

Karlsruher Institut für Technologie <b>Institut für Biomedizinische Messtechnik</b>	
Prof. Dr. rer. nat. O. Dössel Kaiserstr. 12 / Geb. 30.33 Tel.: 0721 / 608-42650	Dipl. Ing. J. Schmid Kaiserstr. 12 / Geb. 30.33 Tel.: 0721 / 608-48035

## Lineare Elektrische Netze

# Matlab-Aufgabe

<b>Vorname:</b>	Niklas
<b>Nachname:</b>	Fauth
<b>Matrikelnummer:</b>	1932872
<b>RZ-Account:</b>	utede
<b>Punkte:</b>	

### 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

<b>1</b>	<b>Einführung</b>	<b>3</b>
<b>2</b>	<b>UDot</b>	<b>3</b>
2.1	Erklärung . . . . .	3
2.2	Quellcode . . . . .	4
2.3	help-Ausgabe . . . . .	5
<b>3</b>	<b>UInt</b>	<b>5</b>
3.1	Erklärung . . . . .	5
3.2	Quellcode . . . . .	6
3.3	help-Ausgabe . . . . .	7
<b>4</b>	<b>Anwendung</b>	<b>8</b>
4.1	Ableiten . . . . .	8
4.2	Ableiten und Integrieren . . . . .	9
4.3	Erklärung des Amplitudenunterschiedes . . . . .	10
<b>5</b>	<b>Simulation</b>	<b>10</b>
5.1	Erklärung . . . . .	10
5.2	Quellcode . . . . .	11
5.3	Kondensator . . . . .	12
5.4	Spule . . . . .	13

Diese Dokumentation sowie dazugehörige Grafiken und Quellcode kann unter  
[https://github.com/NiklasFauth/Private\\_Projects/tree/master/MATLAB](https://github.com/NiklasFauth/Private_Projects/tree/master/MATLAB)  
abgerufen werden.

## 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

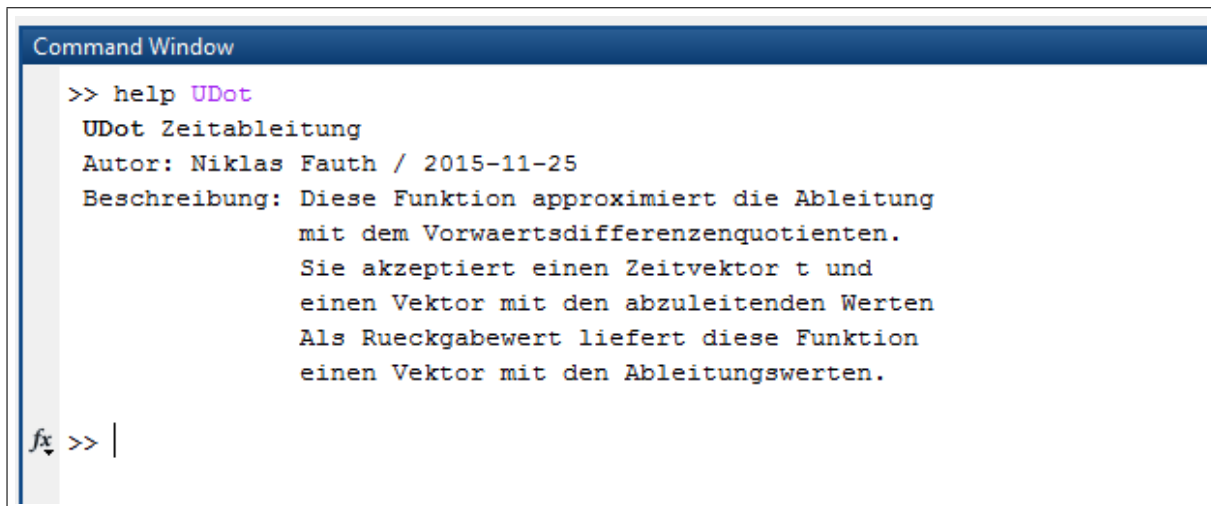
Nachfolgend der Quellcode der Funktion UDot.

```
1 function UDot = UDot(t, U)
2
3 %UDOT Zeitableitung
4 %Autor: Niklas Fauth / 2015-11-25
5 %Beschreibung: Diese Funktion approximiert die Ableitung
6 %               mit dem Vorwaertsdifferenzenquotienten.
7 %               Sie akzeptiert einen Zeitvektor t und
8 %               einen Vektor mit den abzuleitenden Werten
9 %               Als Rueckgabewert liefert diese Funktion
10 %              einen Vektor mit den Ableitungswerten.
11
12 if (length(t) ~= length(U)) %Check if vectors are of the same length
13     vectorLength = min([length(t) length(U)]); %if not, use the shorter
14     one
15     if (vectorLength == length(t))
16         vectorName = 'time';
17     else
18         vectorName = 'input';
19     end
20
21 %display a warning including the name of the used vector
22 warning('The input vectors of UDot do not have the same length. The %s
23 vector will be used.', vectorName);
24 else
25     vectorLength = length(t);
26 end
27
28 % Initialization of the returned vector.
29 UDot = zeros(1, vectorLength);
30
31 for i = 1 : vectorLength % Calculate the derivation.
32
33     % Check for last value in vector.
34     if (i == vectorLength)
35         UDot(i-1) = UDot(i);
36     else
37         % Difference between two time values.
38         dt = t(i+1) - t(i);
39
40         % Difference between two input values.
41         dU = U(i+1) - U(i);
42
43         % calculate the actual derivation.
44         UDot(i) = dU/dt;
45     end
46 end
47 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

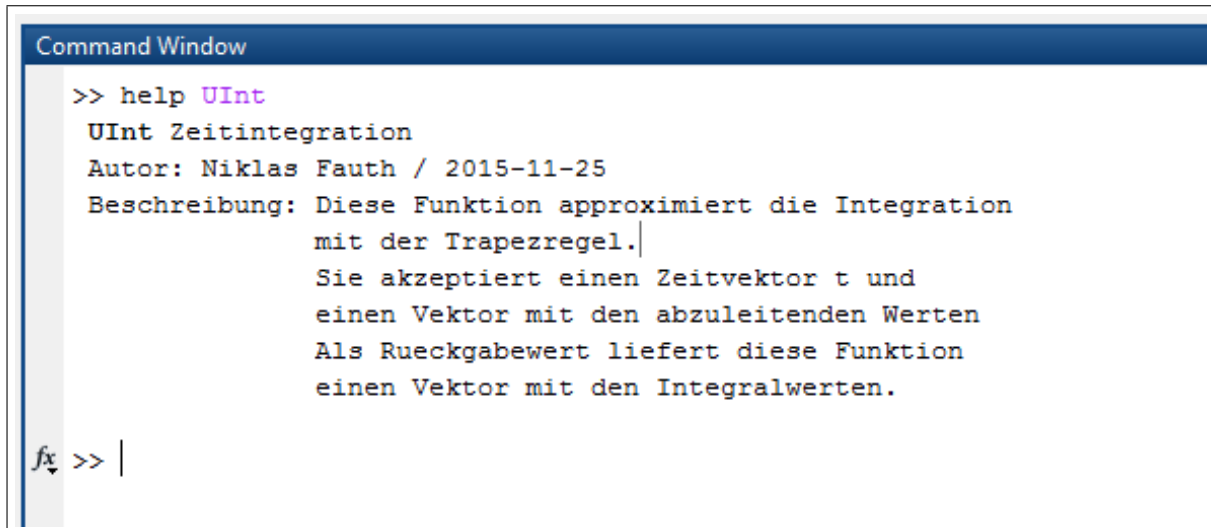
Nachfolgend der Quellcode der Funktion UInt.

```
1 function UInt = UInt(t, U)
2 %UINT Zeitintegration
3 %Autor: Niklas Fauth / 2015-11-25
4 %Beschreibung: Diese Funktion approximiert die Integration
5 %               mit der Trapezregel.
6 %               Sie akzeptiert einen Zeitvektor t und
7 %               einen Vektor mit den abzuleitenden Werten
8 %               Als Rueckgabewert liefert diese Funktion
9 %               einen Vektor mit den Integralwerten.
10
11 if (length(t) ~= length(U)) %Check if vectors are of the same lenght
12     vectorLength = min([length(t) length(U)]); %if not, use the shorter
13     one
14     if (vectorLength == length(t)) used
15         vectorName = 'time';
16     else
17         vectorName = 'input';
18     end
19     %display a warning including the name of the used vector
20     warning('The input vectors of UInt do not have the same length. The %s
21         vector will be used.', vectorName);
22 else
23     vectorLength = length(t);
24 end
25 % Initialization of the returned vector.
26 UInt = zeros(1, vectorLength);
27
28 for i = 1 : vectorLength % Calculate the integration.
29
30     % Check for last value in vector.
31     if(i == vectorLength)
32         break;
33
34     elseif(i == 1)
35         Usum = 0;
36
37     else
38         % Difference between two time values.
39         dt = t(i + 1) - t(i);
40
41         % Sum of two input values.
42         Usum = ((U(i) + U(i + 1)) / 2 * dt) + Usum;
43         UInt(i) = Usum;
44     end
45 end
46 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.

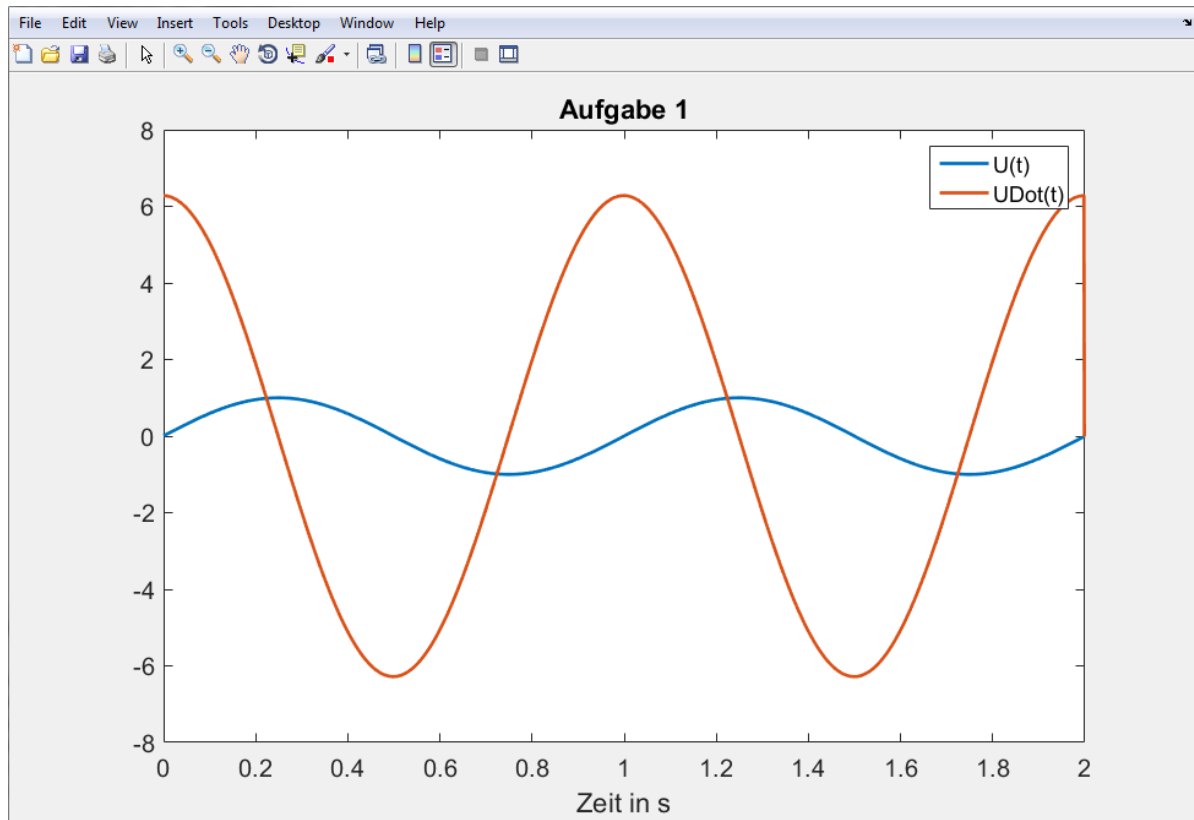


Abbildung 3: Plot der Ausgangsfunktion und ihrer Ableitungsfunktion

Dazu wurden folgende Befehle verwendet:

```
1 samples = 2001;
2 t = linspace(0, 2, samples);           %vector for time
3 U = linspace(0, samples, samples);      %empty vector for voltage
4 %fill the vector with a sine function
5 for i = 1 : samples
6     U(i) = sin((i / samples) * 4 * pi);
7 end
8
9 %plot the original function as well as the derivation of it.
10 plot(t, U, t, UDot(t, U), 'LineWidth', 2);
11 set(gca, 'FontSize', 15); %change fontsize of the axis labeling
12 xlabel('Zeit in s');      %set x-axis label
13 title('Aufgabe 1');       %set title
14 legend('U(t)', 'UDot(t)'); %add legend
```



## 4.2 Ableiten und Integrieren

Nun soll die Sinusfunktion abgeleitet und anschließend wieder integriert werden.

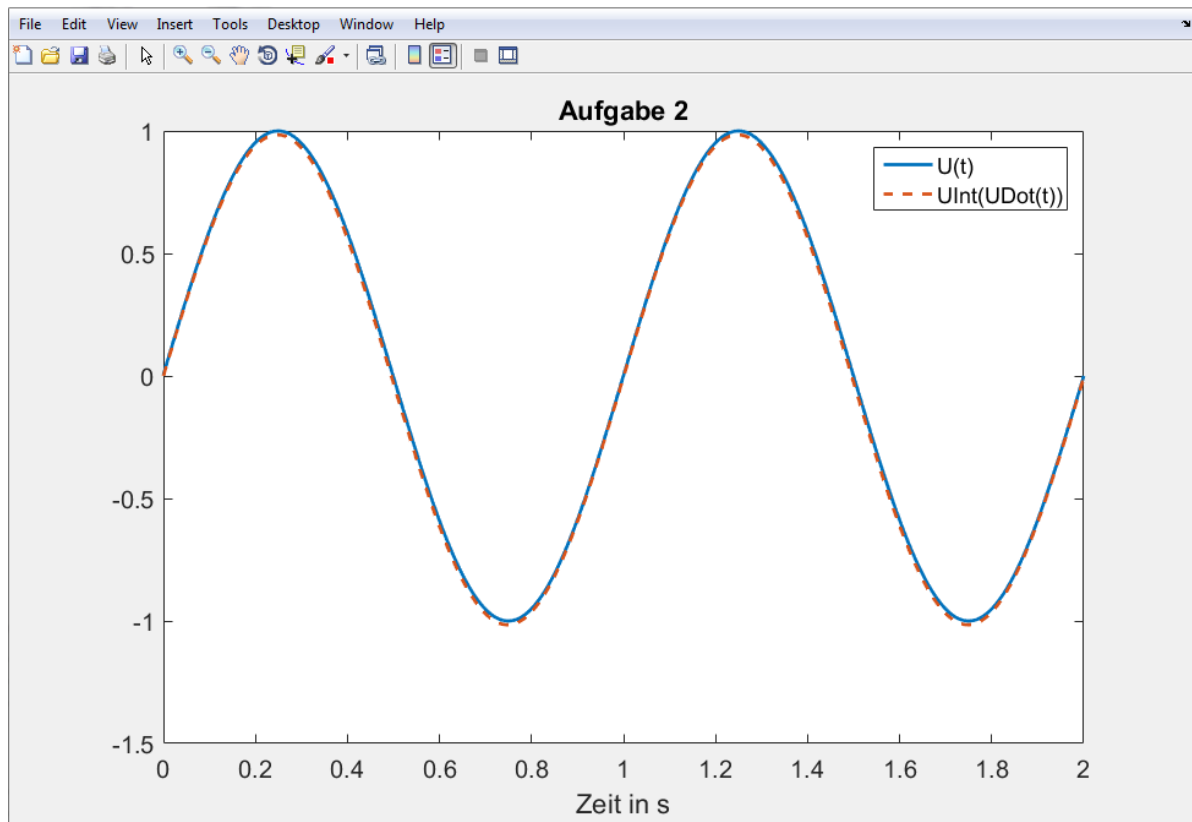


Abbildung 4: Plot

Dazu wurden folgende Befehle verwendet:

```
1 samples = 2001;
2 t = linspace(0, 2, samples);           %vector for time
3 U = linspace(0, samples, samples);     %empty vector for voltage
4
5 %fill the vector with a sine function
6 for i = 1 : samples
7     U(i) = sin((i / samples) * 4 * pi);
8 end
9
10 %plot the original function as well as the integrated derivation of it.
11 plot(t, U, t, UInt(t, UDot(t, U)), '—', 'LineWidth', 2);
12 set(gca, 'FontSize', 15);              %change fontsize of the axis labeling
13 xlabel('Zeit in s');                    %set x-axis label
14 title('Aufgabe 2');                     %set title
15 legend('U(t)', 'UInt(UDot(t))'); %add legend
```

### 4.3 Erklärung des Amplitudenunterschiedes

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

$$U(t) = \sin(2\pi * t) \quad (1)$$

Die Ableitungsfunktion lautet demnach

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

Gemäß der Kettenregel ergibt sich

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

Die Ableitung von  $t$  ist 1

$$UDot(t) = 2\pi * \cos(2\pi * t) \quad (4)$$

Jetzt wird auch ersichtlich, wie es zu dem Amplitudenunterschied kommt. Durch die Ableitung kam der Faktor  $2\pi$  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  $500\mu\text{F}$  Kondensators bzw einer  $500\text{mH}$  Spule bei einem Spannungssprung simuliert werden.

## 5.2 Quellcode

Zum Erzeugen der beiden Plots wurde folgendes Skript verwendet

```
1 samples=2001;
2
3 t = linspace(0, 2, samples);           %vector for time
4 U = zeros(1, samples);                 %empty vector for voltage
5
6 %find the first value in the time vector that is >= 0.5
7 index = find(t >= 0.5, 1);
8
9 %set all values of U to 1 that relate to a time >= 0.5
10 U(index:samples) = 1;
11
12 C = 500e-6; % = 500uF
13 L = 300e-3; % = 500mH
14
15 subplot(1,2,1);                         %subplot, two plots
16 plot(t, U, 'LineWidth', 2);             %plot U(t) with linewidth 2
17 set(gca, 'FontSize', 15);              %fontsize of the axis label
18 xlabel('Zeit in s');                   %set x-axis label
19 ylabel('Spannung');                    %set y-axis label
20 title('Spannung', 'FontSize', 20);      %set title with fontsize 20
21
22 subplot(1,2,2);
23 plot(t, C * UDot(t, U), 'LineWidth', 2); %plot the current of the cap
24 set(gca, 'FontSize', 15);
25 xlabel('Zeit in s');
26 ylabel('Strom');
27 title('Kondensator', 'FontSize', 20);
28
29 figure;                                 %new plot window
30 subplot(1,2,1);                         %subplot, two plots
31 plot(t, U, 'LineWidth', 2);             %plot the voltage
32 set(gca, 'FontSize', 15);
33 xlabel('Zeit in s');
34 ylabel('Spannung');
35 title('Spannung', 'FontSize', 20);
36
37 subplot(1,2,2);
38 plot(t, UInt(t, U) / L, 'LineWidth', 2); %plot I(t) of the inductor
39 set(gca, 'FontSize', 15);
40 xlabel('Zeit in s');
41 ylabel('Strom');
42 title('Spule', 'FontSize', 20);
```

## 5.3 Kondensator

Um zu simulieren, wie sich ein  $500\mu\text{F}$  Kondensator bei schneller Spannungsänderung verhält, kann die UDot Funktion genutzt werden.

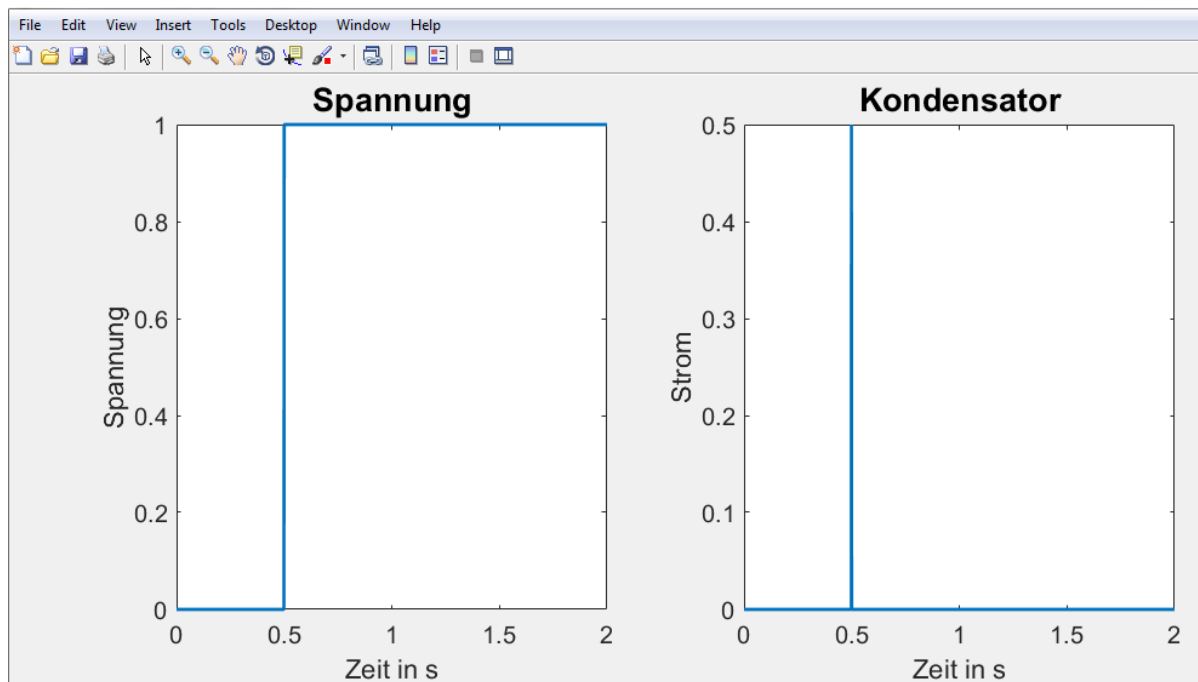


Abbildung 5: Verhalten eines Kondensators bei Spannungsänderung

## 5.4 Spule

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

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

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

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

nun müssen beide Seiten nach  $t$  integriert werden

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

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

$$UInt(t, U)/L \quad (8)$$

Als Plot sieht das für unser Beispiel so aus

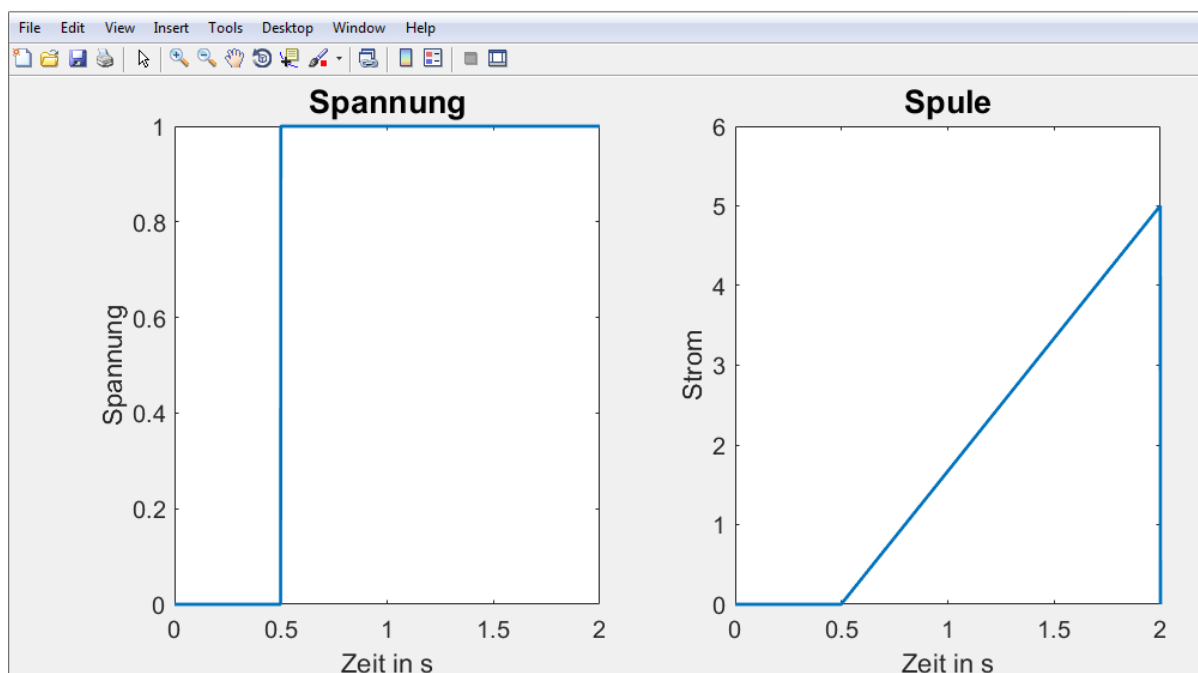


Abbildung 6: Verhalten einer Spule bei Spannungsänderung