Titel	**Gekoppelte Oszillatoren**
Autor	Denis Nordmann
Version	20.03.2018
Zitierung	D. Nordmann. *Gekoppelte Oszillatoren*. URL: `http://physik.co-i60.com` (abgerufen am: `<Datum>`)
Copyright	

Inhaltsverzeichnis

1 Einleitung

Bereits 1912 entwickelte von Laue die Theorie zur Beugung von Röntgenstrahlen an periodischen Strukturen. Die experimentelle Bestätigung dieser Theorie erfolgte durch Friedrich und Knipping, welche die Beugung von Röntgenstrahlung an Kristallen beobachtet haben. Damit wurde bewiesen, dass sich Kristalle aus einer periodischen Anordnung von Atomen zusammensetzen. Anhand dieser Erkenntnis konnten neue Theorien und Modelle entwickelt werden, welche die physikalischen Eigenschaften der (kristallinen) Festkörper erklären (z.B. Wärmeleitung oder Wärmeausdehnung) [4].

Die Gitterschwingungen im Kristall können in erster Näherung klassisch anhand eines Modells von linear angeordneten harmonischen Oszillatoren erklärt werden. Die Atome werden als schwingende Massenpunkte betrachtet, welche über eine Kopplungsfeder miteinander gekoppelt sind. Dieses Modell ist auf mechanische, elektrische und optische Problemstellungen anwendbar.

Im Rahmen dieser Studienarbeit wird das das Modell einer Schwingerkette aus mehreren Feder-Masse-Elementen zunächst analytisch behandelt. Durch zuvor festgelegte physikalische Parameter (Massen und Federkonstanten) wird anschließend das Verhalten der Schwingerkette numerisch simuliert. Ein Vergleich zwischen der analytisch und numerisch berechneten Ergebnissen wird durchgeführt, um eine Aussage über die Genauigkeit und Richtigkeit der Simulation treffen zu können. Die Umsetzung dieser Problemstellung erfolgt mittels der Programmiersprache MATLAB.

2 Grundlagen

2.1 Gekoppeltes Federpendel mit $N = 2$ Massen

In einem einführenden Beispiel wird die Anordnung von Federn und Massen gemäß Abbildung 1 betrachtet. Die Gewichte mit den Massen m_1 und m_2 sind entlang der x-Achse mit Federn verbunden, deren Federkonstanten D_1 und D_2 sowie D_{12} sind. Die Auslenkungen der Massen aus ihrer Ruhelage sind mit x_1 und x_2 gekennzeichnet.

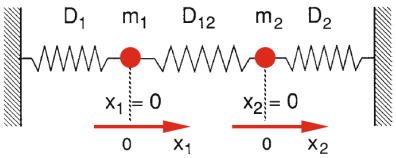

Abbildung 1: Beispiel eines gekoppelten Federpendels [2].

Die Bewegungsgleichungen für dieses System sind nach dem Newtonschen Grundgesetzen der Mechanik unter Vernachlässigung der Reibung gegeben mit

$$m_1\ddot{x}_1 = -D_1x_1 - D_{12}(x_1 - x_2) \tag{1}$$
$$m_2\ddot{x}_2 = -D_2x_2 - D_{12}(x_2 - x_1). \tag{2}$$

Dabei handelt es sich um ein gekoppeltes Differentialgleichungssystem, da in jeder der beiden Gleichungen eine der beiden Variablen x_1 bzw. x_2 vorkommen. Dieses System wird nun entkoppelt, indem Gleichungen 1 und 2 addiert bzw. voneinander subtrahiert werden. Unter der Vereinfachung, dass $m_1 = m_2 = m$ und $D_1 = D_2 = D$ ist, erhält man laut [2] die Gleichungen

$$m(\ddot{x}_1 + \ddot{x}_2) = -D(x_1 + x_2) \tag{3}$$
$$m(\ddot{x}_1 - \ddot{x}_2) = -D(x_1 - x_2) - 2D_{12}(x_1 - x_2). \tag{4}$$

In den neuen Koordinaten

$$\xi^+ = \frac{1}{2}(x_1 + x_2) \quad \text{und} \quad \xi^- = \frac{1}{2}(x_1 - x_2) \tag{5}$$

erhält man die Gleichungen

$$m\ddot{\xi}^+ = -D\xi^+ \tag{6}$$
$$m\ddot{\xi}^- = -(D + 2D_{12})\xi^- \tag{7}$$

mit der allgemeinen Lösung

$$\xi^+(t) = A_1 \cdot \cos(\omega_1 t + \varphi_1) \quad \text{mit} \quad \omega_1^2 = \frac{D}{m} \tag{8}$$
$$\xi^-(t) = A_2 \cdot \cos(\omega_2 t + \varphi_2) \quad \text{mit} \quad \omega_2^2 = \frac{D + 2D_{12}}{m}. \tag{9}$$

Die Transformation auf Normalschwingungskoordinaten erlaubt die Darstellung der gekoppelten Schwingung als Überlagerung von zwei harmonischen Schwingungen mit den Frequenzen ω_1 und ω_2. Für gleiche Amplituden $A_1 = A_2 = A$ lassen sich durch Rücktransformation auf die x-Koordinaten x_1 und x_2 die Schwingungen der Massenpunkte darstellen als

$$x_1 = (\xi^+ + \xi^-) = 2A \cdot \cos\left(\frac{(\omega_1 - \omega_2)}{2}t + \frac{\varphi_1 - \varphi_2}{2}\right) \cdot \cos\left(\frac{(\omega_1 + \omega_2)}{2}t + \frac{\varphi_1 + \varphi_2}{2}\right) \tag{10}$$

$$x_2 = (\xi^+ - \xi^-) = -2A \cdot \sin\left(\frac{(\omega_1 - \omega_2)}{2}t + \frac{\varphi_1 - \varphi_2}{2}\right) \cdot \sin\left(\frac{(\omega_1 + \omega_2)}{2}t + \frac{\varphi_1 + \varphi_2}{2}\right) \tag{11}$$

In Abbildung 2 sind die Lösungsfunktionen der Gleichungen 1 und 2 dargestellt. Sichtbar sind Schwebungen von x_1 und x_2 sowie die entkoppelten Normalschwingungen ξ_1 und ξ_2.

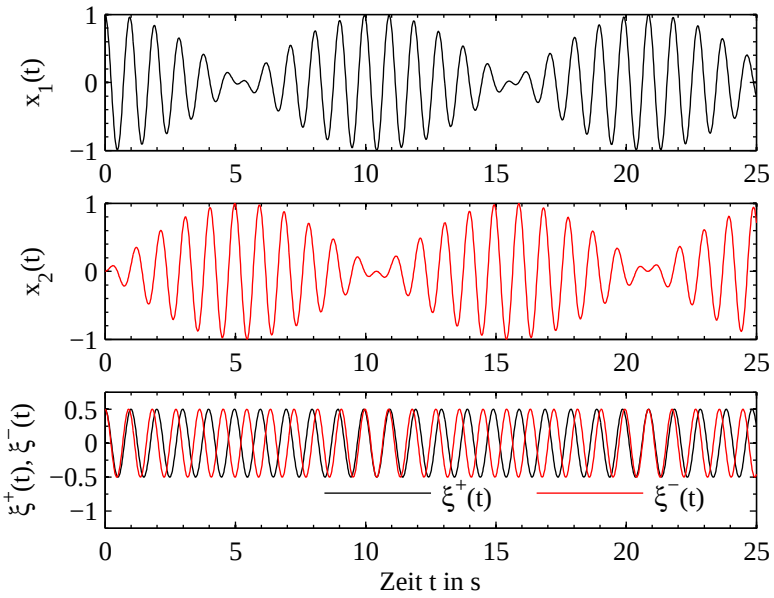

Abbildung 2: Veranschaulichung der Schwingungsamplituden gekoppelter Oszillatoren. Gewählte Parameter sind: $m = 0{,}25$ kg, $D = 10$ N/m, $D_{12} = D/10$, $A = 0{,}5$ m, $\varphi_1 = \varphi_2 = 0$.

2.2 Numerische Lösungsverfahren

Zur numerischen Behandlung von gewöhnlichen Differentialgleichungen (engl.: *ordinary differential equations*, ODE) stehen uns viele Integrationsverfahren zur Verfügung. Eine einfache Methode zur Lösung von ODEs ist das Streckenzugverfahren nach Euler, welche die exakte Lösungskurve stückweise approximiert [9]. Für eine hinreichende Genauigkeit muss die Schrittweite h möglichst klein gewählt werden, was jedoch den Rechenaufwand erhöht.

Im folgenden Abschnitt wird auf die Lösungsmethodik eines Systems von Differentialgleichungen (DGL) eingegangen und das Runge-Kutta-Verfahren 4. Ordnung vorgestellt. Die Lösung von ODEs in MATLAB wird mithilfe der Funktion `ode45` durchgeführt, welche hier kurz vorgestellt wird.

2.2.1 Lösungsmethodik

Problemstellungen, welche die Lösung von ODEs erfordern, lassen sich in ein System von DGL erster Ordnung überführen. So kann beispielsweise die DGL 2. Ordnung

$$\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} + q(x)\frac{\mathrm{d}y}{\mathrm{d}x} = r(x) \tag{12}$$

geschrieben werden als ein System von zwei DGL 1. Ordnung

$$\frac{\mathrm{d}y}{\mathrm{d}y} = z(x) \tag{13}$$

$$\frac{\mathrm{d}z}{\mathrm{d}x} = r(x) - q(x) \cdot z(x) \tag{14}$$

mit einer Hilfsvariablen z. So lassen sich eine Anzahl von N gekoppelten gewöhnlichen DGL 1. Ordnung untersuchen. Deren Form lautet

$$\frac{\mathrm{d}y_i(x)}{\mathrm{d}x} = f_i(x, y_0, y_1, y_2, \ldots, y_{N-1}), \quad \text{mit} \quad i = 0, 1, \ldots, N-1 \tag{15}$$

mit bekannten Funktionen f_i auf der rechten Seite der Gleichung [8]. Für jede Problemstellung müssen die Anfangswerte bzw. Randbedingungen vorliegen, wobei die Art der Randbedingung für die Lösungsmethode entscheidend ist. Die Methodik der Reduktion von DGL-Systemen höherer Ordnung in ein System von DGL 1. Ordnung in MATLAB ist beschrieben in der Dokumentation zum Programm – z.B. in [5] oder [6].

Prinzipiell erfolgt die numerische Lösung einer ODE nach dem folgenden Schema:

- Die Differentialoperatoren dy und dx aus Gleichung 15 werden in finiten Schritten Δy und Δx approximiert und die erhaltenen Gleichungen mit Δx multipliziert
- Man erhält nun algebraische Gleichungen, welche die Änderung der Funktionen um ein Inkrement Δx wiedergeben
- Durch die Verringerung der Schrittweite Δx lässt sich die Genauigkeit der Methode steigern

Drei der am häufigsten angewandten Lösungsverfahren für ODEs sind die Runge-Kutta-Methoden, die Richardson-Extrapolation (auch bekannt als Bulirsch-Stoer-Methode) und die „predictor-corrector“ Methode – ein Mehrschrittverfahren.

2.2.2 Runge-Kutta-Verfahren 4. Ordnung

Beim Streckenzugverfahren von Euler wird die Approximation gemäß

$$y_{n+1} = y_n + h \cdot f(x_n, y_n) \tag{16}$$

durchgeführt, wobei die Lösung von x_n bis $x_{n+1} \equiv x_n + h$ fortschreitet. Die Genauigkeit dieses Verfahrens lässt sich durch Berechnung von Zwischenschritten im Intervall h realisieren. Die Rechenvorschrift für das Runge-Kutta-Verfahren 4. Ordnung lautet:

$$\begin{aligned}
k_1 &= h \cdot f(x_n, y_n) && (17)\\
k_2 &= h \cdot f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) && (18)\\
k_3 &= h \cdot f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) && (19)\\
k_4 &= h \cdot f(x_n + h, y_n + k_3) && (20)\\
y_{n+1} &= y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + O(h^5). && (21)
\end{aligned}$$

In Abbildung 3 ist der RK4-Algorithmus veranschaulicht. Pro Berechnungsintervall werden die vier Hilfsgrößen $k_1 \ldots k_4$ für jeden Rechenschritt neu berechnet [9]. Sie beschreiben näherungsweise das Steigungsverhalten der Lösungskurve $y = y(x)$ in den beiden Randpunkten k_1 und k_4 sowie in der Intervallmitte (k_2, k_3). Anhand der Ergebnisse der jeweiligen Rechenschritte wird der gesuchte Funktionswert y_{n+1} berechnet.

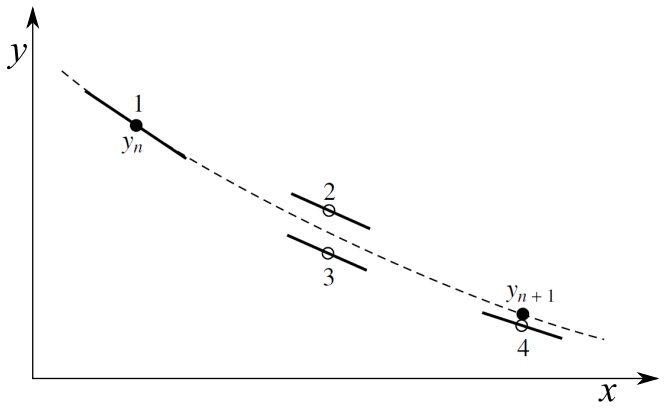

Abbildung 3: Schematische Darstellung des Runge-Kutta-Verfahrens 4. Ordnung [8].

2.2.3 Der `ode45`-Solver in MATLAB

MATLAB bietet eigene Lösungsalgorithmen an, die je nach DGL-Typ angewandt werden können. Hierzu zählt die Funktion `ode45` – eine Implementierung des expliziten Runge-Kutta-Verfahrens mittlerer (4,5) Ordnung. Dieser Solver wird empfohlen, wenn keine Kenntnisse über die Eigenschaften (z.B. Steifigkeit) des Systems vorliegen [6]. Die Syntax von `ode45` lautet wie folgend:

```
[TOUT,YOUT] = ode45(ODEFUN,TSPAN,Y0)
```

- `TOUT`: Ein Spaltenvektor für die Ausgabe der Zeitinkremente

- `YOUT`: Array für die Ausgabe der berechneten y-Funktionswerte und deren Ableitungen
- `ODEFUN`: Eine vom Anwender definierte Funktion (z.B. als `.m`-Datei), welche die Problemstellung in Form eines Systems von DGL'en 1. Ordnung enthält
- `TSPAN`: Spaltenvektor mit Angaben zum Integrationsintervall
- `Y0`: Spaltenvektor mit Anfangs- bzw. Randbedingungen

3 Simulationen in MATLAB

In diesem Abschnitt werden die Problemstellungen aus dem Abschnitt 2 numerisch behandelt. Zur Berechnung wird eine Studentenversion von MATLAB 7.14.0.739 (R2012a) verwendet. Für die Implementierung der in dieser Studienarbeit beschriebenen Simulationen kann nahezu jede moderne wissenschaftliche Rechensoftware verwendet werden. Genannt seien beispielsweise kommerziell erhältliche Programme wie MATLAB, Mathcad, Mathematica, Maple oder auch Open-Source-Programme wie GNU Octave oder Sage. Eine Implementierung der Problemstellung wäre mithilfe von Programmiersprachen wie C, C++, Java, FORTRAN, Python etc. ebenfalls möglich.

3.1 Schwingerkette mit $N = 2$ Massen

In einem einführenden Beispiel wird die Problemstellung des gekoppelten Federpendels mit 2 Massen aus Abschnitt 2.1 behandelt. Die analytische Lösung dieser Problemstellung liegt bereits vor, sodass nun mithilfe von MATLAB ein Simulationsprogramm geschrieben werden kann, welches die Problemstellung numerisch löst und beide Rechnungen miteinander vergleicht. Der für die Berechnungen verwendete MATLAB-Quellcode ist im Anhang A.1 hinterlegt.

3.1.1 Simulationsparameter

Die Massen der Gewichte m_1 und m_2 werden zu $m_1 = m_2 = m$ vereinfacht, wobei die $m = 0{,}25$ kg beträgt. Die Federkonstanten D_1 und D_2 betragen jeweils 10 N/m. Die Federkonstante der Kopplungsfeder sei $D_{12} = 0{,}1 \cdot D_1$, also 1 N/m. Die maximale Auslenkung wird mit $A = 0{,}5$ m angenommen. Die Eigenfrequenzen werden gemäß Gleichung 8 berechnet. Die Phasenwinkel werden mit $\varphi_1 = \varphi_2 = 0$ angenommen. Die Anfangsbedingungen dieses Systems sind $x_1(t = 0) = 1$ m und $x_2(0) = 0$ m sowie $\dot{x}_1(0) = 0 = \dot{x}_2(0) = 0$ m/s.

Die Berechnung erfolgt in einem Zeitintervall von $t = 0 \ldots 25$ s mit einem Zeitschritt von $\Delta t = 0{,}02$ s. Hieraus ergeben sich insgesamt 1251 berechnete Punkte. Die Ergebnisse der Simulation sind in Abbildungen 4, 5 und 6 sichtbar.

3.1.2 Ergebnisse

Abbildungen 4 und 5 zeigen den zeitlichen Verlauf der Amplitude sowie den relativen Fehler als Betrag von

$$\delta x = \left| \frac{x_{\text{analyt.}} - x_{\text{num.}}}{x_{\text{analyt.}}} \right|. \tag{22}$$

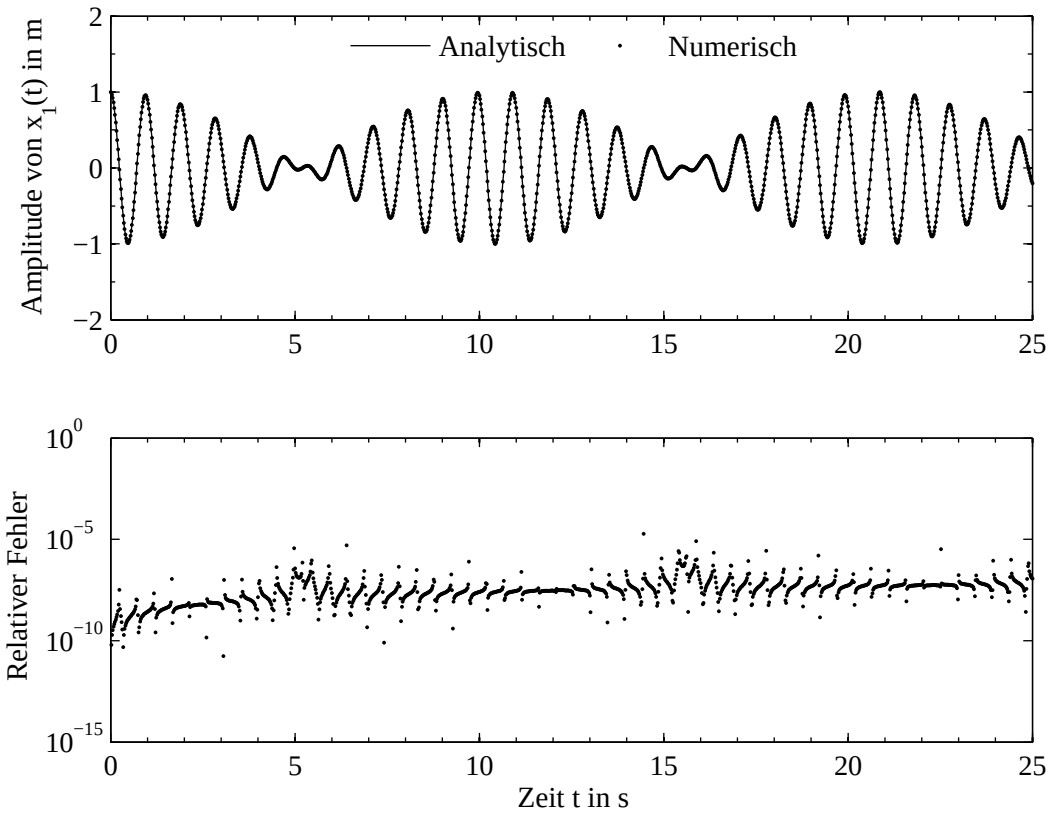

Abbildung 4: Ergebnis der Simulation für $x_1(t)$.

In beiden Grafiken ist erkennbar, dass die numerische und die analytische Rechnung übereinstimmen. Der mittlere relative Fehler beider Rechnungen lässt sich mit $10^{-(7,5\pm0,5)}$ abschätzen. Es ist ebenfalls sichtbar, dass der relative Fehler periodisch zunimmt bzw. abnimmt. Dies geschieht jeweils in der Umgebung einer Nullstelle. Bedingt durch die Division von sehr kleinen $x_{\text{analyt.}}$-Zahlenwerten im Nenner der Gleichung 22 können auch Ausreißer oder ungültige Werte entstehen. Diese Werte werden ignoriert bzw. nicht dargestellt.

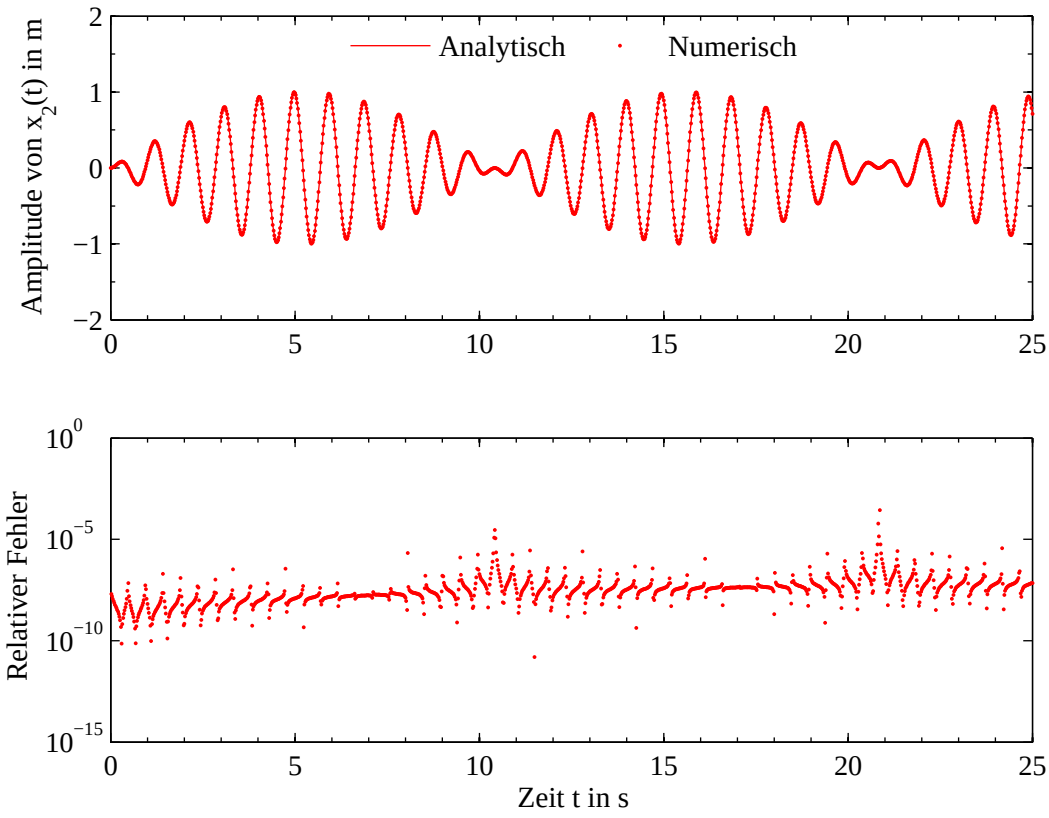

Abbildung 5: Ergebnis der Simulation für $x_2(t)$.

In Abbildung 6 sind entkoppelte Schwingungsamplituden $\xi^+(t)$ und $\xi^-(t)$ nach Gleichung 5 dargestellt. In der oberen Abbildung werden die harmonischen Schwingungen beider Massenpunkte gemeinsam dargestellt, wie bereits in Abbildung 2 angedeutet. Die untere Abbildung zeigt die jeweiligen Komponenten einzeln. Anhand der Diagramme ist erkennbar, dass die analytisch berechneten Werte (gekennzeichnet durch eine durchgezogene Linie) und die numerisch berechneten Werte (Punkte) übereinstimmen.

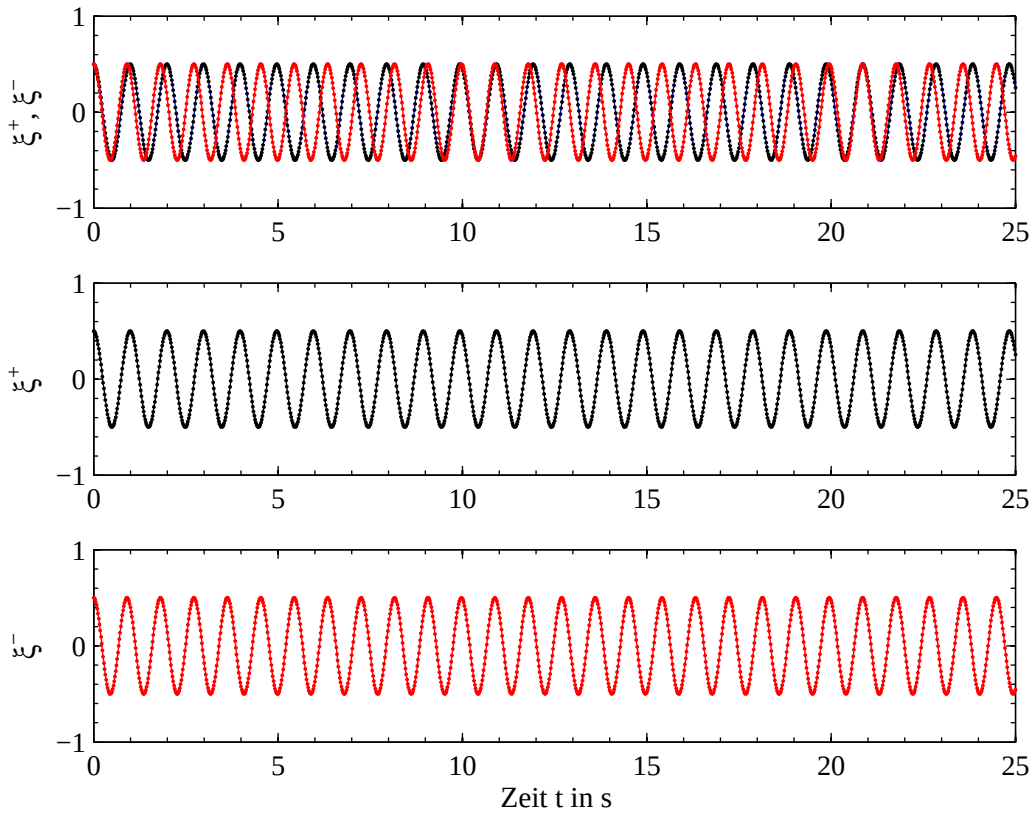

Abbildung 6: Ergebnis der Simulation für $\xi^+(t)$ und $\xi^-(t)$. Linie: analytisch berechnet, Punkte: numerisch berechnet.

3.2 Schwingerkette mit $N = 5$ Massen

Nun wird das Modell der Schwingerkette aus dem vorherigen Abschnitt auf $N = 5$ Massen bzw. 6 Federn erweitert. Ein Modell dieses Systems ist in Abbildung 7 dargestellt.

3.2.1 Simulationsparameter

Die Masse der Gewichte wird mit $m = 0{,}25$ kg festgelegt. Die Federn zwischen der Wand und dem benachbarten Gewicht haben eine Federkonstante von $D_1 = D_6 = 10$ N/m. Die Kopplungsfedern besitzen eine Federkonstante von $D_2 = \ldots = D_5 = \frac{1}{3}D_1$. Die Anfangsbedingungen sind wie folgend gewählt: $x_1(t = 0) = 1$ m sowie $x_2(0) = x_3(0) = x_4(0) = x_5(0) = 0$ m. Die Anfangsbedingungen für die Ableitungen von $x(t)$ sind mit $\dot{x}_1(t = 0) = \dot{x}_2(0) = \dot{x}_3(0) = \dot{x}_4(0) = \dot{x}_5(0) = 0$ m/s gewählt.

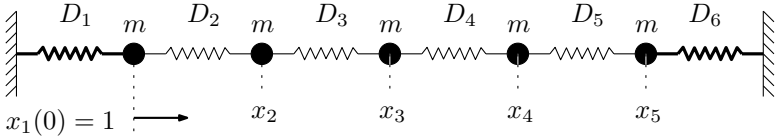

Abbildung 7: Modell einer Schwingerkette mit 5 Massen und 6 Federn.

3.2.2 Mathematische Behandlung der Problemstellung

Die Bewegungsgleichungen für das betrachtete Feder-Masse-Systems lauten in der Matrixform

$$m\cdot\begin{pmatrix}\ddot{x}_1\\ \ddot{x}_2\\ \ddot{x}_3\\ \ddot{x}_4\\ \ddot{x}_5\end{pmatrix}=\begin{pmatrix}-(D_1+D_2) & D_2 & 0 & 0 & 0\\ D_2 & -(D_2+D_3) & D_3 & 0 & 0\\ 0 & D_3 & -(D_3+D_4) & D_4 & 0\\ 0 & 0 & D_4 & -(D_4+D_5) & D_5\\ 0 & 0 & 0 & D_5 & -(D_5+D_6)\end{pmatrix}\cdot\begin{pmatrix}x_1\\ x_2\\ x_3\\ x_4\\ x_5\end{pmatrix}. \tag{23}$$

Zur Lösung dieses gekoppelten DGL-Systems 2. Ordnung wird nach [7] der Ansatz

$$x_i(t) = A_i\cdot e^{\mathrm{i}\omega t} \tag{24}$$
$$\dot{x}_i(t) = \mathrm{i}\omega A_i\cdot e^{\mathrm{i}\omega t} \tag{25}$$
$$\ddot{x}_i(t) = -\omega^2 A_i\cdot e^{\mathrm{i}\omega t} \tag{26}$$

verwendet, wobei der Index $i=1\ldots5$ und A_i eine Amplitude ist. Setzt man den Ansatz aus Gleichungen 24 und 26 in Gleichung 23 ein, so ergibt sich daraus ein Gleichungssystem der Form

$$m\omega^2A_1e^{\mathrm{i}\omega t} = [-D_1A_1-D_2A_1+D_2A_2]e^{\mathrm{i}\omega t} \tag{27}$$
$$\vdots = \vdots \tag{28}$$
$$m\omega^2A_5e^{\mathrm{i}\omega t} = [D_5A_4-D_5A_5-D_6A_5]e^{\mathrm{i}\omega t}. \tag{29}$$

Durch Division beider Seiten mit $e^{\mathrm{i}\omega t}$ und der Umformung in die Matrixform erhalten wir eine Eigenwertgleichung, welche sich schreiben lässt als

$$\underbrace{\begin{pmatrix}-(D_1+D_2) & D_2 & & & \\ D_2 & \ddots & D_3 & & \\ & D_3 & \ddots & D_4 & \\ & & D_4 & \ddots & D_5\\ & & & D_5 & -(D_5+D_6)\end{pmatrix}}_{\text{Matrix D}}\underbrace{\begin{pmatrix}A_1\\ A_2\\ A_3\\ A_4\\ A_5\end{pmatrix}}_{\text{Eigenvektor}\vec{A}} = \underbrace{m\omega^2}_{\text{Eigenwert}\lambda}\begin{pmatrix}A_1\\ A_2\\ A_3\\ A_4\\ A_5\end{pmatrix} \tag{30}$$

Auf die Methodik zur Bestimmung von Eigenwerten (EW) und Eigenvektoren (EV) soll an dieser Stelle an die entsprechende Literatur (vgl. [1, 9]) verwiesen werden. In MATLAB lassen sich Eigenwerte und Eigenvektoren mithilfe der Funktion `eig()` ausrechnen. Dies ist sehr hilfreich, da hierfür die Berechnung der Determinante einer 5×5 Matrix sowie die Bestimmung der Nullstellen des charakteristischen Polynoms 5. Ordnung notwendig wäre. Durch die Eingabe des Befehls

```
1 [EV, EW] = eig(-D/m)
```

berechnet MATLAB die Eigenwerte in Form einer Einheitsmatrix mit N-Diagonalelementen, wobei diese physikalisch dem Quadrat der Kreisfrequenz ω^2 entsprechen. Durch Wurzelziehen erhält man die Eigenfrequenzen als Diagonalelemente

```
omega =

    2.4178         0         0         0         0
         0    4.5982         0         0         0
         0         0    6.3246         0         0
         0         0         0    7.6718         0
         0         0         0         0    7.7988
```

Die berechneten EV sind in der folgenden Matrizen-Ausgabe gelistet:

```
EV =

    0.1441   -0.2706   -0.3536   -0.6533    0.5952
    0.5131   -0.6533   -0.3536    0.2706   -0.3342
    0.6572   -0.0000    0.7071    0.0000    0.2610
    0.5131    0.6533   -0.3536   -0.2706   -0.3342
    0.1441    0.2706   -0.3536    0.6533    0.5952
```

Nun kann die allgemeine Lösung des DGL-Systems konstruiert werden. Sie ist gegeben mit

$$\vec{x}(t) = C_1 \begin{pmatrix} 0.1441 \\ 0.5131 \\ 0.6572 \\ 0.5131 \\ 0.1441 \end{pmatrix} \cdot e^{2.4178\ \mathrm{s}^{-1} \cdot t} + \ldots + C_5 \begin{pmatrix} 0.5952 \\ -0.3342 \\ 0.2610 \\ -0.3342 \\ 0.5952 \end{pmatrix} \cdot e^{7.7988\ \mathrm{s}^{-1} \cdot t} \tag{31}$$

Die Konstanten C_1 bis C_5 werden mithilfe der Anfangsbedingungen berechnet [3]. Diese sind gegeben mit

$$\vec{x}(t=0) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \mathrm{m} \quad \text{sowie} \quad \dot{\vec{x}}(t=0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \frac{\mathrm{m}}{\mathrm{s}}. \tag{32}$$

Zur Ermittlung der Integrationskonstanten muss das folgende Gleichungssystem gelöst werden:

$$\begin{pmatrix} 0.1441 & -0.2706 & -0.3536 & -0.6533 & 0.5952 \\ 0.5131 & -0.6533 & -0.3536 & 0.2706 & -0.3342 \\ 0.6572 & -0.0000 & 0.7071 & 0.0000 & 0.2610 \\ 0.5131 & 0.6533 & -0.3536 & -0.2706 & -0.3342 \\ 0.1441 & 0.2706 & -0.3536 & 0.6533 & 0.5952 \end{pmatrix} \cdot \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{33}$$

Die Berechnung von $\vec{C}$ wird mithilfe von MATLAB durchgeführt. Unter Verwendung des Befehls `C = EV\x0` wird ein Gleichungssystem der Form $\mathbf{EV} \cdot \vec{C} = \vec{x}_0$ gelöst. Mit der Lösung des Gleichungssytems aus Gl. 33 erhält man das folgende Ergebnis:

```
C =

    0.1441   -0.2706   -0.3536   -0.6533    0.5952
```

Unter Verwendung der Eulerschen Relation

$$e^{\mathrm{i}\varphi} = \cos(\varphi) + \mathrm{i}\sin(\varphi) \tag{34}$$

wird nun der Realteil aus Gleichung 24 zur Darstellung von Normalschwingungen verwendet, sodass wir für die spezielle Lösung den folgenden Ausdruck erhalten:

$$\vec{x}(t) = 0.1441 \cdot \begin{pmatrix} 0.1441 \\ 0.5131 \\ 0.6572 \\ 0.5131 \\ 0.1441 \end{pmatrix} \cdot \cos(2.4178\ \mathrm{s}^{-1} t) + \ldots + 0.5952 \cdot \begin{pmatrix} 0.5952 \\ -0.3342 \\ 0.2610 \\ -0.3342 \\ 0.5952 \end{pmatrix} \cdot \cos(7.7988\ \mathrm{s}^{-1} t). \tag{35}$$

Die Ergebnisse dieser Rechnung werden im nächsten Abschnitt mit den numerisch berechneten Ergebnissen verglichen und diskutiert. Der MATLAB-Quellcode ist im Anhang A.2 hinterlegt.

3.2.3 Vergleich der Simulationsergebnisse

In Abbildung 8 sind die nach der analytischen Methode bestimmten Schwingungsamplituden der Massen an den Positionen x_1 bis x_5 dargestellt. Deutlich sichtbar ist eine Schwebung zwischen den Massen x_1 und x_5 mit einer Periodendauer von etwa 50 s. Die Massen an den Positionen x_2, x_3 und x_4 hingegen zeigen keine oder nur sehr schwache Schwebungen sowie anharmonische Schwingungen. Das Verhalten lässt sich damit erklären, dass zwischen den Massen an den Enden ein Energieaustausch stattfindet – die Massen in der Mitte sind nur an der Übertragung beteiligt.

Abbildung 9 zeigt das Simulationsergebnis, welches mithilfe des `ode45` Solvers in MATLAB durchgeführt wurde. Qualitativ stimmen die berechneten Kurven mit dem analytischen Modell überein. Sichtbar sind Schwebungen der Massen bei x_1 und x_5 sowie anharmonische („komplizierte“) Schwingungen der Massen an den Positionen $x_2 \ldots x_5$. Auch andere Eigenschaften wie die Periodendauer oder die Schwingungsamplituden stimmen mit der analytischen Rechnung überein.

Durch einen Vergleich beider Simulationen zur Fehlerabschätzung lässt sich gemäß Abbildung 10 ein mittlerer relativer Fehler von $10^{-(6{,}5 \pm 0{,}5)}$ berechnen. Bestimmte Werte können in Bereiche von etwa 10^{-4} streuen – insbesondere sind die Fehler bei Nulldurchgängen größer, da hier gemäß Gleichung 22 durch kleine Zahlen dividiert wird. Ohne zusätzliche `ode45` Parameter für Toleranzen können die relativen Fehler im Bereich von $\delta x \approx 10^{-2}$ liegen. Dies kann für Überschlagsrechnungen von Vorteil sein, da eine höhere Rechengenauigkeit die Rechenzeit erhöht.

Der Zusammenhang zwischen dem Ort x und der Geschwindigkeit v lässt sich in einem Phasenportrait darstellen. Solche Phasenportraits sind in Abbildung 11 sichtbar. Da es sich um ein Modell des ungedämpften harmonischen Oszillators handelt, sind keine Phänomene wie Attraktoren oder Grenzzyklen sichtbar.

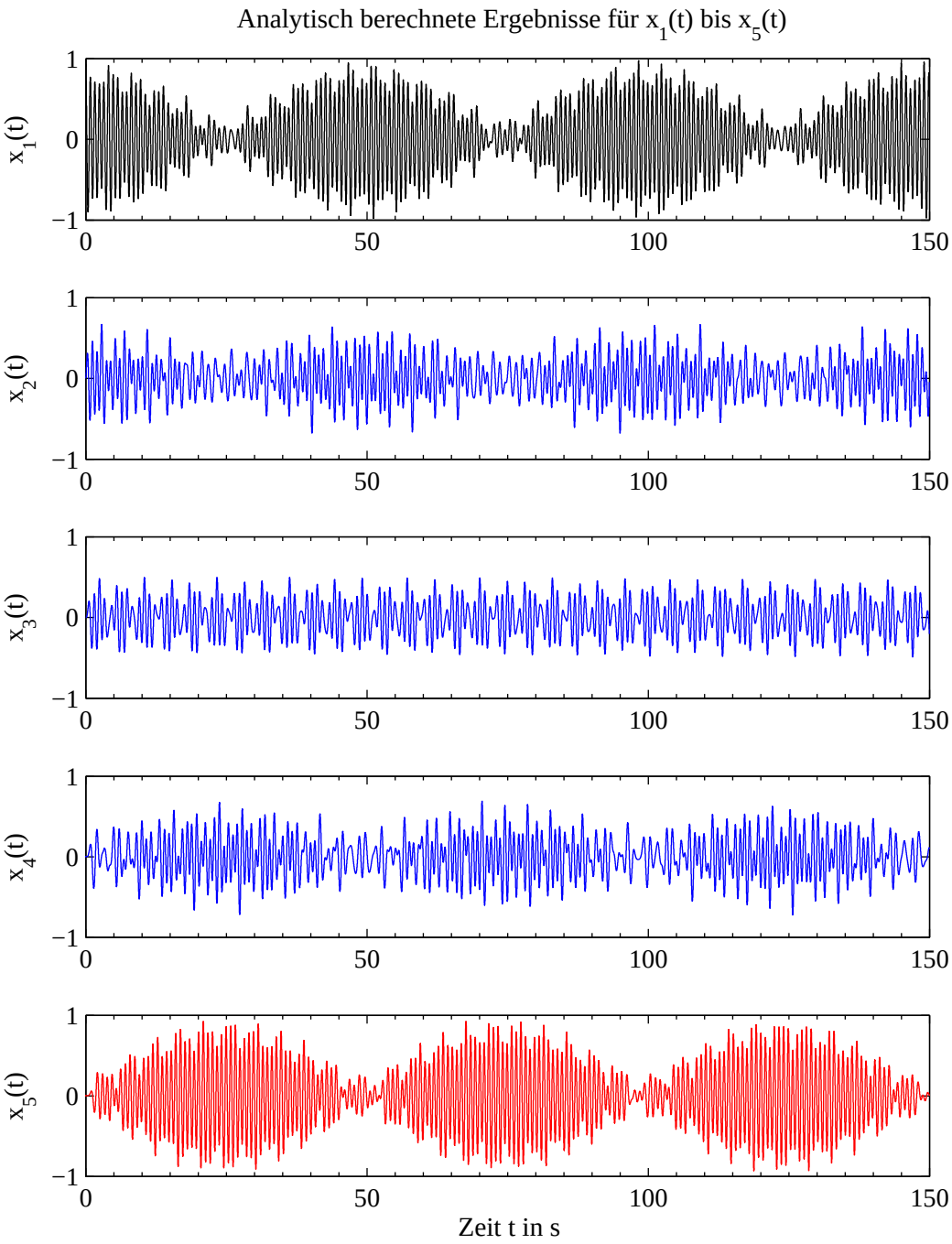

Abbildung 8: Analytisch berechnete Weg-Zeit-Funktionen der Massen an den jeweiligen Positionen x_1 bis x_5.

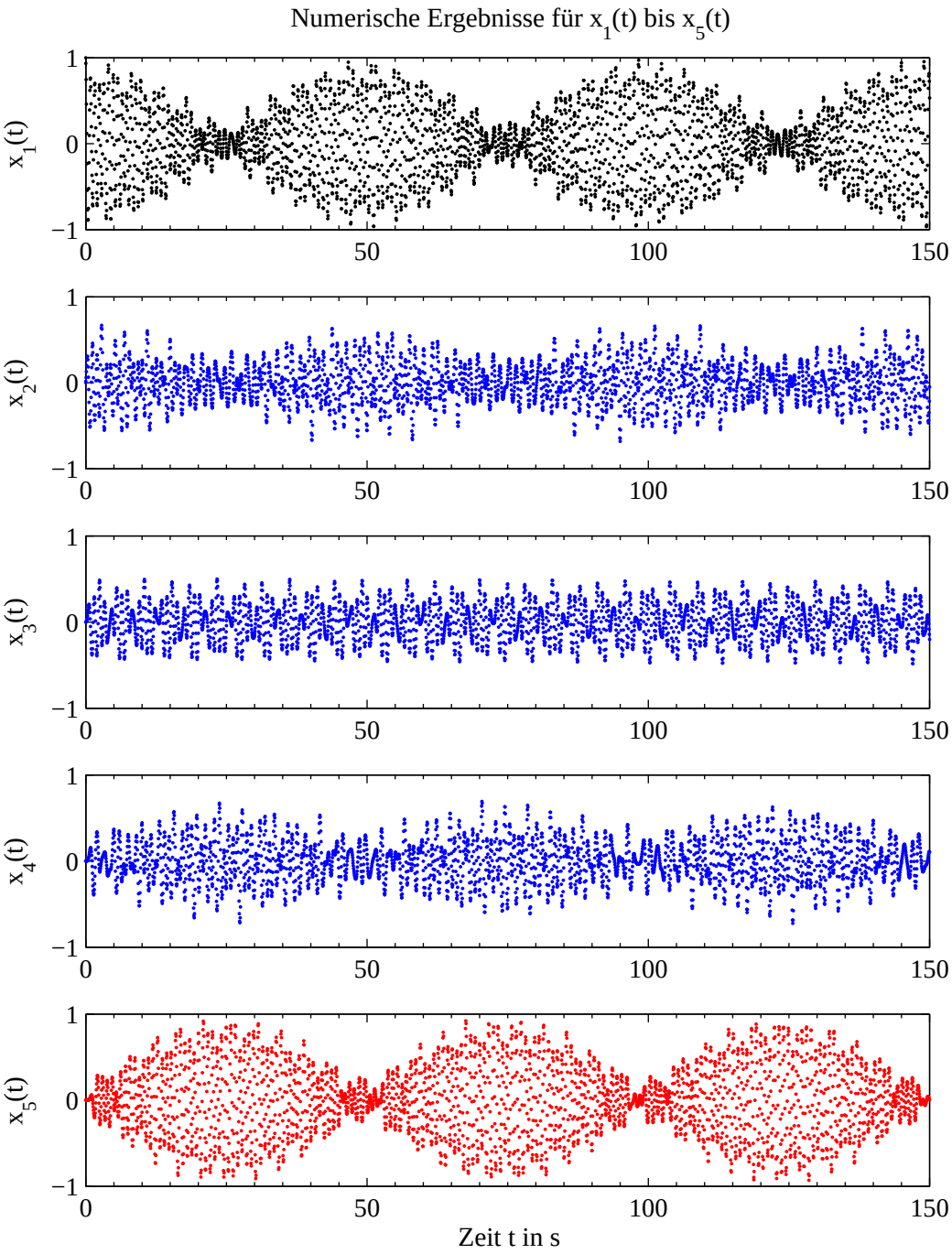

Abbildung 9: Ergebnis der numerischen Simulation für $N = 5$ Massen. Der Verlauf der Messpunkte stimmt mit dem Kurvenverlauf aus Abbildung 8 gut überein.

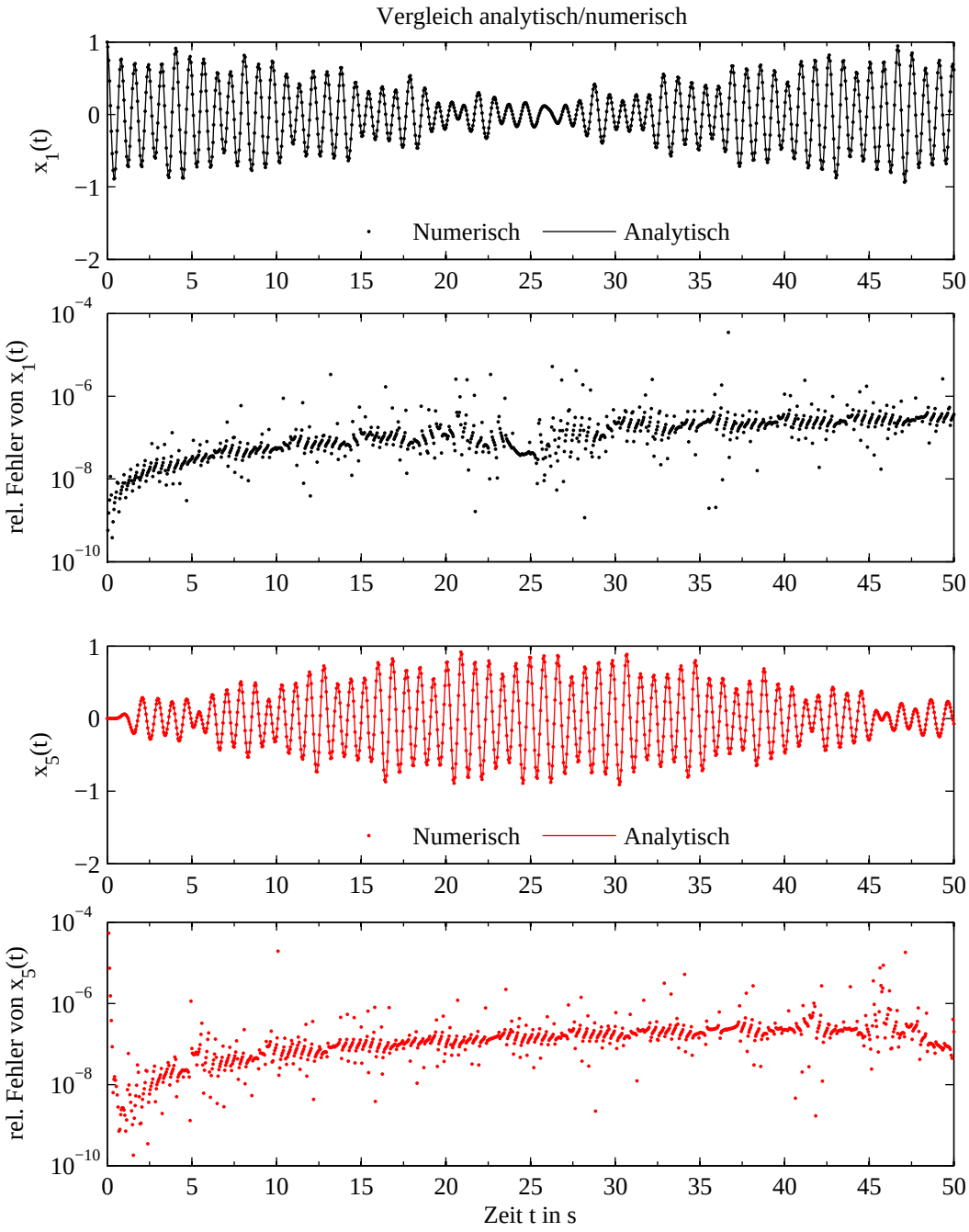

Abbildung 10: Vergleich der analytischen und der numerischen Rechnung. Hierfür wurde das Zeitintervall im Bereich von $t = 0$ s bis $t = 50$ s betrachtet, welches etwa 1000 Datenpunkte darstellt.

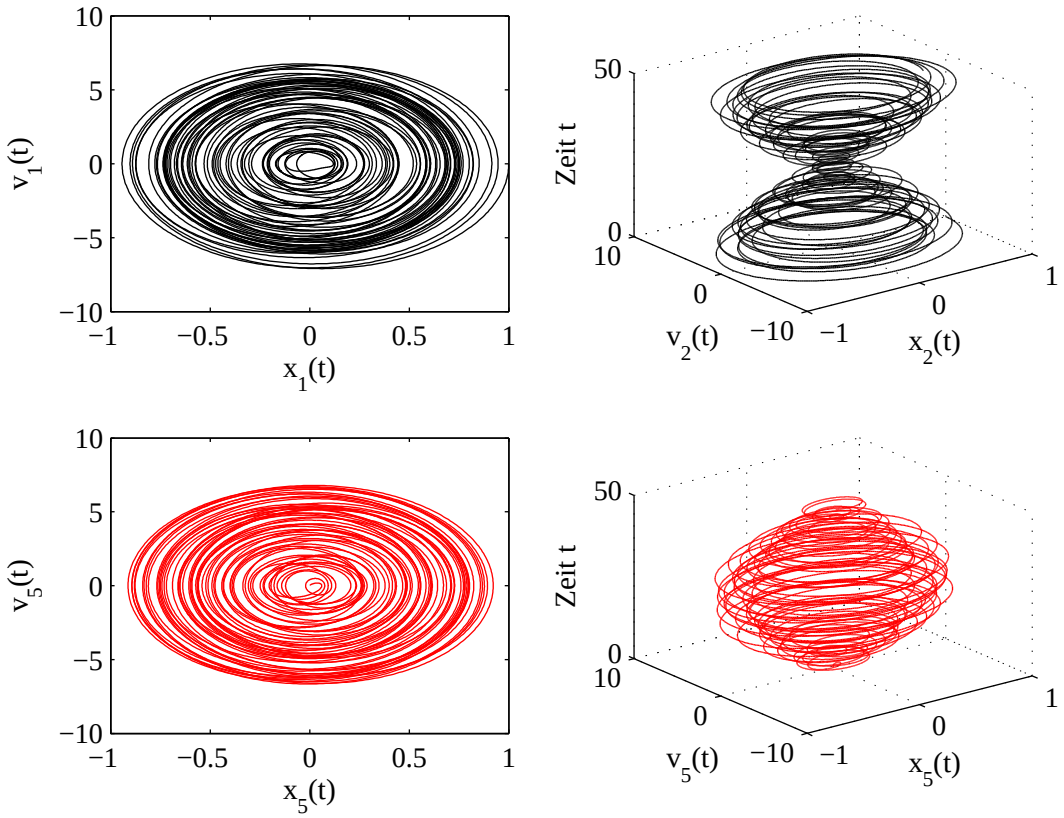

Abbildung 11: Phasenportraits der gekoppelten Oszillatoren an den Positionen x_1 und x_5 im Intervall von $t = 0$ s bis $t = 50$ s.

3.3 Ausblick: $N = 11$ gekoppelte Oszillatoren

Nun soll ein fortgeschritteneres Modell eines Feder-Masse-Systems aus 11 Massen und 12 Federn untersucht werden. Dieses ist in Abbildung 12 schematisch dargestellt. Hierbei sollen die Massen und Federkonstanten so gewählt sein, damit komplizierte Schwingungen der Massen in longitudinaler Richtung simuliert werden können.

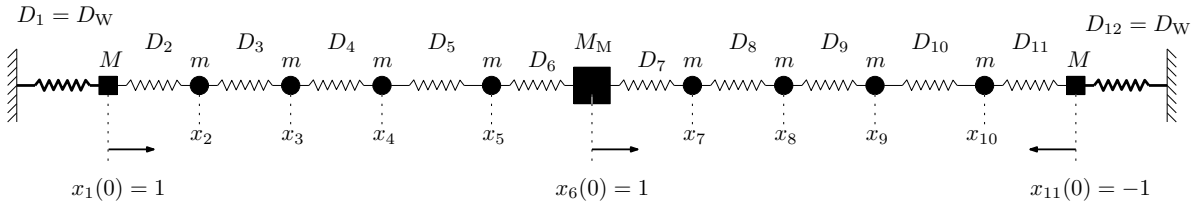

Abbildung 12: Modell einer Schwingerkette mit 11 Massen und 12 Federn.

3.3.1 Simulationsparameter

Die beiden äußeren Gewichte haben eine Masse von $M = 1$ kg, die inneren Gewichte sind etwas kleiner gewählt mit $m = 0{,}25$ kg. In der Mitte des Aufbaus befindet sich ein massereicher Block mit einer Masse von $M_M = 5$ kg, dessen Bewegung durch die Schwingungen der restlichen Massen beeinflusst werden soll.

Die mit den Wänden verbundene Federn besitzen eine Federkonstante von $D_W = 50$ N/m. Die restlichen Federkonstanten ($D_2 \dots D_{11}$) sind niedriger gewählt mit jeweils $D_2 =$

$D_3 = \ldots = D_{11} = \frac{1}{2} D_{\mathrm{W}} = 25$ N/m.

Zu Beginn der Simulation sind die Massen an den Positionen um jeweils $x_1(0) = 1$ m, $x_6(0) = 1$ m und $x_{11}(0) = -1$ m ausgelenkt. Alle Anfangsgeschwindigkeiten betragen Null, d.h. $\dot{x}_1(0) = \dot{x}_2(0) = \ldots = \dot{x}_{11}(0) = 0$ m/s. Nach dem Loslassen führen die Massen jeweils eine oszillierende Bewegung aus, welche sich in 11 Amplituden-Zeit-Diagrammen darstellen lassen.

3.3.2 Ergebnisse der Simulation

In Abbildungen 13 und 14 sind die numerisch berechneten Schwingungsamplituden der Massen an den jeweiligen Positionen x_1 bis x_{11} dargestellt. Die Bewegungen aller Massen sind im Rahmen der gewählten Parameter (Federkonstanten, Massen) anharmonisch und es sind keine eindeutigen Schwebungen sichtbar wie in Abschnitten 3.1 und 3.2 gezeigt. Die Masse M_{M} (mittlere Masse bei x_6) zeigt in Abbildung 13 als einziges Element eine periodische Schwingung, welche jedoch durch Überlagerung vieler Schwingungen verzerrt ist.

In Abbildung 13 sind die numerisch und analytisch ermittelten Kurven dargestellt. Im zeitlichen Verlauf von x_1 und x_{11} gibt es deutliche Unterschiede zwischen beiden Rechnungen. Die Schwingungsamplituden haben eine etwas unterschiedliche Form, geben jedoch den Verlauf der Schwingungen qualitativ wieder. Bei der Berechnung von x_6 stimmen die analytische und numerische Rechnung sehr gut überein.

Eine Diskrepanz findet sich beim Vergleich der Abbildungen 14 und 15. Hier fallen die Schwingungsamplituden der Massen m – also an den Positionen $x_2 \ldots x_5$ und $x_7 \ldots x_{10}$ – in der analytischen Rechnung deutlich geringer aus als bei der numerischen. Eine mögliche Erklärung hierfür ist, dass das analytische Modell seine Grenzen erreicht hat bzw. dass es sich in dieser Form nicht auf beliebig viele gekoppelte Oszillatoren anwenden lässt. Qualitativ betrachtet sieht man in beiden Fällen anharmonische Schwingungen und es treten keine Schwebungen auf. Die Übereinstimmung beider Rechnungen bei x_6 zeigt dennoch, dass zumindest teilweise richtig gerechnet wurde.

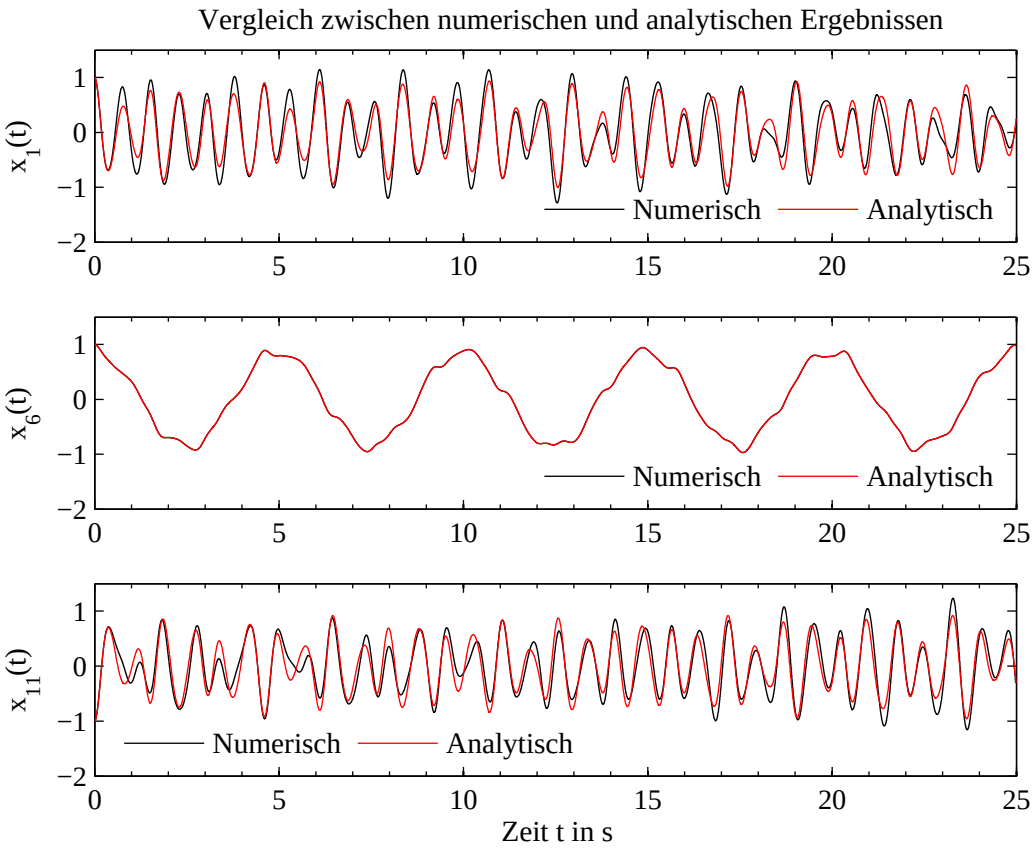

Abbildung 13: Vergleich der Schwingungsamplituden (analyt. und numer.) der Massen an den Positionen x_1 (linke Wand), x_6 (Mitte) und x_{11} (rechte Wand) im Zeitintervall von 0 s bis 25 s.

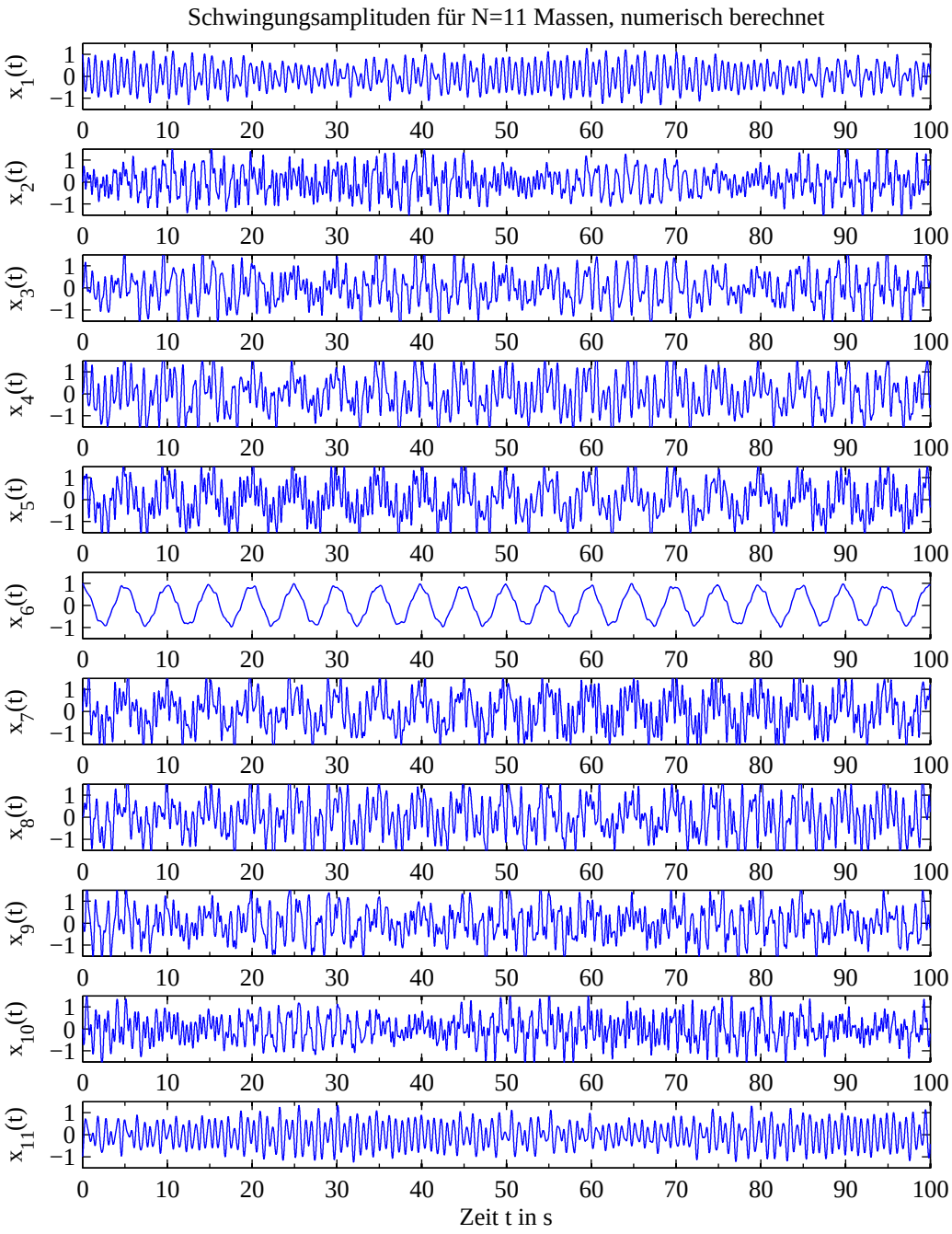

Abbildung 14: Ergebnisse der numerischen Simulation des in Abbildung 12 dargestellten Modells aus 11 Massen und 12 Federn.

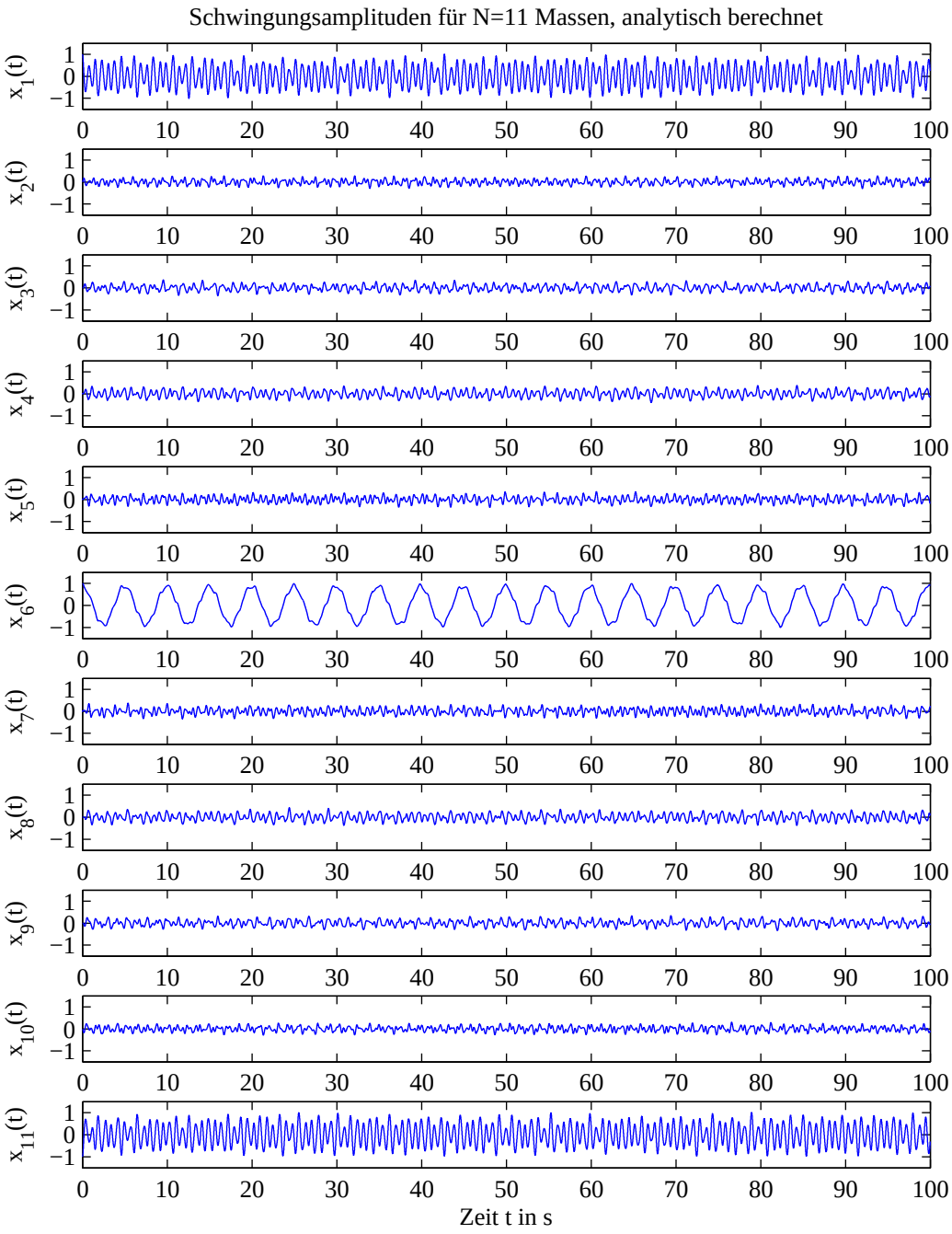

Abbildung 15: Ergebnisse der analytischen Rechnung des in Abbildung 12 dargestellten Modells aus 11 Massen und 12 Federn.

4 Diskussion der Ergebnisse

Im Rahmen dieser Studienarbeit wurde eine Variante des ungedämpften, gekoppelten harmonischen Oszillators analytisch und numerisch simuliert. Ausgehend von den Newtonschen Bewegungsgleichungen aus der Mechanik lässt sich ein mathematisches Modell für Systeme von gekoppelten Oszillatoren erstellen und lösen. Der Schwerpunkt dieser Arbeit lag in der Behandlung von einfachen mechanischen Systemen aus seriell verbundenen Feder-Masse-Schwingern (sog. Schwingerkette). Dabei wurden Schwingungen in longitudinaler Richtung betrachtet.

Es konnte gezeigt werden, dass je nach Wahl der Parameter (Masse, Federkonstante, Anfangsbedingungen) die Massen um ihre Ruhelage schwingen und dabei Energie über die Koppelfeder austauschen. Die Überlagerung der jeweiligen Schwingungen führt zu Schwebungen. Eine Fehlerabschätzung zwischen den numerisch und analytisch berechneten Resultaten zeigte, dass relative Fehler im Bereich von $\delta x \approx 10^{-(7{,}5 \pm 0{,}5)}$ für $N = 2$ Massen und $\delta x \approx 10^{-(6{,}5 \pm 0{,}5)}$ für $N = 5$ Massen liegen. Ohne entsprechend eingestellte Solver-Parameter ist die Genauigkeit der Rechnung deutlich niedriger ($\delta x \approx 10^{-2}$).

Bei komplexeren Problemstellungen wie dem Feder-Masse-System aus $N = 11$ Massen und 12 Federn lässt sich die Dynamik des Systems gut abschätzen. Es konnte gezeigt werden, dass die Massen im Bereich der gewählten Parameter anharmonisch schwingen und keine erkennbare Schwebung auftritt. Die analytischen und numerischen Resultate zeigen bei diesem Modell quantitative Unterschiede – qualitativ wird jedoch das Systemverhalten korrekt durch die analytische Rechnung vorhergesagt.

Abschließend lässt sich sagen, dass sich die betrachteten Problemstellungen gut numerisch simulieren lassen, sofern ein hinreichend genaues mathematisches Modell vorliegt. In diesem Falle müsste das Modell des harmonischen Oszillators erweitert werden. Durch Berücksichtigung der Dämpfung oder eines Erregers können komplexe Systeme mit geringem Programmieraufwand simuliert werden.

Literatur

[1] Bronstejn, I. Semendjajew, K. A. *Taschenbuch der Mathematik.* 3., überarb. und erw. Aufl. der Neubearb. Frankfurt a.M. und Thun: H. Deutsch, 1997 (siehe S. 12).

[2] Wolfgang Demtröder. *Experimentalphysik 1: Mechanik und Wärme.* 5. Aufl. Springer-Lehrbuch. Berlin, Heidelberg: Springer, 2008 (siehe S. 3).

[3] D. Estep. *Using MATLAB To Solve Linear Systems Of Differential Equations.* 2000. URL: `http://www.stat.colostate.edu/~estep/education/M340/matlabeigenvalue.pdf` (besucht am 14.01.2015) (siehe S. 13).

[4] Charles Kittel. *Introduction to solid state physics.* 7th ed. New York: Wiley, 1996 (siehe S. 2).

[5] MathWorks. *MATLAB Mathematics R2012a.* Hrsg. von MathWorks Inc. Natick, MA, 2012. (Besucht am 14.01.2015) (siehe S. 5).

[6] MathWorks. *ode45 - Solve nonstiff differential equations: MATLAB R2014b Dokumentation.* Hrsg. von MathWorks Inc. 2014. URL: `http://de.mathworks.com/help/matlab/ref/ode45.html` (besucht am 03.01.2015) (siehe S. 5 f.).

[7] H. Müther und R. Kleiner. *Skript zur Vorlesung Physik I: Kapitel 7 - Schwingungen und Wellen.* Tübingen, 2003. URL: `http://solid13.tphys.physik.uni-tuebingen.de/muether/physik1/skript/07-schwing.pdf` (besucht am 14.01.2015) (siehe S. 12).

[8] William H. Press. *Numerical recipes: The art of scientific computing.* 3rd ed. Cambridge, UK, New York: Cambridge University Press, 2007. URL: `http://www.netLibrary.com/urlapi.asp?action=summary\&v=1\&bookid=206846` (siehe S. 5 f.).

[9] Lothar Papula. *Mathematik für Ingenieure und Naturwissenschaftler: Ein Lehr- und Arbeitsbuch für das Grundstudium.* 12., überarb. und erw. Aufl. Viewegs Fachbücher der Technik. Wiesbaden: Vieweg, 2009 (siehe S. 5 f., 12).

A MATLAB-Quellcode

A.1 MATLAB-Quellcode aus Abschnitt 3.1

```
%% Gekoppelte Oszillatoren mit N = 2 Massen
% Alte Variablen löschen, Fenster schliessen
clear all; close all; clc;

%% Parameter für die analytische Berechnung
m = 0.25;                          % Masse in kg
D = 10; D12 = D/10;                % Federkonstanten in N/m
A = 0.5;                           % Amplitude in m
phi1 = 0; phi2 = 0;                % Phasenwinkel in rad
omega1 = sqrt(D/m);                % Kreisfrequenzen omega1 und omega2
omega2 = sqrt((D+2*D12)/m);
t = 0:0.02:25; t = t';             % Zeitintervalle generieren

%% Analyt. Berechnung der Schwingungen in Normalkoordinaten
xi_pos = A*cos(omega1.*t + phi1);        % Berechnung von xi^+(t)
xi_neg = A*cos(omega2.*t + phi2);        % Berechnung von xi^-(t)

%% Analyt. Berechnung der Schwingungsamplituden x_1(t) und x_2(t)
x1 = 2*A*cos( (omega1 - omega2).*t/2 + (phi1 - phi2)/2 ) ...
    .*cos((omega1 + omega2).*t/2 + (phi1 + phi2)/2);
x2 = -2*A*sin( (omega1 - omega2).*t/2 + (phi1 - phi2)/2) ...
    .*sin((omega1 + omega2).*t/2 + (phi1 + phi2)/2);

%% Numerische Integration
TS = 0:0.02:25;                    % Zeitschritt
AW = [1 0 0 0];                    % Vektor mit Anfangswerten
OPT = odeset('RelTol',1e-6,'AbsTol',1e-6,'MaxStep',0.01);

% Ausführung des Solvers ode45
[T,X] = ode45(@funktion_modell_N2,TS,AW,OPT);

%% Numerisch berechnete Ergebnisse in Normalkoordinaten
XI_pos = (X(:,1) + X(:,2))./2;
XI_neg = (X(:,1) - X(:,2))./2;

%% Fehlerabschätzung
err1 = abs((x1 - X(:,1))./x1);  % Berechnung des relativen Fehlers
err2 = abs((x2 - X(:,2))./x2);

% Logarithmieren und Mittelwert/StdAbw berechnen
log_err1 = log10(err1(50:1200));
log_err2 = log10(err2(50:1200));
mwerr1 = mean(log_err1); stderr1 = std(log_err1);
mwerr2 = mean(log_err2); stderr2 = std(log_err2);
% Ausgabe im Command Window
fprintf('Fehlerabschätzung␣von␣dx2:␣10^(%g␣+/-␣%g)␣\n',mwerr2,stderr2);
fprintf('Fehlerabschätzung␣von␣dx1:␣10^(%g␣+/-␣%g)␣\n',mwerr1,stderr1);
```

```
48
49 %% Plotten der Ergebnisse
50 % Darstellung von x_1 (Vergleich analyt./numerisch)
51 figure;                                          % Neue Grafik eröffnen
52 subplot(211)                                     % Subplot Nr. 1
53 plot(t,x1,'k-',T,X(:,1),'k.');                   % Plot der Grafik
54 ylim([-2 2]);                                    % y-Achsenskalierung
55 ylabel('Amplitude von x_1(t) in m');             % x-Achse beschriften
56 lc1 = legend('Analytisch','Numerisch',...        % Legende erstellen
57     'Location','Best','Orientation','Horizontal');
58 set(lc1,'Color','none','Box','off');
59
60 % Relativer Fehler von x_1
61 subplot(212)                                     % Subplot Nr. 2
62 semilogy(t,err1,'k.');                           % Plotten
63 ylim([1e-11 1e-5]);                              % y-Achsenskalierung
64 ylabel('Relativer Fehler');                      % Beschriftung y-Achse
65 xlabel('Zeit t in s');                           % Beschriftung x-Achse
66
67 %% Darstellung von x_2 analog zum x_1-Plot
68 figure;
69 subplot(211)
70 plot(t,x2,'r-',T,X(:,2),'r.'); ylim([-2 2]);
71 ylabel('Amplitude von x_2(t) in m');
72 lc2 = legend('Analytisch','Numerisch', ...
73     'Location','Best','Orientation','Horizontal');
74 set(lc2,'Color','none','Box','off');
75 subplot(212)
76 semilogy(t,err2,'r.'); ylim([1e-11 1e-5]);
77 ylabel('Relativer Fehler'); xlabel('Zeit t in s');
78
79 %% Darstellung von xi_pos und xi_neg (Normalkoordinaten)
80 figure;
81 % Überlagerung von xi^+ und xi^-
82 subplot(311)
83 plot(t,xi_pos,T,XI_pos,'k.',t,xi_neg,T,XI_neg,'r.');
84 ylim([-1 1]); ylabel('\xi^+, \xi^-');
85 % Darstellung von xi^+
86 subplot(312)
87 plot(t,xi_pos,T,XI_pos,'k.'); ylim([-1 1]); ylabel('\xi^+');
88 % Darstellung von xi^-
89 subplot(313)
90 plot(t,xi_neg,T,XI_neg,'r.'); ylim([-1 1]);
91 ylabel('\xi^-'); xlabel('Zeit t in s');
```

Zur Ausführung des obigen Codes wird eine MATLAB-Funktion mit dem Dateinamen `funktion_modell_N2.m` benötigt.

```
1 function dx = funktion_modell_N2(t,x)
2 % Dgl-Parameter
3       m = 0.25;           % Masse
4       D1 = 10;            % Federkonstante
5       D2 = D1/10;         % Federkonstante der Kopplungsfeder
```

```
6       D3 = D1;
7       dx = zeros(4,1);    % Erstellen eines 4x1-Nullvektors
8
9   % System von Dgl'en 1. Ordnung aufstellen
10      dx(1) = x(3);
11      dx(2) = x(4);
12      dx(3) = (-D1*x(1) + D2*(x(2) - x(1)))/m;
13      dx(4) = (-D3*x(2) - D2*(x(2) - x(1)))/m;
14  end
```

A.2 MATLAB-Quellcode zum Abschnitt 3.2

```
1   %% Gekoppelte Oszillatoren mit N=5 Massen
2   clear all; close all; clc;       % Löschen alter Variablen
3
4   %% Numerische Berechnung mit ode45
5   TS = 0:0.05:150;                    % Zeitintervall
6   AW = [1 0 0 0 0 0 0 0 0 0];      % Anfangswerte
7   OPT = odeset('RelTol',1e-6,'AbsTol',1e-6,'MaxStep',0.01);
8   [T,X] = ode45(@funktion_modell_N5,TS,AW,OPT);
9
10  %% Analytische Berechnung
11  % Simulationsparameter
12  m = 0.25;                          % Masse der Gewichte
13  D1 = 10; D2 = D1/3; D3 = D1/3;   % Federkonstanten
14  D4 = D1/3; D5 = D1/3; D6 = D1;
15  m = m*eye(5,5);                    % Massenmatrix
16  D = [-(D1+D2) D2 0 0 0;          % Steifigkeitsmatrix
17      D2 -(D2+D3) D3 0 0;
18      0 D3 -(D3+D4) D4 0;
19      0 0 D4 -(D4+D5) D5;
20      0 0 0 D5 -(D5+D6)];
21
22  % Lösen des Eigenwertproblems
23  [EV, EW] = eig(-D/m);    % EW: Eigenwert, EV: Eigenvektor
24  omega = sqrt(EW);        % Berechnung der Eigenfrequenzen
25  x0 = [1; 0; 0; 0; 0];    % Anfangsbed. für x(t=0)
26  C = EV\x0;               % Lösung des Gleichungssystems
27
28  % Spezielle Lösung der DGL
29  t = T';
30  x = C(1)*EV(:,1)*cos(omega(1,1).*t) + ...
31      C(2)*EV(:,2)*cos(omega(2,2).*t) + ...
32      C(3)*EV(:,3)*cos(omega(3,3).*t) + ...
33      C(4)*EV(:,4)*cos(omega(4,4).*t) + ...
34      C(5)*EV(:,5)*cos(omega(5,5).*t);
35  t = t'; x = x';          % t und x transponieren
36
37  %% Diagramme für numerische Ergebnisse von x_1(t) bis x_5(t)
38  % Plot von x_1(t) bis x_5(t)
39  figure
40  yl = {'x_1(t)','x_2(t)','x_3(t)','x_4(t)','x_5(t)'};
```

```
41
42 for i = 1:5
43     subplot(5,1,i)
44     plot(T,X(:,i),'.'); ylabel(yl(i));
45     set(gca,'XMinorTick','on','YLim',[-1 1]);
46     if i == 1
47         title('Numerische Ergebnisse für x_1(t) bis x_5(t)');
48     end
49 end
50 xlabel('Zeit t in s');
51
52 %% Diagramme für analytische Ergebnisse von x_1(t) bis x_5(t)
53 figure
54 for i = 1:5
55     subplot(5,1,i)
56     plot(t,x(:,i),'-'); ylabel(yl(i));
57     set(gca,'XMinorTick','on','YLim',[-1 1]);
58     if i == 1
59         title('Analytisch berechnete Ergebnisse für x_1(t) bis x_5(t)');
60     end
61 end
62 xlabel('Zeit t in s');
63
64 %% Vergleich der analyt. und num. Resultate von x_1 und x_5
65 % Berechnung der relativen Fehler, logarithmieren
66 err1 = abs((x(:,1) - X(:,1))./x(:,1));
67 err5 = abs((x(:,5) - X(:,5))./x(:,5));
68 log_err1 = log10(err1(50:3000));
69 log_err5 = log10(err1(50:3000));
70
71 % Mittelwert und Standardabweichung
72 mwerr1 = mean(log_err1); stderr1 = std(log_err1);
73 mwerr5 = mean(log_err5); stderr5 = std(log_err5);
74
75 % Ausgabe im command window
76 fprintf('Fehlerabschätzung von err1: 10^(%g +/- %g) \n',mwerr1,stderr1);
77 fprintf('Fehlerabschätzung von err5: 10^(%g +/- %g) \n',mwerr5,stderr5);
78
79 %% Vergleichsplots der num. und analyt. Ergebnisse, Fehlerabschätzung
80 figure
81 subplot(4,1,1)                    % Plot von x_1 (analyt./num.)
82 plot(T(1:3000,1),X(1:3000,1),'k.',t(1:3000,1),x(1:3000,1),'k-');
83 title('Vergleich analytisch/numerisch');
84 ylabel('x_1(t)'); ylim([-2 1]);
85 lc = legend('Numerisch','Analytisch', ...
86     'Location','Best','Orientation','Horizontal');
87 set(lc,'Color','none','Box','off');
88 set(gca,'XMinorTick','on');
89 subplot(4,1,2)                    % Plot von err1
90 semilogy(t(1:3000,1),err1(1:3000,1),'k.'); ylim([1e-10 1e-4]);
91 ylabel('rel. Fehler von x_1(t)'); set(gca,'XMinorTick','on');
```

```
92
93  subplot(4,1,3)                    % Plot von x_5 (analyt./num.)
94  plot(T(1:3000,1),X(1:3000,5),'r.',t(1:3000,1),x(1:3000,5),'r-');
95  ylabel('x_5(t)'); ylim([-2 1]);
96  lc = legend('Numerisch','Analytisch', ...
97      'Location','Best','Orientation','Horizontal');
98  set(lc,'Color','none','Box','off');
99  set(gca,'XMinorTick','on');
100 subplot(4,1,4)                    % Plot von err5
101 semilogy(t(1:3000,1),err5(1:3000,1),'r.'); ylim([1e-10 1e-4]);
102 ylabel('rel. Fehler von x_5(t)');
103 xlabel('Zeit t in s'); set(gca,'XMinorTick','on');
104
105
106 %% Phasenportraits
107 figure
108 subplot(221)                      % x_1 gegen v_1
109 plot(X(1:1000,1),X(1:1000,6),'k-');
110 xlabel('x_1(t)'); ylabel('v_1(t)');
111 subplot(222)                      % 3D-Plot von x_1, v_1 und T
112 plot3(X(1:1000,1),X(1:1000,6),T(1:1000,1),'k-');
113 xlabel('x_1(t)'); ylabel('v_1(t)'); zlabel('Zeit t');
114 grid on;
115 subplot(223)                      % x_5 gegen v_5
116 plot(X(1:1000,5),X(1:1000,10),'r-');
117 xlabel('x_5(t)'); ylabel('v_5(t)');
118 subplot(224)                      % 3D-Plot von x_5, v_5 und T
119 plot3(X(1:1000,5),X(1:1000,10),T(1:1000,1),'r-');
120 xlabel('x_5(t)'); ylabel('v_5(t)'); zlabel('Zeit t');
121 grid on;
```

Zur Ausführung des obigen Codes wird eine MATLAB-Funktion mit dem Dateinamen `funktion_modell_N5.m` benötigt.

```
1  function dx = funktion_modell_N5(t,x)
2  % Dgl-Parameter
3      m = 0.25;        % Masse
4      D = 10;          % Federkonstante d. linken u. rechten Feder
5      D25 = D/3;       % Federkonstanten der Koppelfedern
6
7      D = [D D25 D25 D25 D25 D];  % Erstellen des D-Vektors
8      dx = zeros(10,1);           % Vektor für 10x1 Dgl-System
9
10 % Dgl-System
11     dx(1) = x(6);
12     dx(2) = x(7);
13     dx(3) = x(8);
14     dx(4) = x(9);
15     dx(5) = x(10);
16     dx(6) = (-D(1)*x(1) - D(2)*x(1) + D(2)*x(2))/m;
17     dx(7) = (D(2)*x(1) - D(2)*x(2) - D(3)*x(2) + D(3)*x(3))/m;
18     dx(8) = (D(3)*x(2) - D(3)*x(3) - D(4)*x(3) + D(4)*x(4))/m;
19     dx(9) = (D(4)*x(3) - D(4)*x(4) - D(5)*x(4) + D(5)*x(5))/m;
```

```
    dx(10) = (D(5)*x(4) - D(5)*x(5) - D(6)*x(5))/m;
end
```

A.3 MATLAB-Quellcode zum Abschnitt 3.3

```
%% Gekoppelte Oszillatoren mit N=11 Massen
clear all; close all; clc;

%% Parameter für die numerische Simulation
TS = 0:0.02:100;                              % Zeitintervall
AW = [1 0 0 0 0 1 0 0 0 0 -1 ...            % AW für Positionen
    0 0 0 0 0 0 0 0 0 0 0];                  % AW für Geschwindigkeiten
OPT = odeset('RelTol',1e-6,'AbsTol',1e-6,'MaxStep',0.01);
[T,X] = ode45(@funktion_modell_N11,TS,AW,OPT);

%% Gesamtübersicht der num. Resultate, Schwingungsamplituden
% von x_1 bis x_11
figure
yl = {'x_1(t)','x_2(t)','x_3(t)','x_4(t)','x_5(t)','x_6(t)', ...
    'x_7(t)','x_8(t)','x_9(t)','x_{10}(t)','x_{11}(t)'};
for i = 1:11
    subplot(11,1,i)
    plot(T,X(:,i)); ylim([-1.5 1.5]);
    set(gca,'XMinorTick','on');
    if i == 1
        title('Schwingungsamplituden␣für␣N=11␣Massen,␣numerisch␣berechnet');
    end
    ylabel(yl(i));
end
xlabel('Zeit␣t␣in␣s');

%% Analytische Berechnung im Bereich von t=0 bis t=25 sec
% Variablen erstellen
t = 0:0.02:100;             % Zeitintervall der Simulation
M = 1;                      % Massen an der Wand
MM = 5;                     % Masse in der Mitte
m = 0.25;                   % Massen zwischen M und MM

% Federkonstanten
Dw = 50; D1 = Dw;           % Federkonstante an der li. Wand
D2 = Dw/2; D3 = Dw/2;       % zwischen M und MM
D4 = Dw/2; D5 = Dw/2;
D6 = Dw/2;                  % in der Mitte
D7 = Dw/2; D8 = Dw/2;       % zwischen MM und M
D9 = Dw/2; D10 = Dw/2;
D11 = Dw/2;
D12 = Dw;                   % Federkonst. an der re. Wand

%% Erstellen der Steifigkeits- und Massenmatrix
D = [-(D1+D2) D2 0 0 0 0 0 0 0 0 0;
    D2 -(D2+D3) D3 0 0 0 0 0 0 0 0;
    0 D3 -(D3+D4) D4 0 0 0 0 0 0 0;
```

```
48      0 0 D4 -(D4+D5) D5 0 0 0 0 0 0;
49      0 0 0 D5 -(D5+D6) D6 0 0 0 0 0;
50      0 0 0 0 D6 -(D6+D7) D7 0 0 0 0;
51      0 0 0 0 0 D7 -(D7+D8) D8 0 0 0;
52      0 0 0 0 0 0 D8 -(D8+D9) D9 0 0;
53      0 0 0 0 0 0 0 D9 -(D9+D10) D10 0;
54      0 0 0 0 0 0 0 0 D10 -(D10+D11) D11;
55      0 0 0 0 0 0 0 0 0 D11 -(D11+D12)];
56
57  m = [M 0 0 0 0 0 0 0 0 0 0;
58      0 m 0 0 0 0 0 0 0 0 0;
59      0 0 m 0 0 0 0 0 0 0 0;
60      0 0 0 m 0 0 0 0 0 0 0;
61      0 0 0 0 m 0 0 0 0 0 0;
62      0 0 0 0 0 MM 0 0 0 0 0;
63      0 0 0 0 0 0 m 0 0 0 0;
64      0 0 0 0 0 0 0 m 0 0 0;
65      0 0 0 0 0 0 0 0 m 0 0;
66      0 0 0 0 0 0 0 0 0 m 0;
67      0 0 0 0 0 0 0 0 0 0 M];
68
69  %% Lösen des Eigenwertproblems
70  [EV, EW] = eig(-D/m);     % EW: Eigenwert, EV: Eigenvektor
71  omega = sqrt(EW);         % Berechnung der Eigenfrequenzen
72  % Anfangswerte zu x(t=0)
73  x0 = [1; 0; 0; 0; 0; 1; 0; 0; 0; 0; -1];
74  Cx = EV\x0;               % Ermittlung der Integrationskonstenten
75
76  %% Spezielle Lösung der DGL
77  x = Cx(1).*EV(:,1)*cos(omega(1,1).*t) + ...
78      Cx(2).*EV(:,2)*cos(omega(2,2).*t) + ...
79      Cx(3).*EV(:,3)*cos(omega(3,3).*t) + ...
80      Cx(4).*EV(:,4)*cos(omega(4,4).*t) + ...
81      Cx(5).*EV(:,5)*cos(omega(5,5).*t) + ...
82      Cx(6).*EV(:,6)*cos(omega(6,6).*t) + ...
83      Cx(7).*EV(:,7)*cos(omega(7,7).*t) + ...
84      Cx(8).*EV(:,8)*cos(omega(8,8).*t) + ...
85      Cx(9).*EV(:,9)*cos(omega(9,9).*t) + ...
86      Cx(10).*EV(:,10)*cos(omega(10,10).*t) + ...
87      Cx(11).*EV(:,11)*cos(omega(11,11).*t);
88  x = x'; t = t';           % Matrizen/Vektoren transponieren
89
90  %% Gesamtübersicht zur analytischen Rechnung
91  figure
92  yl = {'x_1(t)','x_2(t)','x_3(t)','x_4(t)','x_5(t)','x_6(t)', ...
93      'x_7(t)','x_8(t)','x_9(t)','x_{10}(t)','x_{11}(t)'};
94  for i = 1:11
95      subplot(11,1,i)
96      plot(t,x(:,i)); ylim([-1.5 1.5]);
97      if i == 1
98          title('Schwingungsamplituden␣für␣N=11␣Massen,␣analytisch␣berechnet');
```

```
    end
    ylabel(yl(i));
end
xlabel('Zeit t in s');

%% Vergleich analytisch/numerisch für x_1, x_6 und x_11
yl = {'x_1(t)','x_6(t)','x_{11}(t)'};
Tp = T(1:1250,1);    Xp = X(1:1250,[1 6 11]);
tp = t(1:1250,1);    xp = x(1:1250,[1 6 11]);
figure
for i = 1:3
    subplot(3,1,i)
    plot(Tp(:,1),Xp(:,i),'-',tp(:,1),xp(:,i),'-'); ylabel(yl(i));
    set(gca,'XMinorTick','on','YLim',[-2 1.5]);
    lc = legend('Numerisch','Analytisch', ...
    'Location','Best','Orientation','Horizontal');
    set(lc,'Color','none','Box','off');
        if i == 1
        title('Vergleich zwischen numerischen und analytischen Ergebnissen');
    end
end
xlabel('Zeit t in s');
```

Zur Ausführung des obigen Codes wird eine MATLAB-Funktion mit dem Dateinamen `funktion_modell_N11.m` benötigt.

```
function dx = funktion_modell_N11(t,x)
% Dgl-Parameter
M = 1;           % Äussere Massen an den Wänden
MM = 5;          % Masse in der Mitte
m = 0.25;        % Massen zwischen M und MM

Dw = 50;         % Federkonstante an der Wand
D1 = Dw; D2 = Dw/2; D3 = Dw/2; D4 = Dw/2; D5 = Dw/2; D6 = Dw/2;
D7 = Dw/2; D8 = Dw/2; D9 = Dw/2; D10 = Dw/2; D11 = Dw/2;
D12 = Dw;

D = [D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12];
m = [M m m m m MM m m m m M];
dx = zeros(22,1);

% Dgl-System
    dx(1) = x(12);
    dx(2) = x(13);
    dx(3) = x(14);
    dx(4) = x(15);
    dx(5) = x(16);
    dx(6) = x(17);
    dx(7) = x(18);
    dx(8) = x(19);
    dx(9) = x(20);
    dx(10) = x(21);
    dx(11) = x(22);
```

```
28
29      dx(12) = (-D(1)*x(1) - D(2)*x(1) + D(2)*x(2))/m(1);
30      dx(13) = (D(2)*x(1) - D(2)*x(2) - D(3)*x(2) + D(3)*x(3))/m(2);
31      dx(14) = (D(3)*x(2) - D(3)*x(3) - D(4)*x(3) + D(4)*x(4))/m(3);
32      dx(15) = (D(4)*x(3) - D(3)*x(4) - D(4)*x(4) + D(4)*x(5))/m(4);
33      dx(16) = (D(5)*x(4) - D(3)*x(5) - D(4)*x(5) + D(4)*x(6))/m(5);
34      dx(17) = (D(6)*x(5) - D(3)*x(6) - D(4)*x(6) + D(4)*x(7))/m(6);
35      dx(18) = (D(7)*x(6) - D(3)*x(7) - D(4)*x(7) + D(4)*x(8))/m(7);
36      dx(19) = (D(8)*x(7) - D(3)*x(8) - D(4)*x(8) + D(4)*x(9))/m(8);
37      dx(20) = (D(9)*x(8) - D(3)*x(9) - D(4)*x(9) + D(4)*x(10))/m(9);
38      dx(21) = (D(10)*x(9) - D(3)*x(10) - D(4)*x(10) + D(4)*x(11))/m(10);
39      dx(22) = (D(11)*x(10) - D(11)*x(11) - D(12)*x(11))/m(11);
40  end
```