Institut für Informatik
Dr. Tobias Neckel
Christoph Kowitz
Alfredo Parra

Technische Universität München
WS 2015/2016

Einführung in die wissenschaftliche Programmierung

Übungsblatt 11

1.) Masse-Feder-System

Wir betrachten ein Masse-Feder-System wie in der Abbildung unten dargestellt. Zwei Körper mit Masse m_1 und m_2 sind durch zwei Federn mit Federkonstanten k_1 und k_2 miteinander verbunden. Die Länge der Federn ist L_1 und L_2. Die Massen gleiten auf einer Oberfläche mit Reibungskoeffizienten b_1 und b_2.

Beschrieben wird dies durch ein System von gewöhnlichen Differentialgleichungen zweiter Ordnung

$$\begin{aligned} m_1 x_1'' + b_1 x_1' + k_1 (x_1 - L_1) - k_2 (x_2 - x_1 - L_2) = 0, \\ m_2 x_2'' + b_2 x_2' + k_2 (x_2 - x_1 - L_2) = 0. \end{aligned}$$

Dieses System kann in ein System von Gleichungen erster Ordnung umgewandelt werden

$$x_1' = y_1, \tag{1}$$

$$y_1' = \frac{1}{m_1} \cdot (-b_1 y_1 - k_1 (x_1 - L_1) + k_2 (x_2 - x_1 - L_2)), \tag{2}$$

$$x_2' = y_2, \tag{3}$$

$$y_2' = \frac{1}{m_2} \cdot (-b_2 y_2 - k_2 (x_2 - x_1 - L_2)). \tag{4}$$

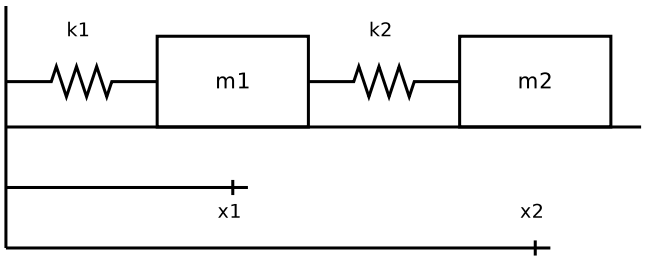
Lösen Sie dieses System mit den Parametern

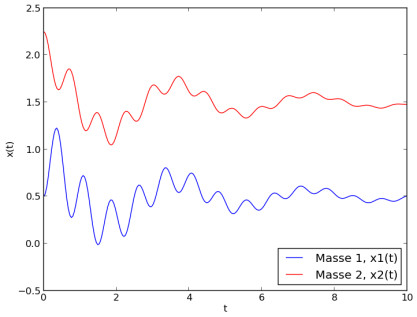
	m	k	L	b
1	1	8	0.5	0.8
2	1.5	40	1.0	0.5

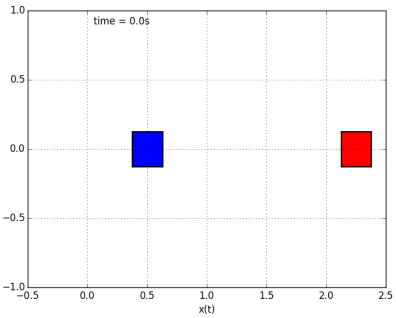
und den Anfangsbedingungen

	x	y
1	0.5	0
2	2.25	0

mit der Funktion `scipy.integrate.odeint`. Sie benötigen hierfür eine Funktion f welche die Rechte Seite des Systems (1)-(4) berechnet. Übergeben Sie dann diese Funktion, Anfangsbedingungen und eine Liste von Zeitschritten an `odeint`. Lösen Sie das System für $t \in [0, 10]$ mit z.B. 100 Zeitschritten. Sie bekommen die Lösung an allen Zeitschritten als Rückgabewert. Stellen Sie die Lösungen $x_1(t)$ und $x_2(t)$ mit `matplotlib` dar.







Fortgeschritten

Verwenden Sie das Paket `matplotlib.animation`, um die Bewegung der Massen zu animieren.

2.) Lineare Konvektionsgleichung (Hausaufgabe)

Wir betrachten die folgende lineare partielle Differentialgleichung

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0, \tag{5}$$

mit Anfangsbedingung $f(x, t_0 = 0) = F(x)$ und konstanter Geschwindigkeit $u = 1.5$. Die analytische Lösung dieser Gleichung ist gegeben durch $f(x, t) = F(x - ut)$.

a.) Diskretisieren Sie Gleichung (5) in Zeit und Raum. Nehmen Sie für die Raumdiskretisierung $x \in [0, 1]$, $x_j = j\Delta x$, $j = 0, 1, \ldots, M-1, M$, mit $M = 100$. Die Zeit kann im Bereich $t \in [0, 1]$ diskretisiert werden durch $t_n = n\Delta t$, $n = 0, 1, \ldots, N$. Der Zeitschritt Δt muss so gewählt werden, dass die Bedingung $u\Delta t/\Delta x \leq 1$ erfüllt wird. Die diskrete Lösung f_j^n soll dann die echte Lösung $f(j\Delta x, n\Delta t)$ annähern. Benutzen Sie für die Zeitabhängigkeit das explizite Euler-Verfahren, sowie den Rückwärtsdifferenzquotienten für die Raumabhängigkeit, der definiert ist durch

$$\frac{\partial f}{\partial x} \approx \frac{f_j - f_{j-1}}{\Delta x}. \tag{6}$$

Formulieren Sie die entstehende Gleichung als Zeit-Update, d.h., schreiben Sie das Gleichungssystem für f_j^{n+1}.

b.) Implementieren Sie die Lösung dieses Gleichungssystems in Python mit $F(x) = e^{-\frac{(x-0.1)^2}{.01}}$ und Randbedingungen $f(x = 0, t) = f(x = 1, t) = 0$. Was beobachten Sie, verglichen zur exakten Lösung?