

Numerische Integration von Anfangswertproblemen

Bei einer Differentialgleichung n -ter Ordnung

$$y^{(n)} = f(x, y, y', y'', \dots, y^{(n-1)})$$

enthält die allgemeine Lösung n Integrationskonstanten. Es können n zusätzliche Bedingungen formuliert werden, mit denen diese Integrationskonstanten zu bestimmen sind, so dass sich die spezielle Lösung des durch Differentialgleichung und Zusatzbedingungen beschriebenen Problems ergibt.

Wenn alle Zusatzbedingungen für die gleichen Stelle x_0 gegeben sind (der Funktionswert y und die Ableitungen bis zur $(n - 1)$ -ten Ordnung sind an dieser Stelle vorgeschrieben), dann spricht man von einem **Anfangswertproblem** (im Gegensatz zum **Randwertproblem**, bei dem diese Bedingungen für unterschiedliche x -Werte gegeben sind). Die mit dem Differenzenverfahren zu lösenden Aufgaben der Biegetheorie sind z. B. typische lineare Randwertaufgaben (Differentialgleichungen **und** Randbedingungen sind linear), für die eine geschlossene Lösung prinzipiell möglich ist, auch wenn komplizierte praxisnahe Probleme eine numerische Lösung nahe legen.

Für **nichtlineare Differentialgleichungen** ist eine geschlossene Lösung nur in ganz seltenen Ausnahmefällen möglich. Auch Näherungsmethoden wie das Differenzenverfahren sind nicht praktikabel, weil sich sehr große nichtlineare Gleichungssysteme ergeben würden. Wenn die Aufgabe jedoch als Anfangswertproblem formuliert ist, lassen sich auf numerischem Wege brauchbare Näherungslösungen gewinnen. Glücklicherweise sind gerade viele nichtlineare Probleme der Ingenieur-Mathematik Anfangswertprobleme, für die hier geeignete Näherungsverfahren behandelt werden.

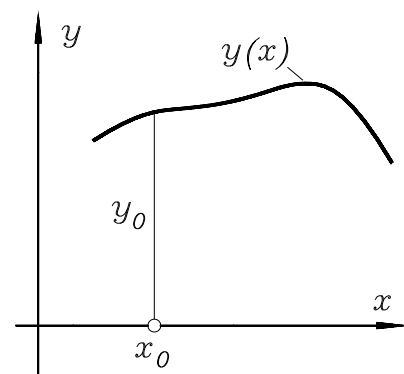
Das Verfahren von EULER-CAUCHY

Die Idee der numerischen Integration soll zunächst am einfachsten Anfangswertproblem mit dem einfachsten Verfahren vorgestellt werden. Für das **Anfangswertproblem 1. Ordnung**

$$y' = f(x, y) \quad , \quad y(x_0) = y_0$$

(eine Differentialgleichung 1. Ordnung und die dazugehörige Anfangsbedingung) ist die Funktion $y(x)$ gesucht, die die Differentialgleichung und die Anfangsbedingung erfüllt (nebenstehende Skizze).

Ausgehend vom einzigen x -Wert, für den der gesuchte y -Wert bekannt ist, dem "Anfangspunkt" x_0 mit dem Wert y_0 , sucht man einen Wert y_1 für die Stelle $x_1 = x_0 + h$ (h ist die "Schrittweite"), um anschließend auf gleiche Weise zum nächsten Punkt zu kommen usw.



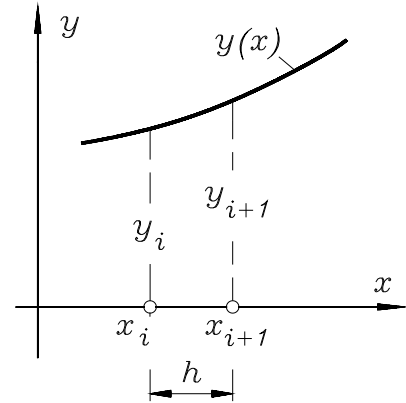
Dieser Prozess sei bis zur Stelle x_i abgelaufen, y_i ist also bekannt. Dann ist das Berechnen von y_{i+1} für $x_{i+1} = x_i + h$ der typische Integrationsschritt des Verfahrens.

Beide Seiten der Differentialgleichung des Anfangswertproblems werden über das Intervall h integriert:

$$\int_{x_i}^{x_{i+1}} y'(x) dx = \int_{x_i}^{x_{i+1}} f(x, y) dx \quad ,$$

$$[y(x)]_{x_i}^{x_{i+1}} = y_{i+1} - y_i = \int_{x_i}^{x_{i+1}} f(x, y) dx \quad ,$$

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y) dx \quad .$$



Das verbleibende Integral auf der rechten Seite, das den Zuwachs des Funktionswertes vom Punkt i zum Punkt $i+1$ repräsentiert, muss näherungsweise gelöst werden, weil die im Integranden enthaltene Funktion $y(x)$ nicht bekannt ist. Die verschiedenen Verfahren der numerischen Integration von Anfangswertproblemen unterscheiden sich im Wesentlichen in der Art und Qualität, wie dieses Integral angenähert wird.

Die grösste Näherung für das Integral ist die Annahme, der Integrand $f(x, y)$ sei im gesamten Integrationsintervall $x_i \leq x \leq x_{i+1}$ konstant und kann durch den Wert $f(x_i, y_i)$ am linken Rand des Integrationsintervalls ersetzt werden (x_i und y_i sind bekannt). Mit

$$\int_{x_i}^{x_{i+1}} f(x, y) dx \approx [x f(x_i, y_i)]_{x_i}^{x_{i+1}} = f(x_i, y_i) (x_{i+1} - x_i) = y_i' h$$

erhält man die

Integrationsformel von EULER-CAUCHY:

$$y_{i+1} = y_i + y_i' h \quad ; \quad x_{i+1} = x_i + h \quad .$$

Dies ist die einfachste Näherungsformel für die numerische Integration eines Anfangswertproblems, die Lösung $y(x)$ wird durch einen Polygonzug approximiert. Die einfache Berechnungsvorschrift verdeutlicht in besonderer Schärfe das Problem aller Integrationsformeln für Anfangswertprobleme: Die Näherungslösung für das Integral erzeugt einen Fehler ("Quadraturfehler"), der in die Berechnung von y' für den nächsten Integrationsschritt eingeht und dabei einen weiteren Fehler (Steigungsfehler) erzeugt.

Andererseits wird auch die Stärke dieser Verfahren deutlich: Eine einfache, immer wieder auf die gerade berechneten Werte angewendete Formel kommt der Programmierung in hohem Maße entgegen. Weil jeder Schritt nur die Ergebnisse seines Vorgängers kennen muss, ist der Speicherplatzbedarf außerordentlich gering, wenn nicht alle berechneten Werte für eine nachfolgende Auswertung gespeichert werden müssen.

An einem einfachen Beispiel soll das Vorgehen demonstriert werden.

Beispiel:

Das Problem, die Funktion $y(x)$ zu bestimmen, die einen Spiegel definiert, der paralleles Licht so ablenkt, dass sich alle Lichtstrahlen in einem Punkt (Fokus) treffen, kann ohne Einschränkung der Allgemeinheit für Lichtstrahlen parallel zur x -Achse und mit dem Nullpunkt als Fokus formuliert werden.

Einige einfache geometrische Überlegungen (nachfolgende Skizze) führen auf die Differenzialgleichung

$$y' = \frac{y}{x + \sqrt{x^2 + y^2}},$$

für die (mit einiger Mühe) die allgemeine Lösung berechnet werden kann, so dass sich die nichtlineare Differenzialgleichung vorzüglich für eine Abschätzung der Genauigkeit einer numerischen Lösung eignet.

Ihre allgemeine Lösung

$$y^2 = C^2 + 2Cx$$

enthält die Integrationskonstante C , die mit einer ziemlich willkürlich festzulegenden Anfangsbedingung bestimmt werden kann. Es gibt unendlich viele Funktionen, die die Forderung der Aufgabenstellung erfüllen, man darf einfach einen Punkt festlegen, durch den die Lösungskurve verlaufen soll. Wählt man z. B. als Anfangsbedingung

$$y(0) = 1,$$

dann erhält man mit $C = 1$ als spezielle Lösung die quadratische Parabel

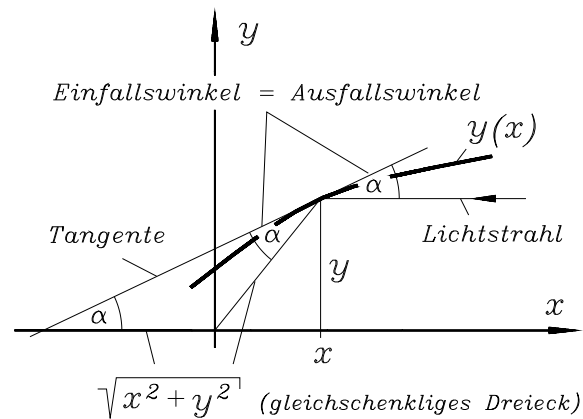
$$y = \pm \sqrt{1 + 2x}$$

("Parabolspiegel"). Die nachfolgende Tabelle zeigt die numerische Lösung nach Euler-Cauchy für das nichtlineare Anfangswertproblem

$$y' = \frac{y}{x + \sqrt{x^2 + y^2}}, \quad y(0) = 1$$

mit der sehr groben Schrittweite $h = 0,1$ im Vergleich mit der exakten Lösung:

i	x_i	y_i	y'_i	$y_{i, \text{exakt}}$
0	0	1,0000	1,0000	1,0000
1	0,1	1,1000	0,9132	1,0954
2	0,2	1,1913	0,8461	1,1832
3	0,3	1,2759	0,7921	1,2649
...
10	1,0	1,7560		1,7321



Schon am Ende des sehr kurzen Integrationsintervalls $x = 0 \dots 1$ zeigt sich eine sichtbare Abweichung, die bei größeren Integrationsintervallen stärker wird, sich durch kleinere Schrittweiten jedoch verringern lässt. Die nachfolgende Tabelle zeigt die Ergebnisse, die sich bei verschiedenen Schrittweiten am Ende des Integrationsintervalls $x = 0 \dots 5$ ergeben:

Schrittweite	$h =$	1,0	0,1	0,01	0,001	Exakte Lösung
Integrationsschritte	$N =$	5	50	500	5000	
$y(5) =$		3,9163	3,3723	3,3221	3,3172	3,3166

Natürlich kann man die Anzahl der Integrationsschritte nicht beliebig erhöhen, weil mit der Anzahl der Rechenoperationen auch der mit jeder Operation unvermeidlich verknüpfte Rundungsfehler das Ergebnis verfälschen wird.

Das nachfolgend angegebene kleine MATLAB-Script (als **ECSpiegel.m** zum Download verfügbar), mit dem diese Berechnung realisiert werden kann, zeigt, wie einfach das Verfahren zu programmieren ist:

```

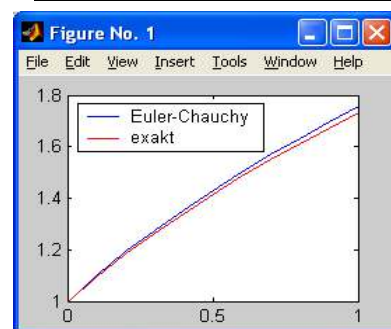
C:\NetObjects Fusion 4.0\User Sites\Tmcu\Dateien\ECSpiegel.m*
File Edit View Text Debug Breakpoints Web Window Help
1  % Differenzialgleichung y' = f(x,y):
2
3  f = inline ('y / (x + sqrt(x^2 + y^2))','x','y') ;
4
5  x0 = 0 ; y0 = 1 ;           % Anfangswert
6  xEnd = 1 ;                 % Ende des Integrationsintervalls
7  n = 10 ;                   % Anzahl der Integrationsschritte
8  h = (xEnd - x0) / n ;
9
10 x = x0 : h : xEnd ;
11 y = zeros (n+1,1) ;
12
13 % Euler-Cauchy-Algorithmus -----
14 y(1) = y0 ;
15
16 for i=1:n
17     y(i+1) = y(i) + h * f (x(i),y(i)) ;
18 end
19 % Euler-Cauchy-Algorithmus -----
20
21 % Exakte Lösung zum Vergleich:
22 C = -x0 + sqrt(x0^2+y0^2) ;
23 yexakt = sqrt (C^2+2*C*x) ;
24
25 y % Ausgabe der Ergebnisse in das Command Window und
26 % gemeinsam mit der exakten Loesung in ein Graphik-Fenster
27 plot (x,y,x,yexakt,'r') ; legend ('Euler-Chauchy','exakt',2) ;
Ready

```

```

Command Window
File Edit View Web Window Help
>>
y =
    1.0000
    1.1000
    1.1913
    1.2759
    1.3551
    1.4299
    1.5009
    1.5686
    1.6335
    1.6958
    1.7560
>> |
Ready

```



Der gesamte EULER-CAUCHY-Algorithmus besteht aus wenigen Zeilen. Die spezielle Differentialgleichung, die gelöst wird, wurde am Anfang als "inline function" definiert, die gegebenenfalls für die Lösung eines anderen Problems ersetzt werden kann. Rechts sieht man das "Command Window" und das Graphik-Fenster mit den Ergebnissen der Rechnung.

- ◆ Bei Berechnung dieses Anfangswertproblem mit negativer Schrittweite (um die Spiegelform auch links vom Fokus zu bestimmen), ist für die Stelle $x = -0,5$ ein Versagen der

Rechnung zu erwarten, weil dort y' unendlich wird. Durch die unvermeidlichen Rundungsfehler bei der Rechnung äußert sich dies unter Umständen "nur" durch unsinnige Ergebnisse.

Dies ist ein generelles Problem bei der Lösung von Differenzialgleichungen in Bereichen, in denen $y' = f(x, y)$ sich "mit y sehr stark ändert". Für die Eindeutigkeit der Lösung einer Differenzialgleichung muss gefordert werden, dass in dem betrachteten Bereich die partielle Ableitung von $f(x, y)$ nach y entsprechend

$$\left| \frac{\partial f(x, y)}{\partial y} \right| \leq k$$

begrenzt ist (so genannte "LIPSCHITZ-Bedingung").

Im Scheitelpunkt der betrachteten Lösungsfunktion ist diese Bedingung nicht erfüllt. Die exakte Lösung verzweigt sich dort auch in einen oberen und einen unteren Zweig.

- ◆ Das Verfahren von EULER-CAUCHY wird nur aus didaktischen Gründen hier so ausführlich behandelt. Für die weitaus meisten praktischen Probleme gibt es keinen Grund, nicht eines der nachfolgend behandelten genaueren Verfahren zu verwenden.

Differenzialgleichungen höherer Ordnung, Differenzialgleichungssysteme

Die EULER-CAUCHY-Formel ist problemlos auf Differenzialgleichungssysteme 1. Ordnung übertragbar, indem sie für jede der zu berechnenden Funktionen aufgeschrieben wird. Für ein Anfangswertproblem mit zwei Differenzialgleichungen 1. Ordnung

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2) & , & & y_1(x_0) &= y_{1,0} & , \\ y_2' &= f_2(x, y_1, y_2) & , & & y_2(x_0) &= y_{2,0} \end{aligned}$$

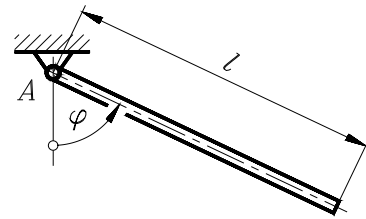
wird die EULER-CAUCHY-Formel in jedem Integrationsschritt zweimal verwendet, so dass $y_{1,i+1}$ und $y_{2,i+1}$ aus $y_{1,i}$ und $y_{2,i}$ berechnet werden können.

Damit ist auch eine Möglichkeit der numerischen Integration von Differenzialgleichungen (und Differenzialgleichungssystemen) höherer Ordnung gegeben: Durch Einführen von zusätzlichen Variablen für die ersten Ableitungen werden Differenzialgleichungen höherer Ordnung in ein Differenzialgleichungssystem 1. Ordnung überführt.

Dies soll am Beispiel einer einfachen Bewegungs-Differenzialgleichung demonstriert werden. Bewegungs-Differenzialgleichungen der Technischen Mechanik sind stets Differenzialgleichungen 2. Ordnung (Beschleunigung ist 2. Ableitung der Wegkoordinate). Für die numerische Integration bedeutet das, dass neben der Wegkoordinate (z. B.: s oder bei Drehbewegungen φ) auch noch die Geschwindigkeit (z. B.: $v = \dot{s}$ bzw. $\omega = \dot{\varphi}$) als Variable auftritt, so dass das Beschleunigungsglied durch die erste Ableitung der Geschwindigkeit ersetzt wird. Die unabhängige Variable in Bewegungs-Differenzialgleichungen ist die Zeit t .

Beispiel:

Ein dünner Stab der Länge l mit konstantem Querschnitt ist an einem Ende reibungsfrei gelagert. Er wird aus der vertikalen Lage um den Winkel φ_0 ausgelenkt und ohne Anfangsgeschwindigkeit freigegeben. Die freie Schwingung wird durch das Anfangswertproblem (vgl. Dankert/Dankert: Technische Mechanik, Seite 529)



$$\ddot{\varphi} = -\frac{3g}{2l} \sin \varphi \quad ; \quad \varphi(t=0) = \varphi_0 \quad ; \quad \dot{\varphi}(t=0) = 0$$

beschrieben (die unabhängige Variable t taucht in der Differentialgleichung gar nicht auf, dies ist sehr häufig bei Bewegungs-Differentialgleichungen). Durch Einführen einer zusätzlichen abhängigen Variablen ω wird aus der Differentialgleichung 2. Ordnung ein Differentialgleichungssystem 1. Ordnung:

$$\begin{aligned} \dot{\omega} &= -\frac{3g}{2l} \sin \varphi & ; & \quad \omega(t=0) = 0 & ; \\ \dot{\varphi} &= \omega & ; & \quad \varphi(t=0) = \varphi_0 & . \end{aligned}$$

Dieses Anfangswertproblem kann zum Beispiel mit dem folgenden EULER-CAUCHY-Formelsatz berechnet werden (zum Problem passend ist die Zeit t die unabhängige Variable, an Stelle der Schrittweite h wird Δt geschrieben und für y_1 und y_2 werden ω bzw. φ verwendet):

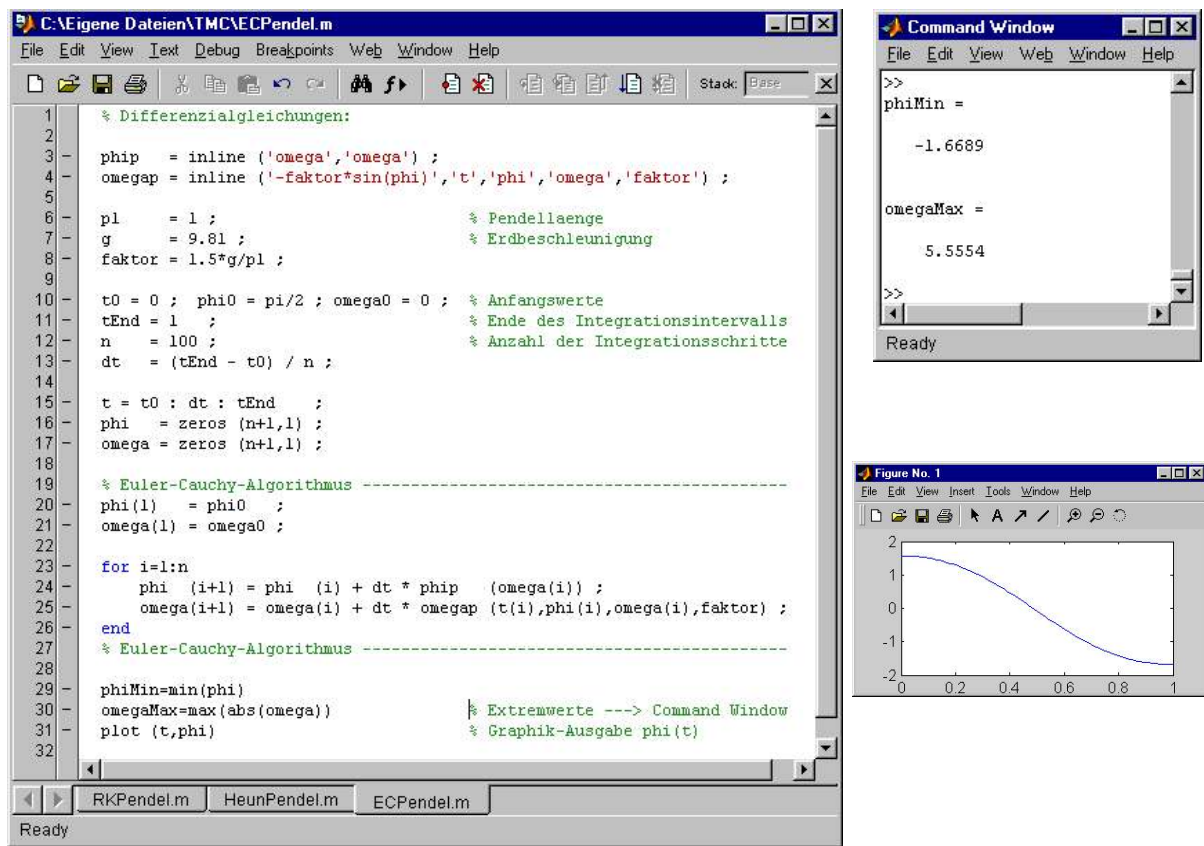
$$\begin{aligned} \omega_{i+1} &= \omega_i + \Delta t \dot{\omega}_i \quad , \\ \varphi_{i+1} &= \varphi_i + \Delta t \dot{\varphi}_i \quad , \\ t_{i+1} &= t_i + \Delta t \quad . \end{aligned}$$

Die Programmierung ist im Vergleich zu einem Problem mit nur einer Differentialgleichung nicht nennenswert aufwendiger (siehe MATLAB-Script auf der nächsten Seite), das als **ECPendel.m** zum Download verfügbar ist.

Im Beispiel wurde mit einer Stablänge $l = 1 \text{ m}$ und einer Anfangsauslenkung $\varphi_0 = \pi/2$ über ein Zeitintervall von nur 1 s gerechnet (es ergibt sich eine halbe Pendelschwingung). Man sieht in diesem Fall dem Ergebnis sofort an, dass es sehr ungenau ist (immerhin wurde das kurze Zeitintervall in 100 Abschnitte unterteilt), denn natürlich kann das Pendel zur anderen Seite nicht über $\varphi = -\pi/2$ hinaus ausschlagen. In der Regel sind solche Ungenauigkeiten dem Ergebnis allerdings nicht sofort anzusehen.

Die Frage, wie man die Ergebnisse bei Problemen kontrollieren kann, für die keine exakten Vergleichslösungen vorliegen (ist der Regelfall, denn sonst bräuchte man ja nicht numerisch zu integrieren), ist ebenso schwierig zu beantworten wie die Frage nach einer geeigneten Schrittweite. An diesem Beispiel sollen einige Möglichkeiten aufgezeigt werden:

- ◆ Man sollte mehrere Rechnungen mit unterschiedlichen Schrittweiten durchführen (man probiere dies für das behandelte Beispiel mit $n = 1000$ und $n = 10000$). Wenn sich die Ergebnisse am Ende des Integrationsintervalls bei halbiertem Schrittweite nicht wesentlich ändern, kann man der Rechnung vertrauen.
- ◆ Für sehr kleine Ausschläge kann die Differentialgleichung linearisiert und exakt gelöst werden. Für die Dauer einer vollen Schwingung erhält man die Formel



$$T = \frac{2\pi}{\sqrt{\frac{3g}{2l}}} = 2\pi \sqrt{\frac{2l}{3g}}.$$

Man sollte eine Testrechnung mit einer kleinen Anfangsauslenkung durchführen und die Ergebnisse vergleichen.

- ◆ Da das Schwingungsproblem ohne Berücksichtigung von Bewegungswiderständen (Reibung, Luftwiderstand) behandelt wird, muss das Pendel bei beliebiger Anfangsauslenkung nach jeder vollen Schwingung wieder die Anfangslage erreichen (sehr gutes Indiz für eine "gesunde" Rechnung, siehe Bemerkung oben).
- ◆ Auch für beliebig große Ausschläge lässt sich die Abhängigkeit der Winkelgeschwindigkeit ω von der Winkelkoordinate φ nach dem Energiesatz exakt berechnen. Speziell liefert diese Betrachtung für die Winkelgeschwindigkeit beim Durchgang durch die tiefste Lage des Pendels ($\varphi = 0$) aus

$$m g \frac{l}{2} (1 - \cos \varphi_0) = \frac{1}{2} \left(\frac{1}{3} m l^2 \right) \omega_{max}^2$$

die maximale Winkelgeschwindigkeit

$$\omega_{max} = \sqrt{\frac{3g}{l} (1 - \cos \varphi_0)},$$

die mit dem numerisch berechneten Wert verglichen werden kann.

Mit den für das berechnete Beispiel gewählten Zahlenwerten wird dieser theoretische Wert

$$\omega_{max} = 5,4249 \text{ s}^{-1}$$

bei einer Einteilung des Intervalls in 100 Abschnitte schon deutlich überschritten und wird erst bei 10000 Abschnitten befriedigend genähert.

Natürlich kann man nicht bei jeder Aufgabe so viele Testmöglichkeiten finden, man sollte jedoch immer bestrebt sein, die Ergebnisse besonders mit "physikalischen und technischen Überlegungen" zu verifizieren.

Verbesserte Integrationsformeln

Prädiktor-Korrektor-Verfahren, das Verfahren von HEUN

Eine Verbesserung der Näherung für das Integral in

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y) dx$$

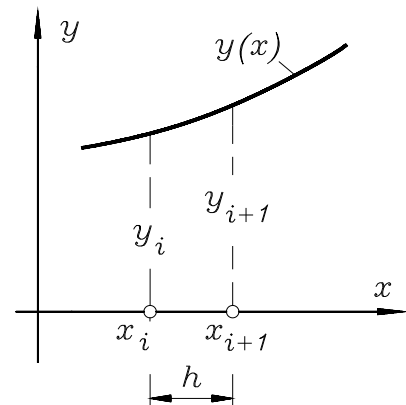
kann nur durch das Einbeziehen weiterer Punkte des Integrationsintervalls h erreicht werden. Wenn der Integrand nicht nur durch einen Funktionswert (am linken Rand des Intervalls wie beim Verfahren von EULER-CAUCHY) ersetzt wird, sondern z. B. auch der Funktionswert am rechten Rand $f(x_{i+1}, y_{i+1})$ in die Näherung einbezogen wird, kann der Integrand als linear veränderliche Größe angenähert werden ("Rechteck"-Näherung wird zur deutlich besseren "Trapez"-Näherung). Mit

$$\int_{x_i}^{x_{i+1}} f(x, y) dx \approx \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1})}{2} (x_{i+1} - x_i) = (y'_i + y'_{i+1}) \frac{h}{2}$$

gelangt man zur wesentlich besseren Integrationsformel

$$y_{i+1} = y_i + (y'_i + y'_{i+1}) \frac{h}{2},$$

die allerdings nicht ohne Vorleistung anwendbar ist, denn auf der rechten Seite geht über $y'_{i+1} = f(x_{i+1}, y_{i+1})$ der Wert y_{i+1} ein, der mit dieser Formel erst ermittelt werden soll. Man berechnet deshalb einen vorläufigen Näherungswert (**Prädiktor**) nach der EULER-CAUCHY-Formel, der dann eine (gegebenenfalls mehrfache) Verbesserung nach der verbesserten Formel erfährt (**Korrektor**-Schritte). Dies ist das



Verfahren von HEUN:

$$\begin{array}{rcl}
 \text{Prädiktor:} & p_{i+1} = y_i + y'_i h & ; \quad x_{i+1} = x_i + h \\
 & \downarrow & \\
 p_{i+1} = y_{i+1} & \Rightarrow & y'_{i+1} = f(x_{i+1}, p_{i+1}) \\
 \uparrow & & \downarrow \\
 \leftarrow \leftarrow \leftarrow \leftarrow & & y_{i+1} = y_i + \left(y'_i + y'_{i+1} \right) \frac{h}{2} \quad (\text{Korrektor})
 \end{array}$$

Das Verfahren kann mit einer festen Anzahl von Korrektorschritten arbeiten oder aber einen Integrationsschritt erst dann beenden, wenn sich y_{i+1} nicht mehr ändert. Wie das EULER-CAUCHY-Verfahren kann auch das Verfahren von HEUN auf ein System von Differentialgleichungen angewendet werden, indem für jede Differentialgleichung der angegebene Algorithmus ausgeführt wird.

Die Programmierung ist nicht wesentlich aufwändiger als beim Verfahren von EULER-CAUCHY. Für das im vorigen Abschnitt behandelte Pendelproblem (2 Differentialgleichungen) wird nebenstehend das modifizierte MATLAB-Script gezeigt (ist als **HeunPendel.m** zum Download verfügbar). In jedem Integrationsschritt werden nach dem Prädiktorschritt genau 3 Korrektorschritte ausgeführt.

Wenn man mit dem auf diese Weise modifizierten Programm die gleichen Testrechnungen ausführt, die mit dem Programm nach EULER-CAUCHY durchgeführt wurden, stellt man fest, dass schon bei wesentlich groberer Schrittweite die Genauigkeit erreicht wird, die nach EULER-CAUCHY eine sehr feine Intervalleinteilung erfordert.

```

1 % Differenzialgleichungen:
2
3 phip = inline ('omega','omega');
4 omegap = inline ('-faktor*sin(phi)','t','phi','omega','faktor');
5
6 pl = 1; % Pendellaenge in m
7 g = 9.81; % Erdbeschleunigung in m/s^2
8 faktor = 1.5*g/pl;
9
10 t0 = 0; phi0 = pi/2; omega0 = 0; % Anfangswerte
11 tEnd = 1; % Ende des Integrationsintervalls
12 n = 100; % Anzahl der Integrations Schritte
13 dt = (tEnd - t0) / n;
14
15 t = t0 : dt : tEnd;
16 phi = zeros (n+1,1);
17 omega = zeros (n+1,1);
18
19 % Heun-Algorithmus -----
20 phi(1) = phi0;
21 omega(1) = omega0;
22
23 for i=1:n
24     phipi = phip (omega(i));
25     ompi = omegap (t(i),phi(i),omega(i),faktor);
26     % * Prädiktorschritt nach Euler-Cauchy: *****
27     prphi = phi (i) + dt * phipi;
28     prom = omega(i) + dt * ompi;
29     for j=1:3 % * Drei Korrektorschritte: *****
30         prphi = phi (i) + (phipi + phip(prom))*dt/2;
31         prom = omega(i) + (ompi + omegap(t(i+1),prphi,prom,faktor))*dt/2;
32     end
33     phi (i+1) = prphi; % ... die endgueltigen Werte des Schritts
34     omega(i+1) = prom;
35 end
36 % Heun-Algorithmus -----
37
38 phiMin=min(phi), omegaMax=max(abs(omega)) % Extremwerte ---> Command Window
39 plot (t,phi) % Graphik-Ausgabe phi(t)
40

```

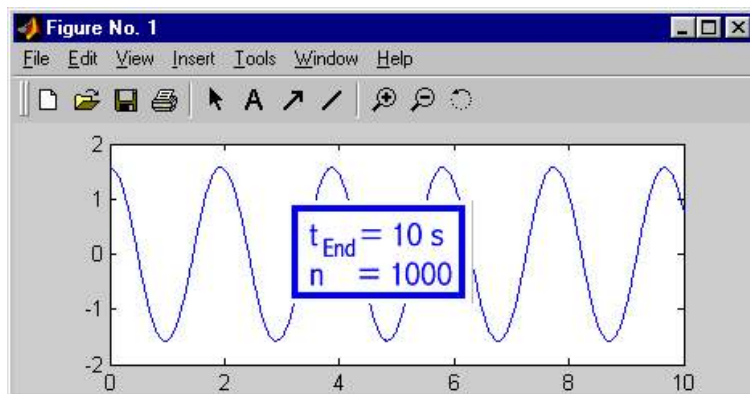
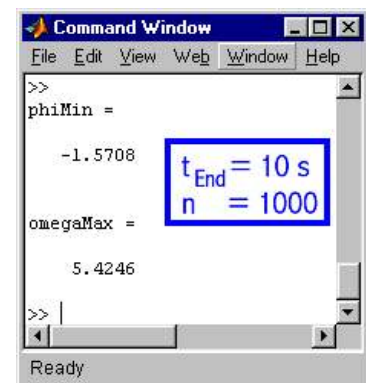
Bei gleicher Schrittweite (100 Abschnitte) und dem gleichen Integrationsintervall (1 s, dies sind die Werte, die im oben gelisteten Script zu sehen sind) zeigt das nebenstehend zu sehende "Command Window" etwa die zu erwartenden Ergebnisse.

Bei einer Einteilung des Integrationsintervalls in 1000 Abschnitte kann man durchaus über ein Integrationsintervall von **10 s** integrieren. Die nachfolgend zu sehenden Graphik-Fenster und "Command Window" bestätigen, dass auch diese Rechnung noch "gesund" ist:



```

Command Window
File Edit View Web Window Help
>>
phiMin =
    -1.5707
omegaMax =
    5.4242
>>
Ready
  
```

```

Command Window
File Edit View Web Window Help
>>
phiMin =
    -1.5708
omegaMax =
    5.4246
>>
Ready
  
```

RUNGE-KUTTA-Verfahren

Mit den Verfahren von EULER-CAUCHY und HEUN wurden zwei einfache Vertreter einer kaum zu überblickenden Anzahl von Integrationsverfahren vorgestellt, an denen aber die typischen Probleme sichtbar werden, die der Anwender beachten muss. Die verschiedenen Verfahren unterscheiden sich im Wesentlichen in der Anzahl der Funktionswertberechnungen für den Integranden in einem Integrationsschritt (beeinflusst die Qualität der Näherung des Integrals), in der Strategie der Berechnung von y_{i+1} (feste oder variable Anzahl von Operationen) und in der Festlegung der Schrittweiten h für die Integrationsschritte (feste oder variable Schrittweite).

Auf dieses weite Feld kann hier nicht weiter eingegangen werden. Nachfolgend werden nur noch die Formeln eines besonders häufig verwendeten Verfahrens einer ganzen Verfahrensklasse angegeben (ist z. B. mit dieser Variante auch im Programm MCALCU des Programmsystems CAMMPUS realisiert). Es ist ein Vertreter der **RUNGE-KUTTA-Algorithm**en, die aus Funktionswerten für den Integranden in

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y) dx$$

an verschiedenen Punkten des Integrationsintervalls h den Wert für y_{i+1} am rechten Intervallrand so ermitteln, dass die Genauigkeit der Berücksichtigung möglichst vieler Glieder der Taylorreihenentwicklung der Lösung entspricht. Der folgende Formelsatz, der für einen Integrationsschritt vier

Funktionswerte des Integranden berechnet (vgl. Dankert/Dankert: Technische Mechanik, Seite 498), ist ein

RUNGE-KUTTA-Verfahren 4. Ordnung:

$$y_{i+1} = y_i + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4) \quad , \quad x_{i+1} = x_i + h$$

mit

$$k_1 = f(x_i, y_i) \quad ,$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right) \quad ,$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right) \quad ,$$

$$k_4 = f(x_i + h, y_i + hk_3) \quad .$$

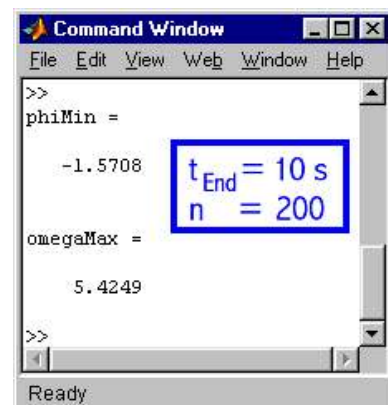
Bei einem Verfahren 4. Ordnung (Übereinstimmung mit den ersten 5 Gliedern der Taylorreihen-Entwicklung) entsteht in jedem Integrationsschritt ein Fehler in der Größenordnung h^5 , der bei genügend kleiner Schrittweite sehr klein ist, andererseits reagiert das Verfahren bei zu großer Schrittweite sehr empfindlich. Gerade für die Runge-Kutta-Algorithmen gibt es eine recht ausgefeilte Theorie zur geeigneten Schrittweitenwahl (und damit auch der Schrittweitenänderung während der Rechnung), die allerdings den gravierenden Mangel hat, dass sie sichere Werte nur dann liefert, wenn die Lösung bekannt ist.

Bei der numerischen Integration eines Anfangswertproblems ist die Wahl einer geeigneten Schrittweite h ebenso wichtig wie schwierig.

Dem Praktiker kann deshalb als effektives Verfahren zur Beurteilung der Qualität einer Rechnung nur empfohlen werden, eine zusätzliche Kontrollrechnung mit halber Schrittweite (und doppelter Anzahl von Integrationsschritten) auszuführen. Die Übereinstimmung beider Lösungen (bei einer vertretbaren Toleranz) ist das sicherste Kriterium für die Bestätigung der Schrittweitenwahl.

Das nachfolgende MATLAB-Script (ist als **RKPendel.m** zum Download verfügbar) zeigt, dass die Programmierung des Runge-Kutta-Algorithmus ebenso einfach ist wie die Programmierung der einfacheren Algorithmen.

Schon bei einer Einteilung des Berechnungsintervalls von **10 s** in 200 Abschnitte erhält man praktisch exakte Ergebnisse (vgl. das nebenstehend zu sehende "Command Window" und die graphische Darstellung der Funktion $\varphi(t)$ auf der folgenden Seite).



```

Command Window
File Edit View Web Window Help
>>
phiMin =
    -1.5708
omegaMax =
    5.4249
>>
Ready
  
```

```

C:\Eigene Dateien\TMC\RKPendel.m
File Edit View Text Debug Breakpoints Web Window Help
Stack: Base
1 % Differenzialgleichungen:
2
3 phip = inline ('omega','omega') ;
4 omegap = inline ('-faktor*sin(phi)','t','phi','omega','faktor') ;
5
6 pl = 1 ; % Pendellaenge
7 g = 9.81 ; % Erdbeschleunigung
8 faktor = 1.5*g/pl ;
9
10 t0 = 0 ; phi0 = pi/2 ; omega0 = 0 ; % Anfangswerte
11 tEnd = 10 ; % Ende des Integrationsintervalls
12 n = 200 ; % Anzahl der Integrationschritte
13 dt = (tEnd - t0) / n ;
14
15 t = t0 : dt : tEnd ;
16 phi = zeros (n+1,1) ;
17 omega = zeros (n+1,1) ;
18
19 % Runge-Kutta-Algorithmus -----
20 phi(1) = phi0 ;
21 omega(1) = omega0 ;
22
23 for i=1:n
24     k1phi = phip (omega(i)) ;
25     k1om = omegap (t(i),phi(i),omega(i),faktor) ;
26     k2phi = phip (omega(i)+k1om*dt/2) ;
27     k2om = omegap (t(i)+dt/2,phi(i)+k1phi*dt/2,omega(i)+k1om*dt/2,faktor) ;
28     k3phi = phip (omega(i)+k2om*dt/2) ;
29     k3om = omegap (t(i)+dt/2,phi(i)+k2phi*dt/2,omega(i)+k2om*dt/2,faktor) ;
30     k4phi = phip (omega(i)+k3om*dt) ;
31     k4om = omegap (t(i+1),phi(i)+k3phi*dt,omega(i)+k3om*dt,faktor) ;
32     phi (i+1) = phi (i) + (k1phi + 2*k2phi + 2*k3phi + k4phi) * dt/6 ;
33     omega(i+1) = omega(i) + (k1om + 2*k2om + 2*k3om + k4om) * dt/6 ;
34 end
35 % Runge-Kutta-Algorithmus -----
36
37 phiMin=min(phi) , omegaMax=max(abs(omega)) % Extremwerte ---> Command Window
38 plot (t,phi) % Graphik-Ausgabe phi(t)
39
ECPendel.m HeunPendel.m RKPendel.m
Ready

```

RUNGE-KUTTA-Verfahren 4. Ordnung: Bewegungsverlauf eines pendelnden Stabes, der seine Bewegung aus der horizontalen Lage ohne Anfangsgeschwindigkeit beginnt (numerische Integration bei Einteilung des Zeitintervalls von **10 s** in 200 Abschnitte).

