Kapitel 3

Lösung der Newtonschen Bewegungsgleichung

3.1 Der Harmonische Oszillator

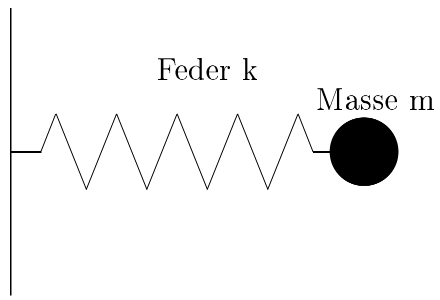

Abbildung 3.1: Schwingungsfeder

Wir gehen davon aus, daß die Masse punktförmig (die Federmasse ist vernachlässigt), die Gravitation gleich Null ist und die Bewegung reibungslos. Dann gilt das Newtonsche Bewegungsgesetz für den harmonischen Oszillator:

$$m\ddot{x} = -kx \tag{3.1}$$

Dabei ist k die Federkonstante. Es gelten die Anfangsbedingungen $x(0) = x_0$ und $\dot{x}(0) = v_0$. Die Lösung dieser Gleichung ist

$$x(t) = A\sin(\omega t + \delta) \qquad \text{mit} \quad \omega^2 = \frac{k}{m} \tag{3.2}$$

Die Anfangsbedingungen ergeben sich durch das Einsetzen von $t = 0$ zu $x_0 = A \sin\delta$ und $v_0 = A\omega \cos\delta$. Daraus werden dann die Konstanten A und δ berechnet:

$$\tan\delta = \frac{x_0\omega}{v_0} \quad \text{und} \quad A = \frac{x_0}{\sin\delta} \tag{3.3}$$

Um eine Funktion in einem Programm verwenden zu können, muß sie diskretisiert werden, d.h. eine kontinuierliche Kurve wird in Teilstücke zerlegt. Dafür benutzen wir ein Feld `x[i]`, wobei `i` nur ganzzahlige Werte annehmen kann. Unsere Zeit t berechnet sich dann durch $t =$`i`dt (dt ist das Zeitintervall zwischen zwei Stützstellen). Damit entspricht der Übergang $t \to t + dt$ dem Übergang `i` $\to$ `i+1`.

Das folgende Programm gibt die exakte Lösungsgleichung (2) wieder:

```
#include<iostream>
using std::cin;
using std::cout;
#include<fstream>
using std::ostream;
#include<cmath>

int main()
{
// define x(t)
   double x[1000], t;  // output-variables

// parameter definition
   double A, delta;     // A=amplitude, delta=phase-angle
   double mass, kfeder; // mass=mass, kfeder=spring-const.
   double t_max, dt;    // t_max=total time, dt=time-interval

// read parameters from std::cin
   cout << "Amplitude      ";  cin >> A;
   cout << "Phasenwinkel   ";  cin >> delta;
   cout << "Masse          ";  cin >> mass;
   cout << "Federkonstante ";  cin >> kfeder;
   cout << "Gesamtzeit     ";  cin >> t_max;
   cout << "Zeitintervall  ";  cin >> dt;
   double omega=sqrt(kfeder/mass);

// loop from t=0 to t=t_max
   t=0.0;
   for( int i=0; i<=t_max/dt; i++ ) {
        x[i]=A*sin(omega*t+delta);  // calculate the funktion
        t=t+dt;                     // increase time
```

```
   }

// output 2 columns to file harmon.dat
   ofstream outfile("harmon.dat");
   t=0.0;
   for( int i=0; i<=t_max/dt; i++ ) {
      outfile << t << " " << x[i] << "\n";
      t=t+dt;
   }
   outfile.close();                  // close file
}
```

Die Funktion plottet man dann am besten mit dem Programm `gnuplot` und dem Befehl `plot 'harmon.dat'`. Auf dem Bildschirm sollte dann die in Abb. 3.2 dargestellte Zeichnung erscheinen.

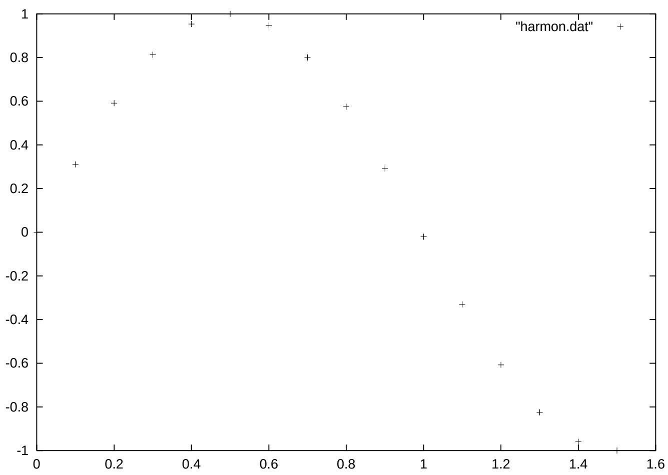

Abbildung 3.2: Ausdruck von gnuplot für $A = 1$, $\delta = 0$, $m = 1$, $k = 10$, $t_{\max} = 1.5$, und $\Delta t = 0.1$, direkt aus der analytischen Lösung berechnet.

3.1.1 Das Euler-Verfahren

Das Euler-Verfahren ist die einfachste Möglichkeit eine Differentialgleichung numerisch zu lösen. Zunächst muß man die Gleichung umformen. Das Newtonsche Bewegungsgesetz lautet $\ddot{x} = -\omega^2 x$. Mit der Geschwindigkeit $v = \dot{x}$ ensteht die Gleichung $\dot{v} = -\omega^2 x$. Die Ableitungen von x und v können für $\Delta t \to 0$ durch den Differenzenquotienten dargestellt werden:

$$\dot{x} = \frac{x(t+\Delta t) - x(t)}{\Delta t} \quad \Rightarrow \quad x(t+\Delta t) = x(t) + v\Delta t \tag{3.4}$$

$$\dot{v} = \frac{v(t+\Delta t) - v(t)}{\Delta t} \quad \Rightarrow \quad v(t+\Delta t) = v(t) - \omega^2 x\Delta t \tag{3.5}$$

Mit Hilfe dieser Gleichungen kann nun eine numerische Approximation im Computer durchgeführt werden. Zuerst werden die Formeln wieder diskretisiert und anschließend wird der Einfachheit halber der Index `i` um Eins verringert. Dies entspricht der Berechnung der aktuellen Werte `i` aus den alten Werten `i-1`. Die Formeln haben einen Fehler in der Größenordnung $O(\Delta t^2)$. Dieser Fehler wirkt sich im Verlauf des Verfahrens durch ansteigende Energie aus. Die Gesamtenergie ist wegen Energieerhaltung aber konstant (Energie = kinetische Energie + Federenergie): $E = \frac{1}{2}mv^2 + \frac{1}{2}kx^2$. Zur Kontrolle der Energieerhaltung wird die Gesamtenergie E ebenfalls im Programm diskret berechnet.

```
// ...
   const int nx=1000;                                    // fieldlength
   double x[nx], v[nx], e[nx], t;
// ...
// make sure that the number of intervals is smaller than nx
   if( t_max/dt >= nx) {
      cout << "ERROR: dt is too small, set to: " << t_max/nx << '\n';
      return -1;
   }
// setup initial conditions
   x[0]=A*sin(delta);
   v[0]=A*omega*cos(delta);
// loop from t=0 to t=t_max
   for( int i=1; i<=t_max/dt; i++ ) {
        v[i]=v[i-1]-omega*omega*x[i-1]*dt;               // next velocity
        x[i]=x[i-1]+v[i-1]*dt;                           // next position
        e[i]=0.5*(mass*v[i]*v[i]+kfeder*x[i]*x[i]);
   }
// output to file
   ofstream outfile("eulerh.dat");
   t=0.0;
   for( int i=0; i<=t_max/dt; i++ ) {
      outfile << t << " " << x[i] << ' ' << v[i] << ' ' << e[i] << "\n";
      t=t+dt;
   }
   outfile.close();
```

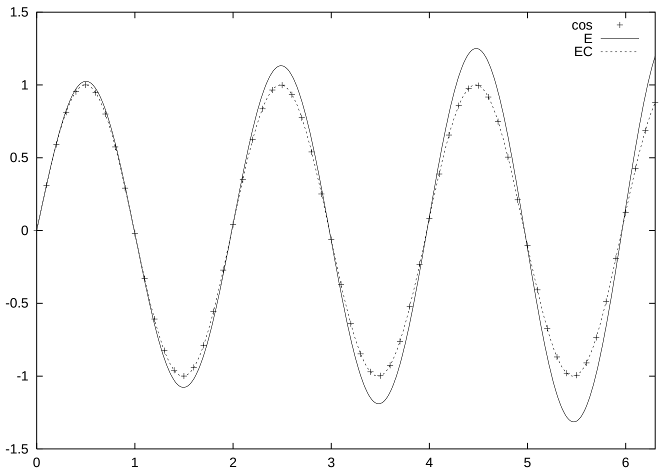

Abbildung 3.3: Ausdruck von gnuplot für $A = 1$, $\delta = 0$, $m = 1$, $k = 10$, $t_{\max} = 6.3$, und $\Delta t = 0.01$, berechnet mit dem Euler-Verfahren (E) im Vergleich mit der exakten Lösung aus Abb. 3.2, und dem Euler-Cromer Verfahren (EC).

3.1.2 Die Euler-Cromer-Methode

Eine besseres Ergebnis erhält man mit der Euler-Cromer-Methode. In der Zeile:

```
x[i]=x[i-1]+v[i-1]*dt;            // next position
```

ersetzt man den alten Wert `v[i-1]` durch den neuen Wert `v[i]`, der ja gerade eben berechnet wurde. Dies ist erlaubt, da durch die Diskretisierung zu jedem Wert ein Bereich von Δt gehört. In diesem Fall ist die numerische Lösung viel besser, wie man leicht sehen kann wenn man die analytische Lösung mit der numerischen vergleicht. Ausserdem ist beim Euler-Cromer-Verfahren die Energie (bis auf kleine Schwankungen) besser erhalten.

3.2 Das Pendel

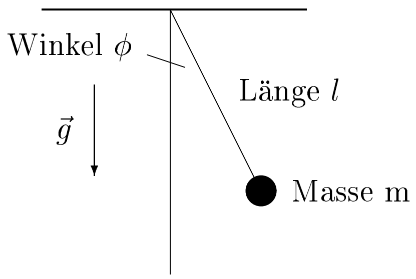

Abbildung 3.4: Das Pendel mit masseloser Aufhängung

Bei einem Pendel ist die (punktförmige) Masse m an einem starren, masselosen Stab der Länge l aufgehängt und schwingt reibungsfrei im Schwerefeld $\vec{g}$. $\varphi(t)$ ist der Auslenkwinkel des Pendels zur Zeit t aus der Ruhelage heraus. Um die Verlaufsfunktion aufzustellen benutzen wir den Energieerhaltungssatz: Die Gesamtenergie E, besthend aus kinetischer und potentieller Energie bleibt erhalten, d.h. $\dot{E} = 0$. Der maximale Auslenkwinkel ist dann $\varphi_{\max}$, mit der zugehörigen Winkelgeschwindigkeit $\omega(\varphi_{\max}) = \dot{\varphi}_{\max} = 0$. Die Gesamtenergie ist also

$$E = \frac{1}{2} m l^2 \dot{\varphi}^2 - mgl\cos\varphi = -mgl\cos\varphi_{\max} \; . \tag{3.6}$$

Die analytische Lösung erhält man durch Auflösen nach $\dot{\varphi}$

$$\dot{\varphi}^2 = \frac{2g}{l}(\cos\varphi - \cos\varphi_{\max}) \; , \tag{3.7}$$

Wurzel ziehen und Inversion

$$\frac{dt}{d\varphi} = \frac{1}{\sqrt{\frac{2g}{l}(\cos\varphi - \cos\varphi_{\max})}} \; , \tag{3.8}$$

Trennung der Variablen t und φ

$$dt = \frac{d\varphi}{\sqrt{\frac{2g}{l}(\cos\varphi - \cos\varphi_{\max})}} \; , \tag{3.9}$$

und anschließender Integration von der Zeit t_0 bis t und den entsprechenden Winkeln φ_0 bis $\varphi(t)$

$$t(\varphi) - t_0 = \sqrt{\frac{l}{2g}} \int_{\varphi_0}^{\varphi(t)} \frac{d\psi}{\sqrt{\cos\psi - \cos\varphi_{\max}}} \; . \tag{3.10}$$

Somit kann man dann die Schwingungsdauer T des Pendels bestimmen. Für eine Periode benötigt das Pendel viermal die Dauer der Bewegung von $\varphi_0 = 0$ nach $\varphi_{\max}$. Damit ergibt sich

$$T = 4\sqrt{\frac{l}{2g}} \int_{0}^{\varphi_{\max}} \frac{d\psi}{\sqrt{\cos\psi - \cos\varphi_{\max}}} \; . \tag{3.11}$$

Der Integrand divergiert, sobald $\psi \to \varphi_{\max}$. Das Integral ist ein sog. elliptisches Integral ohne geschlossene analytische Lösung. In einem der folgenden Kapitel wird die numerische Integration solcher Funktionen behandelt werden. Die Lösungen der elliptischen Integrale sind für die verschiedenen Gattungen tabelliert. Gleichung (3.11) läßt sich durch Variablensubstitution $\sin\psi^* = \frac{\sin(\psi/2)}{\sin(\varphi_{\max}/2)}$ auf ein elliptisches Integral erster Gattung in der Legendreschen Normalform zurückführen:

$$F(k,\varphi) = \int_0^{\varphi} \frac{d\psi}{\sqrt{1 - k^2 \sin^2\psi}} \; , \tag{3.12}$$

wobei $k = \sin(\varphi_{\max}/2)$ ist. Ein weiteres Beispiel ist die Normalform der 2.Gattung

$$E(k,\varphi) = \int_0^\varphi d\psi \sqrt{1 - k^2 \sin^2 \psi} \tag{3.13}$$

Für die numerische Lösung der Pendelbewegung werden wir die Energieerhaltungsgleichung (3.6) umformen, um wieder die Gleichung eines harmonischen Oszillators zu erhalten. Durch Ableitung von Glg. (3.6) ergibt sich

$$\dot{E} = 0 = ml^2 \dot{\varphi}\ddot{\varphi} + mgl\dot{\varphi}\sin\varphi \tag{3.14}$$

und nach Umformung erhält man die nichtlineare Gleichung:

$$\ddot{\varphi} = -\frac{g}{l}\sin\varphi \ . \tag{3.15}$$

In der Nährung für kleine $\varphi \ll 1$ ist $\sin\ \varphi \approx \varphi$ und man erhält die lineare Differentialgleichung

$$\ddot{\varphi} = -\frac{g}{l}\varphi \ , \tag{3.16}$$

die mathematisch identisch zu Gleichung (3.1) ist. Diese kann durch einfaches Umbenennen der Parameter wieder numerisch mit den Verfahren aus dem letzten Abscnitt gelöst werden. Die Anfangsbedingungen seien z.B. $\varphi(0) = \varphi_{\max}$ und $\dot{\varphi}(0) = 0$.

3.2.1 Die Verlet-Methode

Ein verbessertes Verfahren, zur Lösung von Gleichung (3.15) ist die Verlet-Methode. Sie ist allerdings nur gültig, wenn die Kräfte unabhängig von der Geschwindigkeit sind. Zur Herleitung schreibt man die Taylor-Entwicklung von $\varphi(t+\Delta t)$ und $\varphi(t-\Delta t)$ auf:

$$\varphi(t+\Delta t) = \varphi(t) + \Delta t\omega(t) + \frac{\Delta t^2}{2}a(t) + \cdots \tag{3.17}$$

$$\varphi(t-\Delta t) = \varphi(t) - \Delta t\omega(t) + \frac{\Delta t^2}{2}a(t) + \cdots \tag{3.18}$$

wobei $a(t) = \dot{\omega}(t) = -\frac{g}{l}\sin\varphi$ verwendet wurde. Addiert man die beiden Gleichungen (3.17) und (3.18) ergibt sich

$$\varphi(t+\Delta t) = 2\varphi(t) - \varphi(t-\Delta t) + \Delta t^2 a(t) + \cdots \tag{3.19}$$

Der Vorteil liegt darin, daß durch die Addition die ungeraden Potenzen von Δt wegfallen und damit der Fehler auf die Größenordnung $O(\Delta t^4)$ schrumpft, anstatt wie zuvor bei $O(\Delta t^2)$ zu liegen. Für die Formel benötigt man nun zwei Werte der $\varphi(t)$–Funktion, den Anfangswert $\varphi(0) = \varphi_0$ und z.B. den Wert $\varphi(-\Delta t)$, welcher sich wiederum durch den Differenzenquotienten $\dot{\varphi}(0) = \frac{\varphi_0 - \varphi(-\Delta t)}{\Delta t}$ näherungsweise errechnet zu:

$$\varphi(-\Delta t) = \varphi_0 - \Delta t\dot{\varphi}_0 \ . \tag{3.20}$$

Wir haben wieder das übliche Berechnungsprogramm für die numerische Lösung:

```
   const int nx=10001;   // fieldlength
   double p[nx], t;

// parameter definition
   double g, l, m;     // g=gravity, l=length of pendulum, m=mass
   double p0, dp0;     // p0=initial angle, dp0=initial angular velocity
   double t_max, dt;   // t_max=total time, dt=time step

// read parameters
   cout << "Gravitation    ";  cin >> g;
   cout << "Pendellaenge   ";  cin >> l;
   cout << "Masse          ";  cin >> m;
   cout << "Anf.Auslenkung ";  cin >> p0;
   cout << "Anf.Geschw.    ";  cin >> dp0;
   cout << "Gesamtzeit     ";  cin >> t_max;
   cout << "Zeitintervall  ";  cin >> dt;
   double c=-(g/l)*dt*dt;

// make sure that the number of intervals is smaller than nx
   if( t_max/dt >= nx) {
      cout << "ERROR: dt is too small, set to: " << t_max/nx << '\n';
      return -1;
   }

// setup initial conditions
   p[0]=p0-dt*dp0;
   p[1]=p0;
// loop from t=0 to t=t_max
   for( int i=2; i<t_max/dt; i++ ) {
        p[i]=2.*p[i-1]-p[i-2]+c*sin(p[i-1]);
   }

// output to file
   ofstream outfile("verletp.dat");
   t=0.0;
   for( int i=0; i<=t_max/dt; i++ ) {
      outfile << t << ' ' << p[i+1] << '\n';
      t=t+dt;
   }
   outfile.close();
```

Dabei wird der Anfangsindex auf `i=2` verschoben, um negative Indizes zu vermeiden. Mit den Parametern $g = 10$, $l = m = 1$, $\varphi_0 = 0.1$, $\dot{\varphi}_0 = 0$, $t_{\max} = 7.5$ und $\Delta t = 0.01$, erhält man bereits das Bild einer harmonischen Schwingung. Setzt man die Anfangsbedingung höher, z.B. $\varphi_0 = 3$, so ergibt sich ein deutlich anderes Schwingverhalten, das nicht mehr durch einen einfachen Cosinus genähert werden kann, wie in Abb. 3.5 zu sehen.

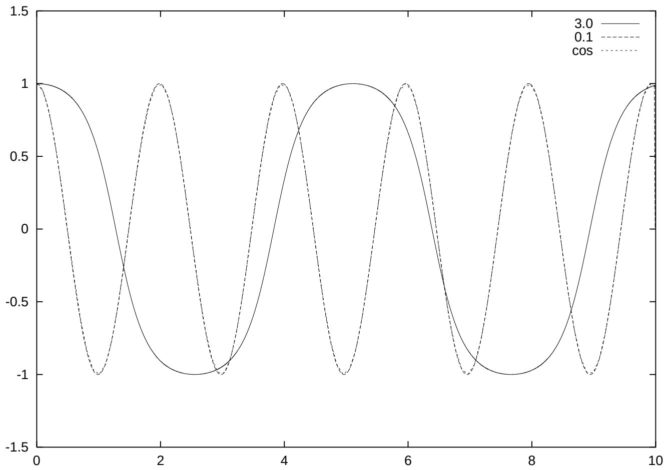

Abbildung 3.5: Ausdruck von $\varphi(t)/\varphi_0$ für $g = 10$, $l = 1$, $m = 1$, $\dot{\varphi}_0 = 0$, $t_{\max} = 10$ und $\Delta t = 0.01$, berechnet mit dem Verlet-Verfahren für $\varphi_0 = 0.1$ und $\varphi_0 = 3$, verglichen mit $\cos(\sqrt{10}t)$.

3.2.2 Das allgemeine Pendel

Zu dem speziellen Pendel aus dem vorherigen Kapitel fügen wir noch eine zur Geschwindigkeit proportionale Reibungskraft $-q\dot{\varphi}$ und eine treibende Kraft $F = F_0 \sin(\Omega t)$ hinzu, wobei die Frequenz Ω nicht gleich der Eigenfrequenz des Pendels sein darf ($\Omega \neq \omega_0 = \sqrt{\frac{g}{l}}$). Somit existieren die zusätzlichen Parameter q, F_0 und Ω. Das dazugehörige Bewegungsgesetz lautet

$$\ddot{\varphi} = -\frac{g}{l}\sin\varphi - q\dot{\varphi} + F_0 \sin\Omega t \tag{3.21}$$

Zur Lösung dieser Gleichung ist das Verlet-Verfahren nicht mehr geeignet, da eine Kraft von der Geschwindigkeit abhängt. Deswegen wird ein neues Verfahren zur numerischen Lösung von Gleichungen vorgestellt.

3.2.3 Das Runge-Kutta-Verfahren

Als Beispiel wird jetzt von einer Gleichung $\dot{\mathbf{x}} = f[\mathbf{x}, t]$ ausgegangen, wobei $\mathbf{x}$, einen Vektor, bestehend aus verschiedenen Größen, darstellen kann. Beim Euler-Verfahren wurde der

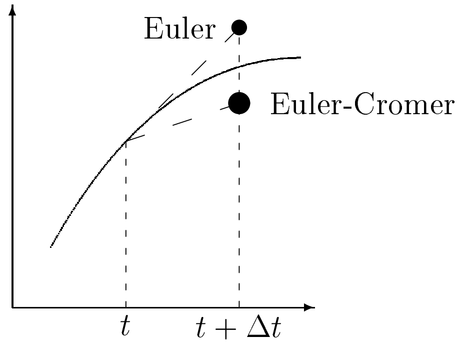

Abbildung 3.6: Fehler beim Euler- / Euler-Cromer-Verfahren

nächste Kurvenstützpunkt mit Hilfe der Steigung des alten Stützpunktes, beim Euler-Cromer-Verfahren mit der Steigung des neuen Stützpunktes berechnet. Beide Verfahren besitzen einen relativ großen Fehler (siehe Abb. 3.6).

Um die Genauigkeit zu verbessern wird beim Runge-Kutta-Verfahren die Steigung in der Mitte des Intervalls bestimmt. Der Funktionswert $\bar{x}$ in der Intervallsmitte wird durch eine Taylorentwicklung berechnet

$$\bar{x} = x\left(t + \frac{\Delta t}{2}\right) = x(t) + \frac{1}{2}\Delta t\ f[x(t), t]. \tag{3.22}$$

Dieser Zwischenwert wird dann in die allgemeine Formel eingesetzt.

$$x(t + \Delta t) = x(t) + \Delta t\ f\left[x\left(t + \frac{\Delta t}{2}\right), t + \frac{\Delta t}{2}\right] \tag{3.23}$$

Für die Anwendung auf das Pendel benötigen wir wieder die beiden Gleichungen

$$\dot{\omega} = f[\varphi, \omega, t] = -\frac{g}{l}\sin\varphi - q\omega + F_0 \sin\Omega t \tag{3.24}$$

$$\dot{\varphi} = \omega \tag{3.25}$$

Für das Runge-Kutta-Verfahren (R-K) werden dann die Zwischenwerte $\bar{\omega}$ und $\bar{\varphi}$ berechnet.

$$\bar{\omega} = \omega(t) + \frac{\Delta t}{2}\ f[\varphi(t), \omega(t), t] \tag{3.26}$$

$$\bar{\varphi} = \varphi(t) + \frac{\Delta t}{2}\ \omega(t) \tag{3.27}$$

Die Lösungen ergeben sich dann durch das R-K-Verfahren.

$$\varphi(t + \Delta t) = \varphi(t) + \Delta t\ \bar{\omega} \tag{3.28}$$

$$\omega(t + \Delta t) = \omega(t) + \Delta t\ f\left[\bar{\varphi}, \bar{\omega}, t + \frac{\Delta t}{2}\right] \tag{3.29}$$