Numerik 2 Projektaufgabe 2- Numerische Integration

Autor: Wahid Far, Okhtay
Mtklnr.: 870485

Inhaltsverzeichnis

1)Einführung numerische Integration
2)Quadratur-Formeln
3)adaptive Quadratur
4)Interpretation Aufgabe 1 und Aufgabe 2

1)Einführung - numerische Integration

Die Motivation der numerischen Integration liegt in der Tatsache, dass in den meisten Fällen nicht jedes bestimmte Integral eine Stammfunktion besitzt. Das heißt, die analytische Bestimmung einer Stammfunktion bzw. des bestimmten Integrals selbst ist nur für wenige Funktionen möglich.

Man nutzt hierbei Näherungsverfahren, um das bestimmte Integral direkt zu bestimmen. Diese Näherungsverfahren bezeichnet man als numerische Quadratur.

Der allgemeine Ablauf ist folgendermaßen:

Schritt 1:
Unterteilung des Intervalls [a,b] des bestimmten Integrals in n Teilintervalle $[t_{k-1},t_k]$ mit der Länge $h = (b-a)/n$, $t_j = a+jh$, $n \in \mathbb{N}$.

Schritt 2:
Seien g_k ein Interpolationspolynom, Q eine Quadratur und $I_Q[f]$ die Approximation eines bestimmten Integrals.

Auf jedem Teilintervall $[t_{k-1},t_k]$ approximieren wir f mithilfe von g_k und verwenden die Addititvität des Integrals bezüglich des Integrationsbereiches:

$$I_Q[f] := \sum_{k=1}^{n} \int_{t_{k-1}}^{t_k} g_k(x)dx \approx \sum_{k=1}^{n} \int_{t_{k-1}}^{t_k} f(x)dx = \int_a^b f(x)dx$$

(Skript „Numerik 2“ auf Seite 25 von Prof. Frank Haußer)

2)Quadratur-Formeln

Die Quadraturverfahren dienen zur Annäherung des bestimmten Integrals bzw. zu dessen analytischen Berechnung. Sei $f(x_1)$, $f(x_2)$,…, $f(x_n)$ mit $n,i,k \in \mathbb{N}$.

Rechteck-Regel:

Hier nutzen wir Rechtecke, die wir ungefähr unterhalb der gewünschten Funktion anlegen. Die Breite jedes Rechtecks bestimmt die Stützstelle x_1 zu seiner nachfolgenden Stützstelle also x_2, wobei jede Stützstelle den gleichen Abstand zueinander aufweist. Die Höhe jedes Rechtecks wird durch jenen Funktionswert bestimmt, der zwischen zwei Stützstellen liegt also zum Beispiel $f(x_2)$ da x_2 zwischen x_1 und x_3 liegt.

Wir nutzen dazu folgende Quadratur-Formel:

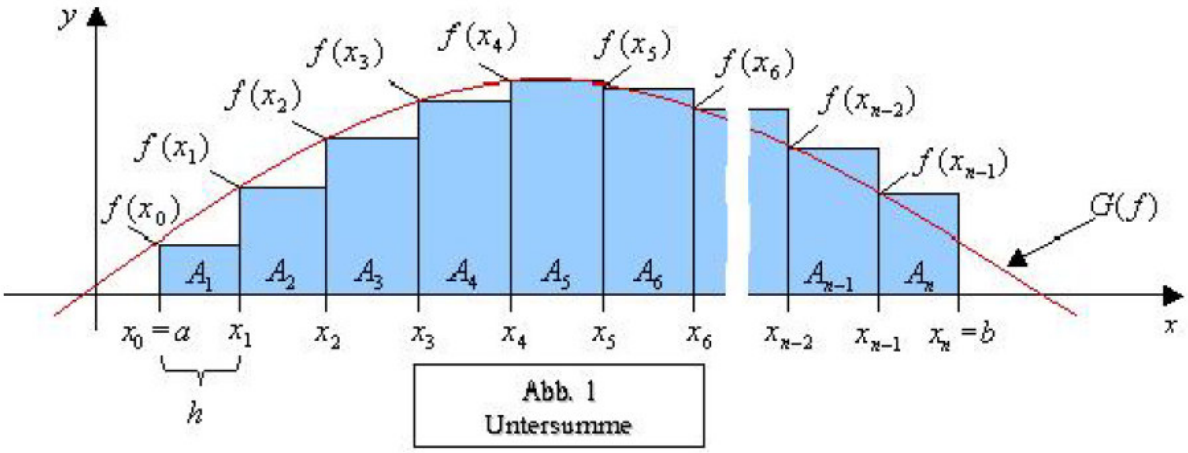
Sei g_k ein Interpolationspolynom mit $g_k(x) := f(t_{k-1})$ mit Interpolationsbedingung am linken Rand so gilt:

$$\int_{t_{k-1}}^{t_k} g_k(x)dx = hf(t_{k-1})$$

Summation als Quadratur:

$$R(h) := I_Q[f] = h\sum_{k=1}^{n} f(t_{k-1})$$

(Quelle: Skript Folie 4 "Numerische Integration zur Einführung" von Prof. Frank Haußer)



(Quelle: Abbildung 2.1 vom Vortrag „Numerische Integration" von Frau Nikola Isenhardt – Universität Heidelberg)

Zur Ergänzung lautet die allgemeine Rechteckregel:

$$\int_a^b f(x)dx \approx \frac{b-a}{n} * \sum_{j=0}^{n-1} f(a + \frac{b-a}{2*n} + \frac{b-a}{n} * i\ )$$

n == Anzahl der Teilintervalle

<u>Trapez-Regel:</u>

Dieses Verfahren ist genauer als das Rechteckverfahren. Hier verbindet man die erste Stützstelle mit der nachfolgenden.

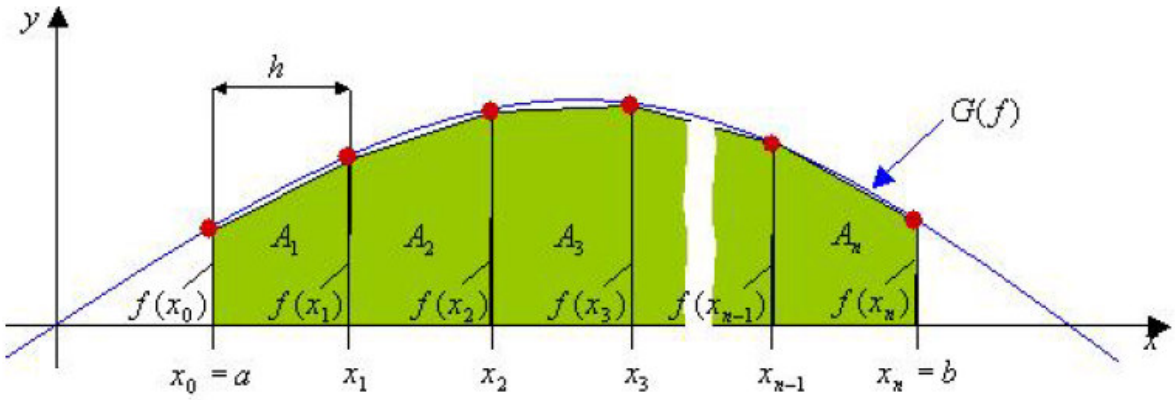
Sei für $g_k(x)$ auf jedem Teilintervall folgende lineare Interpolationsfunktion gegeben:

$$g_k(x) := P(f|t_{k-1}, t_k)(x) = \frac{x - t_{k-1}}{h} f(t_{k-1}) + \frac{t_{k-1} - x}{h} f(t_k)$$

Also gilt: $\int_{t_{k-1}}^{t_k} g_k(x)dx = h * \frac{f(t_{k-1})+f(t_k)}{2}$

Die Summation ergibt dann:

$$T(h) := I_Q[f] = h \sum_{k=1}^{n} \frac{1}{2} * f(t_{k-1}) + \frac{1}{2} * f(t_k)$$



(Abbildung 2.2 vom Vortrag „Numerische Integration“ von Frau Nikola Isenhardt – Universität Heidelberg)

Zur Ergänzung lautet die allgemeine Trapezregel:

$$\int_a^b f(x)dx \approx \frac{b-a}{n} * \sum_{i=0}^{n-1} \frac{f\left(a + i * \frac{b-a}{n}\right) + f(a + (i+1) * \frac{b-a}{n}}{n}$$

<u>*Simpson-Regel:*</u>

Um die Berechnungsfehler eines bestimmten Integrals noch mehr zu verringern, nutzen stückweise Annäherungen mithilfe von kubischen Parabeln. Hierbei muss die Anzahl der Teilintervalle gerade sein also n= 2*k mit n,i,k ∈ ℕ.

Sei für $g_k(x)$ auf jedem Teilintervall folgende quadratische Interpolationsfunktion gegeben:

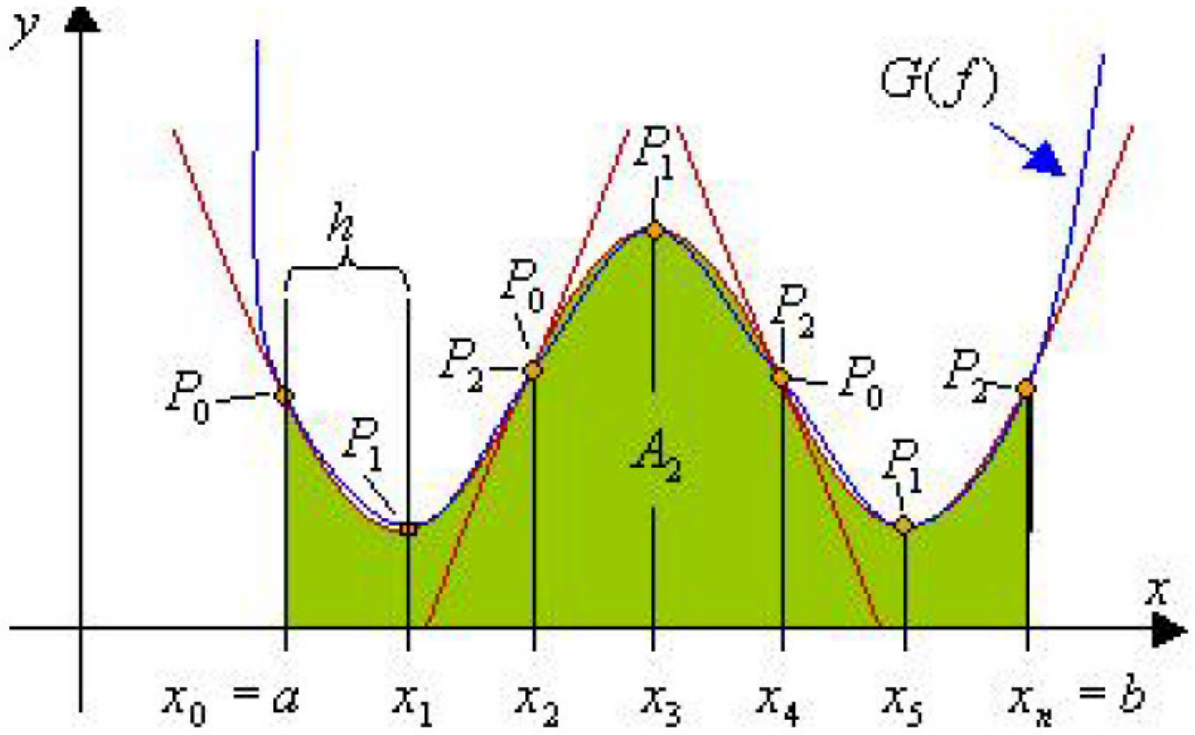
$$g_k(x) := P(f|t_{k-1}, \frac{(t_{k-1} + t_k)}{2}, t_k)(x)$$

Mit dem Lagrange-Basispolynom ergibt das folgendes:

$$\int_{t_{k-1}}^{t_k} g_k(x)dx = h(\frac{1}{6} * f(t_{k-1}) + \frac{4}{6} * f\left(\frac{t_{k-1} + t_k}{2}\right) + \frac{1}{6} * f(t_k))$$

Die Summation ergibt:

$$S(h) := I_Q[f] = h * \sum_{k=1}^{n} (\frac{1}{6} * f(t_{k-1}) + \frac{4}{6} * f(\frac{t_{k-1} + t_k}{2}) + \frac{1}{6} * f(t_k))$$



(Abbildung 2.3 vom Vortrag „Numerische Integration“ von Frau Nikola Isenhardt – Universität Heidelberg)

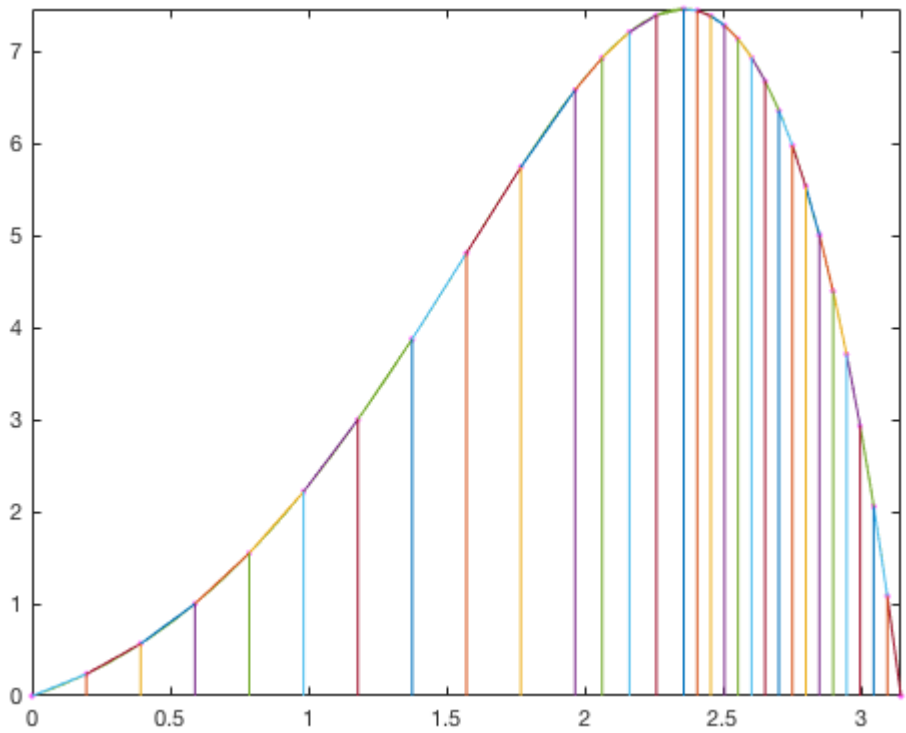
Allgemein kann man sagen, dass die Berechnung eines bestimmten Integrals mit den gleichen Stützstellen durch die Simpson-Regel am genauesten umgesetzt werden kann.
Das heißt, der absolute Fehler ist bei der Simpson-Regel deutlich geringer im Vergleich zu der Rechteck- und Trapezregel. Der Fehler nimmt proportional zu n^3 ab.

3)adaptive Quadratur

Für Funktionen mit großen Ableitungen höherer Ordnung sind die bisherigen Quadraturformeln eher ungeeignet. Hierbei liegt die Fehlerquelle oft in der Länge der Teilintervalle.
Aus diesem Grund ist es sinnvoll bei großen Ableitungen kleine Intervalle und bei kleinen Ableitungen große Intervalle zu nutzen. Die adaptive Quadratur stellt ein Verfahren dar, welches die Problematik großer Ableitungen höherer Ordnung löst.

Um das umzusetzen, werden zwei verschiedene Quadraturmethoden für jedes Teilintervall verwendet. Man überprüft somit lokale Fehler, wobei die eine Methode das Ergebnis überschätzt und das andere unterschätzt. Alternativ kann die eine Methode auch genauer sein als die andere.

Die Annäherung verläuft hier rekursiv und wir können im Code eine Toleranzvariable festlegen. Wir stellen fest, dass bei geraden und glatten Funktionen größere Teilintervalle und bei Krümmungen kleinere Teilintervalle genutzt werden.



4)Interpretation Aufgabe 1 und Aufgabe 2

Zu Aufgabe 1):

Wir betrachten $f(x) = \sqrt{x}$ in den Grenzen 0 bis 1 mit verschiedenen Toleranzen. Es ist zu erkennen, dass am Punkt x=0 die Steigung unendlich hoch wird.
Somit bereitet der Bereich um x=0 der Integration mit Quadraturformeln große Probleme, wodurch diese Umgebung mit kleineren Toleranzen öfter analysiert werden musste.

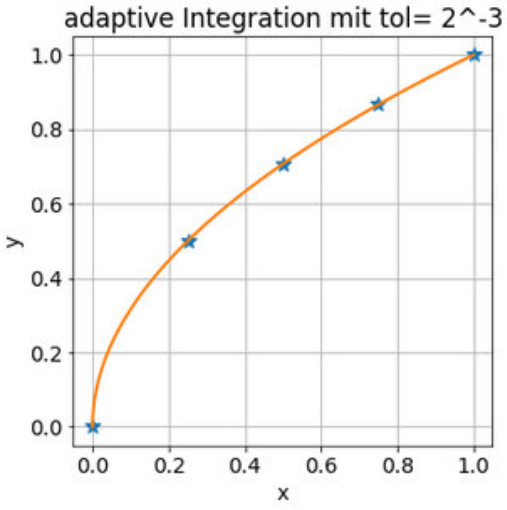
Für die summierte Simpson-Quadratur und die adaptive Quadratur haben wir die identische Anzahl von Untersuchungen verwendet. Zudem wurden die absoluten Fehler getrennt ausgegeben.

Im Vergleich beider Verfahren konnten wir feststellen, dass die adaptive Quadratur weniger Fehler und genauere Ergebnisse ausgibt. Denn die Simpson-Quadratur muss die Schrittweite verkleinern, um die nötige Fehlertoleranz zu erreichen.

Demnach ist der absolute Fehler bei der Simpson-Quadratur deutlich höher.

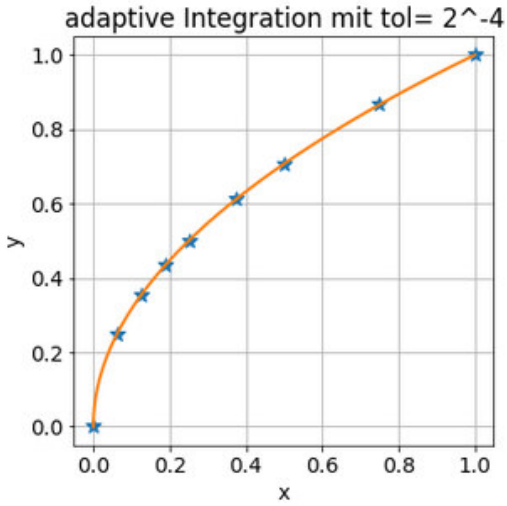
```
a=0;
b=1;
kmax = 5
f = wurzel
# funktionswerte
x = np.linspace(a, b, 1000)
y = f(x)
# fuer Vergleich analitische Loesung
integral_wurzel = 2 / 3
# schleife ueber verschiedene Toleranzen
for k in range(0, kmax):
    # adaptiv
    tol= 1./2.**(k+3)        # Toleranz
    (nodes, Q_adap ) = q.quadAdaptiv(f, a, b, tol, 1.e-9)
    x_adap = np.array(nodes)
    n = len(x_adap)
    y_adap = f(x_adap)
    # absolut fehler bei adaptive Quadratur
    abs_adaptiv = np.abs(integral_wurzel - Q_adap)

    # Simpsonquadrature mit gleichen Anzahl an Auswertungen
    q_simpson = q.quadSimpson(f, a, b, n)
    # absolut fehler bei Simpsonquadratur
    abs_simpson = np.abs(integral_wurzel - q_simpson)
```



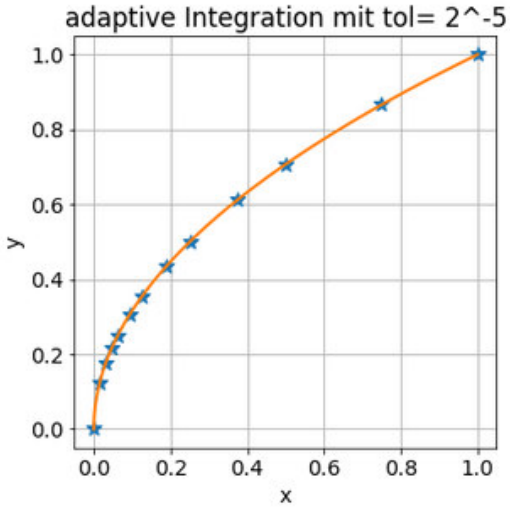
Absolute Fehler bei der Simpsonquadrature: 0.1636346789

Absolute der Fehler der adaptive Quadrature: 0.0631132761



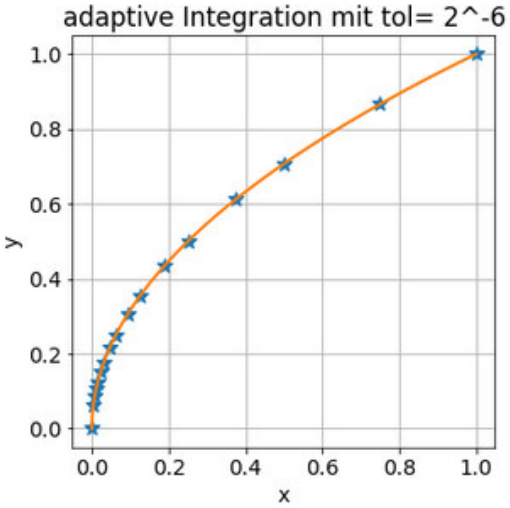
Absolute Fehler bei der Simpsonquadrature: 0.0770566062

Absolute der Fehler der adaptive Quadrature: 0.0135574499



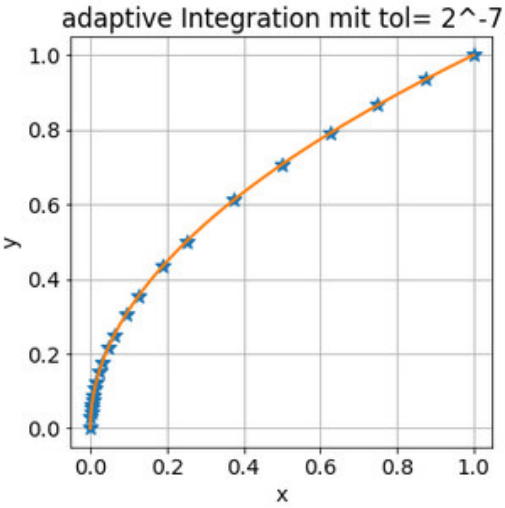
Absolute Fehler bei der Simpsonquadrature: 0.0497963919

Absolute der Fehler der adaptive Quadrature: 0.0073629716



Absolute Fehler bei der Simpsonquadrature: 0.0366088639

Absolute der Fehler der adaptive Quadrature: 0.0065886618



Absolute Fehler bei der Simpsonquadrature: 0.0260967784

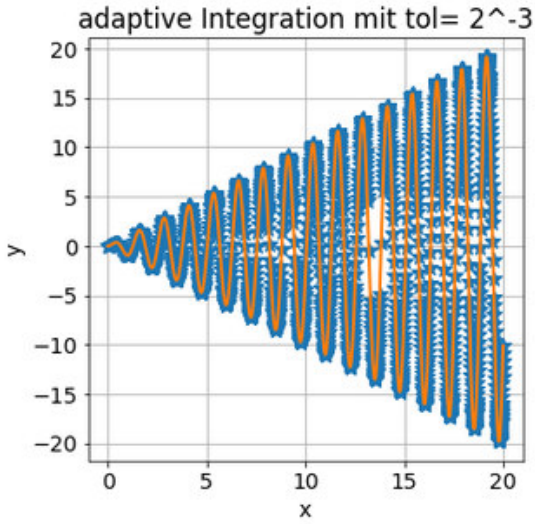
Absolute der Fehler der adaptive Quadrature: 0.0033738698

Zu Aufgabe 2):

Wir nutzen hier wieder die adaptive Quadratur für die Funktion $g(x) = x * \sin(x)$ mit $x \in [0, \dots, 20]$.
Anhand der Grafiken können wir feststellen, dass $g(x)$ stark oszilliert und viele Auswertungen benötigt. Im Vergleich zur Simpson-Quadratur ist aber der absolute Fehler bei der adaptiven Quadratur deutlich geringer.

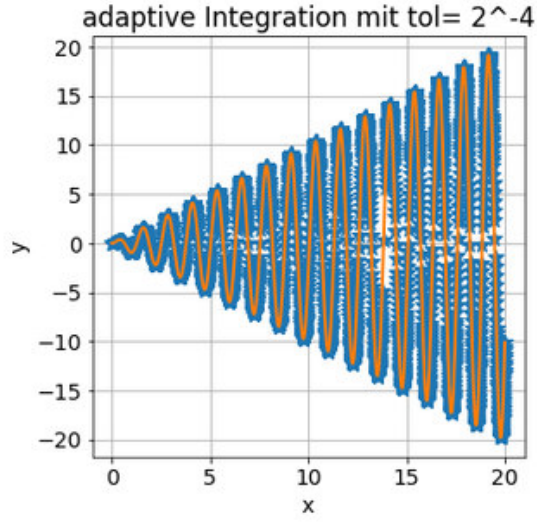
```
def g(x):
    return x*np.sin(5*x)
def I_g(a, b):
    # -1*np.sin(a) + a*np.cos(a) + np.sin(b) - b*np.cos(b)
    return (-1*np.sin(5*a) + 5*a*np.cos(5*a) + np.sin(5*b) - 5*b*np.cos(5*b)) / 25
a=0;
b=20;
kmax = 2
f = g
# funktionswerte
x = np.linspace(a, b, 1000)
y = f(x)
# fuer Vergleich analitische Loesung
integral = I_g(a,b)
# schleife ueber verschiedene Toleranzen
for k in range(0, kmax):
    # adaptiv
    tol= 1./2.**(k+2)       # Toleranz
    (nodes, Q_adap ) = q.quadAdaptiv(f, a, b, tol, 1.e-9)
    x_adap = np.array(nodes)
    n = len(x_adap)
    y_adap = f(x_adap)
    # absolut fehler bei adaptive Quadratur
    abs_adaptiv = np.abs(integral - Q_adap)

    # Simpsonquadrature mit gleichen Anzahl an Auswertungen
    q_simpson = q.quadSimpson(f, a, b, n)
    # absolut fehler bei Simpsonquadratur
    abs_simpson = np.abs(integral - q_simpson)
```



Absolute Fehler bei der Simpsonquadrature: 0.06408

Absolute der Fehler der adaptive Quadrature: 0.00045



Absolute Fehler bei der Simpsonquadrature: 0.04293

Absolute der Fehler der adaptive Quadrature: 0.00098

Quellen:

https://stackoverflow.com/questions/58892251/building-a-recursive-function-in-matlab-to-draw-adaptive-trapezoidal-quadrature

https://www.imn.htwk-leipzig.de/~medocpro/buecher/sedge1/k39t5.html

Buch: „Numerik“ von Andreas Meister und Thomas Sonar
Buch: „Numerik für Ingenieure und Naturwissenschaftler“ von W.Dahmen und A.Reusken