Hochschule Karlsruhe
University of Applied Sciences
Fakultät für Elektro- und Informationstechnik

Labor Regelungstechnik

Einführung in MATLAB und SIMULINK

Laborbericht zum Versuch Nr. 1

Jan Hoegen* Maileen Schwenk†

2. Mai 2024

Abstract

In diesem Laborbericht werden grundlegende Funktionen von MATLAB verwendet, um Systeme zu beschreiben, zu analysieren und grafisch darzustellen. Im ersten Abschnitt werden Sinussignale und ihre Lissajous-Figuren im Zeitbereich dargestellt. Anschließend wird ein Hochpass erster Ordnung simuliert und durch sein Bodediagramm und seine Ortskurve dargestellt. Abschließend wird die Temperaturregelung eines Backofens betrachtet. Mit SIMULINK wird ein Blockschaltbild erzeugt, damit werden die Regelvariablen simuliert und abgebildet.

1 Sinussignale im Zeitbereich

Die Funktionen $x_1(t)$, $x_2(t)$ und $x_3(t)$ mit:

$$x_1(t) = 2 \cdot sin(2\pi \cdot 2\,\text{kHz} \cdot t) \tag{1}$$

$$x_1(t) = 2 \cdot sin(2\pi \cdot 6\,\text{kHz} \cdot t - \frac{\pi}{4}) \tag{2}$$

$$x_3(t) = x_1(t) \cdot x_1(t) \tag{3}$$

aus der Versuchsanleitung [1] werden für den Zeitbereich 0 ms bis 3 ms und 10^3 Abtastpunkten mit MATLAB simuliert und in Abbildung 1 dargestellt. Aus dem Diagramm lässt sich ablesen, dass die Frequenz von $x_3(t)$ genau das doppelte der Frequenz von $x_1(t)$ mit einem DC-Offset ist. Darüber hinaus wird eine Lissajous-Figur erstellt. Dabei werden auf den Diagrammachsen die Funktionen $x_1(t)$ (x-Achse) und $x_2(t)$ (y-Achse) eingetragen. Es entsteht ein geschlossenes Muster, da das Frequenzverhältnis mit 1:3 rational ist.

Wird der Zeitbereich der Lissajous-Figuren jedoch auf 0 s bis 3 s bei gleicher Anzahl an Abtastpunkten gelegt, ändert sich die Abtastrate f_s von

$$f_s = \frac{10^3}{3 \cdot 10^{-3}\,\text{s}} = 333{,}3\,\text{kHz} \tag{4}$$

*Matrikel-Nr. 82358. E-Mail hoja1028@h-ka.de
†Matrikel-Nr. 83802. E-Mail scma1315@h-ka-de

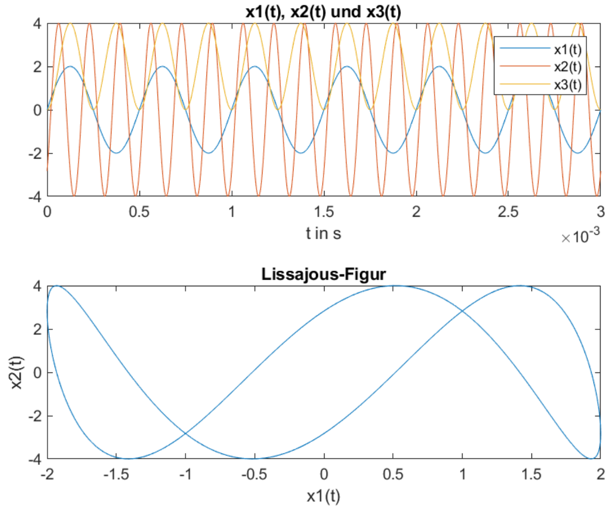


Abbildung 1: Darstellung der Sinussignale
Legende: 10^3 Abtastpunkte

zu der neuen Frequenz

$$f_{s,neu} = \frac{10^3}{3\,\mathrm{s}} = 333{,}3\,\mathrm{Hz} \tag{5}$$

Die Frequenz der Signale $x_1(t)$ und $x_2(t)$ ist deutlich größer als die halbe Abtastfrequenz $f_{s,neu}$, dass Abtasttheorem von Shannon ist verletzt und es kommt zum Aliasing. Da das Frequenzverhältnis der falsch aufgenommen Signale weiterhin 1:3 beträgt, bleibt die Lissajous-Figur jedoch identisch.

Beim Verändern der Abtastpunkte entsteht ein nicht interpretierbares Bild. Das ist darin begründet, dass das Frequenzverhältnis der gemessenen Frequenzen nicht mehr als Bruch zweier ganzer Zahlen dargestellt werden kann. Beide Änderungen sind in Abbildung 2 gezeigt. Der Code zum Erstellen der Grafiken befindet sich im Anhang A.

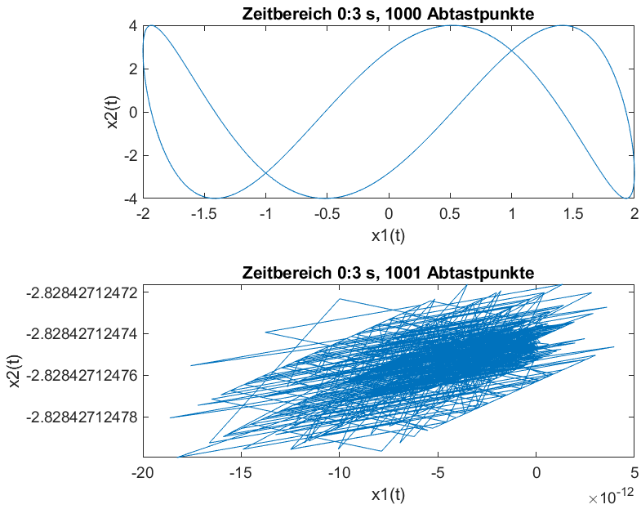


Abbildung 2: Fehlerhafte Lissajous-Figuren

2 Analyse eines Hochpasses

Zunächst werden die Bauteilwerte eines RC-Hochpasses bestimmt, bevor die Analyse mit MATLAB beginnt. Allgemein gilt für den Zusammenhang zwischen Eingangsspannung U_e und Ausgangsspannung U_a bei einem RC-Hochpass erster Ordnung:

$$\frac{U_a}{U_e} = \frac{jwRC}{1 + jwRC} \tag{6}$$

Und die Grenzfrequenz berechnet sich mit

$$f_g = \frac{1}{2\pi RC} \tag{7}$$

Wird der Widerstand R auf 1 kΩ und die Grenzfrequenz zu 10^5 Hz gewählt, berechnet sich die Kapazität zu:

$$C = \frac{1}{2\pi R f_g} = 1{,}59 \cdot 10^{-9}\,\mathrm{F} \tag{8}$$

Nun kann das vollständige Übertragungssystem $G(s)$ in MATLAB verwendet werden.

$$G(s) = \frac{RCs}{RCs + 1} = \frac{1{,}59\,\mu\Omega\,\mathrm{F} \cdot s}{1{,}59\,\mu\Omega\,\mathrm{F} \cdot s + 1} \tag{9}$$

Das erzeugte Bodediagramm findet sich in Abbildung 3 und die zugehörige Ortskurve in Abbildung 4. Da beim Betrachten von negativen Frequenzen auf der Ortskurve die Funktion achsensymmetrisch zur x-Achse ist, kann das Diagramm ohne den Verlust von Informationen um genau diese Spiegelung verkürzt werden. Dies wird durch die Option `ShowFullContour='off'` des `nyquistplot`-Befehls erreicht. Der Code zum Erstellen der Diagramme findet sich in Anhang B.

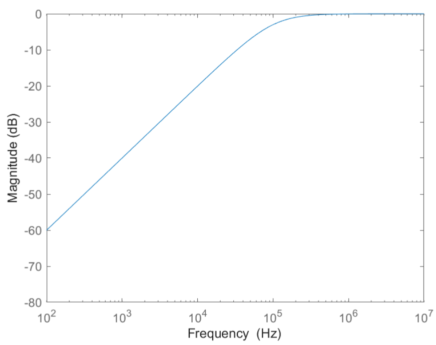


Abbildung 3: Bodediagramm des Hochpasses

2.1 Nachweis der Kreisfunktion

Die Abbildung 4 zeigt einen Kreis mit Radius 0.5 und dem Mittelpunkt 0,5+0i. Folgende Rechnung bringt den Nachweis, dass es sich tatsächlich um einen Kreis handeln muss..

Für einen Kreis mit Mittelpunkt m, sowie Radius r, gilt in der komplexen Zahlenebene mit der Variablen $z \in \mathbb{C}$:

$$|z - m| = r \tag{10}$$

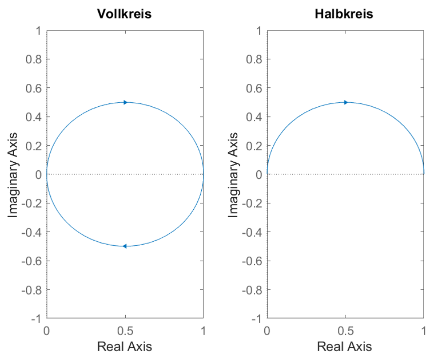


Abbildung 4: Ortskurven des Hochpasses

Einsetzen der Gleichung (6) für z und $m = 0.5 + 0i$ ergibt:

$$\left|\frac{jwRC}{1+jwRC} - 0{,}5 + 0\mathrm{i}\right| = \left|\frac{1}{2} \cdot \frac{-1+jwRC}{1+jwRC}\right| = \left|\frac{1}{2} \cdot \frac{(wRC)^2-1}{(wRC)^2+1} + i\frac{wRC}{(wRC)^2+1}\right| \tag{11}$$

$$= \sqrt{\frac{1}{4}\left(\frac{(wRC)^2-1}{(wRC)^2+1}\right)^2 + \frac{1}{4}\left(\frac{2wRC}{(wRC)^2+1}\right)^2} = \frac{1}{2}\sqrt{\frac{\left((wRC)^2-1\right)^2 + 4(wRC)^2}{\left((wRC)^2+1\right)^2}} = \frac{1}{2} \tag{12}$$

Somit ist gezeigt, dass der Radius tatsächlich 0.5 beträgt und die Kreisgleichung erfüllt ist.

3 Temperaturregler mit SIMULINK

Zuletzt wird der Temperaturregler für einen Backofen betrachtet. Aus der Aufgabenstellung [1] wird das Blockschaltbild in SIMULINK erzeugt (siehe Abbildung 5). Der Puls-Generator am Eingang simuliert das Einschalten auf eine Solltemperatur von 160 °C zu Beginn der Simulation und das Ausschalten nach 30 min. Mit dem Parameter K_1 wird diese Temperatur zu einem Spannungswert übersetzt. Der Zweipunktregler schaltet bei der Spannung u_{ein} ein und gibt eine Spannung von 230 V aus. Bei einer Eingangsspannung von u_{aus} schaltet er ab. Das Heizgerät wird mit einem PT1-Glied simuliert. Dabei wird mit K_2 das Ergebnis wiederum als Temperatur zurückgerechnet. Die Werte der einzelnen Simulationswerte sind in Tabelle 1 gezeigt.

Tabelle 1: Parameter des Temperaturreglers

Parameter	Wert
K_1	0,025 $\frac{\text{V}}{\text{°C}}$
K_2	1,739 $\frac{\text{°C}}{\text{V}}$
u_{ein}	0,2 V
u_{aus}	−0,2 V
timeconst	15 min

Somit ist die Solltemperatur die Führungsgröße, die Eingangsspannung am Zweipunktegler entspricht der Stellgröße und die Ausgangstemperatur des Heizelements ist die Regelgröße. Der zeitliche Verlauf dieser Variablen über 70 min mit einer maximalen Schrittweite von 0,42 s ist in Abbildung 6a dargestellt. Es wurde der schrittweitengesteuerte ode-45-Algorithmus verwendet. Durch Verändern der Schaltschwelle zu ±0,3 V erhält man Abbildung 6b.

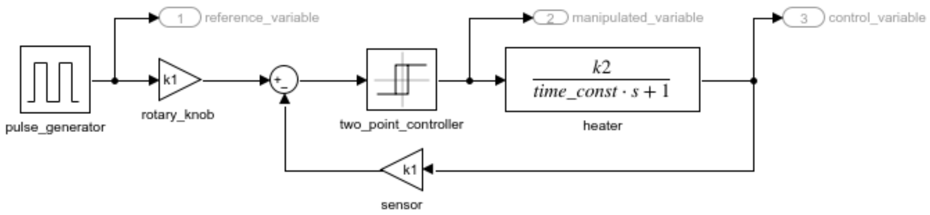
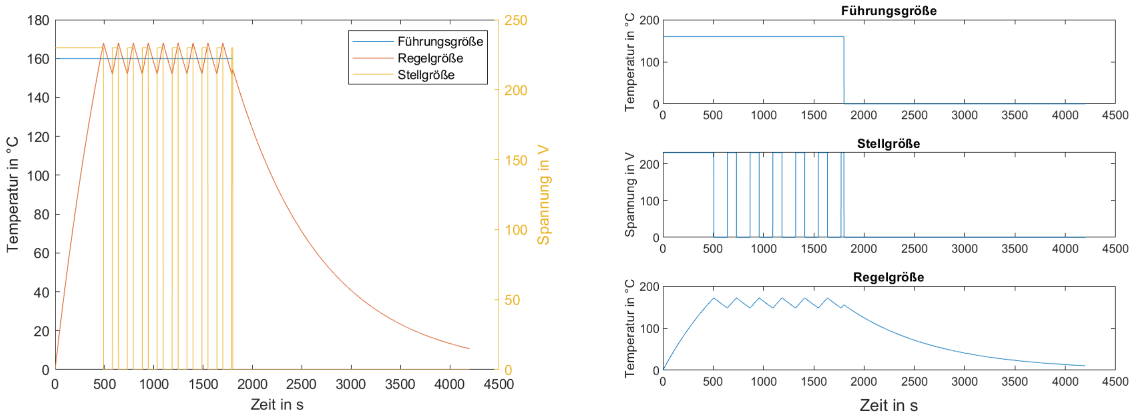


Abbildung 5: Blockschaltbild des Temperaturreglers

Aus den Abbildungen ist zu erkennen: Wird die Differenz zwischen Soll- und Ist-temperatur zu groß, wird die Schaltschwelle u_{ein} überschritten und der Zweipunktegler aktiviert die Heizung. Bei umgekehrten Vorzeichen wird die Heizung ausgeschalten. Durch Verkleinern der Schaltschwelle wird der Regler häufiger umschalten und die Ist-Temperatur weicht weniger von der Führungsgröße ab. Der Code zum Darstellen der Ergebnisse findet sich in C.



(a) Schaltschwelle ±0,2 V

(b) Schaltschwelle ±0,3 V

Abbildung 6: Zeitliche Darstellung der Regelgrößen

4 Literatur

[1] F. Keller, *Labor Regelungstechnik, Einführung in MATLAB/SIMULINK SS2024*, Karlsruhe: Hochschule Karlsruhe, 6. März 2024.

5 Autorenbeiträge

Maileen Schwenk und Jan Hoegen erstellten die Vorbereitung und Messauswertung. Jan Hoegen schrieb den Bericht.

6 Verfügbarkeit des Codes

Der Code zum Auswerten der Daten und Erstellen der Diagramme findet sich unter `https://github.com/JaxRaffnix/Regelungstechnik`. Ebenfalls ist hier der Code zum Erstellen dieser Ausarbeitung hinterlegt.

A MATLAB-Code der Sinussignale

../versuch1/sinus.m

```
clear

LOCAL_DIRECTORY = "C:\Users\janho\Coding\Regelungstechnik\versuch1\";

% x-Axis
time = linspace(0, 3e-3, 1e3);

% declare functions
function f1 = sine1(time)
    f1 = 2 * sin(2 * pi * 2e3 * time - 0);
end
function f2 = sine2(time)
    f2 = 4 * sin(2 * pi * 6e3 * time - pi./4);
end

f1 = sine1(time);
f2 = sine2(time);
f3 = f1 .* f1;

% plot functions
sinplots = tiledlayout(2,1);
nexttile
plot(time, f1);
hold on
plot(time, f2)
hold on
plot(time, f3)
xlabel('t in s')
legend('x1(t)', 'x2(t)', 'x3(t)')
title('x1(t), x2(t) und x3(t)')
nexttile
plot(f1, f2)
xlabel('x1(t)')
ylabel('x2(t)')
title('Lissajous-Figur')

% exportgraphics(sinplots, "sinus.pdf", 'ContentType','vector') % known bug in matlab:
    exportgraphics won't save axis exponent!
saveas(sinplots, LOCAL_DIRECTORY + 'sinus.png')

% plot with wrong paramters
figure
lissplots = tiledlayout(2,1);
time_new = linspace(0, 3, 1e3);

nexttile
f1 = sine1(time_new);
f2 = sine2(time_new);
plot(f1, f2)
xlabel('x1(t)')
ylabel('x2(t)')
title('Zeitbereich 0:3 s, 1000 Abtastpunkte')

% another set of wrong paramters
time_new_new = linspace(0, 3, 1e3+1);
nexttile
f1 = sine1(time_new_new);
f2 = sine2(time_new_new);
plot(f1, f2)
xlabel('x1(t)')
ylabel('x2(t)')
title('Zeitbereich 0:3 s, 1001 Abtastpunkte')
```

```
saveas(lissplots, LOCAL_DIRECTORY + "lissjaou.png")
```

B MATLAB-Code zum Hochpass

../versuch1/hochpass.m

```
clear

LOCAL_DIRECTORY = "C:\Users\janho\Coding\Regelungstechnik\versuch1\";

FREQUENCY = 100e3;
RESISTOR = 1e3;
CAPACITOR = 1 / (2.*pi.*FREQUENCY.*RESISTOR)     % = 1.59e-9

numerator = [RESISTOR*CAPACITOR, 0];
denominator = [RESISTOR*CAPACITOR, 1];
system = tf(numerator, denominator);

figure;
bode = bodeplot(system);
setoptions(bode, ...
    'FreqUnits','Hz', ...
    'PhaseVisible','off', ...
    'xlim', {[1e2, 1e7]} ...
);
title('');

saveas(gcf, LOCAL_DIRECTORY + 'bode.png')

figure;
subplot(1,2,1)      % rows, columns, position
    ny = nyquistplot(system);
    setoptions(ny, ...
        'Xlim', {[0,1]}, ...
        'YLim', {[-1, 1]} ...
    );
    title('Vollkreis');

subplot(1,2,2)
    ny_half = nyquistplot(system);
    setoptions(ny_half, ...
        'ShowFullContour', 'off', ...
        'Xlim', {[0,1]}, ...
        'YLim', {[-1, 1]} ...
    );
    title('Halbkreis');

saveas(gcf, LOCAL_DIRECTORY + 'ortskurve.png')
```

C MATLAB-Code zum Temperaturregler

../versuch1/tempregler.m

```
% global parameters
LOCAL_DIRECTORY = "C:\Users\janho\Coding\Regelungstechnik\versuch1\";
STOPTIME = 70*60 - 1;    % hide last data point to hide second positive flank of the pulse
    generator
POINTS = 1e4;
```

```
STEPSIZE = STOPTIME / POINTS    % = 0.42


% set model parameters
k1 = 10/400;
k2 = 400/230;
time_const = 15*60;
controlller_on = 0.2;
controller_off = -0.2;

% first simulation
%model = LOCAL_DIRECTORY +  'tempregler_modell.slx';
model = "tempregler_modell";
load_system(model);

% open_system(model);
set_param(model,...
    "SolverType","Variable-step", ...
    "SolverName", "VariableStepAuto", ...
    "MaxStep", num2str(STEPSIZE), ...
    "StopTime", num2str(STOPTIME));
output = sim(model);    % 1: control variable, 2: manipulated_variable, 3: reference_variable

% second simulation with changed parameters
controlller_on = 0.3;
controller_off = -0.3;
output_new = sim(model);    % 1: control variable, 2: manipulated_variable, 3:
    reference_variable


% plotting
label = {"Führungsgröße", "Stellgröße", "Regelgröße"};
unit = {"Temperatur in °C", "Spannung in V", "Temperatur in °C"};

figure;
tempregler_plot = tiledlayout("vertical");
nexttile(tempregler_plot);
plot(output.yout{1}.Values.Time, output.yout{1}.Values.Data);
hold on
plot(output.yout{3}.Values.Time, output.yout{3}.Values.Data)
xlabel("Zeit in s")
ylabel("Temperatur in °C")

yyaxis right
plot(output.yout{2}.Values.Time, output.yout{2}.Values.Data);
ylabel("Spannung in V")

legend("Führungsgröße", "Regelgröße", "Stellgröße")

saveas(tempregler_plot, LOCAL_DIRECTORY + "tempregler_plot.png")

figure;
tempregler_plot_new = tiledlayout("vertical");
for i = 1:3
    nexttile
    plot(output_new.yout{1}.Values.Time, output_new.yout{i}.Values.Data);
    title(label(i))
    ylabel(unit(i))
end
xlabel(tempregler_plot_new, "Zeit in s")
saveas(tempregler_plot_new, LOCAL_DIRECTORY + "tempregler_plot_new.png")

% save block diagram
print('-stempregler_modell', '-dpng', LOCAL_DIRECTORY + 'tempregler_block.png')
```