Karlsruher Institut für Technologie
Institut für Biomedizinische Messtechnik

Prof. Dr. rer. nat. O. Dössel	Dipl. Ing. J. Schmid
Kaiserstr. 12 / Geb. 30.33	Kaiserstr. 12 / Geb. 30.33
Tel.: 0721 / 608-42650	Tel.: 0721 / 608-48035

Lineare Elektrische Netze

Matlab-Aufgabe

Vorname:	Niklas
Nachname:	Fauth
Matrikelnummer:	1932872
RZ-Account:	utede
Punkte:	

Angaben zur Bearbeitung der Aufgaben:

Die Aufgaben müssen selbstständig und ohne fremde Hilfe bearbeitet werden.

Der Lösungsweg muss vollständig angegeben und nachvollziehbar sein! Dokumentieren Sie Ihre Überlegungen, geben Sie erläuternde Kommentare!

Die maximale Punktzahl dieser Aufgabe entspricht 3% der Gesamtpunktzahl der Endnote im Fach Lineare Elektrische Netze.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige fremde Hilfsmittel angefertigt habe. Wörtlich oder inhaltlich übernommene Stellen sind als solche kenntlich gemacht und die verwendeten Literaturquellen im Literaturverzeichnis vollständig angegeben. Die „Regeln zur Sicherung guter wissenschaftlicher Praxis im Karlsruher Institut für Technologie (KIT)“ in ihrer gültigen Form wurden beachtet.

Karlsruhe, den ______________________________

Datum und Unterschrift

Inhaltsverzeichnis

1 Einführung

Nachfolgend die erarbeiteten Lösungen. Zum Teil wurde von einer minimalistischen Lösung abgesehen, um für schöneren Code zu Sorgen oder die Benutzerfreundlichkeit zu erhöhen.

2 UDot

2.1 Erklärung

UDot ist eine eigene Implementierung einer Funktion, um (Spannungs)werte gegenüber der Zeit abzuleiten.

2.2 Quellcode

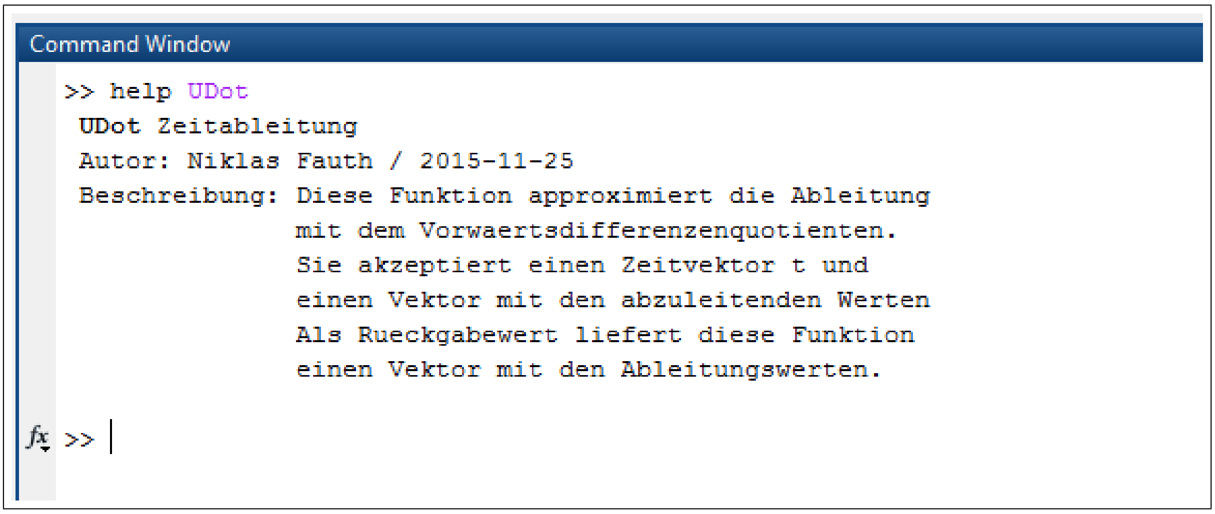
Nachfolgend der Quellcode der Funktion UDot.

```
function UDot = UDot(t, U)

%UDOT Zeitableitung
%Autor: Niklas Fauth / 2015-11-25
%Beschreibung: Diese Funktion approximiert die Ableitung
%             mit dem Vorwaertsdifferenzenquotienten.
%             Sie akzeptiert einen Zeitvektor t und
%             einen Vektor mit den abzuleitenden Werten
%             Als Rueckgabewert liefert diese Funktion
%             einen Vektor mit den Ableitungswerten.

if (length(t) ~= length(U))
    vectorLength = min([length(t) length(U)]);
    if (vectorLength == length(t))
        vectorName = 'time';
    else
        vectorName = 'input';
    end
    warning('The input vectors of UDot do not have the same length. The %s vector will be used.', vectorName);
else
    vectorLength = length(t);
end

% Initialization of the returned vector.
UDot = zeros(1, vectorLength);

for i = 1 : vectorLength  % Calculate the derivation.

    % Check for last value in vector.
    if (i == vectorLength)
        UDot(i-1) = UDot(i);
    else
        % Difference between two time values.
        dt = t(i+1) - t(i);

        % Difference between two input values.
        dU = U(i+1) - U(i);

        % calculate the actual derivation.
        UDot(i) = dU/dt;
    end
end

end
```

Die Funktion erfüllt alle geforderten Bedingungen. Haben die zwei Ausgangsvektoren nicht dieselbe Länge, wird eine entsprechende Warnung ausgegeben. Um diese Ausnahme abzufangen wird im Fehlerfall jedoch nicht einfach abgebrochen, sondern der kürzere der beiden Vektoren zur Berechnung genutzt. Die Angabe, welcher Vektor tatsächlich verwendet wurde, ist in der Warnung enthalten.

2.3 help-Ausgabe

Durch Eingabe des Befehls >help UDot< erscheint folgende Hilfe:



```
Command Window
>> help UDot
  UDot Zeitableitung
  Autor: Niklas Fauth / 2015-11-25
  Beschreibung: Diese Funktion approximiert die Ableitung
                mit dem Vorwaertsdifferenzenquotienten.
                Sie akzeptiert einen Zeitvektor t und
                einen Vektor mit den abzuleitenden Werten
                Als Rueckgabewert liefert diese Funktion
                einen Vektor mit den Ableitungswerten.

fx >>
```

Abbildung 1: help-Ausgabe

3 UInt

3.1 Erklärung

UInt ist eine eigene Implementierung einer Funktion, um (Spannungs)werte gegenüber der Zeit zu integrieren.

3.2 Quellcode

Nachfolgend der Quellcode der Funktion UInt.

```
function UInt = UInt(t, U)
%UINT Zeitintegration
%Autor: Niklas Fauth / 2015-11-25
%Beschreibung: Diese Funktion approximiert die Integration
%              mit der Trapezregel.
%              Sie akzeptiert einen Zeitvektor t und
%              einen Vektor mit den abzuleitenden Werten
%              Als Rueckgabewert liefert diese Funktion
%              einen Vektor mit den Integralwerten.

if (length(t) ~= length(U))
    vectorLength = min([length(t) length(U)]);
    if (vectorLength == length(t))
        vectorName = 'time';
    else
        vectorName = 'input';
    end
    warning('The input vectors of UInt do not have the same length. The %s vector will be used.', vectorName);
else
    vectorLength = length(t);
end

% Initialization of the returned vector.
UInt = zeros(1, vectorLength);

for i = 1 : vectorLength % Calculate the integration.

    % Check for last value in vector.
    if(i == vectorLength)
        break;

    elseif(i == 1)
        Usum = 0;

    else
        % Difference between two time values.
        dt = t(i + 1) - t(i);

        % Sum of two input values.
        Usum = ((U(i) + U(i + 1)) / 2 * dt) + Usum;
        UInt(i) = Usum;
    end
end

end
```

Die Funktion erfüllt alle geforderten Bedingungen. Haben die zwei Ausgangsvektoren nicht dieselbe Länge, wird eine entsprechende Warnung ausgegeben. Um diese Ausnahme abzufangen wird im Fehlerfall jedoch nicht einfach abgebrochen, sondern der kürzere der beiden Vektoren zur Berechnung genutzt. Die Angabe, welcher Vektor tatsächlich verwendet wurde, ist in der Warnung enthalten.

3.3 help-Ausgabe

Durch Eingabe des Befehls >help UInt< erscheint folgende Hilfe:

```
Command Window
>> help UInt
 UInt Zeitintegration
 Autor: Niklas Fauth / 2015-11-25
 Beschreibung: Diese Funktion approximiert die Integration
               mit der Trapezregel.
               Sie akzeptiert einen Zeitvektor t und
               einen Vektor mit den abzuleitenden Werten
               Als Rueckgabewert liefert diese Funktion
               einen Vektor mit den Integralwerten.

fx >>
```

Abbildung 2: help-Ausgabe

4 Anwendung

4.1 Ableiten

Um die Funktionen zu testen soll eine Sinusschwingung mit einer Amplitude von 1 und einer Frequenz von 1Hz abgeleitet werden. Dazu wird die UDot Funktion verwendet.

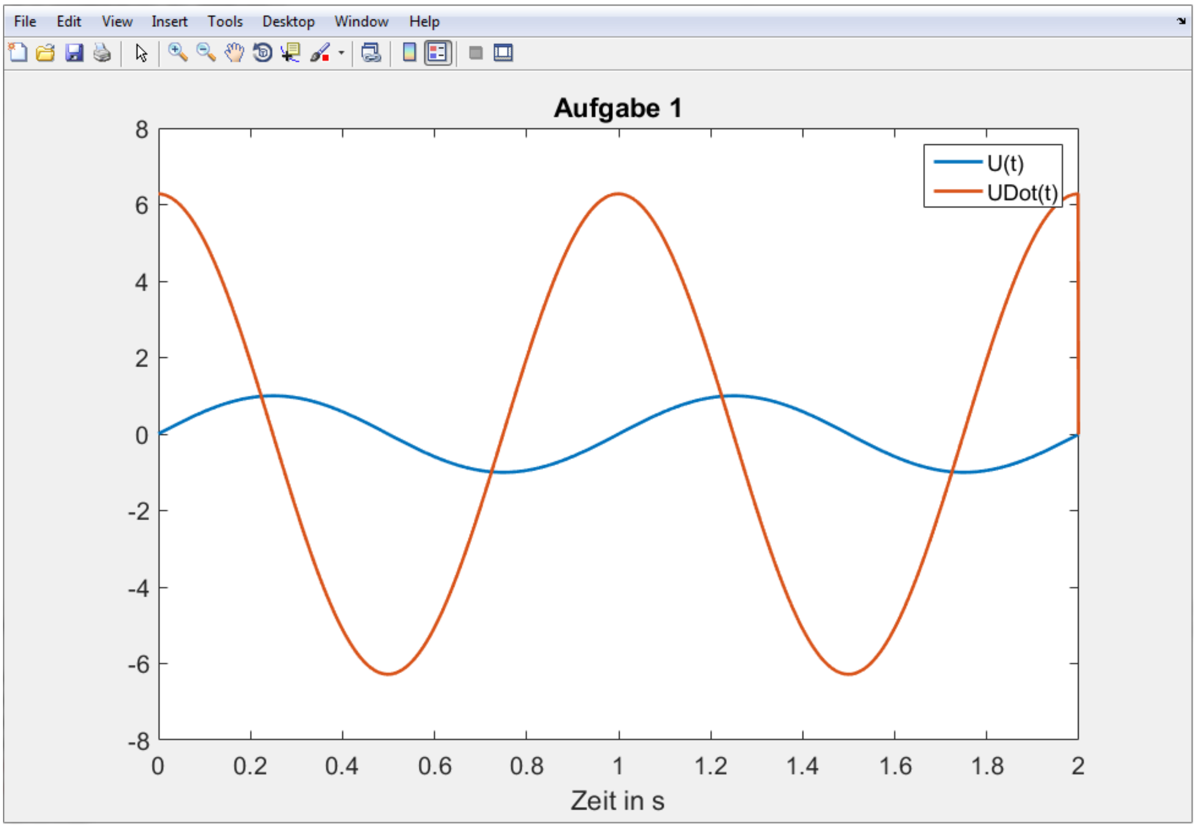


Abbildung 3: Plot der Ausgangsfunktion und ihrer Ableitungsfunktion

Dazu wurden folgende Befehle verwendet:

```
samples = 2001;
t = linspace(0, 2, samples);
U = linspace(0, samples, samples);

for i = 1 : samples
    U(i) = sin((i / samples) * 4 * pi);
end

plot(t, U, t, UDot(t, U), 'LineWidth', 2);
set(gca, 'FontSize', 15);
xlabel('Zeit in s');
title('Aufgabe 1');
legend('U(t)', 'UDot(t)');
```

4.2 Ableiten und Integrieren

Nun soll die Sinusfunktion abgeleitet und anschließend wieder integriert werden.

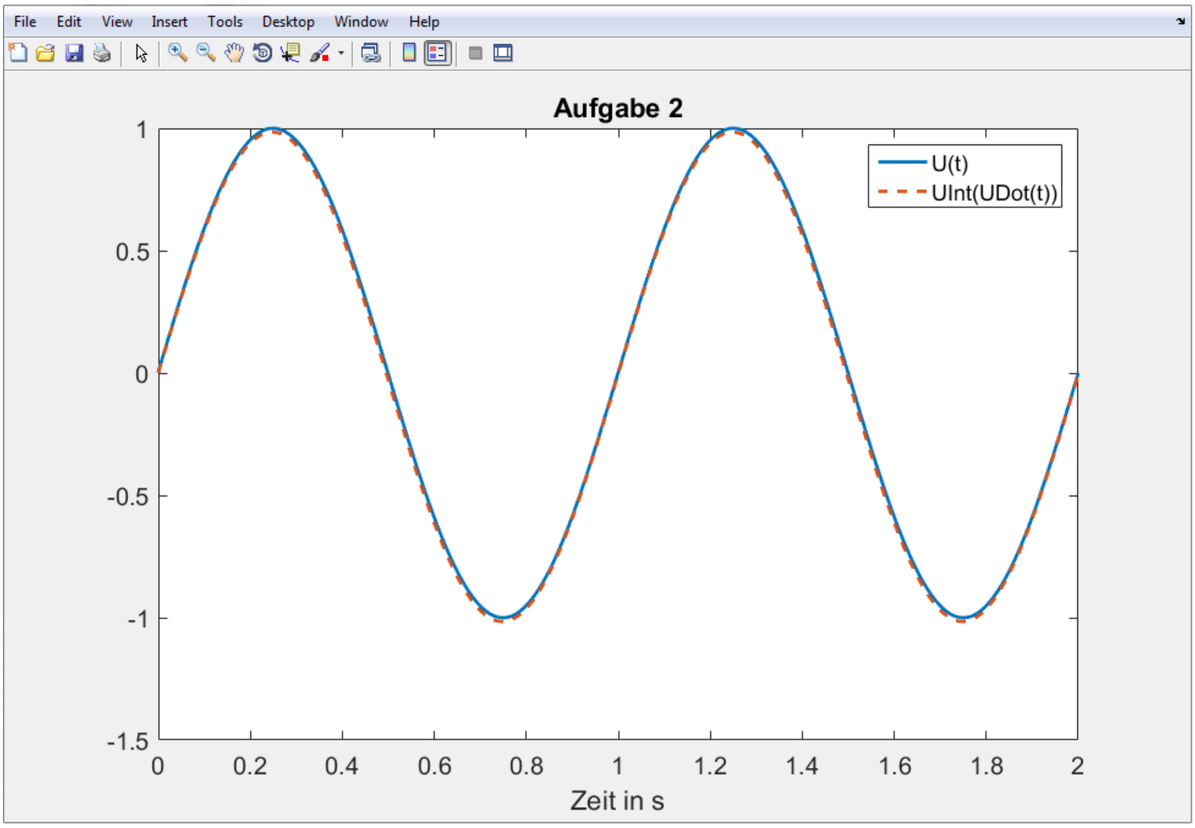


Abbildung 4: Plot

Dazu wurden folgende Befehle verwendet:

```
samples = 2001;
t = linspace(0, 2, samples);
U = linspace(0, samples, samples);

for i = 1 : samples
   U(i) = sin((i / samples) * 4 * pi);
end

plot(t, U, t, UInt(t, UDot(t, U)), '--', 'LineWidth', 2);
set(gca, 'FontSize', 15);
xlabel('Zeit in s');
title('Aufgabe 2');
legend('U(t)', 'UInt(UDot(t))');
```

4.3 Erklärung des Amplitudenunterschiedes

Betrachtet man den Graphen der Ableitung der Sinusfunktion, so fällt auf, dass diese um -90° Phasenverschoben ist, sich aber auch die Amplitude geändert hat. Dies wird ersichtlicher, wenn man die Ausgangsfunktion analytisch ableitet. Die Ausgangsfunktion lautet

$$U(t) = sin(2\pi * t) \tag{1}$$

Die Ableitungsfunktion lautet demnach

$$UDot(t) = \frac{dt}{dU}(sin(2\pi * t)) \tag{2}$$

Gemäß der Kettenregel ergibt sich

$$UDot(t) = \frac{dt}{dU}(2\pi * t) * cos(2\pi * t) \tag{3}$$

Die Ableitung von t ist 1

$$UDot(t) = 2\pi * cos(2\pi * t) \tag{4}$$

Jetzt wird auch ersichtlich, wie es zu dem Amplitudenunterschied kommt. Durch die Ableitung kam der Faktor 2 π dazu. Die Phasenverschiebung lässt sich durch die Kosinusfunktion erklären.

5 Simulation

5.1 Erklärung

Nachdem überprüft wurde, ob sich die beiden Funktionen korrekt verhalten, können diese nun zur Simulation zeitabhängiger Schaltungen, z.B. mit Kondensatoren oder Spulen, verwendet werden. Nachfolgend soll beispielsweise der Stromverlauf eines 500µF Kondensators bzw einer 500mH Spule bei einem Spannungssprung simuliert werden.

5.2 Quellcode

Zum Erzeugen der beiden Plots wurde folgendes Skript verwendet

```
samples=2001;

t = linspace(0, 2, samples);

U = zeros(1, samples);

index = find(t >= 0.5, 1);

U(index:samples) = 1;

C = 500e-6;
L = 300e-3;

subplot(1,2,1);
plot(t, U, 'LineWidth', 2);
set(gca, 'FontSize', 15);
xlabel('Zeit in s');
ylabel('Spannung');
title('Spannung', 'FontSize', 20);

subplot(1,2,2);
plot(t, C * UDot(t, U), 'LineWidth', 2);
set(gca, 'FontSize', 15);
xlabel('Zeit in s');
ylabel('Strom');
title('Kondensator', 'FontSize', 20);

figure;
subplot(1,2,1);
plot(t, U, 'LineWidth', 2);
set(gca, 'FontSize', 15);
xlabel('Zeit in s');
ylabel('Spannung');
title('Spannung', 'FontSize', 20);

subplot(1,2,2);
plot(t, UInt(t, U) / L, 'LineWidth', 2);
set(gca, 'FontSize', 15);
xlabel('Zeit in s');
ylabel('Strom');
title('Spule', 'FontSize', 20);
```

5.3 Kondensator

Um zu simulieren, wie sich ein 500µF Kondensator bei schneller Spannungsänderung verhält, kann die UDot Funktion genutzt werden.

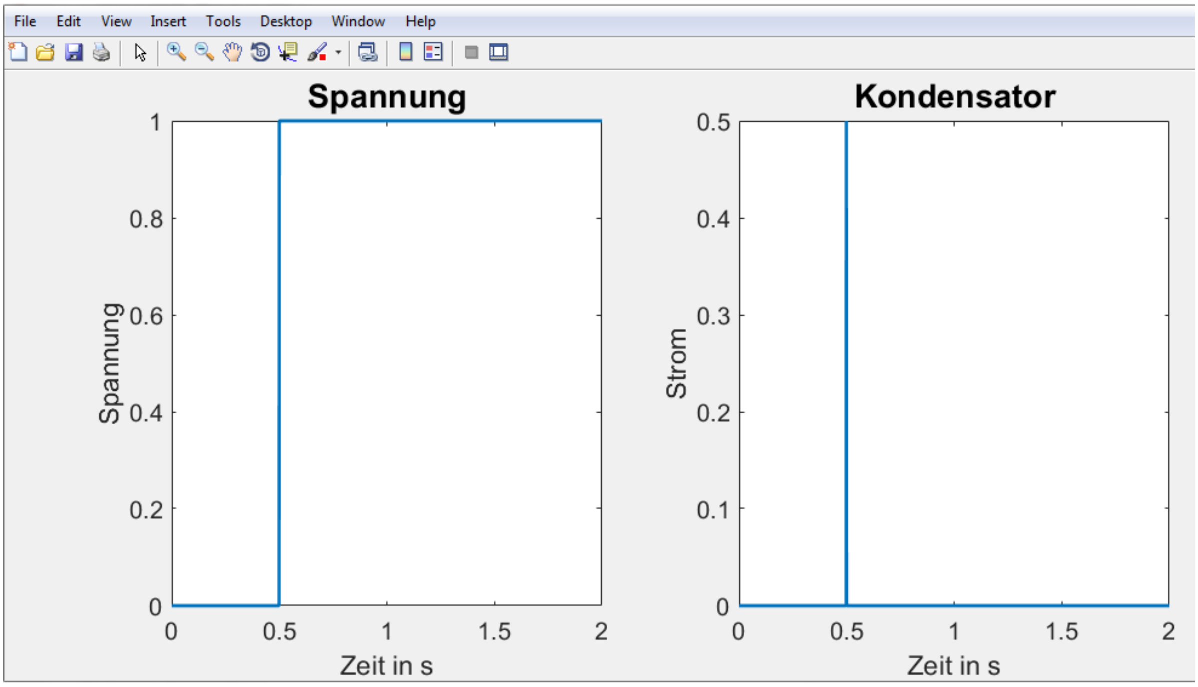


Abbildung 5: Verhalten eines Kondensators bei Spannungsänderung

5.4 Spule

Während beim Kondensator bereits die Gleichung zur Berechnung des Stroms gegeben ist, muss diese für die Spule zunächst umgestellt werden.

$$U(t) = L * \frac{dI(t)}{dt} \tag{5}$$

Dazu muss zunächst die Differentialgleichung dI(t) / dt gelöst werden

$$\frac{dI(t)}{dt} = \frac{U(t)}{L} \tag{6}$$

nun müssen beide Seiten nach t integriert werden

$$I(t) = \int \frac{U(t)}{L} dt = \frac{\int U(t) dt}{L} \tag{7}$$

Zum Integrieren kann die Funktion UInt(t, U) genutzt werde. In Matlab ergibt sich daraus:

$$UInt(t, U)/L \tag{8}$$

Als Plot sieht das für unser Beispiel so aus

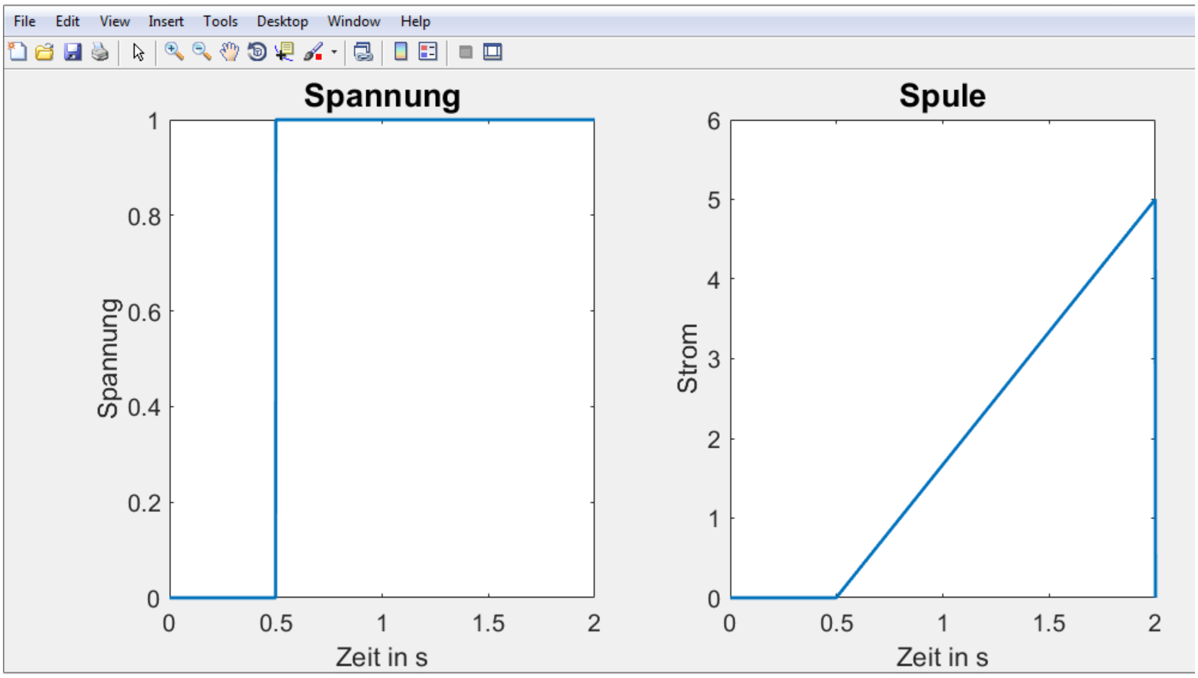


Abbildung 6: Verhalten einer Spule bei Spannungsänderung