

Numerische Integration

Die einfachste Anwendung des Integrals ist wohl die Beantwortung der Frage nach der Fläche zwischen dem Graphen einer Funktion und der Achse über einem gegebenen Intervall ('Quadraturaufgabe'). Die elegante mathematische Beantwortung scheitert oft daran, dass die gegebene Funktion nicht analytisch integrierbar ist.

Beispiel 1:

Die Fläche zwischen dem Graphen von $f(x) = x^7$ und der x -Achse über dem x -Intervall $[0,2]$ ist leicht zu bestimmen. Die Stammfunktion von $f(x)$ ist $x^8/8$ und somit ergibt sich die Antwort 32.

Beispiel 2:

Die Funktion $f(x)$ ist gegeben als $\sin(e^{\cos x})$. Diese Funktion ist nicht 'mit Zettel und Bleistift' integrierbar, wir können ihre Stammfunktion also gar nicht angeben - besser gesagt hat die Mathematik keinen Namen für sie.

In solch problematischen Fällen kann uns vielleicht der Computer weiterhelfen: Wenn wir nur am Zahlenwert des Ergebnisses interessiert sind und nicht an einer angeschriebenen Stammfunktion, könnten wir zu Näherungsverfahren greifen.

Wie wäre eine derartige Aufgabe 'praktisch' lösbar?

Etwa so: Wir nehmen ein großes Stück Papier, zeichnen darauf ein Einheitsquadrat, schneiden es mit der Schere aus, und wiegen es mit einer Waage ab. Dann erstellen wir mithilfe eines Taschenrechners (Computers) eine möglichst umfangreiche Wertetabelle der Funktion $f(x)$ im gegebenen Intervall. Damit können wir nun die x -Achse und den Graph über dem Intervall zeichnen. Diese Figur schneiden wir ebenfalls aus und wiegen sie ab. Der gesuchte Flächeninhalt ist genau der Faktor, wieviel mal diese Figur schwerer ist als das Einheitsquadrat.

Diese Vorgangsweise ließe sich nur dann auf den Computer übertragen, könnte dieser zeichnen, ausschneiden und abwägen. Wir müssen eine Alternative finden.

Vielleicht so: Wir haben doch die Wertetabelle. Das Auswerten der Funktionsgleichung ist problemlos möglich. Wenn wir die gezeichnete Figur in schmale Streifen schneiden, so sind diese ungefähr Rechtecke mit bekannten Maßen: Breite = Streifenbreite, Höhe = ungefähr Funktionswert. Zählen wir die Flächeninhalte all dieser Rechtecke zusammen, so bekommen wir einen Wert, der ungefähr der gesuchten Fläche entspricht.

Warum heißt so etwas ein 'Näherungsverfahren'? Die einzelnen Streifen sind keine echten Rechtecke. Der Graph von $f(x)$ wird vermutlich irgendwie gekrümmt sein. Zur Berechnung der Streifenhöhe müssen wir aber eine einzige geeignete Zahl finden.

Vielleicht der linke Wert? Oder der rechte? Oder ihr Mittelwert? Oder der Funktionswert in der Mitte? Oder eine geschickte Kombination von mehreren Zwischenwerten? All das ist möglich.

Wenn Du mehr mathematischen Hintergrund brauchst, frage Deinen Mathe-Lehrer nach den Integrationsformeln vom Typ 'Newton-Cotes'. Er wird Dir gerne weiterhelfen. (Es gibt noch zahlreiche weitere interessante Methoden, die uns hier aber nicht interessieren sollen).

Für uns sind nur folgende Beobachtungen wesentlich:

- je dünnere Streifen ich schneide, desto genauer wird das Ergebnis stimmen - der Graph hat weniger Platz zum 'Herumwackeln'.
- Wenn ich viele berechnete Zwischenpunkte heranziehe, kann ich vielleicht auch gekrümmte

Verläufe des Graphen gut darstellen - schließlich lässt sich durch k Punkte ein Polynom vom Grad $k-1$ legen. Wenn $f(x)$ ein Polynom von Grad k ist, kann ich es mit $k+1$ Stützstellen exakt fassen.

Hier eine kleine Übersicht gebräuchlicher Formeln. h bezeichnet die Intervalllänge $b-a$, die einzelnen Werte $x_0=a, x_1, x_2, \dots, x_N=b$ sind eine äquidistante Zerlegung des Intervalls $[a,b]$.

Der (lokale) Fehlerterm enthält eine Potenz von h : sie sagt aus, wie der Fehler auf eine Änderung von h reagiert. z.B. bedeutet h^4 , dass eine Halbierung der Intervall-Länge den Fehler auf ein Sechzehntel senkt. Weiters steht eine Ableitung von f : sie gibt die erste Polynom-Ableitung an, die nicht mehr präzise erfasst wird. $f^{(5)}$ bedeutet also, dass ein Polynom 4. Grades noch exakt berechnet würde, eines vom 5. Grad aber nicht mehr (da seine 5. Ableitung nicht Null ist).

Einige beliebte Beispiele:

$\int_{a=x_0}^{b=x_1} f(x) dx = \frac{f(x_0)+f(x_1)}{2} \times h$	Trapez-Regel	$O(h^3 f^{(2)})$
$\int_{a=x_0}^{b=x_1} f(x) dx = f\left(\frac{x_0+x_1}{2}\right) \times h$	Tangententrapez	$O(h^3 f^{(2)})$
$\int_{a=x_0}^{b=x_2} f(x) dx = \frac{f(x_0)+4f(x_1)+f(x_2)}{6} \times h$	Simpson-Regel	$O(h^5 f^{(4)})$
$\int_{a=x_0}^{b=x_3} f(x) dx = \frac{f(x_0)+3f(x_1)+3f(x_2)+f(x_3)}{8} \times h$	Simpsons 3/8 Regel	$O(h^5 f^{(4)})$
$\int_{a=x_0}^{b=x_4} f(x) dx = \frac{7f(x_0)+32f(x_1)+12f(x_2)+32f(x_3)+7f(x_4)}{90} \times h$	Bode-Regel	$O(h^7 f^{(6)})$

Mit diesen Methoden können wir also Integrale numerisch berechnen. Genau genommen alle bestimmten Integrale halbwegs 'braver' Funktionen über endlichen Intervallen. ('brav' bedeutet: keine Unendlichkeitsstellen der Funktion im Intervall, keine bösen Sprünge oder Unendlichkeiten in Ableitungen,... dafür gibt es andere Techniken)

Eine veränderte Sichtweise

Können wir wirklich nur den Zahlenwert des Flächeninhaltes für $f(x)$ ermitteln, oder wäre es uns doch möglich, einen Überblick über den Verlauf der Stammfunktion $F(x)$ zu erhalten?

Schreiben wir den bekannten Zusammenhang $\int_{a=x_0}^{b=x_1} f(t) dt = F(b) - F(a)$ einfach um, wobei wir statt x_0 und x_1 nun x und $x+h$ schreiben, also einen x -Wert und einen darauf folgenden im Abstand h :

$$F(x+h) = F(x) + \int_x^{x+h} f(t) dt, \text{ dabei werten wir das Integral durch eine der obigen Formeln aus.}$$

Dies ergibt die Möglichkeit, schrittweise einen Punkt des Graphen nach dem anderen zu berechnen! Wir bekommen also in Einzelschritten eine Wertetabelle geliefert, die es uns erlaubt, eine beliebig genaue Zeichnung des Verlaufes von $F(x)$ bzw. des Integrals anzufertigen. Damit haben wir die Stammfunktion, wenn auch nicht als mathematischen Term, so doch als Graph gefunden!

Differentialgleichungen

Zugegeben - Das Ermitteln von Flächeninhalten ist keine besonders aufregende Sache. Die wirklich interessanten Zusammenhänge werden durch etwas kompliziertere Gleichungen beschrieben.

Seit Galileo Galilei wissen wir, dass die Geschwindigkeitszunahme eines frei fallenden Körpers gerade der Erdbeschleunigung g (9,81) entspricht. Das formuliert der Mathematiker als

$$v'(t) = g$$

$$v(t) = \int_0^T g \times dt = g \times T$$

Dies ist durch Quadratur lösbar, weil der Ausdruck im Integral konstant ist. Mit dieser Erkenntnis kann man nun sogar die zurückgelegten Fallstrecke bestimmen:

$$s'(t) = v \times t$$

$$s(t) = \int_0^T g \times t \, dt = \frac{g \times T^2}{2}$$

Dies klappt, weil der Integrand nur von t abhängt. Wir kennen ja die Geschwindigkeit zu jedem Zeitpunkt. Diese Aufgabe ist also mithilfe zweier reiner Quadraturen lösbar.

Das war aber kein wirklich spannendes Beispiel. Viel interessanter wäre doch der freie Fall eines Fallschirmspringers, der durch die Luftreibung gebremst wird. Was wird seine Geschwindigkeit nach 20 Sekunden sein? Die Reibungskraft gegenüber der Luft hängt von der Geschwindigkeit des Springers ab. Mit einer (geeignet zu bestimmenden von Springer und Luft abhängenden) Konstante k und dem Wissen, dass die Reibungskraft mit dem Quadrat der Geschwindigkeit wächst, heißt das Newton'sche Gesetz:

$$F = m \cdot a = m \cdot g - k \cdot v^2$$

$$v'(t) = g - \frac{k}{m} \cdot v^2(t)$$

Können wir diese Differentialgleichung lösen? Nein! Wir können die Funktion auf der rechten Seite nicht auswerten: Um die Geschwindigkeit zur Zeit $t+h$ aus der Geschwindigkeitsänderung (die Höhe unserer Papierstreifen) zu erhalten, müssen wir außer t auch v kennen. Doch das versuchen wir gerade erst auszurechnen!

Ein unlösbarer Fall? Zum Glück nein. Ähnlich wie man sich obige Quadraturformeln überlegen kann, gibt es auch Näherungsformeln für die Lösung von Differentialgleichungen. Sie bestimmen den nächstfolgenden Wert der Geschwindigkeit aus der Anfangsgeschwindigkeit und einigen Auswertungen der rechten Gleichungsseite zu gewissen Zeitpunkten und für gewisse Geschwindigkeitswerte (die selbst nicht exakt sind, sondern nur Hilfswerte liefern).

Die bekanntesten derartigen Regeln heißen nach zwei deutschen Mathematikern zu Anfang des 20. Jahrhunderts '**Runge-Kutta-Formeln**'. Es gibt zahlreiche Varianten für unterschiedliche Genauigkeitsanforderungen, unterschiedliche Funktionstypen, unterschiedliche Problemstellungen, mit eingebauter Schrittweitensteuerung, mit automatischer Fehlerabschätzung,....

Runge-Kutta-Verfahren RKp(m)

Zu lösen ist eine Differentialgleichung (d.h. wir suchen eine Funktion $y(t)$, wobei wir nur die Änderung von y kennen, sowie einen Startwert zur Zeit $t = t_0 = 0$)

$$y'(t) = f(t, y) \quad , \quad y(t_0) = y_0$$

Eine der einfachsten Möglichkeiten ist das 'Euler-Verfahren'. Man sollte es niemals für Berechnungen verwenden, doch ist es sehr einfach zu verstehen. Wiederum interessiert uns der Funktionswert ein Stückchen h nach der Startzeit.

$$y(t_0 + h) = y_0 + f(t_0, y_0) \cdot h$$

Man tut also so, als wäre über das gesamte Rechenintervall die Änderung der Funktion gleich groß, man ersetzt die Funktion durch ein Stückchen ihrer Tangente.

Genau wie oben gilt: je kleiner h ist, desto genauer wird das Resultat der echten Lösung entsprechen. Für jedes Verfahren lässt sich eine **Ordnung** angeben, die der Hochzahl von h im Fehlerterm entspricht. Das Euler-Verfahren hat Ordnung 1. Eine Halbierung der Schrittweite h halbiert den begangenen Fehler nur. Dafür ist dieses Verfahren sehr einfach und schnell zu berechnen - wir benötigen die Auswertung der Funktion nur ein einziges Mal. Allzu kleine Schritte dürfen wir aber auch nicht machen, sonst kommen Rundungsfehler zum Tragen.

In der Literatur findest Du zwei charakteristische Buchstaben: p für die Ordnung und m für die Anzahl der Zwischenpunktberechnungen.

Das berühmte 'klassische' Runge-Kutta-Verfahren ist RK4(4) und ein besonders ökonomischer Fall: mit vier Berechnungen erhalten wir ein Verfahren vierter Ordnung. Für höhere Ordnungen wird das Verhältnis schlechter - m muss dann immer echt größer als p sein.

RK4(4)

$$k_1 = f(t_0, y_0)$$

$$k_2 = f\left(t_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right)$$

$$k_3 = f\left(t_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right)$$

$$k_4 = f(t_0 + h, y_0 + k_3)$$

$$y(t+h) = y(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \cdot h$$

Betrachten wir die letzte Zeile, so wird der neue Wert aus dem alten plus dem Mittelwert einiger Hilfsauswertungen berechnet.

Seit dem Auftauchen der Computer gibt es zahlreiche derartige Formeln. Die Auswahl der geeigneten Methode ist wichtig. Nicht immer bringt 'hohe Ordnung' auch genaue Ergebnisse (Speziell für physikalische Anwendungen: in der Nähe eines Gravitationszentrums funktioniert niedrigere Ordnung mit kleinerer Schrittweite besser. Details verrät Dein Physiklehrer). Manche Funktionen sind 'widerrspenstig' und besser mit ganz anderen Verfahren zu berechnen ('steife' Systeme).

Zum Trost: Die Anwendung der klassischen Methode ist eigentlich nie ungeschickt.

Eine schöne Eigenschaft der RK-Verfahren ist, dass sie sofort auf Gleichungssysteme übertragbar sind. Damit lassen sich dann auch etwa Differentialgleichungen zweiter Ordnung im Raum lösen, wie etwa die Berechnung der Flugbahn einer Mondlanderakete.

$$\vec{F} = m \cdot \vec{a} = m \vec{x}''(t) = -G \frac{mM}{r^2(t)} \cdot \frac{\vec{x}}{|\vec{x}|}(t), \quad \vec{x}(t_0), \vec{v}(t_0) \text{ gegeben}$$

Dies spalten wir in zwei Teile erster Ordnung auf

$$\vec{x}' = \vec{v}$$

$$\vec{v}'(t) = -G \frac{M}{r^2(t)} \vec{x}(t)_0$$

und erhalten sechs Zeilen der Form

$$y_1' = f_1(t, y_1, y_2, \dots)$$

$$y_2' = f_2(t, y_1, y_2, \dots)$$

usw.

$$y_1(t_0), y_2(t_0), \text{etc. gegeben}$$

Bei der Auswertung der Runge-Kutta-Formel wird nun jede Berechnungszeile eine Schleife über alle sechs Koordinaten sein.

```
def rk4(t,y,f,h):
    n = len(y)
    ynew = [0]*n

    k1 = f(t,y)
    for i in range(n): k1[i] = k1[i]*h

    for i in range(n): ynew[i] = y[i]+k1[i]/2.0
    k2 = f(t+h/2.0,ynew)
    for i in range(n): k2[i] = k2[i]*h

    for i in range(n): ynew[i] = y[i]+k2[i]/2.0
    k3 = f(t+h/2.0,ynew)
    for i in range(n): k3[i] = k3[i]*h

    for i in range(n): ynew[i] = y[i]+k3[i]
    k4 = f(t+h,ynew)
    for i in range(n): k4[i] = k4[i]*h

    for i in range(n): ynew[i] = y[i]+(k1[i]+2.0*k2[i]+2.0*k3[i]+k4[i])/6.0

    return t+h,ynew
```

Alles, was wir beim Aufruf angeben müssen ist die Startzeit t , der Vektor y der Startkoordinaten $[v_x, v_y, v_z, x, y, z]$ und eine Funktion f , die aus den Übergabeparametern $[t, y]$ den Vektor der Änderungen $[F_x, F_y, F_z, v_x, v_y, v_z]$ bildet.