\left(f(1/2,0) + f(0,1/2) + f(1/2,1/2)\right). \tag{8}$$

3.2. Via Duffy-Transformation

Mit der Duffy-Transformation

$$\Psi : [0,1]^2 \to \hat{T} : (\xi,\eta) \mapsto (\xi, (1-\xi)\eta)$$

lässt sich eine Quadratur auf $[0,1]^2$ auf dem Dreieck definieren. (Diese Transformation lässt die erste Koordinate invariant und die zweite wird entsprechend der Höhe des Dreiecks $\hat{T}$ gestaucht.) Mit der Transformationsformel folgt

$$\int_{\hat{T}=\Psi([0,1]^2)} f(x,y) d(x,y) = \int_{[0,1]^2} f(\xi,(1-\xi)\eta)(1-\xi) d(\xi,\eta) \quad \text{mit } |\det D\Psi| = (1-\xi).$$

Mit zwei gegebenen Quadraturen auf $[0,1]$ gilt mit Fubini:

$$\int_{\hat{T}} f(x,y) d(x,y) \approx \sum_{i=0}^{N_x} \sum_{k=0}^{N_y} \alpha_i \beta_k f(x_i, (1-x_i) y_k)(1-x_i).$$

Um die Ordnung der Quadratur auf $\hat{T}$ zu untersuchen, sei $p \in P_n$ also

$$p(\Psi(x,y))(1-x) = \sum_{k=0}^{n} \sum_{j=0}^{n-k} a_{jk} x^k (1-x)^{j+1} y^j.$$

In x hat p den Grad $k+j+1 = n+1$ und in y also n. Damit p exakt integriert wird, muss $n+1 \le 2N_x+1$ oder $N_x > \lfloor n/2 \rfloor$ und $N_y \ge \lfloor n/2 \rfloor$ erfüllt sein. Die Knotenanzahl muss also in der ersten Koordinate echt größer $\lfloor n/2 \rfloor + 1$ und in der zweiten größer oder gleich $\lfloor n/2 \rfloor + 1$ sein.

Die Duffy-Transformation lässt sich noch für das Dreieck

$$\tilde{T} := \operatorname{conv}\left\{(a,c)^T, (a,d)^T, (b,c)^T\right\}$$

mit $a < b, c < d$ anwenden durch die Variante:

$$\Psi : [a,b] \times [c,d] \to \tilde{T} : (\xi,\eta) \mapsto \left(\xi, \left(1 - \frac{\xi - a}{b-a}\right)\eta\right).$$

Man erhält dann

$$\int_{\tilde{T}} f(x,y)d(x,y) = \int_{[a,b]\times[c,d]} f\left(\xi, \left(1-\frac{\xi-a}{b-a}\right)\eta\right)\left(1-\frac{\xi-a}{b-a}\right)d(\xi,\eta)$$
$$\approx \sum_{i=0}^{N_x}\sum_{k=0}^{N_y} \alpha_i\beta_k f\left(x_i, \left(1-\frac{x_i-a}{b-a}\right)y_k\right)\left(1-\frac{x_i-a}{b-a}\right) \quad \text{mit } |\det D\Psi| = \left(1-\frac{\xi-a}{b-a}\right).$$

3.3. Testen der Implementierung

Auf dem Dreieck $\hat{T}$ wurden analog zum Rechteck Polynome in x und y integriert (also das Produkt zweier Polynome). Dabei wurde hier nur die Implementierung mit Hilfe der Duffy-Transformation betrachtet. Die Anzahl der Stützstellen variiert in beiden Koordinaten simultan von 1 bis 15 (siehe Abbildung 7). Die Ergebnisse entsprechen den Erwartungen. Zum Beispiel werden für das vorletzte Polynom (Grad 12×5, violett) 10 Stützstellen benötigt, um eine hohe Genauigkeit zu erreichen. Denn der größte Grad in x liegt bei $12+5+1 = 18$ (wegen der Nichtlinearität der Duffy-Transformation). Es werden daher $\lfloor 18/2 \rfloor + 1 = 10$ Knoten gebraucht.

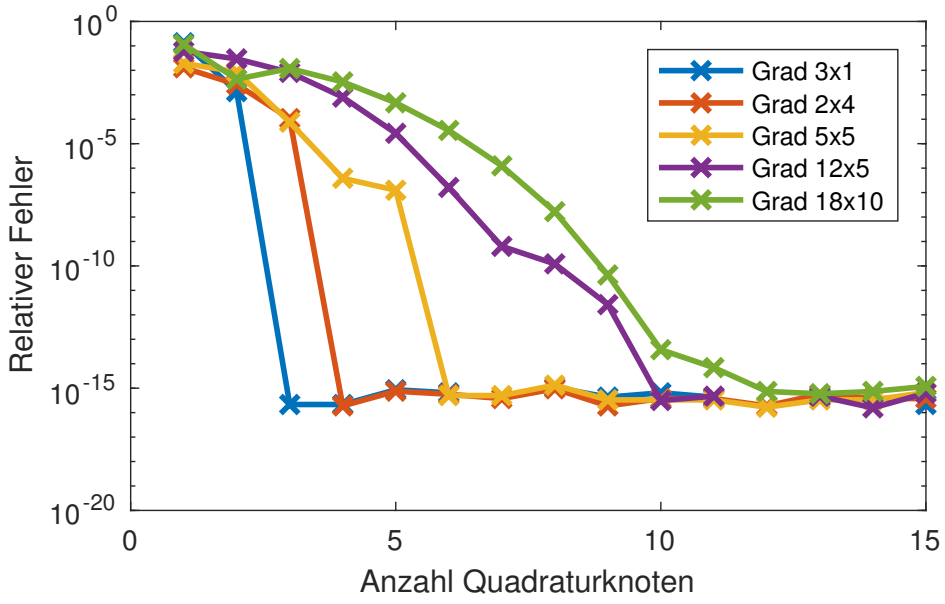

Abbildung 7: Relativer Fehler zur num. Integration von Polynomen in zwei Variablen verschiedenen Grades auf $\hat{T}$ über Anzahl von Quadraturknoten in beiden Koordinaten von 1 bis 15.