

Name: _____

Aufwand (h): _____

Punkte: _____

Aufgabe 1 (8 Pkt): MATLAB

Polynome, eine wesentliche Grundlage bei der Beschreibung des dynamischen Verhaltens von beliebigen Systemen, können als Zeilenvektoren angegeben werden; Polynome von Grad n können allgemein durch Vektoren der Länge $n+1$ dargestellt werden.

Beispiel: Das Polynom

$$p = 5s^5 - 3s^4 + s^2 + 2s$$

lautet in seiner vollständigen Darstellung

$$p = 5s^5 - 3s^4 + 0s^3 + s^2 + 2s + 0s^0$$

und kann durch den Vektor

$$p = [5 \ -3 \ 0 \ 1 \ 2 \ 0]$$

definiert werden.

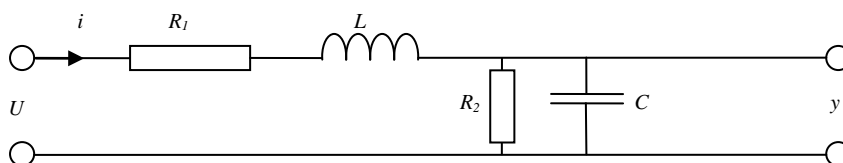
Schreiben Sie eine MATLAB-Funktion, welcher ein Polynom beliebigen Grades als Zeilenvektor übergeben wird plus Angabe des Bereiches, für den dieses Polynom ausgewertet und gezeichnet werden soll; die Schrittweite der Auswertung bzw. graphischen Darstellung soll optional übergeben werden können, wenn nicht definiert soll sie 0.01 betragen.

Achten Sie bei der graphischen Ausgabe auf die Generierung von sinnvollen Diagramm-Überschriften.

(Und rufen Sie für diese Übung nicht einfach eine Polynom-Funktion von MATLAB auf!)

Aufgabe 2 (8 Pkt): Elektrotechnik Basics: Der „Serienresonanzkreis“

Gegeben sei folgende Schaltung:



Zusätzlich wissen wir:

$$u(t) = R_1 * i(t) + L * \frac{\partial i(t)}{\partial t} + y(t)$$

$$i(t) = \frac{1}{R_2} y(t) + C \frac{\partial y(t)}{\partial t}$$

Gesucht ist eine Beschreibung der elektrischen Spannung y als Reaktion auf die angelegte Spannung u .

Bringen Sie dieses System in die (A,B,C)-Normalform und simulieren Sie es in MATLAB, ohne Simulations-Tools (wie etwa SIMULINK) zu verwenden.

Tips:

- Mit Hilfe der Symbolic Math Toolbox für MATLAB dürfte Ihnen dies um einiges leichter fallen
- Modellieren Sie das System aufbauend auf die Zustände $i(t)$ und $y(t)$

Aufgabe 3 (8 Pkt): Theorie der Kontinuierlichen Modellierung

Aufgabe 3a (4 Pkt)

Die Beschreibung von Systemen kann wie in der Vorlesung präsentiert mit Hilfe der sog. (A,B,C) Methode geschehen.

Beschreiben Sie in eigenen Worten in Form eines kurzen Aufsatzes (max. eine Seite), was damit gemeint ist, was die Funktion von A, B und C ist und wie die Dimensionen dieser Matrizen mit der Anzahl an Inputs, Outputs und Zuständen des beschriebenen Systems zusammenhängen.

Nehmen Sie auch zu folgenden Fragen Stellung:

- Wozu sollten wir eine solche mathematische Beschreibung überhaupt brauchen?
- Wozu die Lösung im Zeitbereich, wenn es Simulations-Software gibt?
- Wozu dann Simulations-Software, wenn es die Berechnungsmethode gibt?

Aufgabe 3b (4 Pkt)

Wie hängen die Formeln

$$x'(t) = Ax(t) + Bu(t); x(0) = x_0$$

$$y(t) = Cx(t)$$

und

$$x(t) = e^{tA}x(0) + \int_0^t e^{(t-\tau)A}Bu(\tau)d\tau$$

$$y(t) = Ce^{tA}x(0) + \int_0^t Ce^{(t-\tau)A}Bu(\tau)d\tau$$

mit den in Aufgabe 3a beschriebenen Matrizen zusammen, und wozu braucht man sie beim Modellieren und Simulieren von Systemen?

Hinweise: Geben Sie Ihre Ausarbeitung gedruckt auf Papier ab.
Abgegebene Beispiele müssen in der Übungsstunde präsentiert werden können.

Übung 1

1 Matlab

Dieser Abschnitt beschäftigt sich mit der Aufgabenstellung 1 der ersten Übung. Die folgenden Quelltexte zeigen die *Matlab* Funktionen, die für diesen Teil der Übung implementiert wurden.

In die Funktion *printPolynom* wurden Prüfungen der übergebenen Argumente implementiert wie

- Gültigkeit der Argumentanzahl
- Gültigkeit des Vektors
- Gültigkeit der übergebenen Grenzen und Schrittweite

Wenn die Schrittweite nicht gegeben ist, dann wird der Standardwert 0.01 verwendet.

Listing 1: Funktion zum Plotten eines Polynoms des Grades n

```

1 function printPolynom(poly, tStart, tEnd, tStep)
2 % This function calculates the polynom and plots it
3 % x-axis: tStart:[tStep,0.01]:tEnd
4 % y-axis: polyval(poly, currentStep)
5
6 % ----- BEGIN Prepare -----
7 polySize = size(poly); % get dimensions of matrix
8 if (nargin == 3) % Set default for steps if not given
9     tStep = 0.01;
10 end
11 % ----- END Prepare -----
12
13 % ----- BEGIN Validation -----
14 if(nargin < 3) % Validate parameter count
15     error('At least line vector, tStart and tEnd must be given');
16 elseif(polySize(1) ~= 1) % Validate for line vector
17     error('Only line vectors are allowed');
18 elseif(tStart > tEnd) % Range check
19     error('tStart must not overflow tEnd');
20 elseif((tStep >= 1) || (tStep <= 0)) % Range check
21     error('tStep must be ''0 < tStep < 1''');
22 end
23 % ----- END Validation -----
24
25 % ----- BEGIN Logic -----
26 stepCount = int8((tEnd - tStart) / tStep);
27 x = zeros(1, stepCount); % pre-allocate x
28 y = zeros(1, stepCount); % pre-allocate y
29 idx = 1; % counter variable
30 cols = polySize(2);
31 for t=tStart:tStep:tEnd % loop over range
32     x(idx) = t; % build x-axis
33     yValue = 0;
34     for i=1:1:cols % calculate polynom
35         yValue = yValue + poly(1, i) * t^(cols - i);
36     end
37     y(idx) = yValue; % build y-axis
38     idx = idx + 1; % increase counter
39 end
40
41 figure; % open new window
42 plot(x, y); % plot function over t
43 hold on % hold on
44 xlabel(['t (', num2str(tStep), ')']); % define xlabel

```

Übung 1

```

45 ylabel('y'); % define yLabel
46 legend(strcat('y(t)=', poly2Str(poly))) % define legend
47 % ----- END Logic -----
48 end

```

Listing 2: Funktion zum Umwandeln eines Polynoms des Grades n in eine Zeichenkette

```

1 function value = poly2Str( poly )
2 % This function prints the ploynom represented by the line vector
3 % to its string representation
4
5 polySize = size(poly);
6 cols     = polySize(2);
7 value    = '';
8
9 if(polySize(1) ~= 1) % Validate for line vector
10     error('Only line vectors are allowed');
11 end
12
13 for i=1:1:cols
14     polyVal = poly(1, i);
15     pot     = cols - i;
16
17     if(polyVal ~= 0) % print only value != 0
18         if((i > 1) && (polyVal >= 0)) % add plus sign if positive number
19             value = strcat(value, {' +'});
20         end
21         value = strcat(value, {' ', num2str(polyVal)});
22         if(pot > 1) % add pot only if greather than 1
23             value = strcat(value, {' * x^'}, num2str(pot));
24         elseif(pot == 1)
25             value = strcat(value, {' * x'});
26         end
27     end
28 end
29
30 end

```

Die Funktion aus dem Quelltext 2 zeigt die Funktion, die implementiert wurde, um die Polynomfunktion repräsentiert durch einen Zeilenvektor in seine Repräsentation mit einer Zeichenkette zu konvertieren. Dies war erforderlich, da die Funktion *poly2Sym* nicht zur Verfügung stand.

Übung 1

1.1 Simulation des Polynoms

Folgender Abschnitt beschäftigt sich mit den Simulationen des Polynoms.

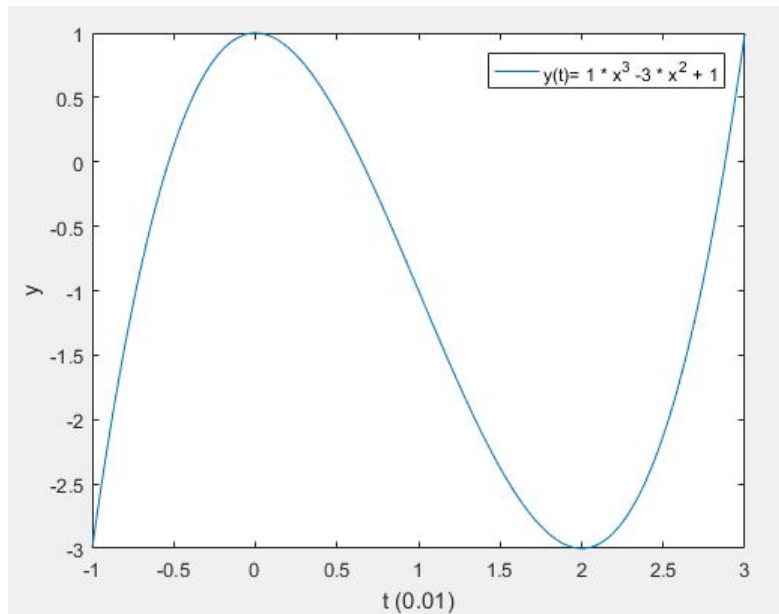


Abbildung 1: Simulation 1 $P=[1 \ -3 \ 0 \ 1]$

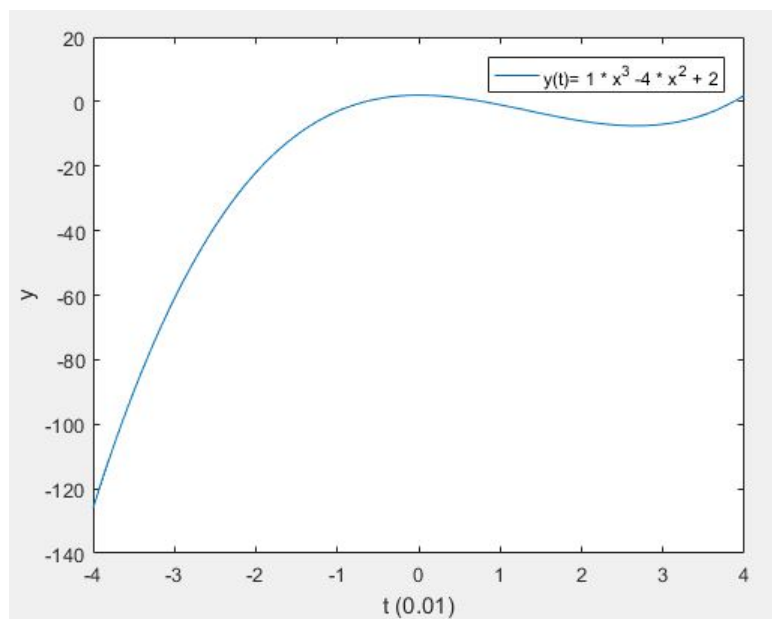


Abbildung 2: Simulation 2 $P=[1 \ -3 \ 0 \ 2]$

Übung 1

2 Der Serienresonanzkreis

Dieser Abschnitt beschäftigt sich mit der Aufgabenstellung 2 der ersten Übung.

Umformen der Gleichung nach $i'(t)$.

$$\begin{aligned} u(t) &= R_1 * i(t) + L * \frac{\delta i(t)}{\delta t} + y(t) \\ u(t) &= R_1 * i(t) + L * i'(t) + y(t) & \frac{\delta i(t)}{\delta t} = i'(t) \\ u(t) - R_1 * i(t) - y(t) &= L * i'(t) & -R_1 * i(t) - y(t) \\ \frac{1}{L} * u(t) - \frac{R_1}{L} * i(t) - \frac{1}{L} * y(t) &= i'(t) & /L \end{aligned}$$

$$i'(t) = \frac{1}{L} * u(t) - \frac{R_1}{L} * i(t) - \frac{1}{L} * y(t)$$

Umformen der Gleichung nach $y'(t)$.

$$\begin{aligned} i(t) &= \frac{1}{R_2} * y(t) + C * \frac{\delta y(t)}{\delta t} \\ i(t) &= \frac{1}{R_2} * y(t) + C * y'(t) & \frac{\delta y(t)}{\delta t} = y'(t) \\ i(t) - \frac{1}{R_2} * y(t) &= C * y'(t) & -\frac{1}{R_2} * y(t) \\ \frac{1}{C} * i(t) - \frac{1}{C * R_2} * y(t) &= y'(t) & /C \end{aligned}$$

$$y'(t) = \frac{1}{C} * i(t) - \frac{1}{C * R_2} * y(t)$$

Substitution von $i(t), y(t)$

$$x_1(t) = i(t)$$

$$x_2(t) = y(t)$$

$$\begin{aligned} x_1'(t) &= -\frac{R_1}{L} * x_1(t) - \frac{1}{L} * x_2(t) + \frac{1}{L} * u(t) \\ x_2'(t) &= \frac{1}{C} * x_1(t) - \frac{1}{C * R_2} * x_2(t) + 0 * u(t) \end{aligned}$$

A, B, C Matrizen aufstellen:

$$A = \begin{bmatrix} \frac{R_1}{L} & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{C * R_2} \end{bmatrix}, B = \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Übung 1

In A, B, C Normalform bringen:

$$X'(t) = \begin{bmatrix} \frac{R_1}{L} & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{C \cdot R_2} \end{bmatrix} * X(t) + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} * U(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} * X(t)$$

Listing 3: Skript zum Simulieren eines Serienresonanzkreises

```

1 % ----- BEGIN Setup -----
2 % Here we initialize the starting test data
3 % ----- BEGIN Setup -----
4 data.R1 = 1;
5 data.R2 = 2;
6 data.C = 1; % 1 / 2 / 3
7 data.L = 2.5; % 2.5 / 1.5
8 data.U = 1;
9 %data.vector.A = [data.R1/data.L -(1/data.L) ;
10 %               1/data.C -(1/(data.C * data.R2))];
11
12 data.vector.A = [(data.R1/data.L) -(1/data.L) ;
13                 (1/data.C) -(1/(data.C * data.R2))];
14 data.vector.B = [(1/data.L); 0];
15 data.vector.C = [0 1];
16 data.vector.X0 = [0 ; 0];
17 data.eval.tStart = 0;
18 data.eval.tEnd = 50;
19 data.eval.tStep = 0.5;
20 % ----- END Setup -----
21
22 % ----- INIT RUN -----
23 % Here we run our simulation runCount times, with the defined movements of
24 % the input variables
25 % ----- INIT RUN -----
26 i = 1; % init i run variable
27 tStart = data.eval.tStart; % local tStart
28 tEnd = data.eval.tEnd; % local tEnd
29 tStep = data.eval.tStep; % local tStep
30 A = data.vector.A; % local A vector
31 B = data.vector.B; % local B vector
32 C = data.vector.C; % local C vector
33 X0 = data.vector.X0; % local X0 vector
34 U = data.U; % local U voltage
35 xAxis = tStart:tStep:tEnd; % allocate x-axis
36 yAxis = zeros(1, int8((tEnd - tStart) / tStep)); % allocate y-axis
37
38 syms tau; % register tau
39 for t=tStart:tStep:tEnd % loop over simulation time frame
40     yAxis(1,i) = C * (double((expm(t * A) * X0) ...
41                     + (int(expm((t - tau) * A) * B * U, tau, 0, t))));
42     i = i + 1; % inc run variable
43 end
44
45 figure; % open new window
46 plot(xAxis, yAxis); % plot function over t
47 hold on % hold on
48 xlabel(['t (', num2str(tStep), ')']); % define xlabel
49 ylabel('y'); % define ylabel
50 legend('Serial resonance cycle');
51 % ----- END RUN -----

```

Übung 1

2.1 Simulationen des Serienresonanzkreises

Folgender Abschnitt behandelt die durchgeführten Simulationen für den Serienresonanzkreis. Die Simulationen wurden mit den folgenden zeitlichen Parametern durchgeführt $tStart = 0$, $tEnd = 50$ und $tStep = 0.5$. Da das Integrieren in Matlab sehr teuer ist, wurde eine nicht so feine Schrittbreite gewählt.

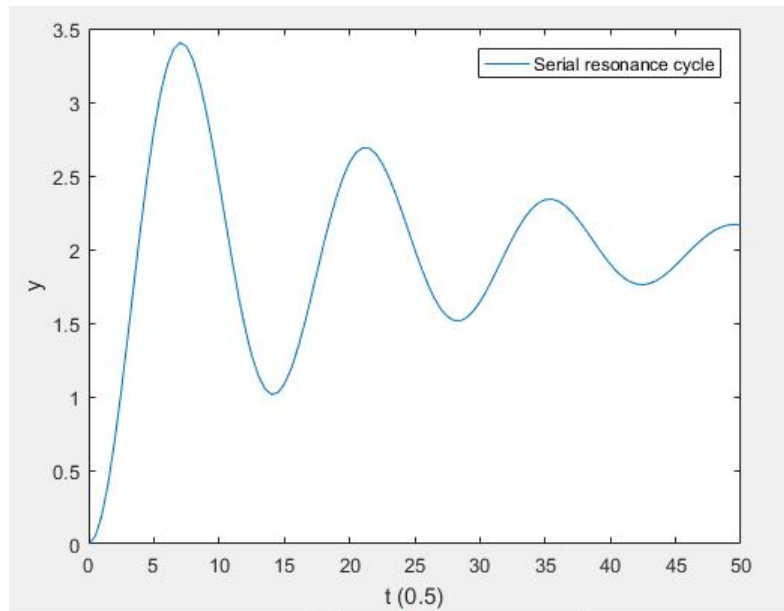


Abbildung 3: Simulation 1 mit $R1=1$, $R2=2$, $C=1$, $L=2.5$, $U=1$

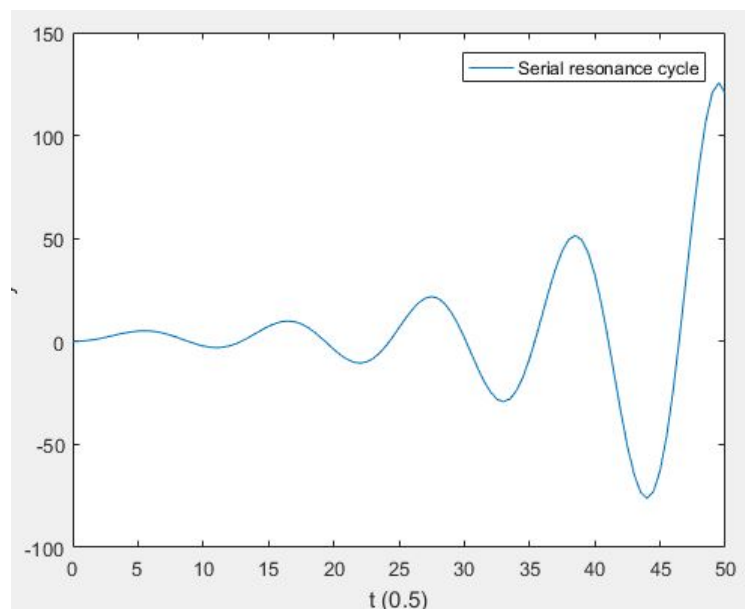
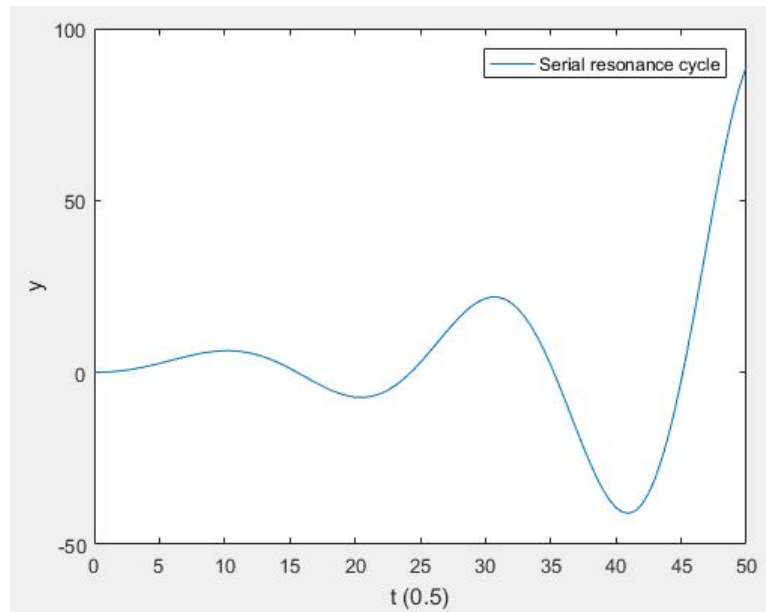
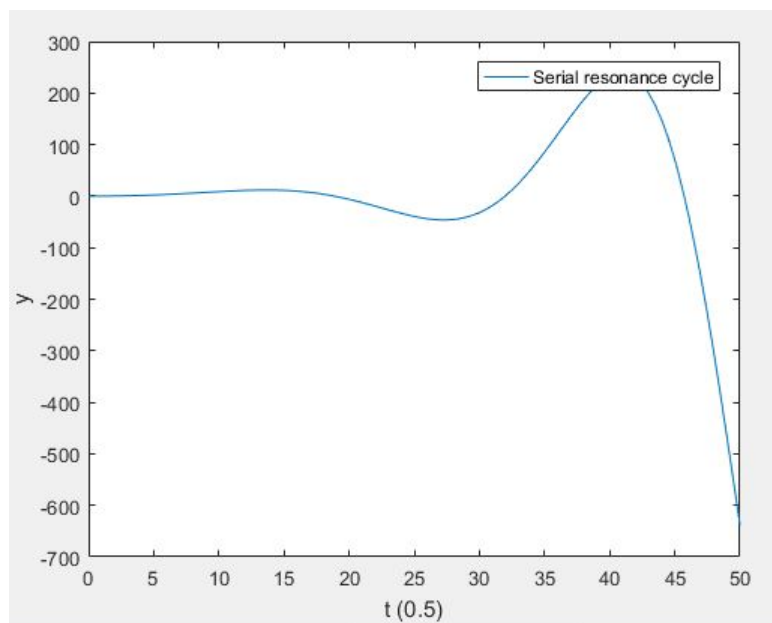


Abbildung 4: Simulation 1 mit $R1=1$, $R2=2$, $C=1$, $L=1.5$, $U=1$

Übung 1

Abbildung 5: Simulation 1 mit $R_1=1$, $R_2=2$, $C=2$, $L=2.5$, $U=1$ Abbildung 6: Simulation 1 mit $R_1=1$, $R_2=2$, $C=3$, $L=2.5$, $U=1$

3 Theorie der Kontinuierlichen Modellierung

Dieser Abschnitt beschäftigt sich mit der Aufgabenstellung 3 der ersten Übung.

3.1 3a Beschreibung der A,B,C Methode

Dieser Abschnitt beschäftigt sich mit der Beschreibung der A, B, C Methode.

Die A, B, C Normalform verwendet die Matrizen A, B und C , welche die Vektoren sind, mit denen die Eingangs-, die Ausgangs- und die Zustandsvariablen beeinflusst werden.

A-Matrix ist der Vektor, der die gegenseitige Beeinflussung der inneren Zustandsvariablen beschreibt. Die A -Matrix hat so viele Spalten wie es innere Zustände gibt, die in der $X(0)$ -Matrix durch die Anzahl der Zeilen gegeben ist.

$$A = \begin{bmatrix} \frac{R_1}{L} & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{C \cdot R_2} \end{bmatrix} X(0) = \begin{bmatrix} 1 \\ 10 \end{bmatrix}$$

B-Matrix ist der Vektor, der die Beeinflussung der Eingangsvariablen beschreibt. Die B -Matrix hat so viele Spalten wie es Eingangsvariablen gibt, die in der $U(t)$ -Matrix durch die Anzahl der Zeilen gegeben ist.

$$B = \begin{bmatrix} \frac{R_1}{L} \\ 0 \end{bmatrix} U(t) = \begin{bmatrix} 1 \end{bmatrix}$$

C-Matrix ist der Vektor, der definiert wird, je nachdem welchen Output man erhalten will. Die C -Matrix wird in der Ausgangsgleichung $y(t) = C * X(t)$ verwendet. Die Anzahl der Spalten ergibt sich aus der Anzahl der Zeilen der Ausgangsmatrix, die angibt wie viele Ausgangsvariablen es gibt. Die Werte 0 und 1 in der C -Matrix werden verwendet um die einzelnen Ausgangsvariablen bei der Betrachtung miteinzubeziehen oder auszuschließen.

$$C * X(t) ; \begin{bmatrix} 1 & 0 \end{bmatrix} * \begin{bmatrix} i(t) \\ y(t) \end{bmatrix} \quad | \quad \begin{bmatrix} 0 & 1 \end{bmatrix} * \begin{bmatrix} i(t) \\ y(t) \end{bmatrix} \quad | \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} i(t) \\ y(t) \end{bmatrix}$$

Hat man sein System auf die A, B, C Normalform gebracht, kann man die einzelnen Werte zum Zeitpunkt t mit dem Integral ausrechnen, das sich durch die Umformung der Eingangs- und Ausgangsgleichung ergibt, wenn es sich um eine lineare Gleichung handelt und wenn der Zustand $X(0)$ bekannt ist. Die aufgestellte Eingangs- und Ausgangsgleichung beschreiben das System, die in einer Simulationssoftware dazu verwendet werden kann, um das System über die Zeit t zu simulieren.

Diese mathematische Beschreibung zeigt, die Zusammenhänge und Beeinflussung der Variablen innerhalb des Systems. Mit der A, B, C Normalform, kann die Veränderung eines Zustands im System zum Zeitpunkt t berechnet werden. Die Berechnung kann auch exakt erfolgen, wenn das System mit einer linearen Differenzialgleichung beschrieben ist.

Bei einer Simulation kann neben der Annäherung auch die exakte Simulation eines Systems erfolgen, das mit der Eingangs- und Ausgangsgleichung in A, B, C Normalform beschrieben ist. Ohne die Berechnungen mit den Gleichungen der A, B, C Normalform kann keine Simulation eines Systems erfolgen, das durch diese Gleichungen beschrieben ist.

3.2 3b Zusammenhang der Formeln der A, B, C Methode

Dieser Abschnitt beschäftigt sich mit der Beschreibung des Zusammenhangs der Formeln der A, B, C Methode.

Ein System in der A, B, C Normalform, kann mit den Eingangs- und Ausgangsgleichungen die Simulationen über die Zeit T erstellen werden. Die aufgestellten Differenzialgleichungen beschreiben das zu testende System, wobei die Vektoren A, B, C die Beeinflussung der Eingangs-, Ausgangs- und inneren Zustandsvariablen beschreiben. Das Integral kann dazu verwendet werden, um zu einem Zeitpunkt t den exakten Zustand eines Systems zu berechnen, wenn die aufgestellte Differenzialgleichung eine lineare Differenzialgleichung ist.