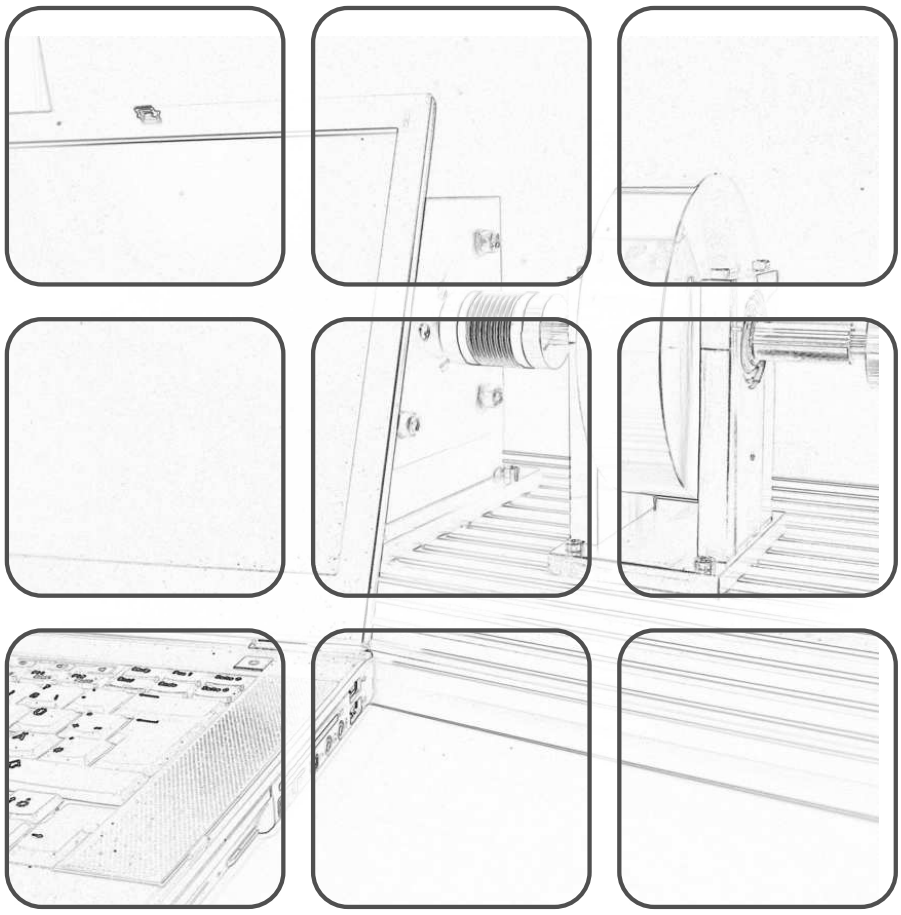
TECHNISCHE UNIVERSITÄT WIEN

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

Laborübung
WS 2016/2017

REGELUNGSTECHNIK

Univ.-Prof. Dr. techn. Andreas KUGI



Regelungstechnik

Laborübung
WS 2016/2017

Univ.-Prof. Dr. techn. Andreas KUGI

TU Wien
Institut für Automatisierungs- und Regelungstechnik
Gruppe für komplexe dynamische Systeme

Gusshausstrasse 27-29
1040 Wien
Telefon: +43 1 58801 – 37615
Internet: http://www.acin.tuwien.ac.at

Inhaltsverzeichnis

0 Organisation

0.1 Inhalt

Die vorliegende Laborübung ist Teil des Moduls Regelungstechnik. Voraussetzung für die Teilnahme an der Laborübung ist der Besuch der zugehörigen VU Automatisierung. Das Ziel dieser Laborübung ist es, einen Einblick in die Modellierung, Analyse, Simulation sowie den Entwurf von Regelungssystemen mittels analytischer Methoden und unter Zuhilfenahme von modernen Simulationswerkzeugen zu erhalten. Es werden drei Übungseinheiten abgehalten. Nach einer Einführung in das Softwarepaket MATLAB/SIMULINK werden in der zweiten und dritten Übungseinheit die erworbenen Kenntnisse an Beispielen zur Modellbildung, Simulation und zum Reglerentwurf angewandt.

0.2 Ablauf

0.2.1 Vorbereitung

Einen Monat vor dem ersten Übungstermin wird das Skriptum ausgegeben. **Zur Vorbereitung auf die jeweilige Übungseinheit sind in Zweier-Gruppen alle in diesem Skriptum gestellten Aufgaben zu lösen.** Selbst wenn die Vorbereitung in Zweier-Gruppen erfolgt, wird davon ausgegangen, dass alle Teilnehmenden die gestellten Aufgaben eigenständig lösen können. Bitte prüfen Sie Ihre Berechnungs- und Simulationsergebnisse auf Plausibilität und achten Sie auf funktionierende Simulationsmodelle.

Wenden Sie sich bei Problemen oder Fragen rechtzeitig an die in den jeweiligen Übungsangaben genannten Ansprechpersonen. Insbesondere wird während der Übung keine Zeit mehr für die Korrektur fehlerhaft oder unvollständig ausgearbeiteter Vorbereitungen zur Verfügung stehen.

Studentenlizenzen für die zur Bearbeitung der Aufgaben benötigten Softwarepakete können Sie z. B. beim Zentralen Informatikdienst der TU Wien (http://www.zid.tuwien.ac.at) beziehen. Die bereitgestellen Beispieldateien setzen MATLAB/SIMULINK ab Version R2016a voraus. Ferner stehen Ihnen im Computerlabor des Instituts (Raum CA0426) von Montag bis Freitag in der Zeit von 9.00 bis 18.00 Uhr Rechner zur Verfügung, sofern der jeweilige Tag nicht vorlesungsfrei ist und der Raum nicht durch Lehrveranstaltungen belegt ist. Der Raum wird bei Bedarf aufgeschlossen.

0.2.2 Laborübung

Während der vierstündigen Übungseinheiten werden die von Ihnen ausgearbeiteten Lösungen der Aufgaben besprochen, die zugrunde liegende Theorie diskutiert und weiterführende

Aufgaben bearbeitet.

0.2.3 Anforderungen und Beurteilung

Während der Übungseinheiten besteht Anwesenheitspflicht. **Die Ausarbeitung aller Aufgaben sowie die Ausführbarkeit aller erstellten Simulationen sind notwendig für eine positive Beurteilung der Vorbereitung und damit für eine Teilnahme am Übungstermin**.

In die positive Beurteilung gehen

- die Richtigkeit Ihrer vorbereiteten Lösungen,
- Ihre Mitarbeit während der Laborübungen und
- für die Übung benötigte Theoriekenntisse

ein.

Für eine **positive Gesamtbeurteilung müssen Sie alle Übungseinheiten positiv** abschließen.

0.3 Termine Wintersemester 2016/17

Die Vorbesprechung zur Lehrveranstaltung findet am 03.11.2016 um 14:00 Uhr im Computerlabor des Instituts (Raum CA0426) statt. Es werden dabei unter anderem die vorzubereitenden Aufgaben besprochen. Daher ist es notwendig, dass alle Teilnehmenden zur Vorbesprechung anwesend sind. Die Gruppeneinteilung in Zweier-Gruppen erfolgt über TISS.

Alle weiteren Übungstermine werden im Computerlabor des Instituts (Raum CA0426) abgehalten. Datum und Uhrzeit entnehmen Sie bitte nachfolgender Tabelle.

Datum		Zeit	Gruppen	Übungseinheit
Donnerstag	03.11.2016	14:00 bis 14:30	alle	Vorbesprechung
Donnerstag	15.12.2016	12:00 bis 16:00	alle	Übung 1
Donnerstag	12.01.2017	14:00 bis 18:00	alle	Übung 2
Donnerstag	26.01.2017	14:00 bis 18:00	alle	Übung 3

0.4 Ansprechperson für organisatorische Belange

Bei Fragen oder Anregungen organisatorischer Natur wenden Sie sich bitte an

- Christian Hartl <hartl@acin.tuwien.ac.at>.

0.5 Weitere Informationen

Aktuelle Informationen zur Lehrveranstaltung sind auf der Instituts-Homepage http://www.acin.tuwien.ac.at abrufbar. Dort sind auch weitere Materialien (vor allem MATLAB-Dateien) für Sie zum Download bereitgestellt.

1 Systemanalyse mit Matlab/Simulink

Ziel dieser Übung ist es, das Computerprogramm MATLAB/SIMULINK für die Systemanalyse einzusetzen. Alle Aufgabenstellungen dieser Übungseinheit sind mit diesem Softwarepaket zu lösen. Im Computerlabor des Instituts steht MATLAB/SIMULINK in der Version R2016a zur Verfügung.

Studieren Sie als Vorbereitung auf die Übung zumindest folgende Kapitel im Skriptum zur VU Automatisierung (WS 2016/17) [1.1]:

- Kapitel 1, vollständig
- Kapitel 2, vollständig
- Kapitel 3, vollständig
- Kapitel 6, vollständig.

Bei Fragen oder Anregungen zu dieser Übung wenden Sie sich bitte an

- Martin Saxinger <saxinger@acin.tuwien.ac.at> oder
- Andreas Ettl <ettl@acin.tuwien.ac.at>.

Für organisatorische Fragen wenden Sie sich bitte an

- Christian Hartl <hartl@acin.tuwien.ac.at>.

1.1 Matlab

MATLAB ist ein Computernumerikprogramm. Der Name ist eine Abkürzung für MATrix LABoratory, womit bereits angedeutet wird, dass das Programm gut zum Rechnen mit Vektoren und Matrizen geeignet ist.

Der MATLAB-Desktop (das eigentliche Programmfenster) enthält in der Standardeinstellung folgende Fenster. (Dies kann jedoch individuell angepasst werden.)

- **Command Window**
  Es stellt den Eingabebereich dar, welcher auf jeden Fall angezeigt werden muss. Hier können alle Befehle hinter der Eingabeaufforderung >> eingegeben und direkt ausgeführt werden. Schließt man eine Befehlssequenz mit einem Semikolon ab, so wird die Ausgabe von MATLAB unterdrückt. Das Drücken der Eingabe-Taste bewirkt die sofortige Ausführung der Befehlszeile. Im Falle mehrzeiliger Befehlseingaben kann mit der Umschalt- und der Eingabe-Taste in eine neue Zeile gesprungen werden.

- **Editor**
  Im Editor können Funktionen, Skripte oder Programm-Code (z. B. `m`-Code oder auch C-Code) erstellt und editiert werden. Es werden die in Programmierumgebungen üblichen Möglichkeiten zum schrittweisen Ausführen der Befehlssequenzen, zum Debuggen, etc. zur Verfügung gestellt. Skripte werden auch `m`-files genannt; sie besitzen die Dateiendung `*.m` und werden zur Laufzeit von einem Interpreter abgearbeitet. Skripte enthalten MATLAB-Befehlssequenzen, wie sie prinzipiell auch direkt im Command Window eingegeben werden können. Funktionen besitzen ebenfalls die Dateiendung `*.m`. Sie erhalten meist Eingangsparameter und geben oft auch Rückgabewerte zurück.

- **Workspace Browser**
  Die aktuellen Variablen werden in MATLAB im so genannten Workspace angezeigt. Sie können durch Anklicken aufgerufen und verändert werden.

- **Current Directory Browser**
  Im Current Directory Browser wird das aktuelle Arbeitsverzeichnis dargestellt. Dateien und Ordner können geöffnet, angelegt, bearbeitet, etc. werden. Für kleine Projekte empfiehlt es sich, alle Dateien, auf die während der Rechnung oder Simulation zugegriffen wird, in einem gemeinsamen, lokalen (aktuellen) Verzeichnis abzulegen.

- **Command History Window**
  Die in der Vergangenheit ausgeführten Befehle sind im Command History Window chronologisch sortiert. Die einzelnen Befehle können herauskopiert bzw. durch einen Doppelklick erneut ausgeführt werden.

Hinweis: Auf der Homepage der Lehrveranstaltung steht für Sie das zip-Archiv `uebung1.zip` zum Download zur Verfügung. In diesem Archiv sind alle für die folgende Übung notwendigen Dateien enthalten.

1.1.1 Grundlagen in Matlab

Die grundlegenden Befehle und deren Anwendung sind in den Dateien `cds_matlab_intro_part1.m` und `cds_matlab_intro_part2.m` gezeigt.

Hinweis: Vor der Bearbeitung der nachstehenden Aufgabe öffnen Sie diese Dateien im MATLAB-Editor und arbeiten Sie alle Befehle schrittweise durch. Beachten Sie, dass die Funktion `mittelwert.m` aufgerufen wird, welche sich daher im aktuellen Arbeitsverzeichnis befinden muss. Zum Ausführen einzelner Befehlssequenzen markieren Sie diese im Editor und drücken F9. Mithilfe der Symbolfolge `%%␣` kann der Programmcode in einzeln ausführbare Sektionen unterteilt werden, welche durch die Tastenkombination *Strg* + *Enter* abgearbeitet werden. Zum Ausführen eines ganzen `m`-files geben Sie entweder dessen Namen (ohne Dateiendung) im Command Window

ein oder öffnen Sie die Datei im Editor und drücken F5. Versuchen Sie alle Befehle zu verstehen und machen Sie gegebenenfalls von der Hilfefunktion Gebrauch.

Aufgabe 1.1. In den folgenden Aufgaben sollen die in `cds_matlab_intro_part1.m` und `cds_matlab_intro_part2.m` beschriebenen Befehle an konkreten Beispielen angewandt werden.

1. Gegeben ist das Gleichungssystem

$$x_1 + 2x_2 + 4x_3 = 5 \tag{1.1a}$$
$$2x_1 + 2x_2 + x_3 = 4 \tag{1.1b}$$
$$3x_1 + 2x_2 = 1\ . \tag{1.1c}$$

   Schreiben Sie dieses Gleichungssystem in Matrixdarstellung an und bestimmen Sie den Lösungsvektor $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^{\mathrm{T}}$. Führen Sie die Rechnung einmal mit dem Befehl `inv()` und einmal mit dem Befehl `mldivide()` (oder in seiner Kurzform \) durch. Untersuchen Sie die Geschwindigkeits- und Genauigkeitsunterschiede der beiden Befehle. Wann würden Sie welchen Befehl verwenden?

   Hinweis: Um die benötigte Rechenzeit zu bestimmen, können die Befehle `tic` und `toc` verwendet werden. Weiters kann mithilfe von `norm`$(\mathbf{Ax} - \mathbf{b})$ der numerische Fehler der Berechnung bestimmt werden.

2. Gegeben ist die Fourier-Reihenentwicklung n-ter Ordnung der Rechteckfunktion

$$\mathrm{rect}(x) \approx A \sum_{k=1}^{n} \frac{4}{\pi(2k-1)} \sin((2k-1)x)\ , \tag{1.2}$$

   wobei $A = 10$ die Amplitude bezeichnet. Stellen Sie diese Rechteckfunktion für $n = 1, 2, \ldots, 100$ im Intervall $x \in [0, 10]$ mithilfe einer `for`-Schleife dar. Nutzen Sie zur schrittweisen grafischen Darstellung der Funktion in der `for`-Schleife den `pause`-Befehl.

3. Gegeben ist die Funktion

$$f(x, y, t) = \sin\left(\frac{x\pi}{10}\right) \sin\left(\frac{y\pi}{10}\right) |\sin(t)|\ . \tag{1.3}$$

   Stellen Sie die zu dieser Funktion gehörige Fläche $z = f(x, y, t_{\mathrm{konst}})$ mithilfe einer `for`-Schleife für verschiedene Zeitpunkte t_{konst} sowie $x \in [0, 10]$ und $y \in [0, 10]$ grafisch dar.

1.1.2 Control System Toolbox

Toolboxen sind Sammlungen von Funktionen, meist in Form von `m`-files, die den Funktionsumfang des Basisprogramms erweitern. Nach der erstmaligen Installation werden Toolboxen automatisch beim Programmstart von MATLAB geladen. Eine Übersicht der installierten Toolboxen erhält man mit dem Befehl `ver`.

Die Toolbox 'Control System' ist häufig bei regelungstechnischen Aufgabenstellungen nützlich. Sie unterstützt bei der Analyse und dem Reglerentwurf von linearen dynamischen Systemen.

Hinweis: Vor der Bearbeitung der nachstehenden Aufgaben öffnen Sie die Datei `cds_matlab_intro_part3.m` im MATLAB-Editor und arbeiten Sie alle Befehle schrittweise durch. Versuchen Sie alle Befehle zu verstehen und machen Sie gegebenenfalls von der Hilfefunktion Gebrauch.

Zur Bestimmung der numerischen Lösung eines (nichtlinearen) Differentialgleichungssystems der Form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1.4}$$

werden in MATLAB zahlreiche Integrationsalgorithmen zur Verfügung gestellt. Die Funktion eines solchen Integrationsalgorithmus soll anhand des expliziten Euler-Verfahrens

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_a \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{1.5}$$

gezeigt werden. Hierbei bezeichnet $\mathbf{x}_k \approx \mathbf{x}(kT_a)$ die Approximation des Zustandsvektors zum Zeitpunkt $t = kT_a$ mit der Abtastzeit T_a.

Aufgabe 1.2. Betrachten Sie die Differentialgleichung zweiter Ordnung zur Beschreibung eines harmonischen Oszillators in der Form

$$m\ddot{s} + c\dot{s} + ks = f \ , \qquad s(0) = s_0, \quad \dot{s}(0) = v_0 \tag{1.6}$$

mit der Masse m, der Auslenkung s, der Dämpfungskonstante c, der Federkonstante k sowie der externen Kraft f. Schreiben Sie (1.6) in Zustandsraumdarstellung

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{1.7a}$$
$$y = \mathbf{c}^{\mathrm{T}}\mathbf{x} \tag{1.7b}$$

mit $\mathbf{x}^{\mathrm{T}} = \begin{bmatrix} x_1, x_2 \end{bmatrix}$ und $\mathbf{u} = u$ an. Wählen sie hierfür $u = f$ und $y = s$.

1. Bestimmen Sie die Eigenwerte der Dynamikmatrix und beurteilen Sie das System hinsichtlich Stabilität unter Annahme positiver Konstanten m, c und k. Berechnen Sie die Transitionsmatrix $\mathbf{\Phi}(t)$ und bestimmen Sie die allgemeine Lösung von (1.7) für den Fall, dass u durch den Einheitssprung $\sigma(t)$ gegeben ist.

2. Schreiben Sie ein `m`-file, in welchem der Integrationsalgorithmus (1.5) auf das Differentialgleichungssystem (1.7) angewendet wird. Verwenden Sie als Inte-

grationszeitraum $[0, 20]$s, als Zeitschrittweite (Abtastzeit) $T_a = 0.05\,\mathrm{s}$ und für die Systemparameter $m = 1\,\mathrm{kg}$, $c = 1\,\mathrm{N\,s/m}$, $k = 2\,\mathrm{N/m}$, $s_0 = 0\,\mathrm{m}$, $v_0 = 0\,\mathrm{m/s}$. Variieren Sie nun die Abtastzeit und untersuchen Sie deren Einfluss auf die Genauigkeit der numerisch berechneten Lösung. Stellen Sie hierzu den Zeitverlauf der Zustände der numerischen, als auch der exakten Lösung in einer Grafik dar.

3. Vergleichen Sie Ihre Ergebnisse mit den mittels Control System Toolbox berechneten Sprungantworten des gegebenen kontinuierlichen Systems und des zugehörigen Abtastsystems (Abtastzeit T_a). Zur Diskretisierung können Sie den Befehl `c2d()` verwenden.

1.2 Simulink

Simulink ist eine Erweiterung von Matlab zur Simulation und Analyse dynamischer Systeme. Simulink-Modelle besitzen die Dateiendung `*.slx`. Die grafische Bedienoberfläche erlaubt die Erstellung von Blockschaltbildern der untersuchten Systeme. Einerseits stellt Simulink eine Bibliothek mit vorgefertigten Funktionsblöcken zur Verfügung, andererseits können benutzerdefinierte Blöcke erstellt werden. Eine flexible Möglichkeit dafür sind so genannte S-function-Blöcke. S-functions können z. B. als Matlab `m`-file oder in C programmiert werden. Sie erlauben die Implementierung dynamischer Modelle in einem Block.

1.2.1 Grundlagen von Simulink

Mit dem Befehl `simulink` wird der 'Simulink Library Browser' geöffnet. Er enthält die Funktionsblöcke, welche per Drag & Drop in das Simulink Modell gezogen werden können. Die Blöcke sind mittels Signalflußleitungen zu verbinden. Besonders häufig benötigte Blöcke sind in der Gliederung des Simulink Library Browsers z. B. in den folgenden Gruppen zu finden.

- `Continuous`: Blöcke zur Simulation zeitkontinuierlicher Systeme, unter anderem der zeitkontinuierliche Integrator `1/s`.
- `Math Operations`: Einige mathematische Operationen, z. B. Addieren, Multiplizieren, Quadrieren.
- `Sinks`: Blöcke, die nur einen Eingang (oder mehrere Eingänge), aber keinen Ausgang besitzen. `Scopes` beispielsweise dienen zur grafischen Darstellung von Signalen. Signale können an beliebiger Stelle direkt von Signalflussleitungen abgegriffen werden. Der Block `To Workspace` erlaubt den Export von Simulationsergebnissen in den Matlab-Workspace, wo sie dann als Variable zur weiteren Auswertung zur Verfügung stehen.
- `Sources`: Blöcke, die nur einen Ausgang (oder mehrere Ausgänge), aber keinen Eingang besitzen, wie etwa Signalgeneratoren.

Zur effizienten Arbeit mit Simulink wird folgendes Vorgehen empfohlen:

1. Parameterwerte werden nicht direkt in SIMULINK eingetragen, sondern zusammengefasst in einem `m`-file definiert, welches vor der Simulation ausgeführt wird. Damit können die Parameter einfach und an zentraler Stelle geändert werden. Zum Update der Parameter muss das `m`-file erneut ausgeführt werden.

   Hinweis: Alle Variablen, die sich im Matlab-Workspace befinden, stehen auch in SIMULINK *zur Verfügung.*

2. Werden die mathematischen Ausdrücke umfangreicher, empfiehlt es sich, die Verschaltung vieler Einzelblöcke durch die Verwendung benutzerdefinierter (programmierter) Blöcke zu umgehen. Die entsprechenden Blöcke finden sich in der Gruppe `User-Defined Functions`. Beispielsweise können im Block `Fcn` beliebige mathematische Ausdrücke aus mathematischen Operatoren und MATLAB Funktionen zusammengesetzt werden. Für dynamische Systeme eignet sich der Block `level 2 MATLAB S-Function`.

3. Eine weitere Möglichkeit die Übersichtlichkeit von Modellen zu verbessern, ist die Verwendung von Subsystemen (`Ports & Subsystems` → `Subsystem`).

4. Um die Anzahl der am Bildschirm dargestellten Signalflussleitungen zu reduzieren, können die Blöcke `From` und `Goto` aus der Gruppe `Signal Routing` verwendet werden.

Simulationseinstellungen können im Menü *Simulation* → *Model Configuration Parameters* vorgenommen werden. Besonders wesentlich ist dabei die Wahl der Simulationsdauer und des Integrationsalgorithmus. Ferner können für den Integrationsalgorithmus Schranken der Zeitschrittweite und Genauigkeitsanforderungen eingestellt werden. Im Rahmen dieser Einführung soll auf eine detaillierte Diskussion der verwendeten Algorithmen verzichtet werden. Einen Überblick über einige in MATLAB zur Verfügung stehende Lösungsverfahren für Anfangswertprobleme erhalten Sie in der Hilfe zu '*ode23, ode45, ode113, ode15s, ode23s, ode23t, ode23tb*' (aufrufbar z. B. mit `doc ode45`) unter der Überschrift *Algorithms*. Diese Algorithmen werden auch von SIMULINK verwendet. Ferner sei auf die Fachliteratur, z. B. [1.2, 1.3], verwiesen.

Die Simulation kann durch Anklicken des Startknopfes ▶ oder durch Drücken der Tasten *Strg* + *T* gestartet werden. Um eine Simulation alternativ aus dem Eingabefenster oder einem `m`-file zu starten, kann der Befehl `sim()` verwendet werden.

1.2.2 Implementierung von dynamischen Systemen

Die Implementierung eines dynamischen Systems, für das ein Modell in Form einer (expliziten) Differentialgleichung existiert, kann als Blockschaltbild, als S-function oder auch als MATLAB-function erfolgen. Die ersten beiden Varianten werden im Folgenden kurz erläutert.

Blockschaltbild

In der Simulationsdatei `cds_blockschaltbild_PT2.slx` und der dazugehörigen Parameterdatei `cds_blockschaltbild_PT2_Parameter.m` ist die Implementierung eines dyna-

mischen Systems in Form von Blockschaltbildern zu finden. Als Beispiel wird hier ein P-T_2-Glied, welches durch die Übertragungsfunktion

$$G(s) = \frac{V}{1 + 2\xi T s + (sT)^2} \tag{1.8}$$

gegeben ist, betrachtet. Um dieses in Form eines Blockschaltbildes mit Integratoren darstellen zu können, muss zunächst die Beschreibung des Systems in Form eines Differentialgleichungssystems erster Ordnung

$$\dot{x}_1(t) = x_2(t) \tag{1.9a}$$

$$\dot{x}_2(t) = \left(-\frac{1}{T^2}x_1(t) - \frac{2\xi}{T}x_2(t) + \frac{V}{T^2}u(t)\right) \tag{1.9b}$$

mit dem Ausgang $y = x_1$ und dem Eingang u oder äquivalent

$$x_1(t) = x_{1,0} + \int_0^t x_2(\tau)\mathrm{d}\tau \tag{1.10a}$$

$$x_2(t) = x_{2,0} + \int_0^t \left(-\frac{1}{T^2}x_1(\tau) - \frac{2\xi}{T}x_2(\tau) + \frac{V}{T^2}u(\tau)\right)\mathrm{d}\tau \tag{1.10b}$$

gegeben sein. Die Integraldarstellung (1.10) lässt sich leicht als Blockschaltbild in Simulink, wie in Abbildung 1.1 gezeigt, realisieren.

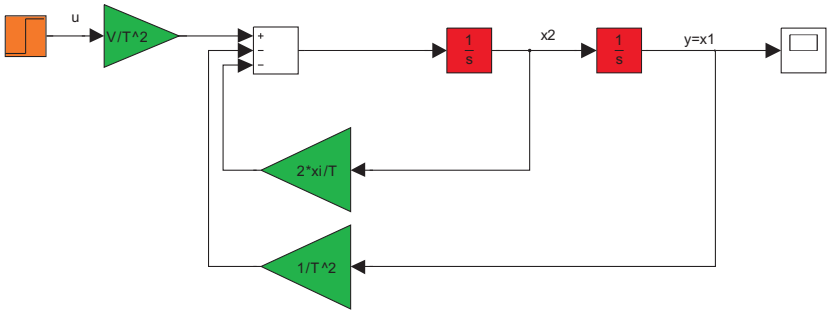


Abbildung 1.1: Implementierung des P-T_2-Gliedes in Simulink.

Aufgabe 1.3. Betrachten Sie für die nachstehenden Teilaufgaben die nichtlineare Differentialgleichung

$$\dddot{x} + \cos(x)^2\ddot{x} + \dot{x} + \mathrm{e}^{-x}u = 0\ . \tag{1.11}$$

1. Ermitteln Sie die numerische Lösung der Differentialgleichung mit dem Zustand x, dem Eingang u und den Anfangsbedingungen $\ddot{x}(0) = \dot{x}(0) = x(0) = 0$. Formen Sie dazu die Differentialgleichung in ein Differentialgleichungssystem erster Ordnung um und implementieren Sie das entsprechende Blockschaltbild. Testen Sie die Simulation für $u(t) = \sin(t)$.

2. Linearisieren Sie das Differentialgleichungssystem erster Ordnung um die Ruhe-

lage $x_R = 0$, $u_R = 0$ und überprüfen Sie die Stabilität des linearen Systems.

Control System Toolbox in Simulink

Eine einfache Möglichkeit der Implementierung linearer zeitinvarianter Systeme bietet die Control System Toolbox. Diese kann für zeitkontinuierliche wie auch zeitdiskrete LTI-Systeme verwendet werden. Dabei können die vordefinierten dynamischen Systeme (siehe auch Abschnitt 1.1.2) sowohl als Übertragungsfunktion, als auch direkt in Zustandsraumdarstellung

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{1.12a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{1.12b}$$

angegeben werden. Es bietet sich dabei insbesondere der Block '`LTI System`' aus der Control System Toolbox an, in welchem das System in beliebiger Darstellung der Control System Toolbox eingetragen werden kann.

S-function

Sollen umfangreichere Systeme simuliert werden, geht die Übersichtlichkeit der in Form von Blockschaltbildern implementierten Modelle relativ rasch verloren. Hier kann die Verwendung von S-functions Abhilfe schaffen, wobei im Rahmen dieser Lehrveranstaltung ausschließlich so genannte `Level 2 MATLAB S-functions` zur Anwendung kommen.

Das in Zustandsraumdarstellung gegebene System

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -x_2 + \sin(u_1) \\ a x_1 + b u_2 \end{bmatrix} \tag{1.13}$$

mit dem Eingangsvektor $\mathbf{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^{\mathrm{T}}$, den Parametern a und b, den Zuständen $\mathbf{x}$ und dem Ausgangsvektor $\mathbf{y}$ wurde in einer S-function in der Beispieldatei `test_sfunc_m.m` implementiert. Die Ausgänge sind durch

$$\mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ e^{x_1}\cosh(x_2) \end{bmatrix} \tag{1.14}$$

gegeben. Das zentrale Element einer `Level 2 MATLAB S-function` ist ein run-time Objekt – eine Instanz der Klasse `Simulink.MSFcnRunTimeBlock`. Das Objekt wird üblicherweise als `block` bezeichnet. Es stellt für die Simulation wichtige Attribute und Methoden zur Verfügung, auf die mithilfe des Punkt-Operators zugegriffen werden kann. Die wichtigsten Methoden sind in Tabelle 1.1 zusammengefasst und erläutert.

Zusätzlich können einer S-function Parameterwerte, z. B. physikalische Parameter des Systems, übergeben werden. Damit ist die Änderung von Parametern direkt in Simulink bzw. von Matlab aus möglich, ohne die S-function selbst zu verändern.

Typischerweise enthält eine S-function für dynamische Systeme die folgenden Funktionen (siehe auch das Beispiel `test_sfunc_m.m`).

Bezeichnung Methode	Erklärung
`block.InputPort`	Liefert die Eingänge $\mathbf{u}$
`block.ContStates`	Liefert die Zustände $\mathbf{x}$
`block.Derivatives`	Liefert die zeitlichen Ableitungen der Zustände $\dot{\mathbf{x}}$
`block.OutputPort`	Liefert die Ausgänge $\mathbf{y}$
`block.DialogPrm`	Liefert die Parameter des Systems

Tabelle 1.1: Die wichtigsten Methoden des Objektes `block`.

1. `function setup(block)`
   In dieser Funktion wird das run-time Objekt `block` initialisiert. Dabei wird z. B. die Länge der Eingangs-, Zustands-, Parameter- und Ausgangsvektoren festgelegt. Es ist möglich, Eingänge und Ausgänge zu so genannten Ports zu gruppieren. In der Beispieldatei `test_sfunc_m.m` wurden zwei Ausgangsports festgelegt, wobei Port 1 die Ausgangssignale x_1 und x_2 zusammenfasst und Port 3 den dritten Ausgang enthält, welcher durch die Funktion $e^{x_1}\cosh(x_2)$ gegeben ist. Die beiden Eingänge werden jeweils über separate Ports an die S-function übergeben. Der Parametervektor ist kein herkömmlicher MATLAB-Vektor, da dessen Elemente unterschiedliche Datentypen aufweisen dürfen. In der Beispieldatei ist das dritte Element ein Vektor. Sollen sehr viele Parameter übergeben werden, ist die Verwendung von `Structures` zu empfehlen. Die verwendeten Feldnamen stehen dann auch in der S-function zur Verfügung, womit der Verwechslung von Parametern vorgebeugt werden kann.

2. `function InitConditions(block)`
   In dieser Funktion werden den Zuständen die Anfangswerte zugewiesen. Günstigerweise werden diese im Parametervektor an die S-function übergeben.

3. `function Output(block)`
   Berechnung der Ausgangsgleichung

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}). \tag{1.15}$$

4. `function Derivatives(block)`
   Diese Funktion kommt nur bei zeitkontinuierlichen dynamischen Systemen vor. Es werden die Ableitungen berechnet, d. h. die Differentialgleichung wird in der Form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1.16}$$

   eingegeben.

5. `function Update(block)`
   Diese Funktion kommt bei zeitdiskreten dynamischen Systemen vor. Es wird die Differenzengleichung in der Form

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{1.17}$$

implementiert.

Die obigen Funktionen werden zur Laufzeit der Simulation von SIMULINK aufgerufen – zum Teil auch mehrmals, wie z. B. die Funktionen `Output(block)` und `Derivatives(block)` oder `Update(block)`. Bei den Funktionen in der Beispieldatei `test_sfunc_m.m` ist zu beachten, dass meist neue Variablen für die Attribute des Objektes `block` eingeführt wurden. Dies dient der besseren Lesbarkeit der Programme und kann den Schreibaufwand verringern.

Beachten Sie im Zusammenhang mit S-functions folgende Hinweise:

- Der Name des `m`-files, welches die S-function beinhaltet, muss mit dem S-function Namen (im vorliegenden Fall: `test_sfunc_m`) identisch sein.
- Das SIMULINK-Modell (`*.slx`) und die `S-function` (`*.m`) müssen verschiedene Namen haben, sich aber im selben Verzeichnis befinden.
- Verwenden Sie in SIMULINK den Block `User-Defined Functions` → `Level 2 MATLAB S-Function`.
- S-function-Blöcke können maskiert werden, so dass man die Parameter direkt in einer Maske eingeben kann (siehe Beispiel `sim_test_sfunc.slx`).

1.3 Anwendungsbeispiele

In diesem Abschnitt wird MATLAB zur Simulation und Analyse konkreter Anwendungsbeispiele genutzt. Die folgenden Beispiele sind so konzipiert, dass die Herleitung der mathematischen Modelle händisch durchgeführt werden kann.

1.3.1 Elektrisches Netzwerk

In der folgenden Aufgabe soll das elektrische System aus Abbildung 1.2 untersucht werden. Das betrachtete Netzwerk besteht aus einem Operationsverstärker, einer nichtlinearen Kapazität $C_1(U_{C1})$, einer linearen Kapazität C_2 sowie den linearen Widerständen R_1, R_2 und R_3. Weiterhin bezeichnet U_s eine auftretende Störung.

Aufgabe 1.4 (Modellierung).

1. Leiten Sie unter der Annahme eines idealen Operationsverstärkers das mathematische Modell des elektrischen Netzwerks aus Abbildung 1.2 in der Form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, d) \tag{1.18a}$$
$$y = \mathbf{h}(\mathbf{x}, u, d) \tag{1.18b}$$

mit der Spannung U_e als Eingang u bzw. der Spannung U_s als Störeingang d sowie der Spannung U_a als Ausgang y her. Wählen Sie dazu die beiden

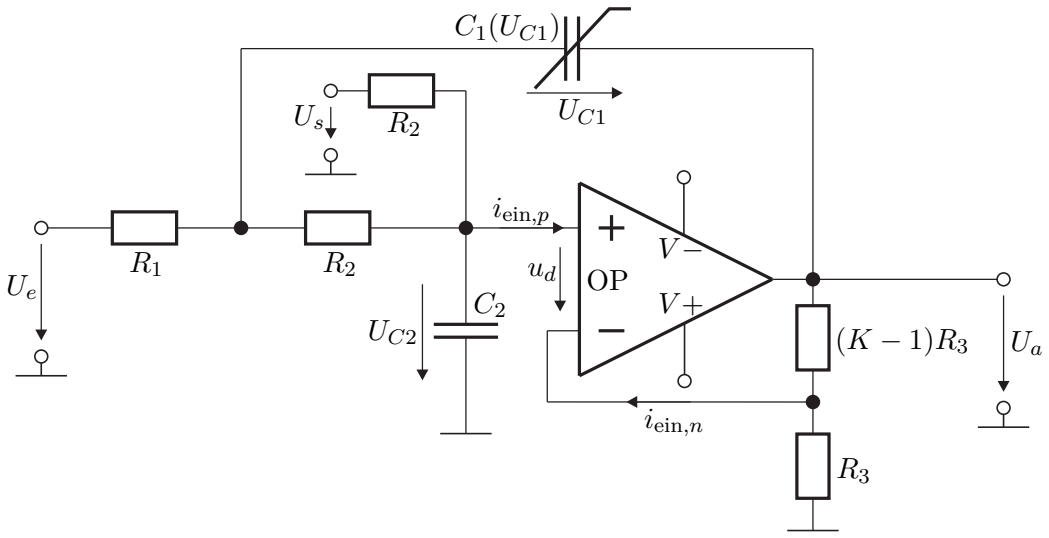


Abbildung 1.2: Elektrisches System.

Kondensatorspannungen als Systemzustände $\mathbf{x} = \begin{bmatrix} U_{C1} & U_{C2} \end{bmatrix}^{\mathrm{T}}$. Setzen Sie für die in der Kapazität C_1 gespeicherte Ladung zunächst den allgemeinen Ausdruck $Q_1(U_{C1})$ an, womit

$$C_1(U_{C1}) = \frac{\mathrm{d}Q_1}{\mathrm{d}U_{C1}}$$

gilt und $C_1(U_{C1})$ als differentielle Kapazität zu verstehen ist. Beachten Sie weiters die eingetragenen Pfeilrichtungen.

Hinweis: Unter einem idealen Operationsverstärker versteht man einen Verstärker, bei dem die beiden Eingangsströme $i_{\text{ein},p}$ und $i_{\text{ein},n}$ sowie die Differenzspannung u_d gleich Null sind.

2. Linearisieren Sie das System (1.18) für den stationären Eingang $u_R = U_{e,R}$ und den stationären Störeingang $d_R = U_{s,R}$ um eine allgemeine Ruhelage $\mathbf{x}_R = \begin{bmatrix} U_{C1,R} & U_{C2,R} \end{bmatrix}^{\mathrm{T}}$. Stellen Sie das linearisierte System in der Form

$$\Delta\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{b}_u\Delta u + \mathbf{b}_d\Delta d \tag{1.19a}$$

$$\Delta y = \mathbf{c}^{\mathrm{T}}\Delta\mathbf{x} + d_u\Delta u + d_d\Delta d \tag{1.19b}$$

mit $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_R$, $\Delta u = u - u_R$, $\Delta d = d - d_R$ und $\Delta y = y - y_R$ dar.

3. Berechnen Sie die Ruhelagen $\mathbf{x}_R = \begin{bmatrix} U_{C1,R} & U_{C2,R} \end{bmatrix}^{\mathrm{T}}$ des Systems (1.18) für den stationären Eingang $u_R = U_{e,R}$ und den stationären Störeingang $d_R = U_{s,R}$.

Hinweis: Laden Sie das zip-Archiv `uebung1.zip` von der Homepage der Lehrveranstaltung herunter. Das M-file `Esys_Parameter.m` stellt ein Grundgerüst für die Lösung der nachfolgenden Aufgaben dar.

4. Bestimmen Sie für das linearisierte System (1.19) in MATLAB die Übertragungsfunktionen $G(s)$ vom Eingang Δu zum Ausgang Δy und $G_d(s)$ von der Störung Δd zum Ausgang Δy. Spezifizieren Sie dazu das nichtlineare Verhalten von $C_1(U_{C1})$ mittels

$$Q_1(U_{C1}) = \left(C_{1,ref} + k_{C1}\left(\frac{U_{C1}}{2} - U_{C1,ref}\right)\right)U_{C1} \tag{1.20}$$

   sowie die Ruhelage durch $U_{e,R} = 5\,\mathrm{V}$ und $U_{s,R} = 5\,\mathrm{V}$. Wählen Sie die Parameter $C_{1,ref} = 1\,\mu\mathrm{F}$, $U_{C1,ref} = -10\,\mathrm{V}$, $k_{C1} = 800\,\mathrm{nFV}^{-1}$, $C_2 = 1\,\mu\mathrm{F}$, $R_1 = 625\,\Omega$, $R_2 = 3500\,\Omega$, $K = 3$.

5. Bei der Übertragungsfunktion $G(s)$ aus Punkt 4 handelt es sich um ein Verzögerungsglied 2-ter Ordnung (P-T$_2$) in der Form $G(s) = \dfrac{V}{1 + 2\xi(sT) + (sT)^2}$. Bestimmen Sie den Verstärkungsfaktor V, den Dämpfungsgrad ξ und die Zeitkonstante T.

6. Zeichnen Sie in MATLAB die Bode-Diagramme der beiden Übertragungsfunktionen und interpretieren Sie diese.

Hinweis: Zur Überprüfung Ihrer Ergebnisse sind an dieser Stelle die numerische Dynamikmatrix und die zugehörigen Eingangsvektoren des Systems (1.19) angegeben

$$\mathbf{A} = \begin{bmatrix} -1885.71 & -5371.43 \\ 285.71 & 285.71 \end{bmatrix}, \quad \mathbf{b}_u = \begin{bmatrix} 1600 \\ 0 \end{bmatrix}, \quad \mathbf{b}_d = \begin{bmatrix} 0 \\ 285.71 \end{bmatrix}.$$

Eine positive Beurteilung setzt voraus, dass Ihre Werte mit den oben angeführten Werten übereinstimmen!

Aufgabe 1.5 (Implementierung in MATLAB/SIMULINK). Das nichtlineare Modell (1.18) soll nun als S-function und das linearisierte Modell (1.19) als `State-Space`-Block implementiert werden. Verwenden Sie hierzu das M-file `Esys_S_m.m`, die Simulationsdatei `Esys.slx` sowie das bereits in Aufgabe 1.4 bearbeitete M-file `Esys_Parameter.m` aus dem zip-Archiv `uebung1.zip`. Verwenden Sie weiterhin die Parameter aus Punkt 4 von Aufgabe 1.4.

1. Öffnen Sie das M-file `Esys_S_m.m` und vervollständigen Sie die Methoden `Output` und `Derivatives` der darin enthaltenen S-function. Unvollständige Einträge

sind durch den den Kommentar `TODO` gekennzeichnet. Führen Sie anschließend die Simulation `Esys.slx` aus und überzeugen Sie sich von der Lauffähigkeit und Plausibilität der fertiggestellten S-function.

2. Das Simulationsmodell `Esys.slx` enthält bereits eine Grundstruktur für die Implementierung des linearisierten Modells (1.19) als `State-Space`-Block. Vervollständigen Sie alle durch `TODO` gekennzeichneten Blöcke. Das Ziel ist es die Zustände $\mathbf{x}$ und den Ausgang y des nichtlinearen System mit den aus dem linearisierten Modell abgeleiteten Größen $\mathbf{x}_R + \Delta\mathbf{x}$ und $y_R + \Delta y$ in den vorbereiteten `Scopes` zu vergleichen.

3. Im Folgenden soll das Verhalten des Systems untersucht werden. Im Simulationsmodell `Esys.slx` stehen Ihnen eine sprungförmige und eine sinusförmige Eingangsgröße $\Delta U_e(t)$ ($U_e(t) = \Delta U_e(t) + U_{e,R}$) zur Verfügung. Variieren Sie Winkelfrequenz und Amplitude der sinusförmigen Eingangsgröße z. B. gemäß der Folgen $[10\,\mathrm{rad/s},\ 10^2\,\mathrm{rad/s},\ 10^3\,\mathrm{rad/s},\ 10^4\,\mathrm{rad/s}]$ und $[0.1\,\mathrm{V},\ 0.5\,\mathrm{V},\ 1\,\mathrm{V}]$. Welches Systemverhalten können Sie hierbei erkennen? Wie verhält sich das System bei Aufschaltung einer sprungförmigen Störung? Wie wirkt sich die nichtlineare Kapazität auf das dynamische Systemverhalten aus? In welcher Weise beeinflusst sie das stationäre Systemverhalten? Dokumentieren und diskutieren Sie ihre Simulationsergebnisse.

1.3.2 Datenkompression

Das in Abbildung 1.3 dargestellte System zur Kompression von Datenströmen wird durch einen Speicher und eine Prozessoreinheit beschrieben. Der eingehende Datenstrom d_{in} wird im Speicher gepuffert, aus dem der Prozessor Daten entnimmt, komprimiert und als Datenstrom d_c an den Speicher zurückgibt. Von dort werden diese mit der Datenrate d_{out} wieder ausgegeben. Das Eingangs-Ausgangsverhalten des Prozessors wird durch ein PT_1-Glied mit der Kompressionsrate $c = 0.5$ und der Zeitkonstanten $T_1 = 1\,\mathrm{ms}$ angegeben. Der Datenstrom d_p am Eingang des Prozessors sei abhängig von der aktuellen Datenmenge S im Speicher und wird durch $d_p = aS$ mit $a = 0.4\,\mathrm{s}^{-1}$ definiert.

Aufgabe 1.6.

1. Schreiben Sie das dynamische Verhalten des Prozessors durch ein zeitdiskretes, lineares System mit dem Eingang $u_k = d_p$, dem Ausgang $y_k = d_c$ in Zustandsraumdarstellung an. Wählen Sie als Abtastzeit $T_a = 1\,\mu\mathrm{s}$.

2. Stellen Sie das Gesamtsystem als zeitdiskretes, lineares System mit dem Eingang $u_k = d_{in} - d_{out}$, dem Ausgang $y_k = S$ sowie geeigneten Zuständen dar.

3. Implementieren Sie das dynamische System der Datenkompression in MATLAB/SIMULINK als Blockschaltbild. Realisieren Sie dabei den Prozessors als **Zustandsraumdarstellung** in einem LTI-Systems-Block.

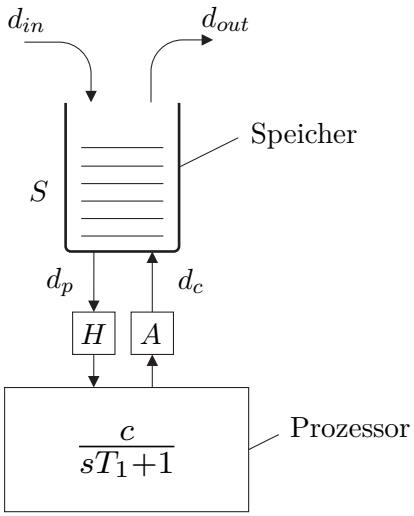


Abbildung 1.3: Datenkompression.

4. Berechnen Sie, welche Datenmenge S_R sich im Speicher einstellt, wenn am Eingang ein konstanter Datenstrom $u_R = 1$ anliegt.

5. Das System befinde sich in einem Arbeitspunkt mit dem konstanten Eingang $u_R = 1$ und der Datenmenge S_R. Berechnen Sie wie groß der Speicher ausgelegt werden muss, damit dieser nicht überläuft, wenn am Eingang für den Zeitraum von 5 ms die Datenrate sprungförmig auf $u_R = 1000$ ansteigt und danach wieder auf u_R zurückgeht? Überprüfen Sie ihr Ergebnis durch Simulation, indem Sie den Sprung der Eingangsdatenmenge zum Zeitpunkt $t = 10$ ms aufschalten.

   Hinweis: Beachten Sie die korrekte Wahl des Anfangszustandes.

1.4 Literatur

[1.1] A. Kugi, *Skriptum zur VU Automatisierung (WS 2016/2017)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2016.

[1.2] A. Angermann, M. Beuschel, M. Rau und U. Wohlfarth, *Matlab - Simulink - Stateflow, Grundlagen, Toolboxen, Beispiele*. München: Oldenbourg Verlag, 2005.

[1.3] H. Schwarz, *Numerische Mathematik*. Stuttgart: B.G. Teubner, 1997.

2 Reglerentwurf und Simulation

Ziel dieser Übung ist der Entwurf einer geeigneten Regelung für dynamische Systeme sowie der Verifikation dieser Regelung mittels Simulation. Dies wird an einer Operationsverstärkerschaltung und an einer Gleichstrommaschine mit Propeller durchgeführt.

Der Reglerentwurf soll mittels Frequenzkennlinienverfahren (FKL) sowohl im s-Bereich als auch im q-Bereich erfolgen. Die für die Zeitdiskretisierung und den Reglerentwurf nach dem Frequenzkennlinienverfahren benötigten Grundlagen wurden in der VU Automatisierung vorgestellt. Studieren Sie daher zur Vorbereitung dieser Übung das folgende Skriptum:

- Skriptum zur VU Automatisierung (WS 2016/17) [2.1]
  - Kapitel 1 bis Kapitel 6

Bei Fragen oder Anregungen zu dieser Übung wenden Sie sich bitte an

- Stefan Flixeder <flixeder@acin.tuwien.ac.at> oder
- Bernhard Bischof <bischof@acin.tuwien.ac.at>.

Für organisatorische Fragen wenden Sie sich bitte an

- Christian Hartl <hartl@acin.tuwien.ac.at>.

2.1 Elektrisches System

In der folgenden Aufgabe soll das elektrische System aus der ersten Übung (Abbildung 2.1) untersucht werden.

Das betrachtete Netzwerk besteht aus einem Operationsverstärker, den Kapazitäten C_1 und C_2 sowie den Widerständen R_1, R_2 und R_3. Weiters, sei die Spannung U_e der Eingang u, die Spannung U_s ein Störeingang d und die Spannung U_a der Ausgang y des Systems. Die Führungsübertragungsfunktion des Netzwerks ist durch

$$G(s) = \frac{V}{1 + 2\xi(sT) + (sT)^2}$$

mit $V = 1.377$, $\xi = 0.802$ und $T = 1.002\,\text{ms}$ gegeben.

Aufgabe 2.1 (Reglerentwurf). In dieser Aufgabe soll ein zeitkontinuierlicher Regler für das elektrische System nach dem Frequenzkennlinienverfahren entworfen werden. Der geschlossene Kreis soll dabei folgende Spezifikationen erfüllen:

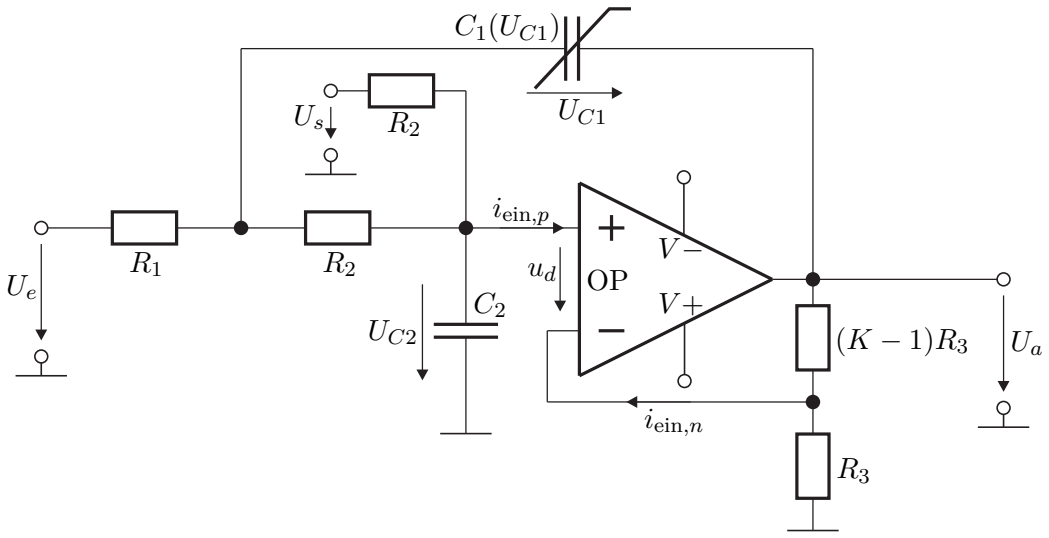


Abbildung 2.1: Elektrisches System.

bleibende Regelabweichung	$e_\infty\|_{r(t)=\sigma(t)} = 0$
Überschwingen	$\ddot{u} \leq 5\,\%$
Anstiegszeit	$t_r = 3\,\text{ms}$

1. Entwerfen Sie in MATLAB einen Regler nach dem Frequenzkennlinienverfahren, der obige Anforderungen erfüllt.

2. Ist das Frequenzkennlinienverfahren ein exaktes Entwurfsverfahren, wenn – wie im vorliegenden Fall – die Strecke ein Verzögerungsglied 2-ter Ordnung ist?

3. Im Weiteren soll zusätzlich zu den oben angeführten Anforderungen die bleibende Regelabweichung
$$e_\infty|_{r(t)=t} = 1 \cdot 10^{-3}$$
gelten. Entwerfen Sie hierfür einen PID-Regler, wobei die Zeitkonstante des Integralterms $T_I = 1\,\text{ms}$ betragen soll.

4. Stellen Sie den geschlossenen Kreis mit dem linearisierten System in Form eines Blockschaltbildes dar. Berechnen Sie die Führungsübertragungsfunktion $T_{r,y}(s)$ und die Störübertragungsfunktion $T_{d,y}(s)$ des geschlossenen Kreises und die zugehörigen Sprungantworten in MATLAB.

5. Implementieren Sie beide Regler im SIMULINK-Modell `E_Netzwerk.slx`, welches in der Datei `uebung2.zip` von der Homepage der Lehrveranstaltung zur Verfügung gestellt wird, und testen Sie diese am linearisierten sowie am vollständigen nichtlinearen Modell. Ist der geschlossene Regelkreis stabil? Begründen Sie Ihre Antwort. Simulieren Sie das Verhalten des geschlossenen Kreises für eine

sprungförmige Eingangsgröße mit und ohne Störung. Werden die Anforderungen erfüllt?

6. Bisher wurde davon ausgegangen, dass dem Regler ein idealer Messwert der Ausgangsgröße y zur Verfügung steht. Da dies in der Realität nicht der Fall ist, ist es sinnvoll in einer Simulation den Einfluss von Messrauschen zu untersuchen. Dazu kann in der Simulation mit einem Schalter zusätzliches Messrauschen auf den Systemausgang aufgeschalten werden. Testen Sie Ihren Regler in der Simulation mit und ohne Messrauschen. Wie verhält sich das System mit Messrauschen im Vergleich zum idealen System?

7. Im Simulationsmodel steht ein weiterer Schalter zur Verfügung, der die zeitliche Änderungsrate des Sollwertes begrenzt. Welche Auswirkung hat die Steigungsbeschränkung auf die Stellgröße und den Ausgang? Welcher Vorteil kann sich aus der Begrenzung der Änderungsrate ergeben?

2.2 Gleichstrommaschine mit Propeller

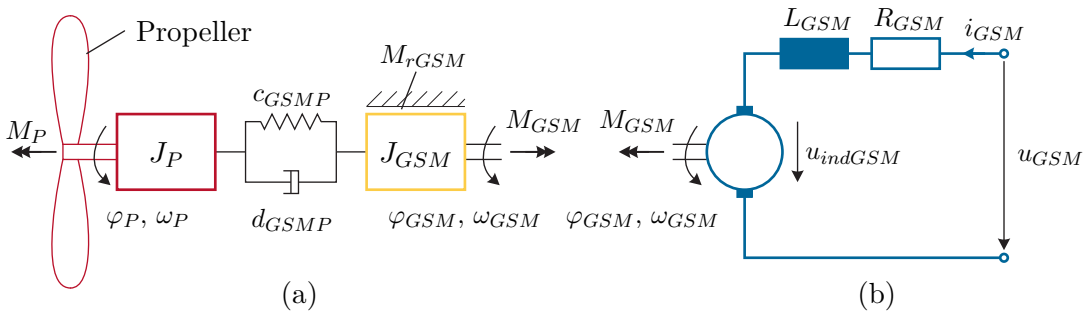


Abbildung 2.2: Gleichstrommaschine mit Propeller, (a) mechanisches Teilsystem, (b) elektrisches Teilsystem.

Abbildung 2.2 zeigt schematisch eine permanenterregte Gleichstrommaschine (Index GSM), die über eine linear elastische und dämpfende Welle (konstante Steifigkeit c_{GSMP}, viskose Dämpfung d_{GSMP}) einen Propeller (Index P) antreibt. Die Freiheitsgrade $\mathbf{q}$ des Systems sind die Drehwinkel $\mathbf{q}^{\mathrm{T}} = \begin{bmatrix}\varphi_{GSM} & \varphi_P\end{bmatrix}$. Die zugehörigen Winkelgeschwindigkeiten werden mit $\dot{\mathbf{q}}^{\mathrm{T}} = \begin{bmatrix}\omega_{GSM} & \omega_P\end{bmatrix}$ benannt. Im Folgenden wird stets von $\omega_{GSM} > 0$ und $\omega_P > 0$ ausgegangen, so dass Haftreibungseffekte beim Nulldurchgang der Geschwindigkeit unberücksichtigt bleiben können.

Auf den Propeller (Massenträgheitsmoment J_P) wirkt ein Lastmoment der Form

$$M_P = d_{cP} + d_{vP}\omega_P + d_{qP}\omega_P^2 + M_{ext} \ , \tag{2.1}$$

wobei d_{cP} die Coulombsche Reibkonstante, d_{vP} die viskose Dämpfungskonstante, d_{qP} der Koeffizient des zum Geschwindigkeitsquadrat proportionalen Dämpfungsanteils und M_{ext}

ein zusätzliches externes Moment ist. Die Lagerung des Ankers verursacht ein Reibmoment der Form

$$M_{rGSM} = d_{cGSM} + d_{vGSM}\omega_{GSM} \tag{2.2}$$

mit der Coulombschen Reibkonstante d_{cGSM} und der viskosen Dämpfungskonstante d_{vGSM}. Das Kopplungsmoment M_{kopp} zwischen Motor und Propeller, also das über die Feder c_{GSMP} und den Dämpfer d_{GSMP} übertragene Moment, berechnet sich zu

$$M_{kopp} = (\omega_{GSM} - \omega_P)d_{GSMP} + (\varphi_{GSM} - \varphi_P)c_{GSMP} \ .$$

Das von der (idealen) Gleichstrommaschine erzeugte elektrische Moment ist $M_{GSM} = k_{GSM}i_{GSM}$, wobei k_{GSM} die Ankerkreiskonstante bezeichnet, die aus der Maschinenkonstante c_A und dem verketteten Fluss der Erregerwicklung Ψ_{EGSM} in der Form $k_{GSM} = c_A\Psi_{EGSM}$ berechnet wird. Aufgrund der Impulserhaltung ergeben sich die Bewegungsgleichungen zu

$$J_{GSM}\dot{\omega}_{GSM} = M_{GSM} - M_{rGSM} - M_{kopp} \tag{2.3a}$$

$$J_P\dot{\omega}_P = M_{kopp} - M_P \ . \tag{2.3b}$$

Dabei bezeichnet J_{GSM} das Massenträgheitsmoment der Gleichstrommaschine und J_P das Massenträgheitsmoment des Propellers.

Die Differentialgleichung für das in Abbildung 2.2(b) dargestellte elektrische Subsystem wird mithilfe der Maschenregel bestimmt. Für die induzierte Spannung gilt $u_{indGSM} = k_{GSM}\omega_{GSM}$. Die Ankerkreisinduktivität wird mit L_{GSM}, der Ankerkreiswiderstand mit R_{GSM} und die Eingangsspannung mit u_{GSM} bezeichnet. Damit erhält man

$$\frac{\mathrm{d}}{\mathrm{d}t}i_{GSM} = \frac{1}{L_{GSM}}(u_{GSM} - R_{GSM}i_{GSM} - k_{GSM}\omega_{GSM}) \ . \tag{2.4}$$

Somit kann das vollständige, nichtlineare mathematische Modell der Gleichstrommaschine in der Form

$$\dot{\mathbf{x}}_m = \mathbf{f}_m(\mathbf{x}_m, u_m, d_m) \tag{2.5}$$

mit den Zustandsgrößen $\mathbf{x}_m^{\mathrm{T}} = \begin{bmatrix} i_{GSM} & \varphi_{GSM} & \omega_{GSM} & \varphi_P & \omega_P \end{bmatrix}$, dem Eingang $u_m = u_{GSM}$ und der Störung $d_m = M_{ext}$ dargestellt werden. Als Ausgang des Systems wird $y = \omega_P$ verwendet.

Aufgabe 2.2 (Vollständiges Modell).

1. Im System (2.5) treten die Größen φ_{GSM} und φ_P stets in Form der Differenz $(\varphi_{GSM} - \varphi_P) = \varphi_{GSMP}$ auf. Mithilfe der nichtregulären Zustandstransformation

$$\mathbf{x}_M = \begin{bmatrix} i_{GSM} \\ \varphi_{GSMP} \\ \omega_{GSM} \\ \omega_P \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_m \tag{2.6}$$

kann daher eine Differentialgleichung eingespart werden. Führen Sie diese

Transformation durch, d. h. bestimmen Sie

$$\dot{\mathbf{x}}_M = \mathbf{f}_M(\mathbf{x}_M, u_M, d_M) \ , \tag{2.7}$$

wobei $u_M = u_m$ und $d_M = d_m$ gelten soll.

2. Bestimmen Sie für stationäre Eingangswerte die Ruhelage des Systems (2.7), linearisieren Sie es bezüglich derselben und stellen Sie es in der Form

$$\Delta\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{b}_u\Delta u + \mathbf{b}_d\Delta d \tag{2.8a}$$
$$\Delta y = \mathbf{c}^{\mathrm{T}}\Delta\mathbf{x} \tag{2.8b}$$

dar. Berechnen Sie weiterhin numerisch die Eigenwerte der Dynamikmatrix des linearisierten Systems in MATLAB, wobei die Parameterwerte aus Tabelle 2.1 und für die stationären Eingangsgrößen $u_{GSM,R} = 5.6\,\mathrm{V}$ und $M_{ext,R} = 0\,\mathrm{N\,m}$ zu verwenden sind.

Parameter	Wert	
L_{GSM}	1.4	mH
R_{GSM}	0.46	Ω
k_{GSM}	0.1	$\mathrm{N\,m\,A^{-1}}$
J_{GSM}	$12.4 \cdot 10^{-3}$	$\mathrm{kg\,m^2}$
d_{cGSM}	0.152	N m
d_{vGSM}	$1.8 \cdot 10^{-3}$	$\mathrm{N\,m\,s\,rad^{-1}}$
J_P	$32.5 \cdot 10^{-3}$	$\mathrm{kg\,m^2}$
d_{cP}	0.169	N m
d_{vP}	$2.7 \cdot 10^{-3}$	$\mathrm{N\,m\,s\,rad^{-1}}$
d_{qP}	$1 \cdot 10^{-4}$	$\mathrm{N\,m\,s^2\,rad^{-2}}$
c_{GSMP}	0.6822	$\mathrm{N\,m\,rad^{-1}}$
d_{GSMP}	$1 \cdot 10^{-5}$	$\mathrm{N\,m\,s\,rad^{-1}}$

Tabelle 2.1: Parameter des Systems Gleichstrommaschine mit Propeller.

Bestimmt man die Eigenwerte λ_i mit $i = 1, \ldots, 4$ der Dynamikmatrix $\mathbf{A}$ des um die Ruhelage linearisierten Systems (2.8), so ergeben sich diese in aufsteigend sortierter Reihenfolge zu $\lambda_1 = -326.809\,\mathrm{s^{-1}}$, $\lambda_{2,3} = -0.727\,\mathrm{s^{-1}} \pm \mathrm{I}\,8.674\,\mathrm{s^{-1}}$ und $\lambda_4 = -0.727\,\mathrm{s^{-1}}$. Offensichtlich ist das System (lokal) asymptotisch stabil. Auffällig ist jedoch, dass

$$|\lambda_1| \gg |\lambda_i| \quad \forall\, i \in \{2, 3, 4\} \ , \tag{2.9}$$

d. h. der Eigenwert λ_1 liegt in der komplexen Ebene sehr viel weiter links als die übrigen

Eigenwerte, was eine relativ *schnelle* Dynamik im zugehörigen Unterraum der Lösung nach sich zieht.

Es lässt sich nun mithilfe der *singulären Störungstheorie* [2.2] zeigen, dass dieser Eigenwert zumindest näherungsweise der Stromdynamik zugewiesen werden darf. Somit kann in weiterer Folge die Dynamik des elektrischen Teilsystems als quasistationär betrachtet werden, womit sich die zugehörige Differentialgleichung zu einer *algebraischen* Gleichung der Form

$$i_{GSM} = \frac{1}{R_{GSM}} u_{GSM} - \frac{k_{GSM}}{R_{GSM}} \omega_{GSM} \tag{2.10}$$

reduziert.

Aufgabe 2.3 (Reduziertes System).

1. Verwenden Sie die Näherung (2.10), um im System (2.7) die zugehörige Differentialgleichung der Stromdynamik zu eliminieren. Damit erhalten Sie ein reduziertes System

$$\dot{\mathbf{x}}_{red} = \mathbf{f}_{red}(\mathbf{x}_{red}, u, d) \tag{2.11a}$$
$$y = h_{red}(\mathbf{x}_{red}) \tag{2.11b}$$

   mit dem neuen Zustandsvektor $\mathbf{x}_{red}^{\mathrm{T}} = \begin{bmatrix} \varphi_{GSMP} & \omega_{GSM} & \omega_P \end{bmatrix}$, wobei Eingang u, Störung d und Ausgang y unverändert bleiben.

2. Bestimmen Sie für stationäre Eingangswerte die Ruhelage des reduzierten Systems (2.11) und linearisieren Sie es bezüglich derselben.

3. Ist die Ruhelage des reduzierten Systems (2.11) gleich jener des vollständigen Systems (2.7)? Begründen Sie Ihre Antwort.

Aufgabe 2.4 (Implementierung in MATLAB/SIMULINK). Erstellen Sie für das System Gleichstrommaschine mit Propeller ein Simulationsmodell. Es soll das vollständige nichtlineare Modell (2.7) (inklusive Stromdynamik) in Form einer Level 2 M-Code s-function enthalten. Wählen Sie hierzu als Eingang der s-function den Vektor $\mathbf{u}^{\mathrm{T}} = \begin{bmatrix} u_{GSM} & M_{ext} \end{bmatrix}$ und als Ausgangsvektoren $\mathbf{y}_1^{\mathrm{T}} = \begin{bmatrix} i_{GSM} & \varphi_{GSMP} & \omega_{GSM} & \omega_P \end{bmatrix}$ bzw. $\mathbf{y}_2^{\mathrm{T}} = \begin{bmatrix} M_{GSM} & M_{Kopp} \end{bmatrix}$. Verwenden Sie als Anfangszustand die Ruhelage für $u_{GSM} = 5.6\,\mathrm{V}$ und $M_{ext} = 0\,\mathrm{N\,m}$.

Implementieren Sie außerdem das vollständige linearisierte Modell und das reduzierte linearisierte Modell in Form von `State-Space`-Blöcken.

Simulieren Sie alle Systeme für eine Eingangsgröße der Form $M_{ext} = 0.25\sigma(t-11)$ und $u_{GSM} = 5.6 - \sigma(t-2) + \sigma(t-5) - 2\sigma(t-8) + 2\sigma(t-13)$ (M_{ext} in $\mathrm{N\,m}$ und u_{GSM} in V). Vergleichen Sie die Ausgangsgrößen der Systeme und dokumentieren Sie Ihre Ergebnisse.

Hinweis: Beachten Sie bei der Implementierung nachfolgende Punkte:

- Alle Systemparameter und Anfangszustände sollen als Parameter von außen an die s-function übergeben werden. Definieren Sie diese Größen in einem M-file, das Sie jeweils vor dem Start der Simulation ausführen.
- Übernehmen Sie dabei die analytischen Ausdrücke der Ruhelagen und des linearisierten Modells aus Aufgabe 2.2 (Teilaufgabe 2) in besagtes M-file und berechnen Sie erst dort die numerischen Werte. Damit ist es später einfach möglich beliebige Ruhelagen zu untersuchen.

Weiterhin steht Ihnen zum Test und Abgleich ihres Modells im zip-Archiv `uebung2.zip` auf der Homepage der Lehrveranstaltung die Datei `GSM_Student_S_m.p` zum Download zur Verfügung. Es handelt sich um eine chiffrierte s-function der Gleichstrommaschine. Die Einbindung erfolgt analog zur Level 2 M-file s-function und ist im Simulationsmodell `Simulation_GSM_out.mdl` gezeigt. Das Modell enthält bereits die oben genannten Verläufe der Eingangsgrößen. Lediglich die berechnete Ruhelage $\mathbf{x}_0$ aus der Aufgabe 2.2 (Teilaufgabe 2) ist in der Maske der s-function des vollständigen Modells (2.7) zu ergänzen. Um Namenskonflikte zu vermeiden, darf die von Ihnen erstellte s-function nicht den Namen `GSM_Student_S_m.m` tragen.

Aufgabe 2.5 (Reglerentwurf). Entwerfen Sie mithilfe des Frequenzkennlinienverfahrens im q-Bereich einen zeitdiskreten Kompensationsregler für das reduzierte linearisierte Modell der Gleichstrommaschine mit Propeller mit der Winkelgeschwindigkeit $\Delta\omega_P$ als Ausgang. Als Arbeitspunkt für den Betrieb des Reglers wählen Sie die von Ihnen berechnete Ruhelage aus Aufgabe 2.3 (Teilaufgabe 2) für $u_{GSM,R} = 5.6\,\mathrm{V}$ und $M_{ext,R} = 0\,\mathrm{N\,m}$. Der geschlossene Kreis soll folgende Spezifikationen für einen Sollsprung $\Delta r = 20\,\mathrm{rad\,s^{-1}}$ erfüllen:

bleibende Regelabweichung	$e_\infty\big\vert_{(r^k)=(1^k)} = 0$
Anstiegszeit	$t_r = 1\,\mathrm{s}$
Überschwingen	$\ddot{u} \leq 0\,\%$
Stellgrößenbeschränkung	$0\,\mathrm{V} \leq u_{GSM} \leq 12\,\mathrm{V}$

Als Reglerstruktur wird

$$R^{\#}(q) = \frac{V_I(1+qT_I)}{q}\,\frac{1+2\xi(qT)+(qT)^2}{\prod_{j=1}^{2}(q-q_{r,j})} \tag{2.12}$$

gewählt, wobei $1+2\xi(qT)+(qT)^2$ das konjugiert komplexe Polpaar der Streckenübertragungsfunktion als Nullstellen besitzt und $q_{r,j}$ die gewünschten Realisierungspole bezeichnen. Überlegen Sie, warum die obige Reglerstruktur gewählt wird. Sie kön-

nen dazu zusätzlich einen PI-Regler ohne Kompensationsterm entwerfen und das Verhalten der beiden Regelkreise miteinander vergleichen!

Um den Regler später am Laborversuch testen zu können, muss der zeitdiskrete Regler $R(z)$ in Proportional-, Integral- sowie Kompensationsteil aufgespalten werden,

$$R(z) = \left(P + \frac{I}{z-1}\right) R_{komp}(z) \ . \tag{2.13}$$

Sie können dazu den MATLAB-Befehl `residue` verwenden.

Hinweis: Verwenden Sie als Abtastzeit $T_a = 50$ ms. Weiterhin sei erwähnt, dass in MATLAB folgende Prozeduren für die Transformationen zwischen s-, z- und q-Bereich zur Verfügung stehen:

$G(s) \overset{T_a}{\to} G(z)$	MATLAB-Befehl:	`Gz=c2d(Gs,Ta,'zoh')`
$G(z) \to G^{\#}(q)$	MATLAB-Befehl:	`Gq=d2c(Gz,'tustin')`
$G^{\#}(q) \overset{T_a}{\to} G(z)$	MATLAB-Befehl:	`Gz=c2d(Gq,Ta,'tustin')`

Aufgabe 2.6 (Verifikation). Testen Sie durch Simulation in SIMULINK, ob der so entworfene Regelkreis die Spezifikationen auch tatsächlich erfüllt. Vergleichen Sie die Ergebnisse für die drei betrachteten Systemmodelle.

Aufgabe 2.7 (Einfluss des Messrauschens). Testen Sie Ihren Regler in der Simulation mit Messrauschen[a]. Wie verhält sich das System mit Messrauschen im Vergleich zum idealen System? Welchen Einfluss hat die Wahl der Anstiegszeit t_r auf Rauschanteile im Eingang u und im Ausgang y? Dokumentieren Sie aussagekräftige Ergebnisse für verschiedene Parameter und analysieren Sie dies formal anhand der Übertragungsfunktionen vom Sensorrauschen n zum Eingang u bzw. zum Ausgang y.

[a] Üblicherweise wird Messrauschen als bandbeschränktes, weißes Rauschen modelliert.

2.3 Literatur

[2.1] A. Kugi, *Skriptum zur VU Automatisierung (WS 2016/2017)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2016.

[2.2] P. Kokotovic, H. K. Khalil und J. O'Reilly, *Singular pertubation methods in control: Analysis and design*. USA, Philadelphia: Society for Industrial Mathematics, 1999.

3 Zustandsregler

Ziel dieser Übung ist der Entwurf einer geeigneten Regelung für ein dynamisches System. Im Unterschied zur vorherigen Übung sollen nun ein Zustandsregler für das im vorigen Kapitel eingeführte Modell einer Gleichstrommaschine entworfen werden.

Die benötigten Grundlagen wurden in der VU Automatisierung vorgestellt. Studieren Sie daher zur Vorbereitung dieser Übung folgende Unterlagen:

- Skriptum zur VU Automatisierung (WS 2016/17) [3.1]
  - Kapitel 7 bis Kapitel 8

Bei Fragen oder Anregungen zu dieser Übung wenden Sie sich bitte an

- Christian Hartl `<hartl@acin.tuwien.ac.at>` oder
- Alexander Zeiler `<zeiler@acin.tuwien.ac.at>`.

3.1 Zustandsreglerentwurf für die Gleichstrommaschine mit Propeller

Aufgabe 3.1. In dieser Aufgabe soll für die Gleichstrommaschine mit Propeller eine Drehzahlregelung mithilfe eines zeitdiskreten Zustandsreglers entworfen werden. Der Reglerentwurf erfolgt dabei am linearisierten, reduzierten Modell aus Aufgabe 2.3. Als Abtastzeit für den Regler gelte $T_a = 10$ ms. Bearbeiten Sie dazu die nachfolgenden Teilaufgaben:

1. Eine wesentliche Systemanforderung für den Zustandsreglerentwurf ist das Vorhandensein der vollständigen Erreichbarkeit. Berechnen Sie die zeitdiskrete Zustandsraumdarstellung des linearisierten, reduzierten Systemmodells aus Aufgabe 2.3 für die Abtastzeit $T_a = 10\,\text{ms}$ und weisen Sie nach, dass das System vollständig erreichbar bezüglich des Systemeingangs Δu ist [3.1].

   Hinweis: Mithilfe des MATLAB-Befehls `ctrb` können Sie die Erreichbarkeitsmatrix des zeitdiskreten Systems berechnen. Weiters existieren in MATLAB die nützlichen Befehle `ss` und `c2d`.

2. Entwerfen Sie einen zeitdiskreten Zustandsregler der Form

$$\Delta u_k = \mathbf{k}^{\mathrm{T}} \Delta \mathbf{x}_{red,k} + g \Delta r_k, \tag{3.1}$$

   mit dem die Propeller-Drehzahl $\Delta\omega_P$ der Sollgröße Δr folgt. Alle weiteren Anforderungen für den Reglerentwurf sind Aufgabe 2.5 zu entnehmen. Bestimmen

Sie den Rückführungsvektor $\mathbf{k}$ und den Parameter g so, dass der geschlossene Kreis die Pole $p_j = \exp(\lambda_0 T_a), j = 1 \dots 3$ mit einem geeigneten $\lambda_0 \in \mathbb{R}$ besitzt und begründen Sie die Wahl dieser Pole.

Hinweis: Zur Bestimmung des Rückführungsvektors $\mathbf{k}$ kann in MATLAB der Befehl `acker` verwendet werden. Achten Sie bei der Verwendung auf die vom Skriptum abweichende Vorzeichenkonvention.

Aufgabe 3.2. Implementieren Sie den Zustandsregler aus Aufgabe 3.1 in SIMULINK am linearen Entwurfsmodell. Achten Sie darauf, dass die Abtastzeit des Reglers entsprechend gesetzt ist und der Regler tatsächlich zeitdiskret berechnet wird. Betrachten Sie dazu einen Sollgrößensprung und achten Sie insbesondere auf die Einhaltung der Stellgrößenbegrenzung sowie auf das Verschwinden der bleibenden Regelabweichung. Falls Ihr Regler die gestellten Anforderungen nicht erfüllt, führen Sie den Entwurf in Aufgabe 3.1 erneut durch.

Implementieren Sie nun den Zustandsregler am nichtlinearen Modell (2.7) der Gleichstrommaschine und achten Sie auf eine korrekte Aufschaltung der Ruhelagen. Prüfen Sie erneut die Einhaltung der Anforderungen. Überlegen Sie sich, warum der Regler die Anforderung in Bezug auf die bleibende Regelabweichung prinzipiell nicht erfüllen kann.

Aufgabe 3.3. Um auch im Falle von Abweichungen der Strecke vom nominellen System sowie von auftretenden Störungen die bleibende Regelabweichung zu unterdrücken, ist in dieser Aufgabe ein zeitdiskreter PI-Zustandsregler der Form

$$x_{I,k+1} = x_{I,k} + (\Delta r_k - \underbrace{\mathbf{c}^{\mathrm{T}} \Delta \mathbf{x}_{red,k}}_{\Delta y_k}) \tag{3.2a}$$

$$\Delta u_k = \begin{bmatrix} \mathbf{k}_x^{\mathrm{T}} & k_I \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}_{red,k} \\ x_{I,k} \end{bmatrix} + k_P (\Delta r_k - \underbrace{\mathbf{c}^{\mathrm{T}} \Delta \mathbf{x}_{red,k}}_{\Delta y_k}) \tag{3.2b}$$

zu entwerfen. Für den Reglerentwurf gelten die gleichen Anforderungen wie in Aufgabe 3.1. Welche Anforderungen hinsichtlich Erreichbarkeit werden an das System gestellt und sind diese erfüllt? Orientieren Sie sich bei der Bestimmung der Reglerparameter $\mathbf{k}_x$, k_P und k_I an der Vorgehensweise im Skriptum [3.1].

Prinzipiell könnte der PI-Zustandsregler der Form (3.2) in SIMULINK als zeitdiskretes Blockdiagramm aufgebaut werden. Eine übersichtlichere Darstellung des Regelalgorithmus erhält man durch eine Realisierung in Form eines MATLAB Function Blocks. Um einen reibungslosen Ablauf der Reglerimplementierung am zugehörigen Laboraufbau zu ermöglichen, realisieren Sie Ihren Regler in Form eines MATLAB Function Blocks, bei dem Sie die folgende Schnittstelle vorsehen:

```
function du = fcn(dx, dr, sw, parZR)
```

Die Blockeingänge `dx` und `dr` bezeichnen den Zustand $\Delta\mathbf{x}_{red}$ und die Sollgröße Δr und der Blockausgang `du` die berechnete Stellgröße Δu. Achten Sie auf eine korrekte Aufschaltung der Ruhelagen *außerhalb* des Blocks. Sämtliche Reglerparameter müssen als Parameter in der Struktur `parZR` an den Block übergeben werden. Dazu muss diese im „Model Explorer“ (Rechtsklick auf den Block – „Explore“) des MATLAB Function Blocks als „Parameter“ gekennzeichnet werden, womit sie direkt aus dem MATLAB-Workspace entnommen wird. Berücksichtigen Sie weiters die zeitdiskrete Form des Zustandsreglers, indem Sie die Abtastzeit für den MATLAB Function Block vorgeben. Der Blockeingang `sw` dient zum Deaktivieren des Reglers: Gilt `sw=0`, so muss der Reglerausgang `du=0` zurückgeben. Weiters muss in diesem Fall die Integration des Fehlers angehalten werden und ein Reset auf 0 durchgeführt werden, damit der Regler bei einem erneuten Aktivieren (`sw=1`) keine unzulässigen Stellamplituden ausgibt. Verifizieren Sie den Reglerentwurf anschließend am nichtlinearen Modell (2.7) der Gleichstrommaschine.

Hinweis: Eine beispielhafte Verwendung von MATLAB Function Blöcken zeigt Ihnen das SIMULINK Modell `matlabfunction.slx`, welches Ihnen auf der Homepage in der Datei `uebung3.zip` zur Verfügung steht. Dort ist gezeigt, wie ein zeitdiskretes LTI-System mit einem MATLAB Function Block und unter Verwendung von `persistent`-Variablen simuliert werden kann. Weiters ist im „Model Explorer“ die Abtastzeit für die Ein- und Ausgänge gesetzt.

Aufgabe 3.4 (Störverhalten). Vergleichen Sie den Kompensationsregler aus Aufgabe 2.5 mit dem PI-Zustandsregler von Aufgabe 3.3 am nichtlinearen Modell (2.7).

1. Bewerten Sie das Führungsverhalten der verschiedenen Regelungsstrategien für einen Sollgrößensprung $\Delta r = 20\,\mathrm{rad\,s^{-1}}\sigma(t-1)$.
2. Untersuchen Sie das Verhalten der Regler für eine Störung der Form $M_{ext} = -0.5\,\mathrm{N\,m}\sigma(t-5)$.

3.2 Literatur

[3.1] A. Kugi, *Skriptum zur VU Automatisierung (WS 2016/2017)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2016.