

## PRAKTIKUM NUMERISCHE SIMULATION IN DER TECHNIK

### Dynamische Systeme

**Stichworte:** Stabilität, Chaos

**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

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \quad (1)$$

betrachtet, ausführlich geschrieben in der Form

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, \dots, x_n), \\ \dot{x}_2 &= f_2(x_1, x_2, \dots, x_n), \\ &\vdots \\ \dot{x}_n &= f_n(x_1, x_2, \dots, x_n), \end{aligned} \quad (2)$$

wobei  $x_i = x_i(t)$  zeitabhängige Funktionen sind, welche zur vektorwertigen Funktion  $\mathbf{x} = (x_1, \dots, x_n)$  zusammengefasst werden, ebenso wie die Funktionen  $f_i$  zum Vektorfeld  $\mathbf{f} = (f_1, \dots, f_n)$  zusammengefasst werden. Mit  $\dot{x}_i = \dot{x}_i(t) = x'_i(t)$  ist die Ableitung von  $x_i$  gemeint. Das System (1) beziehungsweise (2) wird ergänzt durch Hinzunahme einer **Anfangsbedingung**  $\mathbf{x}(t_0) = \mathbf{x}_0$ , man erhält dann ein **Anfangswertproblem** (AWP)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{x}_0. \quad (3)$$

Unter geeigneten Bedingungen an die Funktionen  $f_i$  existiert eine eindeutige Lösung  $\mathbf{x} : [t_0, \infty) \rightarrow \mathbb{R}^n$  des Anfangswertproblems (3). Thema des Praktikums ist die Lösung des AWP (3) in konkreten technischen Anwendungen und die Untersuchung der Lösung insbesondere hinsichtlich ihres Langzeitverhaltens für  $t \rightarrow \infty$ . Die nachfolgenden Anwendungen sind zu betrachten.

**Anwendung 1: Van der Pol – DGL.** Der holländische Physiker van der Pol untersuchte Triodenschaltungen wie die in Abbildung 1 angegebene. (Bildautor: Kraaiennest, Quelle: [https://commons.wikimedia.org/wiki/File:Van\\_der\\_pol\\_triode.svg](https://commons.wikimedia.org/wiki/File:Van_der_pol_triode.svg)).

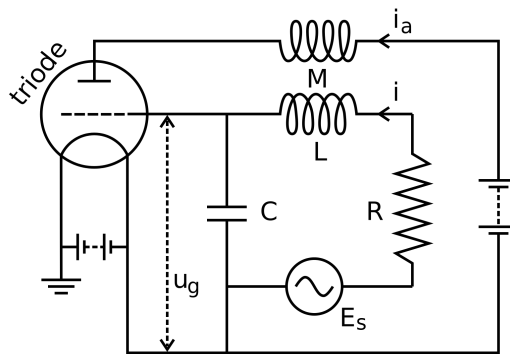


Abbildung 1: Triodenschaltung mit Störterm  $E_S$ .

Für den Fall  $E_S = 0$  (kein Störterm) stelle man die DGL zur Beschreibung der Gitterspannung  $u_g = y$  auf.

$$\ddot{y} = \mu(1 - y^2)\dot{y} - y \quad (4)$$

auf (durch den Parameter  $\mu$  wird die Kennlinie der Triode modelliert), welche durch die Umbenennungen  $x_1 := y$  und  $x_2 := \dot{y}$  in das DGL-System

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= \mu(1 - x_1^2)x_2 - x_1 \end{aligned} \quad (5)$$

übergeht, welches die Form (1) beziehungsweise (2) hat.

**Anwendung 2: Lorenz-Gleichungen.** Der amerikanische Meteorologe Edward Lorenz beschrieb durch aufsteigende Warmluft verursachte Turbulenzen in der Erdatmosphäre mithilfe des folgenden DGL-System

$$\begin{aligned} \dot{x}_1 &= \sigma(x_2 - x_1), \\ \dot{x}_2 &= rx_1 - x_2 - x_1x_3, \\ \dot{x}_3 &= x_1x_2 - bx_3, \end{aligned} \quad (6)$$

wobei  $\sigma > 0$ ,  $r > 0$  und  $b > 0$  vorausgesetzt wird. Auch dieses System hat die Form (1) beziehungsweise (2).

**Anwendung 3: Schwingkreis mit Tunnel diode.** Der in Abbildung 2 skizzierte Schwingkreis enthält eine Tunnel diode (Esaki, 1957), welche durch eine Kennlinie

$$i = g(u) = 2u^3 - u^2 + 0.13u$$

charakterisiert sei (vergleiche: Meyberg, Vachenauer, *Höhere Mathematik 2*, Springer). Für Stromstärke und Spannung im Serienschwingkreis gelten dann die Gleichungen

$$\begin{aligned} C\dot{U} &= I - g(U), \\ L\dot{I} &= E - RI - U \end{aligned} \quad (7)$$

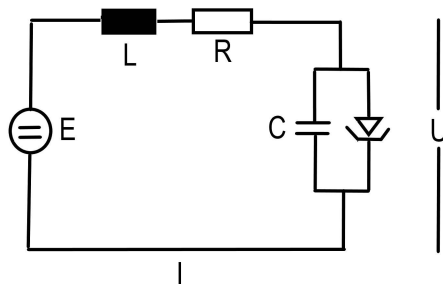


Abbildung 2: RCL-Schwingkreis mit Tunneldiode.

und dieses DGL-System lässt sich ebenfalls leicht in die Form (1) beziehungsweise (2) bringen.

**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.

**1. Aufgabe** (Für Tipps zu dieser Aufgabe: siehe Anhang 1). Informieren Sie sich über die numerische Lösung gewöhnlicher DGL-Systeme in der Programmierumgebung Matlab. Lesen Sie insbesondere die Matlab-Dokumentation zur Funktion ode45. Schreiben Sie eine Testfunktion test1, mit der Sie ode45 am AWP

$$\dot{x} = -10\sqrt{x}, \quad x(0) = 4 \tag{8}$$

testen, dessen exakte Lösung durch die Funktion

$$x(t) = \begin{cases} (2 - 5t)^2, & 0 \leq t \leq 2/5 \\ 0, & t > 2/5 \end{cases}$$

gegeben ist. Im Anhang 1 finden Sie eine ausgearbeitete Matlab-Funktion test0 für ein ähnliches Beispiel. Dort wird auch illustriert, wie der Funktion ode45 durch Übergabe eines *function handle* mitgeteilt wird, wie die „rechte Seite der DGL“, hier also die Funktion  $x \mapsto -10\sqrt{x}$  zu berechnen ist. Berechnen Sie die Lösung numerisch an 101 äquidistanten Stellen zunächst zwischen 0 und 0.35 und zeichnen Sie die Lösung mithilfe der Funktion plot in einem beschrifteten  $t$ - $x$ -Diagramm, vergleiche Abbildung 3. Wie genau ist die berechnete Lösung (Vergleich mit der exakten Lösung) ? Steigern Sie die Genauigkeit der numerischen Berechnung durch

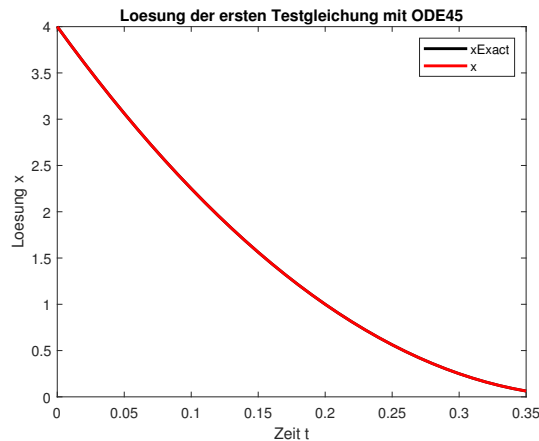


Abbildung 3: Ergebnis von Aufgabe 1.

Verwendung der Optionen `AbsTol` und `RelTol` von `ode45` (vergleiche die Dokumentation zu `odeset`).

**2. Aufgabe** (Für Tipps zu dieser Aufgabe: siehe Anhang 3). Die rechte Seite der DGL (8) ist nur sinnvoll, solange  $x(t) \geq 0$  gilt. Für positive Werte  $x(t) > 0$  ist  $\dot{x}(t) = -10\sqrt{x(t)} < 0$ , das heißt die Lösung von (8) ist eine streng monoton fallende Funktion, bis sie den Wert 0 annimmt. Dies passiert zum Zeitpunkt  $t = T := 2/5$ . In diesem Zeitpunkt ist  $x(T) = 0$  und  $x(t) \equiv 0$  ist für  $t \geq T$  eine konstante Lösung der DGL – eine sogenannte **Gleichgewichtslage** (GGL), weil sich  $x(t)$  ab da nicht mehr ändert. Schreiben Sie eine Matlab-Funktion `test2`, um die Lösung von (8) an 101 äquidistanten Stellen zwischen 0 und 1 zu berechnen. Beobachten Sie die Schwierigkeit, auf die `ode45` stößt. Zur Vermeidung dieser Schwierigkeit kann das Lösungsverfahren gestoppt werden, sobald der Zustand  $x(t) = 0$  eintritt, denn es ist klar, dass dann eine GGL erreicht ist. Im Allgemeinen ist der entsprechende Zeitpunkt  $t = T$  unbekannt, das heißt das Lösungsverfahren muss zu einem vorab unbekanntem Zeitpunkt gestoppt werden. Informieren Sie sich in der Matlab-Dokumentation über die sogenannte *ODE Event Location* und definieren Sie eine geeignete Funktion (einen *event handler*), mit deren Hilfe das Lösungsverfahren gestoppt wird, sobald der Fall  $x(t) = 0$  eintritt. Geben Sie diesen vom *event handler* berechneten Zeitpunkt geeignet aus (Funktion `fprintf`). Ein Beispiel zum Programmieren mit einem `event handle` finden Sie im Anhang 3.

**3. Aufgabe** (Für Tipps zu dieser Aufgabe: siehe Anhang 2). Mit `ode45` können auch DGL-Systeme gelöst werden, es muss dazu beim Aufruf von `ode45` ein *handle* auf eine Funktion übergeben werden, welche die nunmehr vektorwertige rechte Seite des DGL-Systems auswertet (die Funktion `odefun` ist jetzt vektorwertig). Folgendes AWP wird betrachtet:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 - \rho \cdot x_2 |x_2|, \quad \mathbf{x}_0 = \begin{pmatrix} 0 \\ 0.6 \end{pmatrix}. \end{aligned} \quad (9)$$

Verfassen Sie eine Matlab-Funktion `test3` zur numerischen Lösung von (9). Berechnen Sie die Lösung an äquidistanten Stellen mit Abstand  $h = 0.01$  zwischen  $t = 0$  und  $t = 20$  und für  $\rho = 4$ , achten Sie jedoch darauf, den Wert  $\rho$  beim Auswerten der rechten Seite als Parameter zu behandeln und nicht „hart zu kodieren“ (vergleiche hierzu den Punkt *Parameterizing Functions* in der Matlab-Dokumentation von `ode45`). Zeichnen Sie die berechnete Lösung als ein sogenanntes **Phasendiagramm**: Eine Achse gibt  $x_1$  (erste Komponente der Lösung), die andere gibt  $x_2$  (zweite Komponente der Lösung) an, vergleiche etwa Abbildung 4. Aus einem Phasendiagramm können die Zustände  $(x_1(t), x_2(t))$  abgelesen werden, welche die Lösung von (9) im Lauf der Zeit annimmt – dies ist die sogenannte **Phasenbahn** –, aber im Gegensatz zu einem  $t$ - $x_1$ - und einem  $t$ - $x_2$ -Diagramm ist nicht mehr ersichtlich, zu welchem Zeitpunkt  $x_1$  und  $x_2$  bestimmte Werte annehmen. Zum Test: Für die exakte Lösung gilt für  $\rho = 4$ , für  $c = 0.6^2 - 1/32$ , für  $t_i = i/100$  und  $i = 1, \dots, 100$

$$x_2(t_i)^2 = c \exp(-8 \cdot \operatorname{sgn}(x_2(t_i))x_1(t_i)) - \frac{x_1(t_i)}{4 \cdot \operatorname{sgn}(x_2(t_i))} + \frac{1}{32},$$

wobei  $\operatorname{sgn}(x) := 1$  für  $x \geq 0$  und  $\operatorname{sgn}(x) := -1$  für  $x < 0$ .

**4. Aufgabe.** Für DGL-Systeme (1) ergeben sich GGL aus den Stellen  $\mathbf{x} = \mathbf{a}$ , an denen  $\mathbf{f}(\mathbf{a}) = 0$  gilt, weil dann die konstante (also sich nicht mehr ändernde) Funktion  $\mathbf{x}(t) \equiv \mathbf{a}$  eine Lösung von (1) ist. Das durch  $\mathbf{x} = \mathbf{x}(t)$  modellierte System verharrt für alle Zeit  $t \geq t_0$  in unveränderlichem Zustand, sobald es sich einmal in einer GGL befindet:  $\mathbf{x}(t_0) = \mathbf{a}$ . Interessant ist die Frage, wie sich ein System verhält, wenn es „aus dem Gleichgewicht gebracht“ wird, also in einen Zustand nahe einer GGL  $\mathbf{a}$  gebracht wird. Kehrt das System ins Gleichgewicht zurück? Man interessiert sich also für das Verhalten der Lösung des AWP

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{a}_0 \approx \mathbf{a} \quad (10)$$

für  $t \rightarrow \infty$ . Verifizieren Sie, dass  $\mathbf{a} = (0, 0)$  die einzige GGL des Systems (9) ist. Schreiben Sie eine Funktion `test4`, welche die Lösung des (10) entsprechenden AWP für  $t_0 = 0$  und aussagekräftige Startwerte  $\mathbf{a}_0$  berechnet und in einem Phasendiagramm ausgibt. Berechnen Sie die Lösung vom Startzeitpunkt  $t_0 = 0$  bis  $t = 20$ ,  $t = 100$ ,  $t = 1000$ . Was beobachten Sie? Welche Vermutung ergibt sich für das Verhalten der Lösung  $\mathbf{x}(t)$  im Grenzfall  $t \rightarrow \infty$ ? Informieren Sie sich in diesem Zusammenhang über die Klassifikation von GGL als **stabil**, **asymptotisch stabil** oder **instabil** (zum Beispiel in Kapitel 9, Abschnitt 10.3 des Buchs von Meyberg und Vachenaer). Welcher Art ist die GGL von (9) vermutlich? In manchen Fällen lässt sich eine GGL  $\mathbf{a}$  klassifizieren, wenn man die Eigenwerte der Jacobimatrix

$$A = \mathbf{f}'(\mathbf{a}) \quad (11)$$

(Ableitungsmatrix der Funktion  $\mathbf{f}$ ) an der Stelle  $\mathbf{a}$  untersucht. Eine Klassifikation ist möglich, wenn  $\mathbf{f}$  stetig differenzierbar ist und alle Eigenwerte von  $A$  entweder negativen Realteil haben oder wenn mindestens ein Eigenwert positiven Realteil hat. Lesen

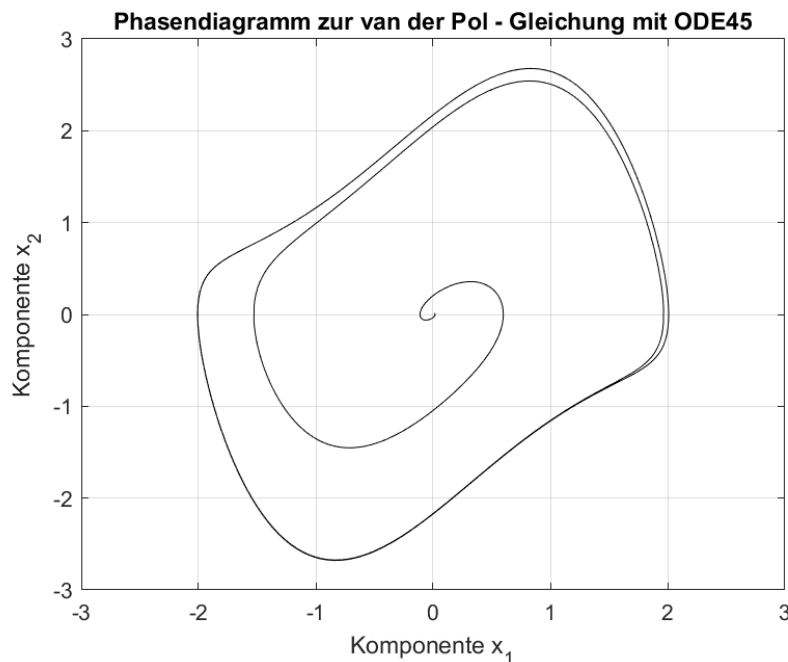


Abbildung 4: Phasenbahn des van der Pol - Oszillators für  $\mu = 1$ , Start in  $(0.01, 0.01)$ .

Sie hierzu die entsprechenden Aussagen in Satz 10.3 („Linearisierungstheorem“) des 9. Kapitels von Meyberg und Vachnauer sowie das anschließende „Rechenschema“. Lässt sich die GGL von (9) mittels dieser Untersuchung klassifizieren?

**5. Aufgabe.** Bestimmen Sie die einzige GGL des Systems (5). Berechnen Sie die Eigenwerte der Matrix  $A$  gemäß (11) in Abhängigkeit von  $\mu$ . Bestätigen Sie, dass die GGL für  $\mu < 0$  asymptotisch stabil, für  $\mu > 0$  instabil ist. Schreiben Sie eine Matlab-Funktion `vanDerPol`, mit der Sie die theoretischen Ergebnisse überprüfen (Wahl verschiedener Parameter  $\mu \neq 0$ , Anfangswerte in der Nähe der GGL wählen, Lösungen in einem längeren Zeitbereich und für verschiedene Rechengenauigkeiten `RelTol` und `AbsTol`, graphische Darstellung einiger typischer Phasenbahnen). In Abbildung 4 ist eine typische Phasenbahn einer Lösung von (5) für Parameter  $\mu = 1$  dargestellt. Experimentieren Sie auch mit dem Parameter  $\mu = 0$ . Welche Vermutung ergibt sich hinsichtlich der Klassifikation der GGL in diesem Fall? Anmerkung: Im Fall  $\mu = 0$  ist das System (5) linear und die Art der GGL kann dann mittels des „Stabilitätssatzes für lineare Systeme“, siehe etwa Satz 10.2 im Kapitel 9 des Buchs von Meyberg und Vachnauer, bestimmt werden.

**6. Aufgabe.** Für das System (5) ergeben sich im Fall  $\mu = 0$  **periodische Phasenbahnen**, das heißt es gilt

$$\mathbf{x}(t + T) = \mathbf{x}(t) \quad \text{für alle } t \geq 0$$

mit einem (kleinsten) Wert  $T > 0$ . Bestimmen Sie  $T$  numerisch. [Bemerkung:

im Fall  $\mu = 0$  ist das System (5) linear und in diesem Fall werden alle möglichen Phasenbahnen in der gleichen Zeit durchlaufem – unabhängig von ihrer Länge.]  
Anleitung: Stellen Sie fest, zu welchem frühesten Zeitpunkt eine Phasenbahn wieder zu ihrem Startwert  $\mathbf{x}(0) = \mathbf{x}_0$  zurückkehrt. Dazu kann man das Abstandskadrat

$$d(t) = \|\mathbf{x}(t) - \mathbf{x}_0\|_2^2 = \langle \mathbf{x}(t) - \mathbf{x}_0 | \mathbf{x}(t) - \mathbf{x}_0 \rangle$$

des Bahnpunkts  $\mathbf{x}(t)$  vom Startwert betrachten. Die Ableitung dieser Funktion lautet

$$\dot{d}(t) = 2\langle \mathbf{x}(t) - \mathbf{x}_0 | \dot{\mathbf{x}}(t) \rangle.$$

Die Bahn erreicht ihren Ausgangspunkt  $\mathbf{x}_0$  für den ersten Wert  $T > 0$ , für den ein Minimum von  $d$  erreicht wird. Dies passiert, wo  $\dot{d}(T) = 0$  und die Werte  $\dot{d}(t)$  links von  $T$  negativ und rechts von  $T$  positiv sind (Matlab nennt das *zero crossing in positive direction*). Mithilfe eines *event handlers* kann dieser Zeitpunkt abgefangen werden. Sie können sich hierbei durch die in Matlab zur Verfügung gestellte Funktion `orbitode` inspirieren lassen.

**7. Aufgabe.** Bestimmen Sie die GGL von (6) (im Fall  $r \leq 1$  eine GGL, im Fall  $r > 1$  drei GGL). Setzen Sie dann die folgenden Parameterwerte an

$$\sigma = 10, \quad r = 28, \quad b = 8/3 \tag{12}$$

und klassifizieren Sie für diesen Fall die GGL durch Berechnung der Eigenwerte der Matrix  $A$  gemäß (11). Anleitung: Bei der Berechnung der Eigenwerte von  $A$  ergeben sich die charakteristischen Polynome

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

für die eine 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. Im ersten Fall können die Eigenwerte analytisch berechnet und die GGL damit klassifiziert werden. Im zweiten Fall kann mithilfe des Routh-Hurwitz-Kriteriums (Satz 10.4 im 9. Kapitel des Buchs von Meyberg und Vachenaer) entschieden werden, ob die Nullstellen des charakteristischen Polynoms alle negativen Realteil haben oder ob ein positiver Realteil auftritt – somit kann auch in diesem Fall die GGL klassifiziert werden. Alternativ können Sie die Eigenwerte der Matrix  $A$  in Matlab numerisch mittels der Funktion `eig` berechnen.

Schreiben Sie eine Matlab-Funktion `lorenzTest` zur Lösung der Lorenz-Gleichungen für die Parameterwerte (12) und zum Startwert  $\mathbf{x}_0 = (20, 5, -5)$ . Beschreiben Sie das Verhalten des Systems qualitativ – warum spricht man hier von einem „deterministischen Chaos“? Abbildung 5 zeigt die zu berechnende Phasenbahn. Erstellen Sie eine gleichartige Graphik mithilfe der Funktion `plot3`.

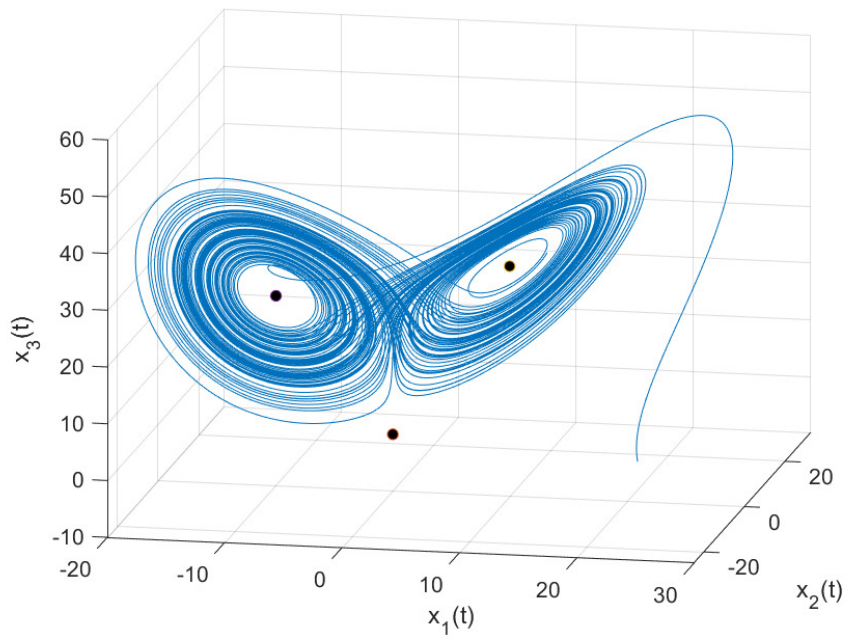


Abbildung 5: Phasenbahn zur Lorenz-Gleichung mit markierten Gleichgewichtslagen.

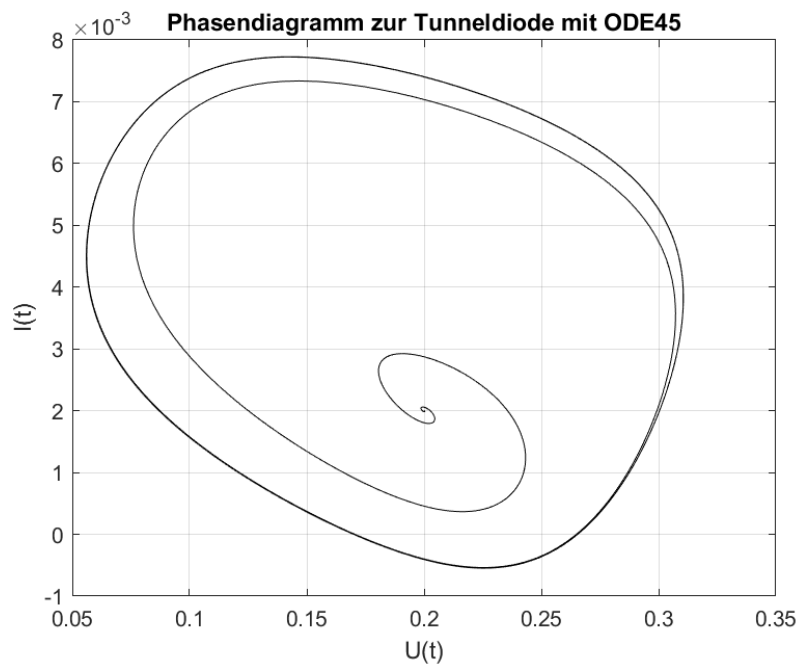


Abbildung 6: Phasenbahn zum Schwingkreis mit Tunnel diode.



**8. Aufgabe.** Formulieren Sie die Gleichungen (7) so um, dass diese die Form des Systems (2) erhalten. Wählen Sie die Werte  $R = 10 \Omega$ ,  $C = 100\mu\text{F}$ ,  $L = 0.1 \text{ H}$ ,  $E = 0.22 \text{ V}$  und die Startwerte  $U(0) = 0.2 \text{ V}$  sowie  $I(0) = 0.002 \text{ A}$ , erstellen Sie numerisch ein Phasendiagramm und zeichnen Sie dieses. Abbildung 6 zeigt das Resultat. Anmerkung: Die Phasenbahn mündet in einen sogenannten **Grenzyklus** ein und wird periodisch, das heißt das System beginnt zu oszillieren. Freiwillige Zusatzaufgabe: Bei Interesse können Sie sich im Buch von Meyberg und Vachenauer (Abschnitt 10.4) über Grenzyklen und den Satz von Poincaré-Bendixson informieren.

**Anhang 1 zur Programmierung mit Matlab.** Eine Matlab-Funktion `test0` (oder ein Matlab-Skript `test0`) muss in einer Datei mit Namen `test0.m` stehen. Nachfolgend der Inhalt eines Files `test0.m`, welches eine Funktion `test0` zur Integration des AWP

$$\dot{x} = \sin(x), \quad x(0) = 1$$

enthält.

```
% Testfunktion test0 zur Lösung des
% AWP
%   x' = sin(x), x(0) = 1

function test0
%   Festlegung äquidistanter Stellen,
%   an denen die Lösung berechnet werden soll
tspan=0:0.01:2*pi;

%   Definition des Anfangswerts
x0 = 1;

% Aufruf der Matlab-Funktion ode45 zur
% Lösung des AWP
% Aufrufparameter:
%   @test0_func: Handle (Zeiger) auf die Funktion test0_func,
%               welche die rechte Seite der DGL auswertet. Die Funktion
%               test0_func ist in die Funktion test0 integriert
%               (nested function), Definition folgt.
%   tspan: Die Punkte, an denen die Lösung zu berechnen ist
%   x0 : Anfangswert an der ersten Berechnungsstelle tspan(1)
% Rückgabewerte:
%   t: Vektor aller Berechnungsstellen. Für tspan könnte auch ein
%      Intervall tspan = [a b] übergeben werden. Dann würde sich
```

```

%         ode45 selbst seine Auswertestellen aussuchen. Im
%         vorliegenden Fall ist t = tspan
%     x: Vektor der berechneten Werte der Lösungsfunktion
[t, x] = ode45(@test0_func,tspan,x0);

% Zeichnen der berechneten Lösung als Funktion der Zeit
figure;
plot(t,x,'k','Linewidth',2);
title('Loesung der Einführungsgleichung mit ODE45');
xlabel('Zeit t');
ylabel('Loesung x');
legend('x');

% ----- Funktion zur Auswertung der rechten Seite
function dxdt = test0_func(t, x)
    dxdt = sin(x);
end
end

```

**Anhang 2 zur Programmierung mit Matlab.** Ein Beispiel einer Funktion test0v, welche ein DGL-System

$$\begin{aligned} \dot{x}_1 &= -(a - bx_2)x_1 \\ \dot{x}_2 &= (c - dx_1)x_2 \end{aligned}, \quad \begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

löst. Der nachfolgende Programmcode muss in einer Datei test0v.m stehen.

```

% Testfunktion test0v zur Lösung des
% vektoriiellen AWP
%     x_1' = -(a-b*x_2)*x_1,   x_1(0) = 1
%     x_2' = (c-d*x_1)*x_2,   x_2(0) = 2
%
% Hierbei sind a,b,c,d positive reelle Parameter

function test0v

%     Festlegung äquidistanter Stellen,
%     an denen die Lösung berechnet werden soll
tspan=0:0.01:5;

%     Definition des Anfangswerts als Spaltenvektor
x0 = [1; 2];

%     Festlegung der Parameter
a = 2; b = 1;
c = 1; d = 2;

```

```

% Festlegung erhöhter Genauigkeitsparameter
% für den Aufruf von ode45
opts = odeset('RelTol',1e-6,'AbsTol',1e-9);

% Aufruf der Matlab-Funktion ode45 zur
% Lösung des AWP
% Änderungen gegenüber Funktion test0:
% Die veränderten Genauigkeitsoptionen werden
% als zusätzliches Argument übergeben
[t, x] = ode45(@test0v_func,tspan,x0,opts);

% Zeichnen der Phasenbahn der Lösung
figure;
plot(x(:,1),x(:,2),'k');
title('Phasendiagramm zum Einführungsbeispiel mit ODE45');
xlabel('Komponente x1');
ylabel('Komponente x2');
grid on;

% ----- Funktion zur Auswertung der rechten Seite
function dxdt = test0v_func(t, x)

    dxdt = [-(a-b*x(2)).*x(1); (c-d*x(1)).*x(2)];
end

end

```

**Anhang 3 zur Programmierung mit Matlab.** Eine Event-Funktion wird benutzt, um die Integration der Lösungsfunktion des ersten Anhangs zu stoppen, sobald ein Wert  $x(t) \geq 3$  erreicht ist.

```

% Testfunktion test0e zur Lösung des
% AWP
%  $x' = \sin(x)$ ,  $x(0) = 1$ 
%
% Die numerische Integration wird für  $t=T$  gestoppt,
% sobald  $x(T) \geq 3$  erreicht wird

function test0e

% Festlegung äquidistanter Stellen,
% an denen die Lösung berechnet werden soll
tspan=0:0.01:2*pi;

% Definition des Anfangswerts
x0 = 1;

```

```

% Vereinbarung erhöhter Genauigkeiten
% sowie eines event handlers für den späteren
% Aufruf von ode45
opts = odeset('RelTol',1e-6,'AbsTol',1e-9,'Events',@EventFunction);

% Aufruf der Matlab-Funktion ode45 zur
% Die Mitteilung des event handlers erfolgt über den Parameter opts
[t, x] = ode45(@test0_func,tspan,x0,opts);

% ----- Funktion zur Auswertung der rechten Seite
function dxdt = test0_func(t, x)

    dxdt = sin(x);
end

% ----- Event Function
function [position,isterminal,direction] = EventFunction(t,x)

    position = x(1)-3; % Der Event, um den es geht:
                       % x(1)-3 wird null
    isterminal = 1;   % Die Aktion, um die es geht:
                       % Stopp der Integration
    direction = 1;    % Der Wert x(1)-3 soll
                       % vom Negativen ins Positive gehen

end
end

```