Modellierung und Simulation **Übung 01**
WS 2016/17 **Abgabefrist: Siehe elearning bzw. exchange**

Name: _____ **Aufwand (h):** _____

Punkte: _____

Aufgabe 1 (8 Pkt): MATLAB

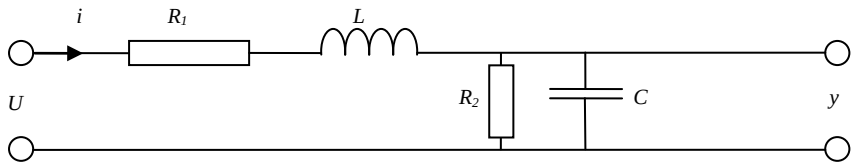
Polynome, eine wesentliche Grundlage bei der Beschreibung des dynamischen Verhaltens von beliebigen Systemen, können als Zeilenvektoren angegeben werden; Polynome von Grad n können allgemein durch Vektoren der Länge n+1 dargestellt werden.

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

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

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

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.

1 Matlab

Dieser Abschnitt beschäftigt sich mit der Aufgabenstellung 1 der ersten Übung. Folgende 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

```
function printPolynom(poly, tStart, tEnd, tStep)
% This function calculates the polynom and plots it
% x-axis: tStart:[tStep,0.01]:tEnd
% y-axis: polyval(poly, currentStep)

% ------------------------- BEGIN Prepare -------------------------------
polySize = size(poly);       % get dimensions of matrix
if (nargin == 3)             % Set default for steps if not given
    tStep = 0.01;
end
% ------------------------- END Prepare ---------------------------------

% ------------------------- BEGIN Validation ----------------------------
if(nargin < 3)                          % Validate parameter count
    error('At least line vector, tStart and tEnd must be given');
elseif(polySize(1) ~= 1)                % Validate for line vector
    error('Only line vectors are allowed');
elseif(tStart > tEnd)                   % Range check
    error('tStart must not overflow tEnd');
elseif((tStep >= 1) || (tStep <= 0))  % Range check
    error('tStep must be ''0 < tStep < 1''');
end
% ------------------------- END Validation ------------------------------

% ------------------------- BEGIN Logic ---------------------------------
stepCount = int8((tEnd - tStart) / tStep);
x    = zeros(1, stepCount);             % pre-allocate x
y    = zeros(1, stepCount);             % pre-allocate y
idx  = 1;                               % counter valriable
cols = polySize(2);
for t=tStart:tStep:tEnd                 % loop over range
    x(idx) = t;                         % build x-axis
    yValue = 0;
    for i=1:1:cols                      % calculate polynom
        yValue = yValue + poly(1, i) * t^(cols - i);
    end
    y(idx) = yValue;                     % build y-axis
    idx    = idx + 1;                    % increase counter
end

figure;                                  % open new window
plot(x, y);                              % plot function over t
hold on                                  % hold on
xlabel(['t (', num2str(tStep), ')']);    % define xlabel
```

```
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 Zeichenklette

```
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.

2 Der Serienresonanzkreis

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

Umformen der Gleichung nach $\check{i}(t)$.

$$\begin{array}{lll} u(t) & = & R_1 * i(t) + L * \frac{\delta i(t)}{\delta t} + y(t) \\ u(t) & = & R_1 * i(t) + L * \check{i}(t) + y(t) \qquad \frac{\delta i(t)}{\delta t} = \check{i}(t) \\ u(t) - R_1 * i(t) - y(t) & = & L * \check{i}(t) \qquad -R_1 * i(t) - y(t)) \\ \frac{1}{L} * u(t) - \frac{R_1}{L} * i(t) - \frac{1}{L} * & = & \check{i}(t) \qquad /L \\ y(t) & & \end{array}$$

$$\check{i}(t) = \frac{1}{L} * u(t) - \frac{R_1}{L} * i(t) - \frac{1}{L} * y(t)$$

Umformen der Gleichung nach $\check{y}(t)$.

$$\begin{array}{lll} i(t) & = & \frac{1}{R_2} * y(t) + C * \frac{\delta y(t)}{\delta t} \\ i(t) & = & \frac{1}{R_2} * y(t) + C * \check{y}(t) \qquad \frac{\delta y(t)}{\delta t} = \check{y}(t) \\ i(t) - \frac{1}{R_2} * y(t) & = & C * \check{y}(t) \qquad -\frac{1}{R_2} * y(t) \\ \frac{1}{C} * i(t) - \frac{1}{C * R_2} * y(t) & = & \check{y}(t) \qquad /C \end{array}$$

$$\check{y}(t) = \frac{1}{C} * i(t) - \frac{1}{C * R_2} * y(t)$$

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

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

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

$$\check{x_1}(t) = -\frac{R_1}{L} * x_1(t) - \frac{1}{L} * x_2(t) + \frac{1}{L} * u(t)$$
$$\check{x_2}(t) = \frac{1}{C} * x_1(t) - \frac{1}{C * R_2} * x_2(t) + 0 * u(t)$$

A, B, C Matrizen aufstellen:

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

In A, B, C Normalform bringen:

$$\check{X}(t) = \begin{bmatrix} \frac{R_1}{L} & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{C*R_2} \end{bmatrix} * X(t) + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} * U(t)$$

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

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 beschreibt *Linear Differential Systems*. Die Matrizen A, B, C enthalten die Faktoren mit denen Eingangs-, Ausgangs- und Zustandsvariablen beeinflusst werden.

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

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

B-Matrix ist die Matrix, welche die Faktoren der Beeinflussung der Eingangsvariablen enthält. Diese Matrix beschreibt die Beeinflussung der Eingangsvariablen, 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 die Matrix, die aufgestellt 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 Ausgangsvariabeln bei der Betrachtung miteinzubeziehen oder auszuschließen.

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

Hat man sein System auf die A, B, C Normalform gebracht, kann man die einzelnen Werte mit dem Integral ausrechnen, das sich durch die Umformung der A, B, C Normalform ergibt, wenn der Zustand $X(0)$ bekannt ist.

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 Simulationssoftware verwendet die Berechnungsmethoden intern um auf die graphische Darstellung zu kommen, also kann es keine Simulationssoftware geben, die nicht auf Grundlage dieser Berechnungsmethoden arbeitet.

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.