Was haben Beschleunigungs-Apps mit der *Quadratur des Kreises* zu tun?

Teilnehmer:

Jonathan Geuter	Herder-Oberschule, Berlin
Leonard Hackel	Herder-Oberschule, Berlin
Paul Hagemann	Herder-Oberschule, Berlin
Maximilian Kuch	Herder-Oberschule, Berlin
Amber Lucas	Andreas-Oberschule, Berlin
Tobias Thieme	Heinrich-Hertz-Oberschule, Berlin
Tobias Thiesse	Heinrich-Hertz-Oberschule, Berlin
Niko Wolf	Herder-Oberschule, Berlin

Gruppenleiter:

Caren Tischendorf	Humboldt-Universität zu Berlin Mitglied im Forschungszentrum MATHEON
René Lamour	Humboldt-Universität zu Berlin Mitglied im Forschungszentrum MATHEON

Sensoren in Smartphones können deren Beschleunigung messen. Uns interessieren die Geschwindigkeiten und Wegstrecken. Mithilfe numerischer Integration (Quadratur) lassen sich die gesuchten Werte für beliebige Messkurven approximieren. Durch Experimente und einige Vorüberlegungen können wir daraus verschiedene Berechnungsmethoden ableiten. Mithilfe von Python wurden diese auch implementiert und erfolgreich getestet. Dabei haben wir uns folgende Fragen gestellt:

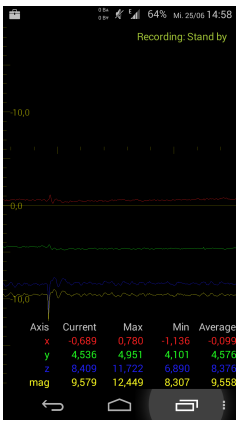
(1) Wie sieht eine App aus, die die Beschleunigung messen kann?

(2) Wie hängen diese Größen physikalisch zusammen?

(3) Welche Möglichkeiten der Berechnung gibt es?

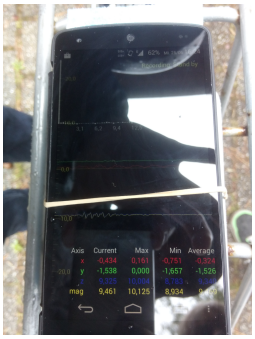
(4) Wie lassen sich diese in Python implementieren?

1 Accelogger & Experiment




```
2014/06/23 16:56:12 1403535372838349175 0,380142    -1,219330   10,420410
2014/06/23 16:56:12 1403535372843293023 -0,205429   -1,574005   10,203796
2014/06/23 16:56:12 1403535372848236870 -0,876694   -1,712067   9,960999
2014/06/23 16:56:12 1403535372853180718 -1,200424   -1,435944   9,649170
2014/06/23 16:56:12 1403535372858094048 -0,955246   -0,843231   9,508728
2014/06/23 16:56:12 1403535372863037896 -0,576767   -0,257660   9,418274
2014/06/23 16:56:12 1403535372867981743 -0,429184   0,061310    8,958862
2014/06/23 16:56:12 1403535372872925591 -0,479172   0,111298    8,506592
2014/06/23 16:56:12 1403535372877869439 -0,705307   -0,067230   8,611328
2014/06/23 16:56:12 1403535372882813286 -0,876694   -0,317169   9,180237
2014/06/23 16:56:12 1403535372887726616 -0,771957   -0,579010   9,515869
2014/06/23 16:56:12 1403535372892670464 -0,438705   -0,771820   9,549194
2014/06/23 16:56:12 1403535372897614312 -0,269699   -0,921783   9,701538
2014/06/23 16:56:12 1403535372902558159 -0,298264   -1,093170   9,994324
```

Die App Accelogger für Android misst die Beschleunigung mithilfe der Beschleunigungssensoren. In den obigen Bildern lassen sich verschiedene Kurven und verschiedene Werte in der Tabelle erkennen. Diese sind unterteilt in die Beschleunigungskomponenten in x, y, z-Richtung und die Gesamtkoordinate *mag*. Accelogger nimmt standardmäßig mit einer Frequenz von 100Hz auf, sie bietet aber auch die Option an, diese Frequenz zu ändern. Accelogger ist kostenlos im Google Play Store erhältlich. Die Zeit und Beschleunigungswerte werden in einem txt-File gespeichert, wobei die Zeit in Nanosekunden gespeichert wird.



Wir haben die App beim Fahrradfahren einer Teststrecke von ca. 200m getestet. Dazu haben wir das Smartphone auf dem Gepäckträger befestigt und haben die Beschleunigung mithilfe von Accelogger gemessen.

2 Weg, Geschwindigkeit, Beschleunigung

Ziel ist es, aus der Beschleunigung den Weg und die Geschwindigkeit zu berechnen. Dabei ist die Ableitung des Weges nach der Zeit die Geschwindigkeit und

die Ableitung der Geschwindigkeit nach der Zeit die Beschleunigung:

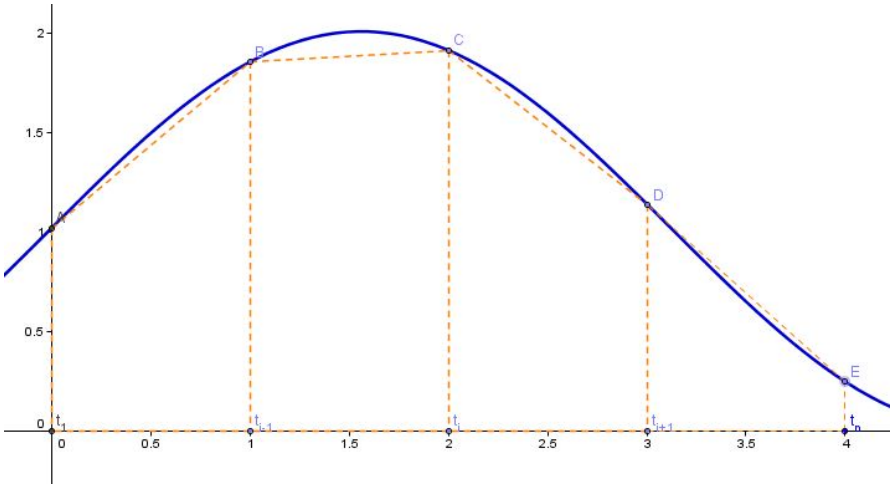
$$s'(t) = \frac{\mathrm{d}}{\mathrm{d}t}s(t) = v(t),$$
$$v'(t) = \frac{\mathrm{d}}{\mathrm{d}t}v(t) = a(t).$$

Entsprechend ist das Integral über der Beschleunigung die Geschwindigkeit und das über der Geschwindigkeit der Weg:

$$v(T) - v(0) = \int_0^T a(t)\,\mathrm{d}t,$$
$$s(T) - s(0) = \int_0^T v(t)\,\mathrm{d}t.$$

Bei einer beschleunigten Bewegung berechnet man die Fläche unter dem Graphen der Funktion, um die Geschwindigkeit zu erhalten. Dazu nutzen wir die nachfolgend erläuterte Trapezregel.

3 Trapezregel



Allgemein lässt sich das Integral durch Trapeze annähern. Dabei gilt für den Flächeninhalt eines einzelnen Trapezes folgende Formel, wobei $a(t)$ die Beschleunigung ist:

$$A_t = \frac{a(t_{i-1}) + a(t_i)}{2}(t_i - t_{i-1}).$$

Um das gesamte Integral zu berechnen, müssen alle Trapeze addiert werden:

$$v(T) - v(0) = \int_0^T a(t)dt \approx \sum_{i=1}^{n} \frac{a(t_{i-1}) + a(t_i)}{2}(t_i - t_{i-1}).$$

Analog kann aus der Geschwindigkeit die Strecke berechnet werden. Nach dem Hauptsatz der Integral- und Differentialrechnung lässt sich das Integral aus $v(t)$ als $s(T) - s(0)$ schreiben:

$$s(T) \approx s(0) + \sum_{i=1}^{n} \frac{v(t_{i-1}) + v(t_i)}{2}(t_i - t_{i-1}).$$

Das Integral wird mit 1 multipliziert, um die partielle Integration anwenden zu können:

$$\int_{t_{i-1}}^{t_i} f(t)dt = \int_{t_{i-1}}^{t_i} f(t)g'(t)dt$$

mit $g(t) = t - c$ und $c = \frac{t_{i-1} + t_i}{2}$.
Es gilt allgemein für die partielle Integration:

$$\int_a^b u'v = \Big[uv\Big]_a^b - \int_a^b uv'.$$

Die partielle Integration angewandt auf das Integral:

$$\int_{t_{i-1}}^{t_i} f(t)g'(t)dt = \Big[f(t)g(t)\Big]_{t_{i-1}}^{t_i} - \int_{t_{i-1}}^{t_i} f'(t)g(t)dt$$

$$\text{mit } \int_{t_{i-1}}^{t_i} f'(t)g(t)dt = R_i \text{ und } \sum_{i=1}^{n} R_i = R \text{ ergibt}$$

$$= \frac{f(t_i) + f(t_{i-1})}{2}(t_i - t_{i-1}) - R.$$

Durch die eingesetzten Grenzen und die umgeformte Gleichung ergibt sich die Trapezregel:

$$\int_0^T f(t)dt = \sum_{i=1}^{n} \frac{f(t_i) + f(t_{i-1})}{2}(t_i - t_{i-1}) - R.$$

3.1 Iterierte Trapezregel

Die iterierte Trapezregel ist ein Spezialfall der Trapezregel, bei dem eine äquidistante Unterteilung vorgenommen wird. Der Abstand zwischen zwei Punkten ist konstant, sodass man ihn durch ein h ersetzen kann. Beim Summieren der Einzeltrapeze wird dann h ausgeklammert.

$$T(h) = \sum_{i=1}^{n} \frac{f_{i-1} + f_i}{2} \cdot h$$

$$= (f_0 + f_1 + f_1 + f_2 + f_2 + \ldots + f_n) \cdot \frac{h}{2}.$$

Da dann alle Punkte zwischen den Randpunkten doppelt vorkommen, wird die Gleichung wie folgt zusammengefasst

$$T(h) = \frac{h}{2} \cdot (f_0 + 2\sum_{i=1}^{n-1} f_i + f_n).$$

4 Implementierung der Trapezregel

```
for i in range(b):
    #in sec umwandeln
    t=(float(datentxt[i,1])-
       float(datentxt[0,1]))/1000000000.0
    vektor_t[i]=t
```

Abbildung 1: Import der Messpunkte in Sekunden

```
for i in range(b):
    #Korrekturwert = -r/(vektor_t[b-1])
    a= float(datentxt[i,2].replace(",",".")) #+K
    vektor_a[i]=a
```

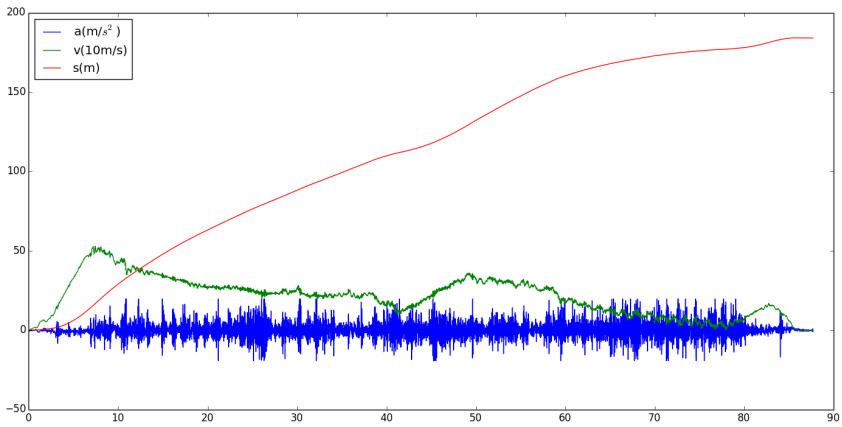
Abbildung 2: Erstellen des Beschleunigungsvektors

```
r=0
vek=zeros(b)
for i in range(1,b) :
    r=r+((vektor_a[i-1]+vektor_a[i])/2)*
        (vektor_t[i]-vektor_t[i-1])
    vek[i]=r
```

Abbildung 3: Integration mit Trapezregel

Als erstes werden die Messwerte aus der Datei „daten.txt“ importiert. Nun werden die Zeiten in Sekunden umgewandelt und als Differenz zum Anfangswert in einem Vektor gespeichert (siehe Abb. 1). Als nächstes erfolgt die Umrechnung der Beschleunigungswerte in float-Zahlen (siehe Abb. 2). Diese werden zunächst in einem Beschleunigungsvektor gespeichert. Diese Beschleunigungswerte werden nun mithilfe der Trapezregel „integriert“ (siehe Abb. 3). Anschließend wird der

Weg als Integral der Geschwindigkeit nach der Zeit nach dem selben Muster berechnet. Ausgehend von den Messungenauigkeiten des Smartphones ist noch ein Korrekturwert nötig. Aufgrund der Tatsache, dass die Geschwindigkeit am Ende Null ist, kann man diesen über die Abweichung der berechneten Geschwindigkeit ermitteln (siehe Abb. 2).



Hier lässt sich klar erkennen, dass die Beschleunigungswerte große Sprünge aufweisen, jedoch sieht der Weg dabei sehr realistisch aus. Die Ergebnisse beider Versuche sind relativ dicht am eigentlichen Ergebnis von 200m; $s_1 = 184m$ und $s_2 = 169m$.

5 Bernoulli-Polynome

5.1 Allgemeine Definition

Das k-te Bernoulli Polynom $B_k(x)$ ist das Polynom, das auf [0;1] mit Werten in $\mathbb{R}$ definiert ist und folgende Eigenschaften besitzt:

$$B_0(x) = 1 \qquad \forall x \in [0;1] \tag{5.1}$$

$$B_k'(x) = k * B_{k-1}(x) \qquad \forall x \in [0;1]\ \forall k \geq 1 \tag{5.2}$$

$$\int_0^1 B_k(x)\,\mathrm{d}x = 0 \qquad \forall k \geq 1 \tag{5.3}$$

Die Eigenschaften (5.1)-(5.3) sind eine Bildungsvorschrift für weitere Bernoulli-Polynome, wobei stets aus einem $B_n(x)$ das nächste Glied $B_{n+1}(x)$ bestimmt wird.

5.2 Berechnung der Polynome

$$B_1(x) = x + c$$
$$0 = \int_0^1 (x + c)\,\mathrm{d}x = \left[\frac{x^2}{2} + xc\right]_0^1$$
$$c = -\frac{1}{2}.$$

Hier wird $B_1(x)$ gebildet. Es gilt $\int_0^1 B_0(x) = x + c$. Um die Konstante c zu bestimmen, muss das Polynom integriert werden (nach Eigenschaft (5.3)).

$$B_2'(x) = 2 * B_1(x)$$
$$B_2' = \int_0^1 2\left(x - \frac{1}{2}\right)\,\mathrm{d}x = \left[x^2 - x + c\right]_0^1$$
$$0 = \int_0^1 (x^2 - x + c)\,\mathrm{d}x = \left[\frac{x^3}{3} - \frac{x^2}{2} + c\right]_0^1$$
$$c = \frac{1}{6}.$$

Aus $B_1(x)$ wird nun das nächste Polynom $\mathrm{B}_2(x)$ ermittelt. Die Bernoulli-Zahl bezeichnet die Konstante c, welche im 3. Bernoulli-Polynom $c = \frac{1}{6}$ beträgt. $B_3(x)$ bis $B_5(x)$ sind die nächsten Bernoulli-Polynome:

$$B_3(x) = x^3 - \frac{3}{2}x^2 + \frac{1}{2}x,$$
$$B_4(x) = x^4 - 2x^3 + x^2 - \frac{1}{30},$$
$$B_5(x) = x^5 - \frac{5}{2}x^4 + \frac{5}{3}x^3 - \frac{1}{6}x.$$

5.3 Eigenschaften

Für alle $k > 1$ mit k ungerade, gilt $C = 0$ und folglich $B_0 = 0$. Für alle $j \geq 2$ gilt $B_j(1) = B_j(0)$ beziehungsweise $B_j(1) - B_j(0) = 0$. Dies lässt sich mittels des Hauptsatzes der Infinitesimalrechnung und Eigenschaft (5.2) zeigen.

5.4 Weitere Anwendung

Da $B_0(x) = 1$ gilt, lässt sich ein Integral über einer Funktion $g(x)$ mit Hilfe partieller Integration umformen zu:

$$\int_0^1 g(x)dx = \Big[g(x)B_1(x)\Big]_0^1 - \int_0^1 g'(x)B_1(x)dx = \dots$$
$$= \sum_{i=1}^{n} \left[\frac{1}{i!}g^{(i-1)}(x)B_i(x)\right]_0^1 * (-1)^{i+1} + (-1)^n \int_0^1 \frac{1}{n!}B_n(x)g^{(n)}(x)dx.$$

Dabei wurde partielle Integration n mal angewandt; das am Ende stehende Integral stellt das Restglied dar.

6 Reihenentwicklung in h für Trapezregel

Um das Romberg-Verfahren anwenden zu können, benötigen wir eine weitere Formel, welche für $T(h)$ einen Term mit geraden Potenzen von h, der äquidistanten Schrittweite beim Trapezverfahren, und Koeffizienten a_n ausgibt:

$$T(h) = a_0 + a_1h^2 + a_2h^4 + a_3h^6 + \dots \ .$$

6.1 Beweis der Existenz dieser Darstellung von $T(h)$

Es gilt

$$\int_0^T f(t)dt = \sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} f(t)dt.$$

Nun sei $x = \dfrac{t - t_{i-1}}{h}$ und $h = t_i - t_{i-1}$. Wir definieren eine Funktion $g_i(x)$:

$$g_i(x) = h * f(t) = h * f(t_{i-1} + hx).$$

Nach Integration durch Substitution gilt:

$$\begin{aligned}\int_{t_{i-1}}^{t_i} f(t)dt &= \int_0^1 hf(t_{i-1} + hx)dx \\ &= \int_0^1 g_i(x)dx \\ &= \sum_{j=1}^{n}(-1)^{j+1}\left[\frac{1}{j!}g_i^{(j-1)}(x)B_j(x)\right]_0^1 + R_i.\end{aligned}$$

Setzt man in die Grenzen und die (j-1)te Ableitung von $g_i(x)$ ein, so erhält man:

$$\int_0^1 g_i(x)dx = \sum_{j=1}^{n}(-1)^{j+1}\frac{1}{j!}h^j\left(f^{(j-1)}(t_i)B_j(1) - f^{(j-1)}(t_{i-1})B_j(0)\right) + R_i.$$

Aus den Eigenschaften der Bernoulli-Polynome folgt nun:

$$\int_0^1 g_i(x)dx = h\left[\frac{f(t_i)}{2} + \frac{f(t_{i-1})}{2}\right] - \sum_{k=1}^{m}\frac{h^{2k}}{(2k)!}b_{2k}\left(f^{(2k-1)}(t_i) - f^{(2k-1)}(t_{i-1})\right) + R_i.$$

Dabei stellt der Term $h\left[\frac{f(t_i)}{2} + \frac{f(t_{i-1})}{2}\right]$ den Term $T(h)$ dar. Betrachtet man nun anstelle des Integrals $\int_0^1 g_i(x)dx$ das ursprüngliche gesamte Integral $\int_0^T f(t)dt$, so kürzen sich die meisten der Faktoren $f^{(2k-1)}(t_i) - f^{(2k-1)}(t_{i-1})$ raus und die Reste R_i addieren sich zu dem Gesamtrestglied R:

$$\int_0^T f(t)dt = T(h) - \sum_{k=1}^{m}\frac{h^{2k}}{(2k)!}b_{2k}\left(f^{(2k-1)}(T) - f^{(2k-1)}(t_0)\right) + R.$$

Geht m gegen ∞, so geht R gegen 0. Die Umstellung nach $T(h)$ ergibt:

$$\begin{aligned} T(h) =& a_0 + \sum_{k=1}^{m}\frac{h^{2k}}{(2k)!}b_{2k}\left(f^{(2k-1)}(T) - f^{(2k-1)}(t_0)\right) \\ =& a_0 + a_1h^2 + a_2h^4 + a_3h^6 + \dots \end{aligned}$$

Hierbei lässt sich die obige Summe als Summe aus Produkten mit Potenzen von h und von h unabhängigen Faktoren darstellen. Damit erhalten wir die geforderte Darstellung von $T(h)$.

7 Romberg-Verfahren

Aus der Reihenentwicklung folgt, dass die Trapezformel auch als Reihe, abhängig von der äquidistanten Schrittweite h, geschrieben werden kann. Hierbei ist $T(h)$ der Wert der Trapezformel, a_0 der Wert des Integrals und die restlichen Terme sind Fehlerterme.

$$\begin{aligned} T(h) = T_h^{[0]} &= a_0 + a_1h^2 + a_2h^4 + \dots \\ T_{\frac{h}{2}}^{[0]} &= a_0 + a_1(\tfrac{h}{2})^2 + a_2(\tfrac{h}{2})^4 + \dots \end{aligned}$$

Da $a_1 h^2$ der größte Fehler ist, wird dieser durch geschicktes Umformen eliminiert. Dabei werden $T(h)$ und $T(\frac{h}{2})$ mithilfe von q (=0.5) verrechnet, sodass bei der Subtraktion beider Gleichungen a_1 mit 0 multipliziert wird.

$$T^{[0]}_{\frac{h}{2}} - q^2 \cdot T^{[0]}_h = a_0 - q^2 a_0 + a_1(\tfrac{h^2}{4} - \tfrac{h^2}{4}) + a_3(\tfrac{h^4}{16} - \tfrac{h^4}{4}) + \dots$$

Nun teilen wir durch $(1 - q^2)$, damit wir wieder a_0 als Ergebnis erhalten und definieren es als $T^{[1]}_h$

$$\frac{T^{[0]}_{\frac{h}{2}} - q^2 \cdot T^{[0]}_h}{1 - q^2} = a_0 - a_1 \cdot 0 + a_2^* h^4 + a_3^* h^6 + \dots =: T^{[1]}_h.$$

Somit fällt der größte Fehlerterm (a_1) heraus, sodass wir einen genaueren Wert erreichen.
Diese Methode lässt sich beliebig oft fortsetzen, um Fehler zu eliminieren und einen genaueren Wert zu erreichen, solange es genügend Startwerte (z.B. $T^{[0]}_h$) gibt. Es ergibt sich folgende Tabelle, wobei zwei Werte nach erwähnter Methode einen neuen ergeben.

$$\begin{array}{lllll}
T^{[0]}_h & & & & \\
 & \searrow & & & \\
T^{[0]}_{\frac{h}{2}} & \rightarrow & T^{[1]}_h & & \\
 & \searrow & & \searrow & \\
T^{[0]}_{\frac{h}{4}} & \rightarrow & T^{[1]}_{\frac{h}{2}} & \rightarrow & T^{[2]}_h \dots
\end{array}$$

Ein weiterer Vorteil des Romberg-Verfahrens ist, dass es gleichzeitig auch eine gute Fehlerabschätzung liefert, indem man jeweils die zwei genauesten Werte, den neu erzeugten und den vorher genaueren Wert, subtrahiert.
So würde man z.B., um die Genauigkeit des Ergebnisses zu testen, $T^{[2]}_h$ von $T^{[1]}_{\frac{h}{2}}$ subtrahieren. Dabei entfällt der Wert des Integrals a_0, sodass lediglich die Differenz des Fehlers übrigbleibt.
Beispiel:

$$T^{[0]}_{\frac{h}{2}} - T^{[1]}_h = a_0 - a_0 + a_1^* \frac{h^2}{4} - a_1^* h^2 \dots$$

7.1 Implementierung am Beispiel

Um das Romberg-Verfahren in Python zu implementieren, definieren wir uns eine Funktion. Da wir als Ergebnis π erhalten möchten, machen wir das in 1. mit der Kreisfunktion, welche sich über Umformen des Satzes des Pythagoras herleiten lässt.

1. Definition der Funktion:

```
def f(x):
        return sqrt(1-x**2)
```

Diese Funktion übergeben wir mit folgenden Parametern an unsere Funktion *unser_romberg*: unser_romberg(vorher angegebene Funktion, untere Grenze, obere Grenze, Toleranz).

2. Ausführung der Funktion:

```
unser_romberg(f,0.0,1.0,1e-10)
```

Die Grenzen müssen in Dezimalschreibweise angegeben werden, da sie sonst als *int* interpretiert und dementsprechend gerundet werden. Dies möchten wir aber verhindern, weshalb wir sie in dieser Weise als *float* angeben.
Die Werte werden nun von unserer Funktion *unser_romberg* verarbeitet, sodass wir ein genaues Ergebnis erhalten.

3. Funktion Definieren:

```
def unser_romberg(f,a,b,TOL):
if TOL<1e-10:  TOL=1e-10
q=0.5; n=2;T=zeros(n);T[0]=0;T[1]=2*TOL
while ((abs(T[0]-T[1]))>TOL):
    T=zeros(n)
    for i in range(n):
        T[i]=Tt(f,a,b,(2**i))
    for j in range(n-1):
        Tz=zeros(n-j-1)
        for k in range(n-j-1):
            Tz[k]=(T[k+1]-((q**2)*T[k]))/(1-q**2)
        for k in range(n-j-1):
            T[k]=Tz[k]
    n=n+1
```

Dabei wird als erstes die Toleranz korrigiert, wenn diese zu hoch ist, da sonst der Rechenaufwand zu hoch wäre. Danach wird die oben bereits angesprochene Konstante q definiert und in den folgenden Zeilen ein Vektor T konstruiert, welcher nur den Sinn hat, in die while-Schleife zu gelangen. Nun wird als erstes ein neuer n-dimensionaler Vektor T erzeugt, welcher dann als Speicher für die durch das Trapez-Verfahren errechneten Werte genutzt wird. In der nächsten Schleife wird ein neuer Vektor Tz für den nächsten Iterationsschritt erzeugt. Dieser wird dann mithilfe des Rombergverfahrens wie oben beschrieben.

Danach werden die alten Vektoren T mit den Werten aus Tz überschrieben, damit die aktuellen Werte weiter genutzt werden können. Dabei bleibt einer der Werte aus den alten T-Vektoren übrig, was genutzt wird, wenn danach für die while-Schleife kontrolliert wird, ob die Toleranz erreicht ist. Falls dem so ist, wird der Wert ausgegeben (nicht im Quelltext zu sehen) oder die Schleife wiederholt, wobei vorher $n = n + 1$ gesetzt wurde, also diesmal eine Ebene mehr berechnet wird.

8 Quadratur des Kreises

Abschließend wurden diese Verfahren mit Hilfe der Kreisfunktion $\sqrt{1 - x^2}$ im Intervall [0,1] getestet:

- $\pi(Trapez) = 3.14159254841$; bei h =1e-5
- $\pi(Romberg) = 3.14159265358$57523 ; bei 23 Iterationen
- $\pi = 3.141592653589793\ldots$

Am Ende stellt sich natürlich noch die Frage, was diese Verfahren nun mit der Quadratur des Kreises und mit der Berechnung von π zu tun hat. Es lässt sich sagen, dass das Romberg-Verfahren bei ungefähr gleichem Rechenaufwand deutlich genauer ist. Außerdem kann man mit dem Romberg-Verfahren, im Gegensatz zum Trapez-Verfahren, genau abschätzen, wie groß der Fehler am Ende ist.