

Labor Regelungstechnik

Einführung in MATLAB und SIMULINK

Laborbericht zum Versuch Nr. 1

Jan Hoegen*

Maileen Schwenk†

1. 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:

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

$$x_2(t) = 2 \cdot \sin(2\pi \cdot 6 \text{ kHz} \cdot t - \frac{\pi}{4}) \quad (2)$$

$$x_3(t) = x_1(t) \cdot x_2(t) \quad (3)$$

aus der Versuchsanleitung [1] werden für den Zeitbereich 0 ms bis 3 ms mit MATLAB simuliert und in Abbildung 1 dargestellt. Darüber hinaus wird eine Lissajous-Figur mit $x_1(t)$ auf der x-Achse und $x_2(t)$ auf der y-Achse abgebildet. Es ist zu erkennen, dass die Frequenz von $x_3(t)$ das doppelte der Frequenz von $x_1(t)$ mit einem DC-Offset beträgt.

Wird der Zeitbereich der Lissajous-Figuren jedoch auf 0 s bis 3 s gelegt und somit die Größenordnung um 10^3 erhöht, bleibt das Bild aufgrund des Aliasing-Effekts identisch. Wird der Zeitbereich leicht verschoben, entsteht ein nicht interpretierbares Bild. Beide Änderungen sind in Abbildung 2 gezeigt. Der Code zum Erstellen der Grafiken befindet sich im Anhang A.

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:

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

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

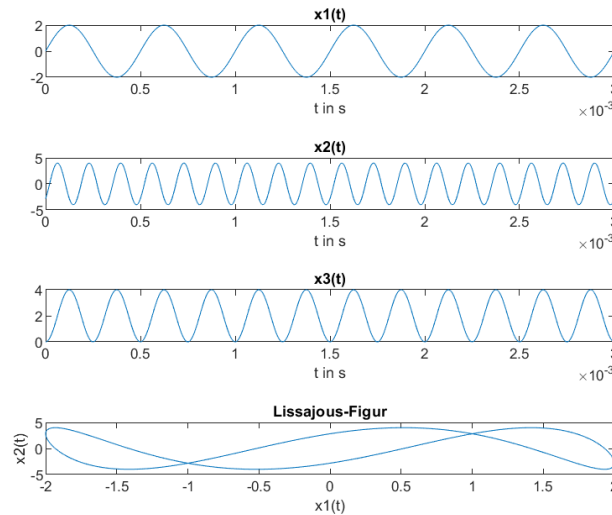


Abbildung 1: Darstellung der Sinussignale

Legende: 10^3 Abtastpunkte

$$\frac{U_a}{U_e} = \frac{j\omega RC}{1 + j\omega RC} \quad (4)$$

Und die Grenzfrequenz somit zu

$$f_g = \frac{1}{2\pi RC} \quad (5)$$

Wird der Widerstand R auf $1 \text{ k}\Omega$ und die Grenzfrequenz zu $1 \cdot 10^5 \text{ Hz}$ gewählt, berechnet sich die Kapazität zu:

$$C = \frac{1}{2\pi R f_g} = 1,59 \cdot 10^{-9} \text{ F} \quad (6)$$

Nun kann das vollständige Übertragungssystem in MATLAB verwendet werden. Das erzeugte Bodediagramm findet sich in Abbildung 3 und die zugehörige Ortskurve in Abbildung 4. Da die Ortskurve 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.

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 $z \in \mathbb{C}$:

$$|z - m| = r \quad (7)$$

Einsetzen der Gleichung (4) für z und $m = 0,5 + 0i$ ergibt:

$$\begin{aligned} \left| \frac{j\omega RC}{1 + j\omega RC} - 0,5 + 0i \right| &= \left| \frac{1}{2} \cdot \frac{-1 + j\omega RC}{1 + j\omega RC} \right| = \left| \frac{1}{2} \cdot \frac{(\omega RC)^2 - 1}{(\omega RC)^2 + 1} + i \frac{\omega RC}{(\omega RC)^2 + 1} \right| \\ &= \sqrt{\frac{1}{4} \left(\frac{(\omega RC)^2 - 1}{(\omega RC)^2 + 1} \right)^2 + \frac{1}{4} \left(\frac{2\omega RC}{(\omega RC)^2 + 1} \right)^2} = \frac{1}{2} \sqrt{\frac{((\omega RC)^2 - 1)^2 + 4(\omega RC)^2}{((\omega RC)^2 + 1)^2}} = \frac{1}{2} \end{aligned} \quad (8)$$

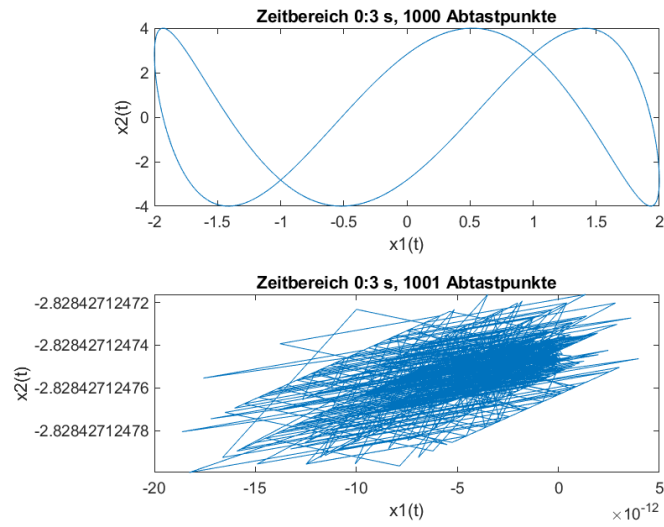


Abbildung 2: Fehlerhafte Lissajous-Figuren

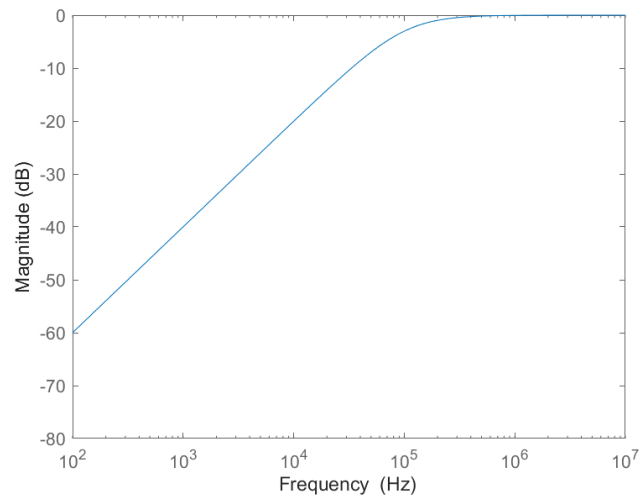


Abbildung 3: Bodediagramm des Hochpasses

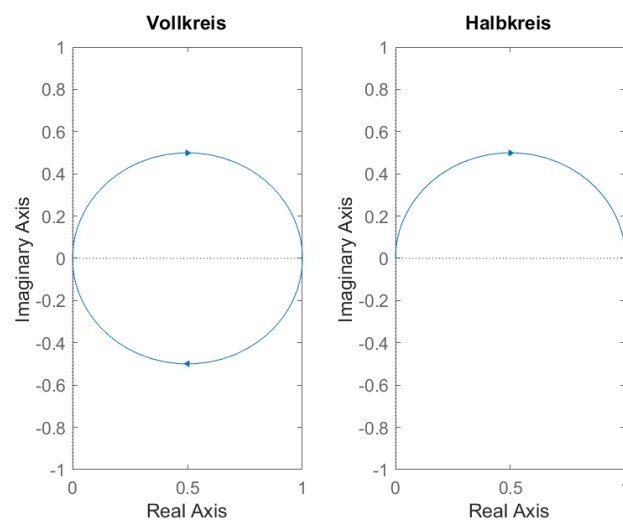


Abbildung 4: Ortskurven des Hochpasses

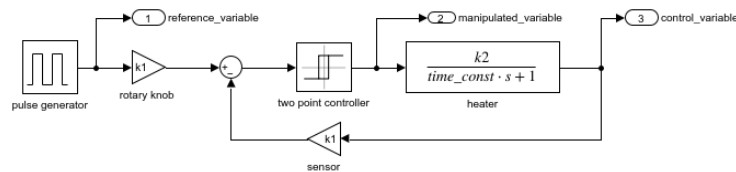


Abbildung 5: Blockschaltbild des Temperaturreglers

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 Simulationen sind in Tabelle 1 gezeigt.

Tabelle 1: Parameter des Temperaturreglers

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

Somit ist die Solltemperatur die Führungsgröße, die Eingangsspannung am Zweipunktregler entspricht der Stellgröße und die Ausgangstemperatur des Heizelements ist die Regelgröße. Der zeitliche Verlauf dieser Variablen über 70 min ist in Abbildung 6a dargestellt. Durch Verändern der Schaltschwelle zu $\pm 0,01$ V erhält man Abbildung 6b.

Aus den Abbildungen ist zu erkennen: Wird die Differenz zwischen Soll- und Isttemperatur zu groß, wird die Schaltschwelle u_{ein} überschritten und der Zweipunktregler aktiviert die Heizung. Bei umgekehrten Vorzeichen wird die Heizung ausgeschaltet. 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.

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.

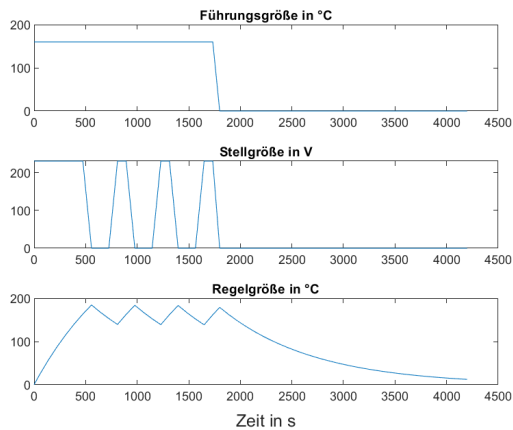
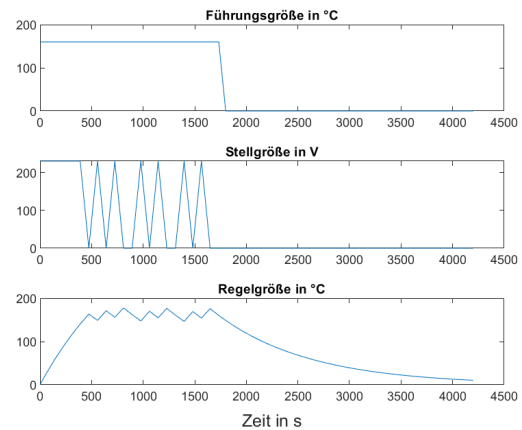
(a) Schaltschwelle $\pm 0,2$ V(b) Schaltschwelle $\pm 0,01$ V

Abbildung 6: Zeitliche Darstellung der Regelgrößen

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(4,1);
nexttile
plot(time, f1)
xlabel('t in s')
title('x1(t)')
nexttile
plot(time, f2)
xlabel('t in s')
title('x2(t)')
nexttile
plot(time, f3)
xlabel('t in s')
title('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 parameters
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 parameters
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 Tiefpass

../versuch1/hochpass.m

```

clear

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

FREQUENCY = 100e3;
RESISTOR = 1e3;
CAPACITOR = 1 / (2.*pi.*FREQUENCY.*RESISTOR)    % = 1.5915e-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 = 1e3;

STEPSIZE = STOPTIME / POINTS

% set model parameters
model = LOCAL_DIRECTORY + 'tempregler_modell.slx';
k1 = 10/400;
k2 = 400/230;
time_const = 15*60;

% run first simulation
controlller_on = 0.2;
controlller_off = -0.2;
% output = sim(model, "StopTime", num2str(STOPTIME), 'FixedStep', num2str(STEPSIZE)); % 1: control variable, 2: manipulated_variable, 3:
reference_variable
output = sim(model, "StopTime", num2str(STOPTIME)); % 1: control variable, 2: manipulated_variable, 3: reference_variable

% second simulation with changed parameters
controlller_on = 0.01;
controlller_off = -0.01;
output_new = sim(model, "StopTime", num2str(STOPTIME), 'FixedStep', num2str(STEPSIZE)); % 1: control variable, 2: manipulated_variable, 3:
reference_variable

```

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

figure;
tempregler_plot = tiledlayout("vertical");
for i = 1:3
    nexttile
    plot(output.yout{1}.Values.Time, output.yout{i}.Values.Data);
    title(label(i))
end
xlabel(tempregler_plot, "Zeit in s")
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))
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')
```
