Professur für Mathematik
Fakultät für Elektrische Energiesysteme
und Informationstechnik

Universität der Bundeswehr München

Praktikum Numerische Simulation in der Technik

Synchronisation von chaotischen Systemen

Stichworte: numerische Integration von nichtlinearen Dynamiken, Stabilität, Chaos, Synchronisation

Betreuer: Richter

Termine: WT oder FT.

Thema:
Ein dynamisches System wird mathematisch durch $n \in \mathbb{N}$ gekoppelte Differentialgleichungen (DGL) beschrieben. Speziell werden autonome Systeme der Form

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}) \tag{1}$$

betrachtet, ausführlich geschrieben in der Form

$$\begin{aligned} \frac{dx_1}{dt} &= f_1(x_1, x_2, \ldots, x_n)\,, \\ \frac{dx_2}{dt} &= f_2(x_1, x_2, \ldots, x_n)\,, \\ \vdots \quad & \quad \vdots \\ \frac{dx_n}{dt} &= f_n(x_1, x_2, \ldots, x_n)\,, \end{aligned} \tag{2}$$

wobei $x_i = x_i(t)$ zeitabhängige Funktionen sind, welche zum zeitabhängigen Vektor $\mathbf{x} = (x_1, \ldots, x_n)^T$ (genannt **Zustandsvektor**) zusammengefasst werden, ebenso wie die Funktionen f_i zum Vektorfeld $\mathbf{f} = (f_1, \ldots, f_n)^T$ zusammengefasst werden. Das System (1) beziehungsweise (2) wird ergänzt durch Hinzunahme einer Anfangsbedingung $\mathbf{x}(t_0) = \mathbf{x}_0$, man erhält dann ein **Anfangswertproblem** (AWP)

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{x}_0. \tag{3}$$

Unter geeigneten Bedingungen an die Funktionen f_i existiert eine eindeutige Lösung $\mathbf{x} : [t_0, \infty) \to \mathbb{R}^n$ des Anfangswertproblems (3).

Ist die Anzahl an Komponenten n größer als 2 und sind einige **Nichtlinearitäten** in den Funktionen f_i vorhanden (dafür reichen schon einfache Nichtlinearitäten der

Form $x_i x_j$ aus) kann **Chaos** auftreten. Jedoch ist Chaos alles anderes als ein zufälliges Verhalten, wie es in diesem Praktikum untersucht wird. Daher ist das Thema des Praktikums die numerische Lösung des AWP (3) und die Untersuchung des Langzeitverhaltens $t \to \infty$ bei dem folgenden nichtlinearen, autonomen DGL-System

$$\begin{aligned}
\frac{dx_1}{dt} &= -\sigma x_1 + \sigma x_2 \,, \\
\frac{dx_2}{dt} &= -x_2 + r x_1 - x_1 x_3 \,, \\
\frac{dx_3}{dt} &= -b x_3 + x_1 x_2 \,, \\
\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} &= \begin{pmatrix} -\sigma x_1 + \sigma x_2 \\ r x_1 - x_2 - x_1 x_3 \\ x_1 x_2 - b x_3 \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2, x_3) \\ f_2(x_1, x_2, x_3) \\ f_3(x_1, x_2, x_3) \end{pmatrix} , \qquad (4)
\end{aligned}$$

wobei konstante positive Parameter $\sigma > 0$, $r > 0$ und $b > 0$ vorausgesetzt werden. Diese Dynamik wird Lorenz-System oder Lorenz-Gleichungen genannt, nach Edward Norton Lorenz (1917-2008), dem Begründer der Chaos-Theorie, und ist eine vereinfachte Beschreibung der zweidimensionalen Konvektion-Strömung in der Atmosphäre zwischen einer warmen und einer kalten Schicht. Für das Praktikum wird immer mit den Standardparametern $\sigma = 10$ und $b = 8/3$ (bei unterschiedlichen r-Parametern) gearbeitet.

Literatur:

- Kurt Meyberg, Peter Vachenauer, *Höhere Mathematik 2*, Springer Verlag, dort insbesondere Kapitel 9 (Gewöhnliche Differentialgleichungen), Paragraph 10 (Stabilität).

- Claus Hillermeier, *Dynamische Systeme*, Skriptum zur Vorlesung an der Universität der Bundeswehr München.

- Steven Strogatz, *Nonlinear Dynamics and Chaos*, Perseus Books, Kapitel 9, insbesondere Abschnitt 9.6 ("Using Chaos to Send Secret Messages").

1. Aufgabe. Auch wenn die Haupttätigkeit wird sich darauf konzentrieren, das zeitabhängige Verhalten der Dynamik zu untersuchen, ist die erste Aufgabe immer die Suche nach konstanten Lösungen der DGL, die sogenannten **Gleichgewichtslagen** (GGL) (auch Ruhelagen bezeichnet): Konstante Werte in den Komponenten von $\mathbf{x}_{GG}$, welche die DGL erfüllen und somit folgendes algebraisches Gleichungssystem $\frac{d}{dt}\mathbf{x}_{GG} = \mathbf{0} = \mathbf{f}(\mathbf{x}_{GG})$ genügen. Mit anderen Worten: Einmal ist die Gleichgewichtslage für $t = t_0$ erreicht worden, verhärrt das durch $\mathbf{x} = \mathbf{x}(t)$ modellierte System für alle Zeit $t \geq t_0$ in diesem unveränderlichem Zustand.

Wichtig ist jetzt die Frage, wie sich ein System verhält, wenn es "durch eine kleine Störung aus dem Gleichgewicht gebracht" wird, also in einen Zustand nahe einer GGL $\mathbf{x}_{GG}$ gebracht wird. Kehrt das System ins Gleichgewicht zurück (d.h. ist die entsprechende GGL stabil) oder die anfängliche Störung wächst und wächst (die entsprechende GGL ist instabil)? Man interessiert sich also für das zeitliche Verhalten der Lösung des AWP

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{x}_{GG} + \overbrace{\Delta\mathbf{x}}^{\text{kleine Abweichungen}} \tag{5}$$

für $t \to \infty$. Oder äquivalenterweise, welches ist das Langzeitverhalten der Dynamik der ursprnglich kleinen Abweichungen $\Delta\mathbf{x}$? Solange diese Abweichungen aus der untersuchten GGL $\mathbf{x}_{GG}$ klein bleiben, kann die Dynamik von $\Delta\mathbf{x}$ durch Linearisierung der (im Allgemeinen) nichtlinearen Dynamik (1) um die Ruhelage gewonnen werden

$$\begin{aligned}
\frac{d}{dt}(\mathbf{x}_{GG} + \Delta\mathbf{x}) = \overbrace{\mathbf{0}}^{\text{konst. } \mathbf{x}_{GG}} + \frac{d}{dt}\Delta\mathbf{x} \quad &= \quad \mathbf{f}(\mathbf{x}_{GG} + \Delta\mathbf{x}) \\
&\overset{\Delta\mathbf{x}\,\text{klein}}{\approx} \overbrace{\mathbf{f}(\mathbf{x}_{GG})}^{=\mathbf{0},\,\text{weil GGL}} + \underbrace{\left.\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}\right|_{\mathbf{x}_{GG}}}_{\text{Jacobimatrix } A} \Delta\mathbf{x}\,,
\end{aligned}$$

$$\frac{d}{dt}\Delta\mathbf{x} \approx A\,\Delta\mathbf{x}\,. \tag{6}$$

Für den Fall, dass alle Komponenten in $\mathbf{f}$ stetig differenzierbar sind, ist eine Klassifikation der **(asymptotischen) Stabilität** *vs.* **Instabilität** der Gleichgewichtslage gegenüber kleinen Störungen möglich, wenn alle Eigenwerte von A entweder einen strikt negativen Realteil haben oder wenn mindestens ein Eigenwert einen strikt positiven Realteil hat. Lesen Sie hierzu die entsprechenden Aussagen in Satz 10.3 ("Linearisierungstheorem") des 9. Kapitels von Meyberg und Vachenauer sowie das anschließende "Rechenschema" (Stichwort: Routh-Hurwitz-Kriterium).

Für das Lorenz-System (4) mit $\sigma = 10$ und $b = 8/3$ beweisen Sie analytisch folgende Eigenschaften

- Für $r < 1$ ist $\mathbf{x}_{GG,0} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ die einzige asymptotisch stabile Gleichgewichtslage.
- Für $1 < r < \sigma\dfrac{\sigma + b + 3}{\sigma - b - 1} = 24.7368$ existieren zwei asymptotisch stabilen Gleichgewichtslagen $\mathbf{x}_{GG,\pm} = \begin{pmatrix} \pm\sqrt{b(r-1)} \\ \pm\sqrt{b(r-1)} \\ r-1 \end{pmatrix}$.

- Für $r > \sigma \dfrac{\sigma + b + 3}{\sigma - b - 1} = 24.7368$ existiert keine asymptotisch stabile Gleichgewichtslage mehr, hier treibt Chaos sein Unwesen (aber nicht nur Chaos).

Anleitung: Bei der Berechnung der Eigenwerte von A (6) ergeben sich die charakteristischen Polynome

$$\chi(\lambda) = (\lambda + b) \cdot \Big(\lambda^2 + \lambda(1 + \sigma) + \sigma(1 - r)\Big)$$

für die erste GGL sowie

$$\chi(\lambda) = \lambda^3 + \lambda^2(1 + \sigma + b) + \lambda \cdot b(r + \sigma) + 2\sigma b(r - 1)$$

für die beiden anderen GGL. Mithilfe des Routh-Hurwitz-Kriteriums (Satz 10.4 im 9. Kapitel des Buchs von Meyberg und Vachenauer) kann entschieden werden, ob die Nullstellen des charakteristischen Polynoms alle negativen Realteil haben oder ob ein positiver Realteil auftritt – somit kann in diesem Fall die Stabilität der GGL klassifiziert werden. Sie können die charakteristischen Polynome symbolisch mittels Matlab rechnen lassen.

2. Aufgabe. (Für Tipps zu dieser Aufgabe: siehe Anhang 1). Auch wenn in Matlab die Funktion `ode45` zur numerischen Integration von DGL-Systemen zur Verfügung steht, wird hier die Dynamik des Lorenz-Systems mittels des einfacheren klassischen Runge-Kutta-Verfahren (RK4) selbst programmiert. Dafür muss zunächst die Vektorfunktion $\mathbf{f}(\mathbf{x})$ für das Lorenz-System als "anonymous function" definiert werden, so wie im Anhang 1 erklärt wird. Diese Funktion soll mehrere unterschiedliche Zustände gleichzeitig bearbeiten, damit mehrere Bahnen verfolgt werden können.

Die numerische Integration eines DGL-Systems (besonders nützlich wenn Nichtlinearitäten vorhanden sind und eine exakte Lösung unmöglich macht) ist ein angenahertes Lösungverfahren, womit die Trajektorie des Zustandsvektors $\mathbf{x}(t)$ an gleichmäßig entfernten, diskreten Zeitpunkten $t_k = k\,h$ bestimmt, mit k einem ganzzähligen Index und h als Dauer eines einzelnen Zeitschritts. Im Fall des klassischen Runge-Kutta-Verfahrens (RK4) für die autonome Dynamik $\dfrac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x})$ wird aus dem bekannten, aktuellen Zustand $\mathbf{x}(k)$ der unbekannte Zustand im nächsten Zeitschritt $\mathbf{x}(k+1)$ folgendermaßen angenähert

$$\begin{aligned}
\mathbf{K}_1 &= h\,\mathbf{f}\left(\mathbf{x}(k)\right) \,, \\
\mathbf{K}_2 &= h\,\mathbf{f}\left(\mathbf{x}(k) + 0.5\mathbf{K}_1\right) \,, \\
\mathbf{K}_3 &= h\,\mathbf{f}\left(\mathbf{x}(k) + 0.5\mathbf{K}_2\right) \,, \\
\mathbf{K}_4 &= h\,\mathbf{f}\left(\mathbf{x}(k) + \mathbf{K}_3\right) \,, \\
\mathbf{x}(k+1) &\approx \mathbf{x}(k) + \frac{1}{6}\left(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4\right) \,. \qquad (7)
\end{aligned}$$

Für die untersuchte Dynamik des Lorenz-Systems wird mit einem Zeitschritt $h = 0.01$

gearbeitet und von $t = 0$ bis ungefähr [1] $t = \texttt{t_end} = 100$ Sekunden integriert, so dass die Zeitkoordinaten durch folgendes Matlab-Array

```
%% Zeit-Aufloesung
h = 0.01;
t_end = 100;
t_end = Dt*(2^nextpow2(t_end/h)-1);
t = 0:h:t_end;
```

gegeben ist. Mittels des RK4-Algorithmus integrieren Sie numerisch die Lorenz-Dynamik für 3 unterschiedliche Anfangszustandsvektoren, z.B.

```
x0 = [[-20; -11; 2], [6; -1; -10], [-10; 8; 13]];
```

so dass 3 unterschiedliche Bahnen entstehen. Diese numerische Integration soll bei den folgenden 4 verschiedenen r-Werten getrennt durchgeführt werden:

- $r = 0.8$,
- $r = 10$,
- $r = 23$,
- $r = 150$,
- und schließlich $r = 28$.

Für jeden r-Fall soll die zeitliche Entwicklung der drei Komponenten des ersten Zustandsvektors dargestellt werden, so wie eine 3-dimensionale Darstellung der Bahnen der 3 betrachteten Zustandsvektoren. Hinweis: Sei `dimx` die Anzahl an Komponenten aller Anfangszustände (in dieser Aufgabe lautet $\texttt{dimx} = 3 * 3$ für die 3 untersuchten 3-dimensionalen Anfangszustände), so dass die zeitliche Entwicklung aller Zustandskomponenten im Matlab-Array `x` mit `dimx` Zeilen (für die Komponenten der Zustandsvektoren) und `length(t)` Spalten (für die aufgelösten Zeitkoordinaten) gespeichert wird, dann enthält `x(1:3,:)` die Evolution des ersten 3-dimensionalen Zustandsvektors (mit `x(1:3,1)` dem ersten Anfangszustandsvektor), `x(4:6,:)` die Evolution des zweiten 3-dimensionalen Zustandsvektors, und `x(7:dimx,:)` die Evolution des dritten 3-dimensionalen Zustandsvektors. Das Plotten der Ergebnisse erfolgt mittels des folgenden Matlab-Codes

```
figure(10);
plot(t,x(1,:),'b-',t,x(2,:),'r-',t,x(3,:),'g-');
xlabel('t');
ylabel('x_1, x_2, x_3');
legend('x_1','x_2','x_3');
figure(11);
for c=1:dimx/3
```

[1] Die Dauer `t_end` der gesamten Simulation wird justiert so, dass das Zeit-Array `t = 0:h:t_end;` eine Länge gleich einer Potenz von 2 ist, damit die spätere Fast-Fourier-Transformation effizienter berechnet werden kann.

```
    plot3(x((c-1)*3+1,:),x((c-1)*3+2,:),x((c-1)*3+3,:),'-');
    hold on;
end
hold on;
for c=1:dimx/3
    plot3(x((c-1)*3+1,1),x((c-1)*3+2,1),x((c-1)*3+3,1),'ko');
    hold on;
end
xlabel('x_1');
ylabel('x_2');
zlabel('x_3');
```

Im dritten r-Fall $r = 23 < \sigma\dfrac{\sigma+b+3}{\sigma-b-1} = 24.7368$ spielen Sie ein bisschen mit den Anfangszuständen bis folgende Situation eintritt: Auch wenn am Ende die Trajektorien an einer der beiden asymptotischen GGL $\mathbf{x}_{GG,\pm} = \begin{pmatrix} \pm\sqrt{b(r-1)} \\ \pm\sqrt{b(r-1)} \\ r-1 \end{pmatrix}$ [2] enden, so wie in 1. Aufgabe behandelt wurde, kann recht lange dauern, bis dieser endgültige GGL-Zustand erreicht wird. Die zeitbeschränkte Zwischenzeit, wo die Komponenten des Zustandsvektors hin und her "oszillieren" ohne ein periodisches Muster aufzuweisen, wird **Transientes Chaos** genannt. Für den Fall $r = 28 > \sigma\dfrac{\sigma+b+3}{\sigma-b-1} = 24.7368$ sehen die Bahnen wie in Abb. 1 aus, charakterisiert durch den berühmten "Schmetterling-Attraktor", diesmal jedoch ohne eine endgültige GGL dauerhaft zu erreichen (1. Aufgabe). Die Zwischenzeit des Transienten Chaos bei $r = 23$ dauert nun die ganze modellierte Zeit, wo das "Oszillieren" ohne periodisches Muster nie aufhört: Dies ist das **chaotische Verhalten**, welches in der nächsten Aufgabe genauer charakterisiert wird. Jedoch nicht alles ist Chaos sobald $r > \sigma\dfrac{\sigma+b+3}{\sigma-b-1} = 24.7368$ gilt: Bei $r = 150$ sind die Bahnen doch periodisch, keine Spur vom Chaos, und man redet von einem **Grenzzyklus**.

Wichtig: Die Vektorfunktion $\mathbf{f}(\mathbf{x})$, so wie im Anhang als "anonymous function" `f = @(x,r) (...)` implementiert worden ist, berechnet mehrere 3-dimensionalen Spaltenvektoren auf einmal. Somit kann `x(:,k)` nicht direkt auf `f` übertragen werden, sondern es muss zunächst als Gruppe von mehreren 3-dimensionalen Spaltenvektoren umgeformt werden

```
x_aux = reshape(x(:,k),3,dimx/3);
```

wobei `dimx` die Anzahl an Komponenten aller Anfangszustände darstellt; wie bereits erwähnt gilt für diese Aufgabe `dimx` $= 3 * 3$, weil 3 unterschiedliche 3-dimensionalen Vektoren verfolgt werden.

[2] Welche der beiden GGL $\mathbf{x}_{GG,+}$ oder $\mathbf{x}_{GG,-}$ angepeilt wird, ist vom Anfangszustand abhängig.

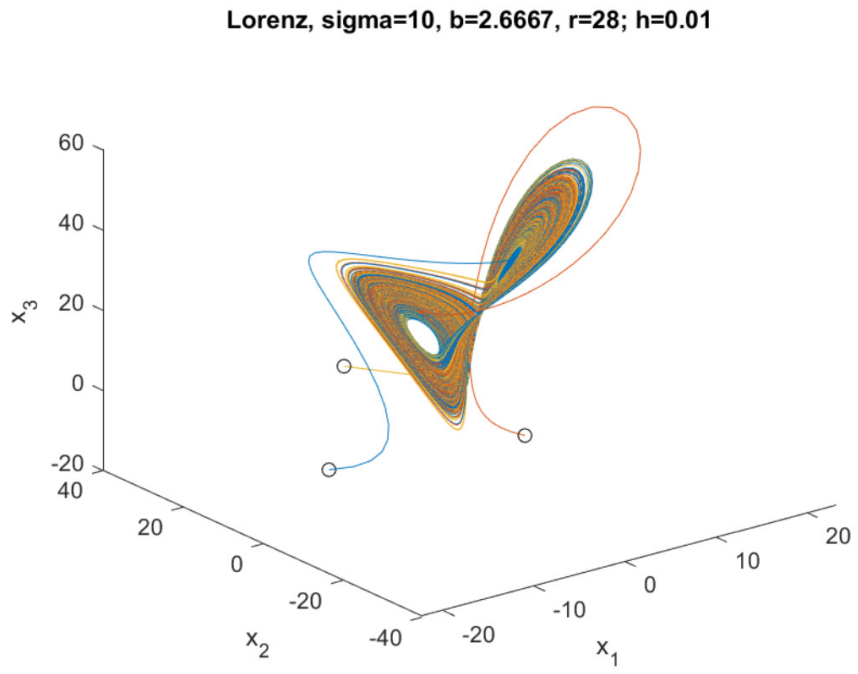

Abbildung 1: Teilergebnis von Aufgabe 2.

3. Aufgabe. Für $r = 28$ sollen nun 2 recht nahe Anfangszustände betrachtet werden, z.B.

```
x0 = [[1; 2; 3], [1; 2; 3+1e-8]];
```

und anschließend, wie in der vorherigen 2. Aufgabe, die entsprechenden 2 Trajektorien numerisch berechnet werden; hier gilt `dimx` $= 2 * 3$. Das Ergebnis soll in `xc` gespeichert werden (bitte das `x` für $r = 28$ aus 2. Aufgabe nicht überschreiben, dies wird später in 4. Aufgabe benötigt): die ersten 3 Komponenten `xc(1:3,:)` gehören der ersten Trajektorie $\mathbf{x}(t)$, die letzten 3 Komponenten `xc(4:6,:)` der zweiten Trajektorie $\mathbf{x}'(t)$. Zum Schluss berechnen Sie die Entfernung zwischen beiden Trajektorien an jedem Zeitpunkt

$$d_c(t) = \sqrt{(x_1(t) - x_1'(t))^2 + (x_2(t) - x_2'(t))^2 + (x_3(t) - x_3'(t))^2}\,. \tag{8}$$

Dies wird in `dc` gespeichert, welches logarithmisch als Funktion der Zeit dargestellt werden soll. Beachten Sie wie $d_c(t)$ nach kurzer Zeit exponentiell wächst (oder in der logarithmischen Darstellung linear wächst). Dies ist die zweite Haupterscheinung vom Chaos: Die Extremempfindlichkeit im Verhalten auf winzige nderungen im Anfangszustand, welche aus den Nichtlinearitäten in der Dynamik stammt; die erste Haupterscheinung ist das bereits untersuchte Hin-und-Her-Springen ohne wiederholendes, periodisches Muster.

Bestimmen Sie die positive Steigung des anfänglichen Anstiegs bei der logarithmischen Darstellung von $d_c(t)$ als Funktion der Zeit: Diese Steigung entspricht der Skala $\kappa_L > 0$ beim exponentiellen Wachstum $d_c(t) \sim e^{+\kappa_L t}$ und wird Lyapunov-Exponent genannt. Der Grund, wieso sich irgendwann das exponentielle Wachstum in $d_c(t)$ "sättigt" hat mit der Dissipation der darunter liegenden physikalischen Konvektionsströmung zu tun und wird nicht näher behandelt: Es genügt zu erwähnen, dass die räumliche Ausdehnung des "Schmetterling-Attraktors" beschränkt ist (wie in der vorherigen Aufgabe gesehen wurde) und somit Entfernungen zwischen beliebigen Punkten auf diesem Attraktor nicht unendlich groß werden können.

4. Aufgabe. (Für Tipps zu dieser Aufgabe: siehe Anhang 2). Auch wenn nicht sehr genau, kann der fehlende Periodizitäts-Charakter bei Chaos [3] durch eine spektrale Fourier-Darstellung charakterisiert werden. Dafür wird der sogenannte Fast-Fourier-Transformation-Algorithmus verwendet, welcher in Matlab bereits vorhanden ist. Für $r = 28$ berechnen Sie in doppellogarithmischen Darstellung die Fourier-Transformation (im Betrag) der zeitlichen Entwicklung für jede der 3 Komponenten in `xc(1:3,:)` aus der vorherigen 3. Aufgabe (Details dazu entnehmen Sie aus dem Anhang 2). Belegen Sie damit, dass keine charakteristische Frequenz bei den Trajektorien des Lorenz-Systems im chaotischen Betrieb vorhanden ist.

5. Aufgabe. Auch wenn Chaos umgangsprachlich mit Zufall gleichgesetzt wird, soll nun einen recht kuriosen Effekt untersucht werden, welche solch eine Zufälligkeit wiederlegt: Die Synchronisation von Lorenz-Systemen im chaotischen Betrieb (Abschnitt 9.6 des Buchs von Strogatz). Für $r = 28$ wird die erste Komponente des Zustandsvektors (aus 2. Aufgabe) auf ein zweites Lorenz-System (von nun an als Lorenz-Empfänger bezeichnet und mit dem Index $_r$ markiert) übertragen, wobei die Dynamik des Empfängers leicht modifiziert ist: In den zweiten und dritten Bewegungsgleichungen mit den nichtlinearen Termen wird das eigene $x_{r\,1}$ durch das aus dem ersten Lorenz-System empfangene x_1 ersetzt

$$\text{Sender:}\quad \begin{cases} \dfrac{dx_1}{dt} &= \sigma\,(x_2 - x_1) \\ \dfrac{dx_2}{dt} &= r x_1 - x_2 - x_1\,x_3 \\ \dfrac{dx_3}{dt} &= -b x_3 + x_1\,x_2 \end{cases} \quad \xrightarrow{\underline{u}=x_1} \quad \text{Empfänger:}\quad \begin{cases} \dfrac{dx_{r,\,1}}{dt} &= \sigma\,(x_{r\,2} - x_{r\,1}) \\ \dfrac{dx_{r,\,2}}{dt} &= r\underline{u} - x_{r\,2} - \underline{u}\,x_{r\,3} \\ \dfrac{dx_{r,\,3}}{dt} &= -b x_{r\,3} + \underline{u}\,x_{r\,2} \end{cases} \tag{9}$$

Programmieren Sie dafür die entsprechende "anonymous function" $\mathbf{f}_r(\mathbf{x})$ zur Beschreibung der Dynamik für den Empfänger

```
fr = @(x,u,r) ([ sigma*(x(2,:)-x(1,:)); r*u...  ]);
```

(2-te und 3-te Komponente bei ... ergänzen) und definieren Sie einen zufälligen Anfangszustand für den Empfänger, z.B.

```
xr0 = 5*rand(3,1);
```

[3] Eine genauere Begründung wurde bereits 1963 durch Lorenz recht elegant (mittels der heute als Lorenz-Kurve bezeichneten Methode) plausibel gemacht. Dies jedoch wird nicht in diesem Praktikum behandelt.

Anschließend berechnen Sie die zeitliche Evolution des Zustandsvektors $\mathbf{x}_r(t)$ für den Empfänger und zeigen Sie graphisch, dass $\mathbf{x} - \mathbf{x}_r \stackrel{t\to\infty}{\longrightarrow} \mathbf{0}$ gilt, d.h., das Empfänger-System synchronisiert sich mit dem Sender, obwohl der Empfänger keine Information über den Anfangszustand des Senders hat. Sowohl der Sender wie der Empfänger werden bei den gleichen Parameter σ, b und r betrieben. Sie werden sicherlich die kleinen Abweichungen merken, welche aus der Ungenauigkeit in der numerischen Integration stammt: Können Sie identifizieren/erraten, wo die Ursache liegt?

6. Aufgabe. Diese Synchronisation von chaotischen Systemen kann sogar verwendet werden, um versteckte Nachrichten vom Lorenz-Sender zum Lorenz-Empfänger zu übertragen. Dabei wird das Signal $\underline{u}(t)$ vom Sender zum Empfänger, welches letzterer zum Synchronisieren einsetzt, mit einem zusätzlichen schwachen (=versteckten) Signal $y(t)$ überlagert, so dass $y(t)$ nicht im Frequenz-Spektrum sichtbar wird. Als verstecktes Signal wird nun die Superposition von zwei beliebigen sinusförmigen Oszillationen angenommen, mit Frequenzen $f_{1/2}$ (unterhalb von $50\ \mathrm{Hz}$, die maximal aufgelöste Frequenz bei dem gewählten Zeitschritt `h`), beliebigen Phasen $\varphi_{1/2}$ und kleinen Amplituden $A_{1/2}$, wobei die Amplituden so gewählt werden sollen, dass die FFT von $\underline{u}(t)$ diese zwei Frequenzen nicht als Peaks aufweist (typischerweise $A_{1/2} \leq 0.05$)

$$\begin{aligned} \overbrace{y(t)}^{\text{versteckte Nachricht}} &= A_1 \sin\left(2\pi f_1\, t + \varphi_1\right) + A_2 \sin\left(2\pi f_2\, t + \varphi_2\right)\ , \\ \underline{u}(t) &= \underbrace{x_1(t)}_{\text{Lorenz-Sender}} + y(t) \quad \text{übertragen zum Lorenz-Empfänger}\,. \end{aligned}$$

Ihre Aufgabe: Modifizieren Sie die letzte 5. Aufgabe so, dass die Synchronisation des Lorenz-Empfängers nun mit dem modifizierten $\underline{u}(t)$ erfolgt. Die resultierende Komponente $x_{r,1}(t)$ wird von $\underline{u}(t)$ abgezogen

$$y_r(t) = \underline{u}(t) - x_{r,1}(t)\,,$$

und mit dem versteckten Signal verglichen. Sie können für den Vergleich sowohl im Zeitbereich oder im Frequenzbereich (FFT) arbeiten, wie es Ihnen lieber ist. Welche ist die Schlussfolgerung aus diesem Vergleich?

Anhang 1 zur Programmierung mit Matlab. Implementierung von $\mathbf{f}(\mathbf{x})$ für das Lorenz-System (4) als "anonymous function", mit $\mathbf{x}$ als (aktueller) Zustandsvektor und r als variabler Parameter:

```
sigma = 10;    b = 8/3;

 f = @(x,r)    ([ sigma*( x(2,:) - x(1,:) ); r*x(1,:) - x(2,:) - x(1,:).*x(3,:); ... ]);
```

Formulieren Sie die dritte Komponente `...` der Vektorfunktion und achten Sie dabei auf das `.*` bei dem Produkt der verschiedenen Komponenten von $\mathbf{x}$. Diese Funktion berechnet gleichzeitig die Dynamik von mehreren unterschiedlichen Zustandsvektoren, damit mehrere Bahnen gleichzeitig verfolgt werden können.

Anhang 2 zur Programmierung mit Matlab. In Matlab ist die Fast-Fourier-Transformation (FFT) einer Zeitreihe recht einfach mittels der bereits eingebauten Funktion `fft()` zu berechnen. Anbei finden Sie ein Beispiel für eine FFT eines Signals von Dauer ungefähr [4] `t_end` $= 100$ Sekunden bei einer zeitlichen Auflösung von `Dt` $= 10^{-3}$ Sekunden, bestehend aus der Superposition von zwei beliebigen sinusförmigen Oszillationen mit Frequenzen $f_{1/2}$ (unterhalb von 500 Hz, die maximal aufgelöste Frequenz bei dem gewählten `Dt`), beliebigen Phasen $\varphi_{1/2}$ und kleinen Amplituden $A_{1/2}$

$$y(t) = A_1 \sin(2\pi f_1 t + \varphi_1) + A_2 \sin(2\pi f_2 t + \varphi_2) + \eta(t)\,,$$

zusammen mit einem Rauschanteil $\eta(t)$, welcher als gleichmäßig verteilte Zufallszahl zwischen 0 und 0.1 bei jedem Zeitschritt generiert wird

```
%% Zeit-Aufloesung
Dt = 0.001;
t_end = 100;
t_end = Dt*(2^nextpow2(t_end/Dt)-1);
t = 0:Dt:t_end;

%% verrauschtes Signal
f1 = 4.83;
A1 = 0.6;
phi1 = -0.3;
f2 = 38.17;
A2 = -1.7;
phi2 = 1.27;
y = A1*sin(2*pi*f1*t+phi1) + A2*sin(2*pi*f2*t+phi2) + 0.1*rand(1,length(t));

%% maximal aufgeloeste Frequenz und Frequenz-Array
Fs = 1/Dt;
f = Fs/length(t)*(0:length(t)/2);

%% FFT
y_fft = fft(y);
absFFT = abs(y_fft/length(t));
absFFT = absFFT(1:length(t)/2+1);

%% Plots
figure(1)
loglog(f,absFFT,'LineWidth',1)
xlabel('f (Hz)');
ylabel('|FFT(y)|');
title( sprintf('f_1=%gHz, f_2=%gHz',f1,f2) );
```

[4] Die Dauer `t_end` des betrachteten Zeit-Intervalls wird justiert so, dass das Zeit-Array `t = 0:Dt:t_end;` eine Länge gleich einer Potenz von 2 ist, damit die Fast-Fourier-Transformation effizienter berechnet werden kann.