Ausarbeitung Übung 2

Studienarbeit von Dominik Schiller, Constanze Kramer, Simon Arnold & Tobias Lingenberg
Datum: 25. November 2020

Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Ausarbeitung Übung 2

Studienarbeit von Dominik Schiller, Constanze Kramer, Simon Arnold & Tobias Lingenberg

Datum: 25. November 2020

Darmstadt

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit beschäftigt sich mit dem Übungsblatt 2 des Faches "Einführung in die numerische Berechnung elektromagnetischer Felder". Zunächst wird eine analytische Berechnung des dynamischen Verhaltens eines harmonischen Oszillators durchgeführt, indem die sich aus den Maschengleichungen ergebenden Differentialgleichungen gelöst werden. Anschließend wird das dynamische Verhalten des Oszillators numerisch berechnet und mit den analytischen Ergebnissen verglichen. Zur numerischen Simulation wird die Software "LT-Spice" verwendet. In der letzten Aufgabe wird die in LT-Spice aufgebaute Schaltung in Matrizenform umgewandelt. Diese Umwandlung wird durch eine Routine in der Octave-Software durchgeführt. Die Schaltung wird anschließend durch das BDF-Zeitintegrationsverfahren $dassl$ in Octave gelöst.

2 Hauptteil

2.1 Harmonischer Oszillator, analytisch

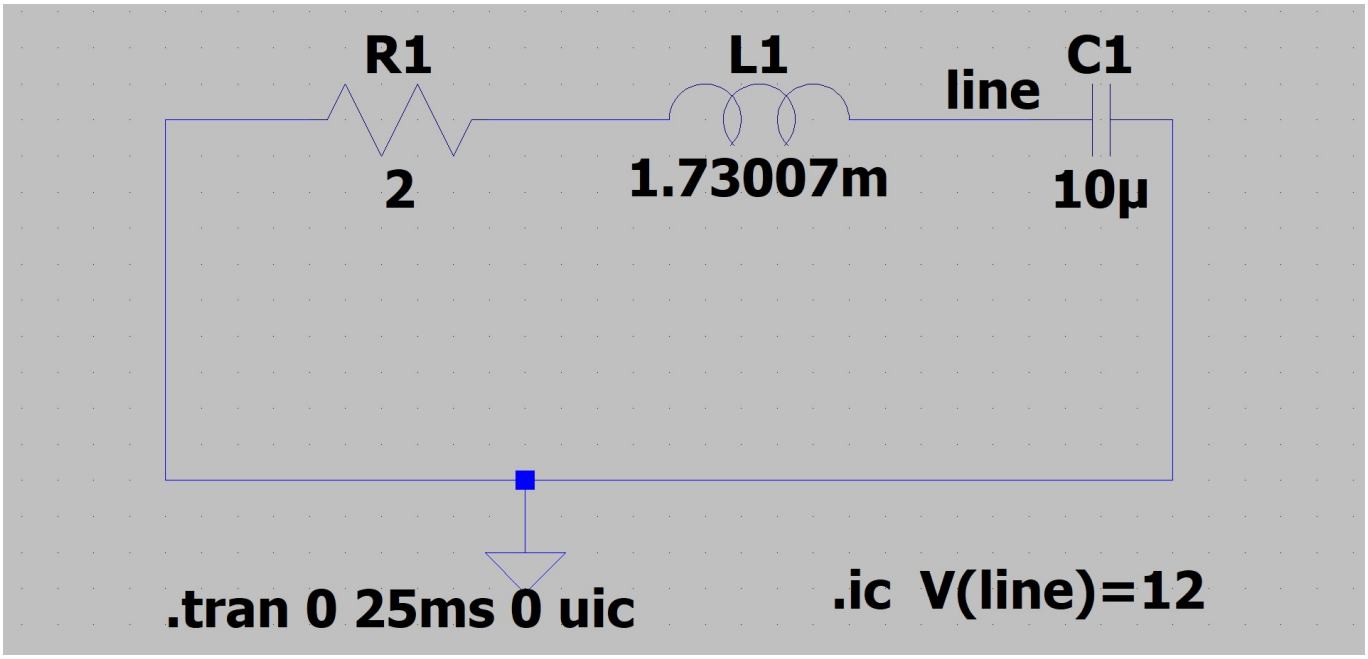


Abbildung 2.1: Gegebener harmonischer Oszillator in LT-Spice implementiert

Für die vorliegende Schaltung wird angenommen, dass am Kondensator zum Zeitpunkt $t = 0$ eine Spannung $u_\mathrm{C}(0) = 12\,\mathrm{V}$ anliegt und der Strom durch den Reihenschwingkreis $i(0) = 0\,\mathrm{A}$ beträgt. Nach dem Kirchhoff'schen Gesetz ergibt sich die Maschengleichung für den Schwingkreis zu $u_\mathrm{C}(t) + u_\mathrm{R}(t) + u_\mathrm{L}(t) = 0$. Aus den Spannungen für den Widerstand $u_\mathrm{R}(t) = R\,i_\mathrm{R}(t)$, die Spule $u_\mathrm{L}(t) = L\,\frac{\mathrm{d}i_\mathrm{L}}{\mathrm{d}t}(t)$ und den Kondensator $u_\mathrm{C}(t) = \frac{1}{C}\int i_\mathrm{C}(t)dt$ ergibt sich daraus die Maschengleichung in Abhängigkeit des Stromes zu

$$R\,i(t) + L\,\frac{\mathrm{d}i}{\mathrm{d}t}(t) + \frac{1}{C}\int i_\mathrm{C}(t)dt = 0.$$

Nach einmaligem Differenzieren nach der Zeit t folgt daraus die Differentialgleichung 2. Ordnung

$$R\frac{\mathrm{d}i}{\mathrm{d}t}(t) + \frac{1}{\mathrm{C}}i(t) + L\,\frac{\mathrm{d}^2i}{\mathrm{d}t^2}(t) = 0. \tag{2.1}$$

Die Lösung der Differentialgleichung im Falle einer gedämpften Schwingung lässt sich aus dem allgemeinen Ansatz

$$i(t) = a\mathrm{e}^{(-\delta+j\omega_e)t} + b\mathrm{e}^{(-\delta-j\omega_e)t} \tag{2.2}$$

ermitteln. Hierbei bezeichnet $\delta = \frac{R}{2L}$ die Dämpfung, $\omega_0 = \sqrt{\frac{1}{LC}}$ die Resonanzkreisfrequenz und $\omega_e = \sqrt{\omega_0^2 - \delta^2}$ die gedämpfte Resonanzkreisfrequenz. Die Konstanten a und b werden bestimmt durch einsetzen des Anfangswertes $i(0) = 0\,\mathrm{A}$ folgt $a + b = 0$ und somit $a = -b$.
Da zum Zeitpunkt $t = 0$ noch kein Strom fließt liegt noch keine Spannung u_R an dem Ohmschen Widerstand R an und somit vereinfacht sich 2.1 zu $u_\mathrm{C}(0) + L\,\frac{\mathrm{d}i}{\mathrm{d}t}(0) = 0$ und durch einsetzen von $u_\mathrm{C}(0) = 12\,\mathrm{V}$ folgt daraus $L\frac{\mathrm{d}i}{\mathrm{d}t}(0) = -12\,\mathrm{V}$. Einmaliges differenzieren von 2.2 sowie einsetzen von $i(0) = 0$ und $a = -b$ ergibt $\frac{\mathrm{d}i}{\mathrm{d}t}(0) = -j2\omega_e b$. Gleichsetzen und nach b auflösen und man erhält $b = -j\frac{u_\mathrm{C}(0)}{2L\omega_e}$ und somit $a = j\frac{u_\mathrm{C}(0)}{2L\omega_e}$. Der Strom ergibt sich damit nun zu

$$i(t) = j\frac{u_\mathrm{C}(0)}{2L\omega_e}\mathrm{e}^{(-\delta+j\omega_e)t} - j\frac{u_\mathrm{C}(0)}{2L\omega_e}\mathrm{e}^{(-\delta-j\omega_e)t}. \tag{2.3}$$

Durch einfaches ausmultiplizieren und nutzen der Euler'schen Formel

$$\sin(x) = \frac{\mathrm{e}^{ix} - \mathrm{e}^{-ix}}{2i}$$

vereinfacht sich 2.3 noch weiter zu

$$i(t) = -\frac{u_\mathrm{C}}{L\omega_e}\mathrm{e}^{-\delta t}\sin(\omega_e t)$$

Mit den eingebauten Bauelementen $L = 1.73007\,\mathrm{mH}$, $R = 2\,\Omega$ und $C = 10\,\mu\mathrm{F}$ kommt man auf eine gedämpfte Resonanzkreisfrequenz von $\omega_e = 7580.701\,\mathrm{s}^{-1}$ und eine Dämpfung von $\delta = 578.011\,\mathrm{s}^{-1}$.

Um eine eindeutige Lösung für eine lineare gewöhnliche Differentialgleichung n-ter Ordnung zu ermitteln sind auch n Anfangsbedingungen nötig da auch n Integrationen nötig sind um die Differentialgleichung zu lösen und somit auch n Integrationskonstanten bestimmt werden müssen.

2.2 Schaltungssimulationen in LTSpice

Der in Aufgabe 2.1 vorgestellte harmonische Oszillator soll nun mit Hilfe des Programms „LT-Spice“ zur Simulation elektrischer Schaltungen nachgebaut und berechnet werden. Da es keine Strom- bzw. Spannungsquelle in der Schaltung gibt, wird am Kondensator eine initiale Spannung zum Zeitpunkt $t = 0$ von $12\mathrm{V}$angelegt. Es handelt sich um eine transiente Simulation, also eine Simulation, bei der zeitabhängige Einflüsse berücksichtigt werden. Der Simulationsbereich bewegt sich zwischen 0 und $25\ \mathrm{ms}$.
Die in LT-Spice erzeugte Schaltung wird nun mit der analytisch berechneten Funktion aus Aufgabenteil 1 verglichen. Hierzu sind an verschieden Zeitpunkten t , $t \in [0, 25]\ \mathrm{ms}$ Stromwerte aus dem Graphen in LT-Spice abgelesen worden, sowie die Werte mit Hilfe der Funktion zu gleichen Zeitpunkt t händisch berechnet worden.

Daraus ergibt sich die folgende Tabelle

Zeitpunkt t in ms	LT-Spice Wert in mA	Berechneter Wert in mA
1	-487,18	-494,26
2	-159,73	-149,67
3	102,45	110,23
6	-28,39	-28,46
9	4,57	3,91
12	-0,356	-0,122
15	-0,048	-0,090

Zusätzlich zu den Messwerten wurde der Graph der Funktion aus Aufgabenteil 2.1 in Matlab geplottet. Vergleich man nun sowohl die Funktionswerte in der Tabelle, als auch die beiden entstandenen Graphen 2.2 und 2.3 , so wird deutlich, dass die Funktion aus 2.1 den gleichen Graphen erzeugt wie die Simulation in LT-Spice. Außerdem ist zu erkennen, dass die Amplitude bei fortlaufender Zeit kleiner wird und der Strom gegen $0\,\mathrm{mA}$ konvergiert. Die Unterschiede in den Funktionswerten der Tabelle lassen sich auf Rundungsfehler, sowie Ablesefehler in LT-Spice zurückführen.

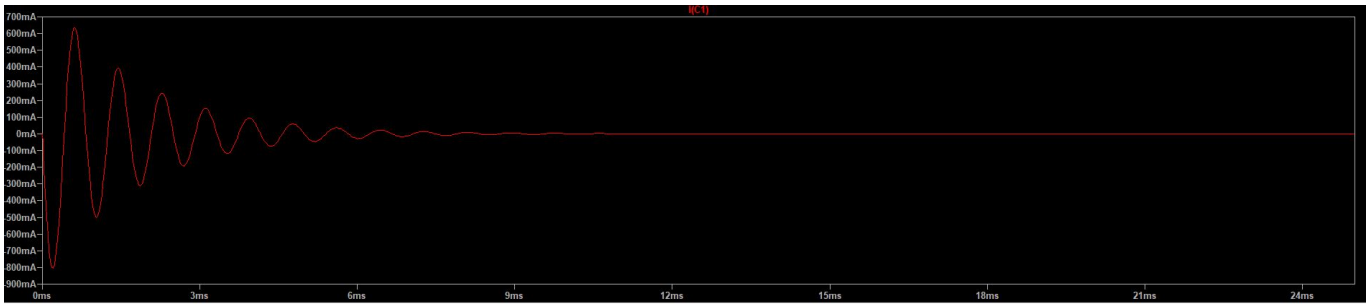


Abbildung 2.2: Simulationsgraph aus LT-Spice

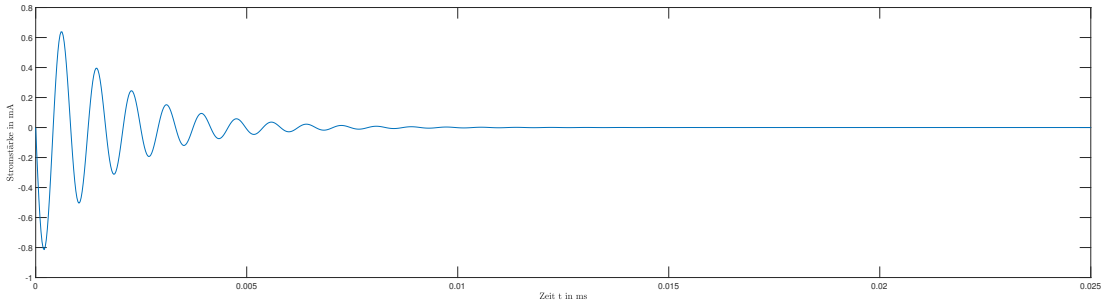


Abbildung 2.3: Simulationsgraph aus Matlab

Des Weiteren soll untersucht werden, inwiefern sich ein kleinerer Widerstand R auf den Funktionsgraphen auswirkt. Hierzu wurde der Widerstand R in dem harmonischen Oszillator auf den Wert $0,5\ \Omega$ gesetzt. Aus der Grafik 2.4 geht hervor, dass die Amplituden durch einen kleineres R beeinträchtigt wird. Wegen der verkleinerten Widerstand nimmt die Auslenkung langsamer ab, gleichzeitig ist der maximal fließende Strom, höher als der Maximalstrom bei einem größeren Widerstand. Die Periodendauer bleibt unbeeinträchtigt.

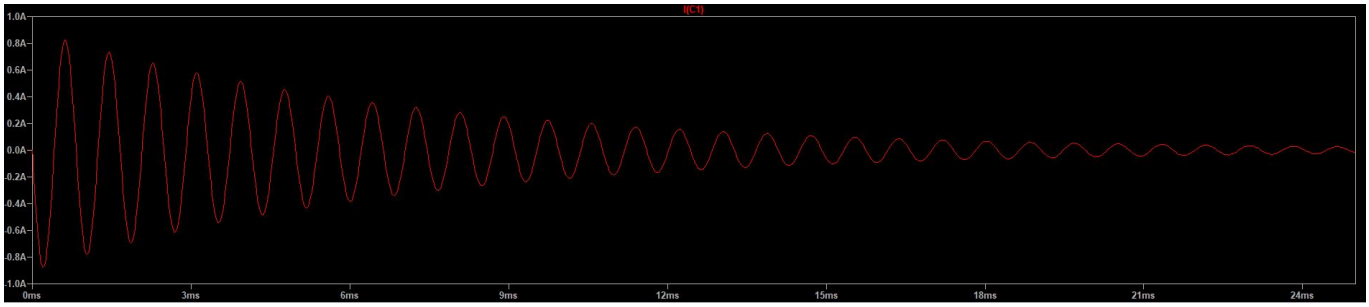


Abbildung 2.4: harmonischer Oszillator mit $R = 0,5\ \Omega$

Neben dem harmonischen Oszillator wird eine weitere Schaltung in LT-Spice betrachtet. Bei dieser Schaltung handelt es sich um einen Spannungsverdoppler. Diese ist auch unter dem Namen Villard-Schaltung bekannt.

Wie der Name besagt, ist das Ziel der Schaltung die Ausgangsspannung, im Vergleich zur Eingangsspannung, zu verdoppeln. Bei der gegebenen Schaltung 2.5 handelt es sich nicht um einen Standard Spannungsverdoppler, der nur aus einer Diode und einem Kondensator besteht, sondern um einen von Greinacher erweiterten Spannungsverdoppler . Bei diesem ist der Villard-Schaltung, die aus dem Kondensator C1 und der Diode D1 besteht, noch ein Kondensator C2 und eine Diode D2 nachgeschaltet. Zusätzlich wurde der Widerstand R1 in Reihe zur Spannungsquelle geschaltet. Voraussetzung für Villard-Schaltung ist eine Quelle, die eine Wechselspannung erzeugt.

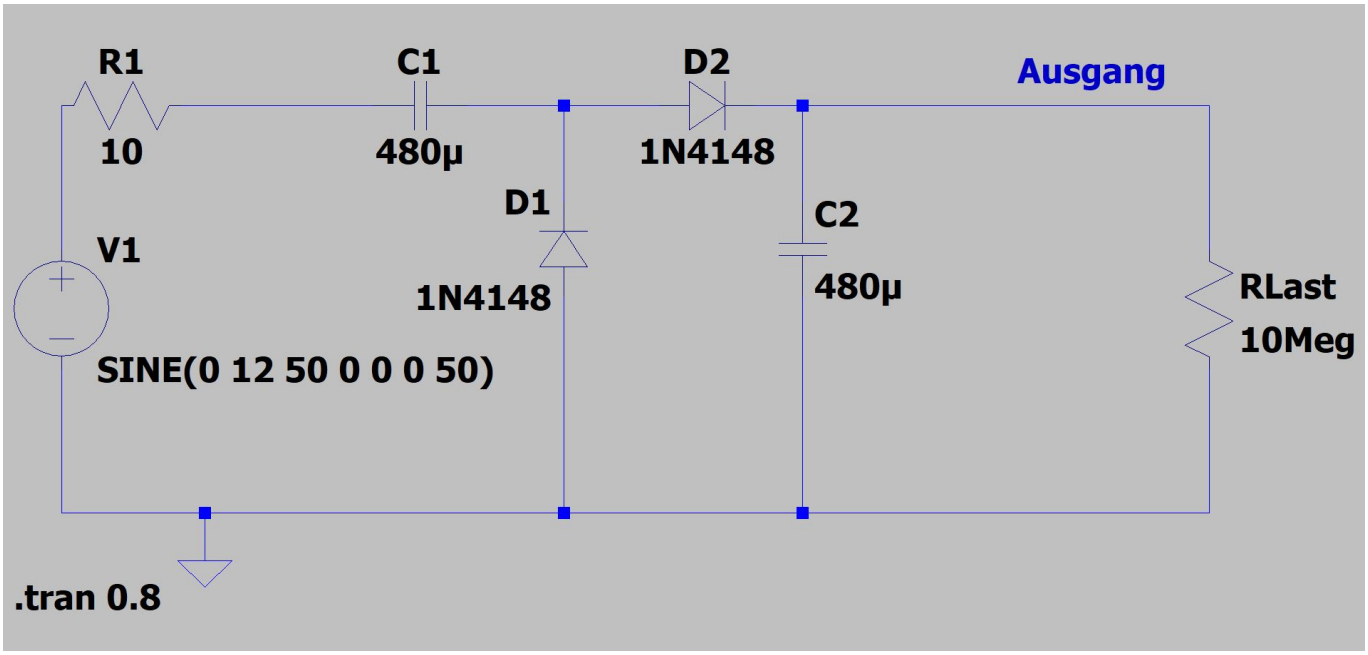
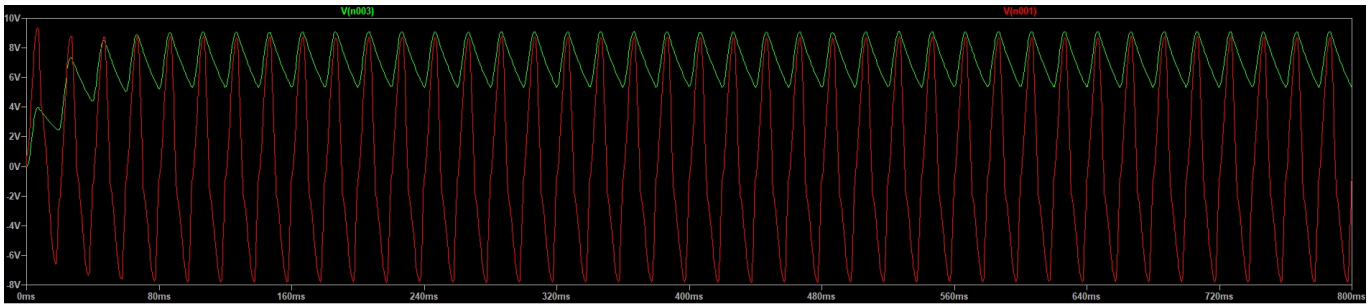


Abbildung 2.5: Spannungsverdoppler

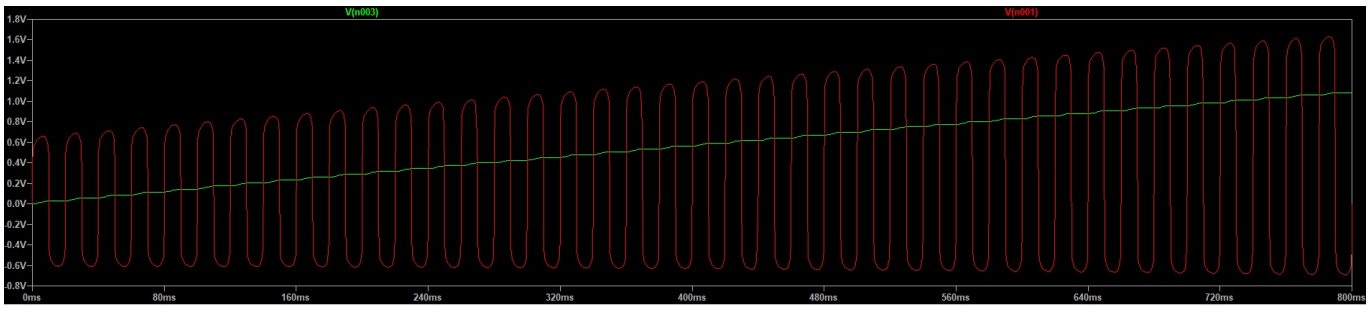
Der Kondensator C1 wird während der ersten positiven Halbperiode auf die Eingangsspannung, also die Spannung u_e, aufgeladen. In der darauf folgenden negativen Halbperiode findet nun ein Ladungsaustausch zwischen der Kondensatoren C1 und C2 statt. Die Diode D1 schließt sich und die Diode D2 öffnet sich. Dadurch fließt ein Strom über C2 zurück zur Spannungsquelle. C1 wird hierbei entladen und lädt den Kondensator C2 auf. Die Spannung an C2 beträgt am Ende der Periode den Wert der Eingangsspannung u_e.
Bei der nächsten positiven Halbperiode wird der Kondensator C1 wieder auf die Eingangspannung geladen, beim Wechsel zur negativen Halbperiode wird C1 auf die halbe Spannung entladen, C2 jedoch auf den 1,5-fachen Wert. Dies führt sich so weiter fort, bis die Spannung an C2 den doppelten Wert der Eingangsspannung erreicht hat.

Der doppelte Wert der Eingangsspannung wird nicht erreicht, da der Spannungsabfall bei einer Silizium Diode ca. $0,7\,\mathrm{V}$ beträgt. In der gegeben Schaltung sind zwei Dioden verbaut, deshalb liegt der Spannungsabfall bei ca. $1,4\,\mathrm{V}$.

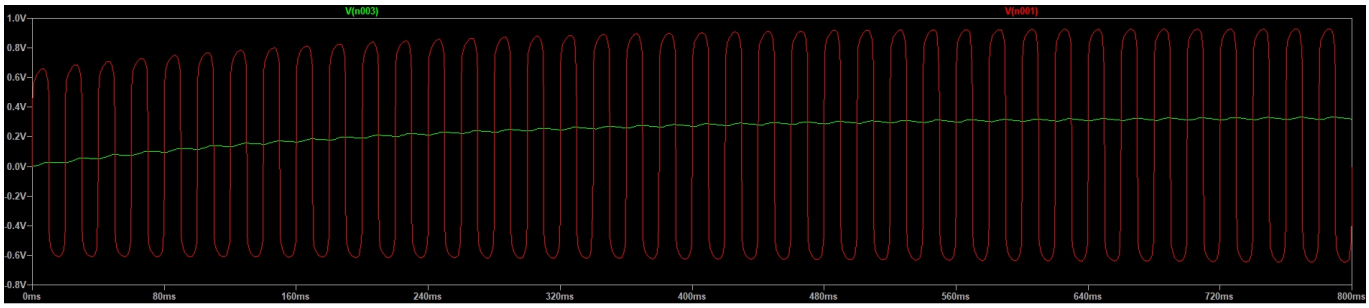
Vergrößert man bei der Schaltung den Ladewiderstand, so beobachtet man in der Abbildung 2.6b, dass die Spannung am Ausgang bzw. am Kondensator C2 langsamer steigt. Dies ist dadurch zu begründen, dass durch den erhöhten Ladewiderstand ein geringerer Stromfluss möglich ist und damit der Ladungsausgleich verlangsamt wird. Durch die Verkleinerung des Lastwiderstands R2 kommt es bei der negativen Halbperiode auch zu einem Spannungsabfall am Kondensator C2, siehe Abbildung 2.6a. Die Auswirkungen eines kleinen Lastwiderstandes mit großem Ladewiderstand sind in Abbildung 2.6c graphisch dargestellt.



(a) Spannungsverlauf bei kleinem Lastwiderstand



(b) Spannungsverlauf bei großem Ladewiderstand



(c) Spannungsverlauf bei kleinem Lastwiderstand und großem Ladewiderstand

Abbildung 2.6: Verschiedene Spannungsverläufe, rot ist Spannung hinter dem Widerstand, grün die Ausgangsspannung

2.3 Schaltungssimulation in Octave

Eine numerische Simulation von linearen Schaltungen kann auch in Octave vor genommen werden. Hierzu wurden bereits die beiden Methoden `prs_spice` (Methode zum Auslesen der Netzliste) und `dassl` (Methode zum nummerischen Lösen von Differenzialgleichungen) bereit gestellt. Die Aufgabe wurde nun in 3 einzelne Skripte unterteilt:

Methode `circuit_matrices`

Methode `circuit_matrices` (siehe 3.1) berechnet alle Matrizen, die für die modifizierte Knotenanalyse benötigt werden. Rückgabewerte sind die Matrizen $\mathbf{A_C}, \mathbf{A_L}, \mathbf{A_R}, \mathbf{A_V}, \mathbf{A_I}, \mathbf{C}, \mathbf{L}, \mathbf{G}$ sowie die Vektoren $\mathbf{v}$ und $\mathbf{i}$. Hierbei stellen die Variablen $\mathbf{A}$ so genannte Inzidenzmatrizen dar, also Matrizen, die Auskunft über die Beziehungen der Knoten und Kanten eines Graphen geben. In diesem Fall wird für jeden Typ Bauteil (Kondensator C, Spule L, Widerstand R, Spannungsquelle V und Stromquelle I) eine Matrix erstellt, deren Einträge folgender Eigenschaft genügen:

$$a_{ij} = \begin{cases} +1 & \text{falls Zweig } j \text{ mit Bauteil von Knoten } i \text{ weg führt} \\ -1 & \text{falls Zweig } j \text{ mit Bauteil zu Knoten } i \text{ hin führt} \\ 0 & \text{sonst} \end{cases} \tag{2.4}$$

$\mathbf{C}, \mathbf{L}$ und $\mathbf{G}$ stellen Diagonalmatrizen dar, die die Zahlenwerte der Bauteile enthalten (Induktivitäten in Henry, Kapazitäten in Farad, Leitfähigkeiten in Siemens). Zuletzt gibt die Methode die Vektoren $\mathbf{v}$ und $\mathbf{i}$ zurück, welche konstante Ströme oder Spannungen der Quellen enthalten.

Methode `calculate_matrices`

Methode `calculate_matrices` (siehe 3.2) setzt die zuvor berechneten Matrizen zu größeren Matrizen zusammen, die die Differenzialgleichung

$$\mathbf{M}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} + \mathbf{K}\mathbf{x} = \mathbf{r} \tag{2.5}$$

beschreiben. Rückgabewerte sind die Matrizen $\mathbf{M}, \mathbf{K}$ und der Spaltenvektor $\mathbf{r}$, die sich nach folgender Rechenvorschrift ergeben:

$$\underbrace{\begin{bmatrix} \mathbf{A_C C A_C^T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{:=\mathbf{M}} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \boldsymbol{\varphi} \\ \mathbf{i_L} \\ \mathbf{i_V} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{A_R G A_R^T} & \mathbf{A_L} & \mathbf{A_V} \\ -\mathbf{A_L^T} & \mathbf{L} & \mathbf{0} \\ -\mathbf{A_V^T} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{:=\mathbf{K}} \begin{bmatrix} \boldsymbol{\varphi} \\ \mathbf{i_L} \\ \mathbf{i_V} \end{bmatrix} = \underbrace{\begin{bmatrix} -\mathbf{A_I i_S} \\ \mathbf{0} \\ -\mathbf{v_S} \end{bmatrix}}_{:=\mathbf{r}} \tag{2.6}$$

Finales Skript

Das finale Skript (siehe 3.3) löst nun die in 2.5 beschriebene Differenzialgleichung auf nummerischen Weg. Dazu wird eine bereits in Octave implementierte Methode namens `dassl` verwendet. Diese bekommt den Term der Form $\mathbf{M}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} + \mathbf{K}\mathbf{x} - \mathbf{r}$ übergeben. Ebenso benötigt die Funktion die Anfangswerte $\mathbf{x_0}$ und $\dot{\mathbf{x}}_\mathbf{0}$ sowie einen Vektor $\mathbf{t}$ mit Zeitpunkten, für die die Lösung berechnet werden sollen.

Durch nummerische Annäherungsverfahren bestimmt die Methode den gesuchten Vektor $\mathbf{x}$ zum Zeitpunkt t. Dieser Lösungsvektor $\mathbf{x}$ enthält in diesem Fall die Werte

$$\mathbf{x} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ i_L \end{bmatrix}, \tag{2.7}$$

also die beiden Potenziale φ_1, φ_2 an Knoten 1 und 2, sowie den Strom durch die Spule i_L (siehe 2.7). Durch einzeichnen der verschiedenen Lösungen mithilfe der Methode `plot`, lässt sich der zeitliche Verlauf des Systems betrachten:

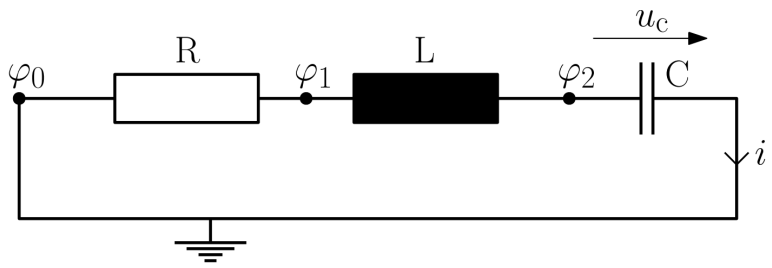


Abbildung 2.7: Schaltplan des harmonischen Oszillators mit $R = 2\Omega$, $L = 1.73007\mathrm{mH}$ und $C = 10\mu\mathrm{F}$

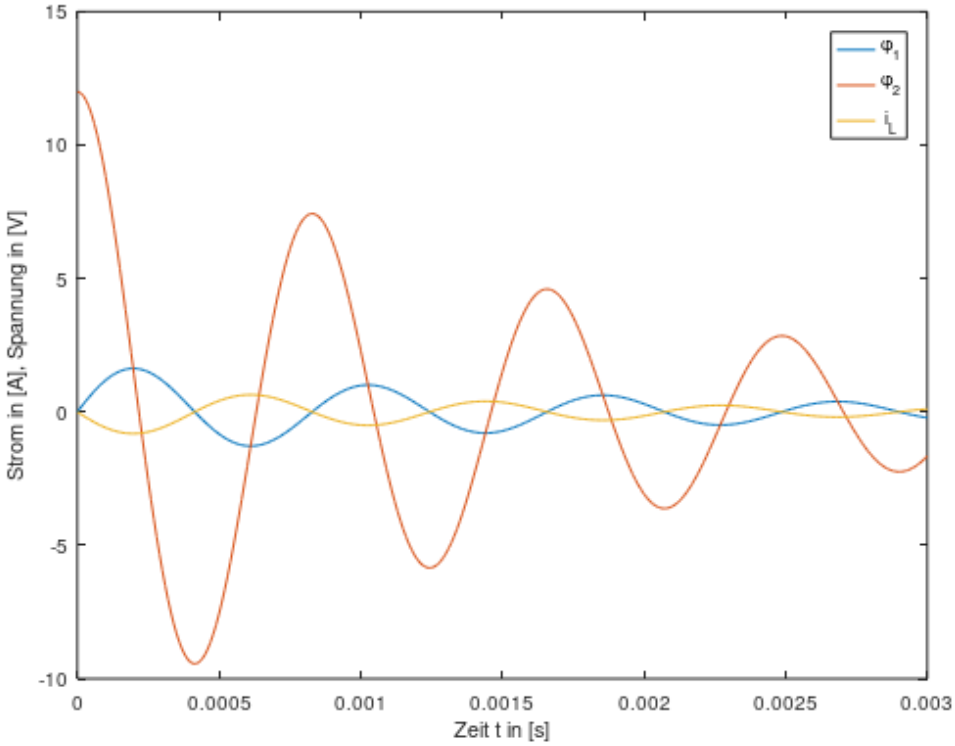


Abbildung 2.8: Schaltungssimulation des harmonischen Oszillators in 2.7. Initialisiert mit Anfangsspannung $u_C = 12V$ am Kondensator zum Zeitpunkt $t = 0$

Hierbei entspricht $\varphi_1 = -u_R$ (der negativen Spannung am Widerstand), $\varphi_2 = u_C$ (der Spannung am Kondensator) und $i_L = i$ (Identisch zum Gesamtstrom, da es nur einen Strom im System gibt).

Hinweis zu den in 3.3 berechneten Anfangswerten $\mathbf{x_0}$ und $\dot{\mathbf{x}}_\mathbf{0}$:

Diese müssten theoretisch per Hand für jede neue Schaltung und Art der Initialisierung des Systems neu berechnet werden. Jedoch scheinen auch inkorrekte Anfangswerte zum selben Ergebnis zu führen, was uns daraus schließen ließ, dass das numerische Annäherungsverfahren der `dassl`-Funktion, auch bei teilweise falschen Anfangswerten, zum richtigen Ergebnis konvergieren kann.

3 Fazit

Das dynamische Verhalten komplexer Schaltungen kann mit Hilfe numerischer Simulationswerkzeuge ausreichend genau berechnet werden. Dies zeigt unter anderem der Vergleich von analytischen mit numerischen Ergebnissen für das Zeitverhalten des harmonischen Oszillators. Auch bei analytisch schwer berechenbaren Schaltungen wie dem Spannungsverdoppler liefern die Simulationswerkzeuge wichtige Informationen über deren zeitliches Verhalten. Unerwartete Effekte können somit früh erkannt werden.

Anhang

Methode circuit_matrices

```
function [AC,AL,AR,AV,AI,C,L,R,V,I] = circuit_matrices(filename)

% PEMCE Uebung 2: Routine um Schaltunsmatrizen zu erzeugen:
%  [AC,AL,AR,AV,AI,C,L,R,V,I] = circuit_matrices(filename)
%
% Eingabe:
%  filename  - Dateiname der Netzliste
%
% Ausgaben:
%  AC,...,AI - reduzierte Inzidenzmatrizen fuer Kapazitaten, ..., Stromquellen
%  C,...,I   - Matrizen der Parameter (Kapazitaten, ..., Stromquellen)

  %# lese Schaltung aus netzliste
   cirstruct = prs_spice (filename);

  %# initialisiere variablen
  n = cirstruct.totextvar;
  AC=AL=AR=AV=AI=sparse(n,0);
  C=L=R = zeros(0,0);
  V=I= zeros(0,1);

  % Durchlaufe alle Elementtypen (R,L,C,...)
  for i=1:size(cirstruct.LCR,2)
    typ=cirstruct.LCR(i).parnames{1,1};
    sz=size(cirstruct.LCR(i).pvmatrix,1);

    % Durchlaufe alle Bauteile eines Typen
    for j=1:size(cirstruct.LCR(i).pvmatrix,1)
      val=cirstruct.LCR(i).pvmatrix(j,:);
      pins=cirstruct.LCR(i).vnmatrix(j,:);

      if typ == 'L'
        matrix = AL;
        diagonal = L;
      endif
      if typ == 'R'
        matrix = AR;
        diagonal = R;
      endif
      if typ == 'C'
        matrix = AC;
        diagonal = C;
      endif
      if typ == 'V'
        matrix = AV;
        diagonal = V;
      endif
      if typ == 'I'
        matrix = AI;
        diagonal = I;
      endif

```

```
53        is_voltage_or_current = typ == 'V' || typ == 'I';
54
55        matrix = [matrix,sparse(n,1)]
56        %Diagonalmatrix Elemente einfuegen
57
58        if is_voltage_or_current == false
59          diagonal = [diagonal,zeros(sz,1)];
60
61          if typ == 'R'
62            val = 1/val;
63          endif
64
65          diagonal(j,j) = val;
66        endif
67
68        %Richtung des Stroms bestimmen
69        direction = pins(1) - pins(2);
70
71        %Strom fliesst von 1 nach 2
72        if direction < 0
73          %Nur hinzufuegen wenn es sich nicht um den Masseknoten handelt
74          if pins(1) != 0
75          matrix(pins(1),j)=1;
76        endif
77        matrix(pins(2),j)=-1;
78
79        if is_voltage_or_current == true
80          diagonal = [diagonal;val];
81        endif
82
83        %Strom fliesst von 2 nach 1
84        else
85          matrix(pins(1),j)=1;
86          %Nur hinzufuegen wenn es sich nicht um den Masseknoten handelt
87          if pins(2) != 0
88            matrix(pins(2),j)=-1;
89          endif
90
91          if is_voltage_or_current == true
92            diagonal = [diagonal;-val];
93          endif
94        endif
95
96        if typ == 'L'
97          AL = matrix;
98          L = diagonal;
99        endif
100       if typ == 'R'
101         AR = matrix;
102         R = diagonal;
103       endif
104       if typ == 'C'
105         AC = matrix;
106         C = diagonal;
107       endif
108       if typ == 'V'
109         AV = matrix;
110         V = diagonal;
```

```
      endif
      if typ == 'I'
        AI = matrix;
        I = diagonal;
      endif

    endfor
  endfor
endfunction
```

Listing 3.1: Methode `circuit_matrices` in Octave

Methode calculate_matrices

```
function [M,K,r] = calculate_matrices(filename)

%Berechnet die Matrizen M,K und r
  [AC,AL,AR,AV,AI,C,L,R,V,I] = circuit_matrices(filename);
  bl = size(L,1);
  bv = size(V,1);
  n = size(AR,1);

  M = [AC*C*AC' zeros(n,bl) zeros(n,bv);
  zeros(bl,n) L zeros(bl,bv);
  zeros(bv,n) zeros(bv,bl) zeros(bv,bv)];

  K = [AR*R*AR' AL AV;
  -AL' zeros(bl,bl) zeros(bl,bv);
  -AV' zeros(bv,bl) zeros(bv,bv)];

  r = [-AI*I; zeros(bl,1); -V];
endfunction
```

Listing 3.2: Methode `calculate_matrices` in Octave

Finales Skript zum Berechnen der Differenzialgleichung

```
t = [0:1e-5:0.003];

% Bestimmung des Gleichungssystems:
[M,K,r] = calculate_matrices('oszilator2.1.net');
res = @(y, yd,t) (M*yd+K*y-r);

% Bestimmung der Anfangswerte:

% x0:
% phi_1 = 0
% phi_2 = 12
% i_L   = 0
U_c = 12;
L = 0.00173007;
R = 2;
x0 = zeros(size(r));
```

```
17 x0(2) = U_c;
18
19 % x0_Strich:
20 % phi_1' = 13872,27
21 % phi_2' = 0
22 % i_L'   = -6936,16
23 x0_Strich = zeros(size(r));
24 x0_Strich(1) = R*U_c/L;
25 x0_Strich(3) = -U_c/L;
26
27 % Nummerische Berechnung des Gleichungssystems:
28 x = dassl(res, transpose(x0), zeros(size(r)), t);
29
30 % Erstellen des Plots:
31 plot(t,x(:,1:3));
32 xlabel('Zeit t in [s]', 'interpreter', 'tex')
33 ylabel('Strom in [A], Spannung in [V]', 'interpreter', 'tex')
34 legend('\phi_1', '\phi_2', ' i_L')
```

Listing 3.3: `FinalesSkript` in Octave

Abbildungsverzeichnis