



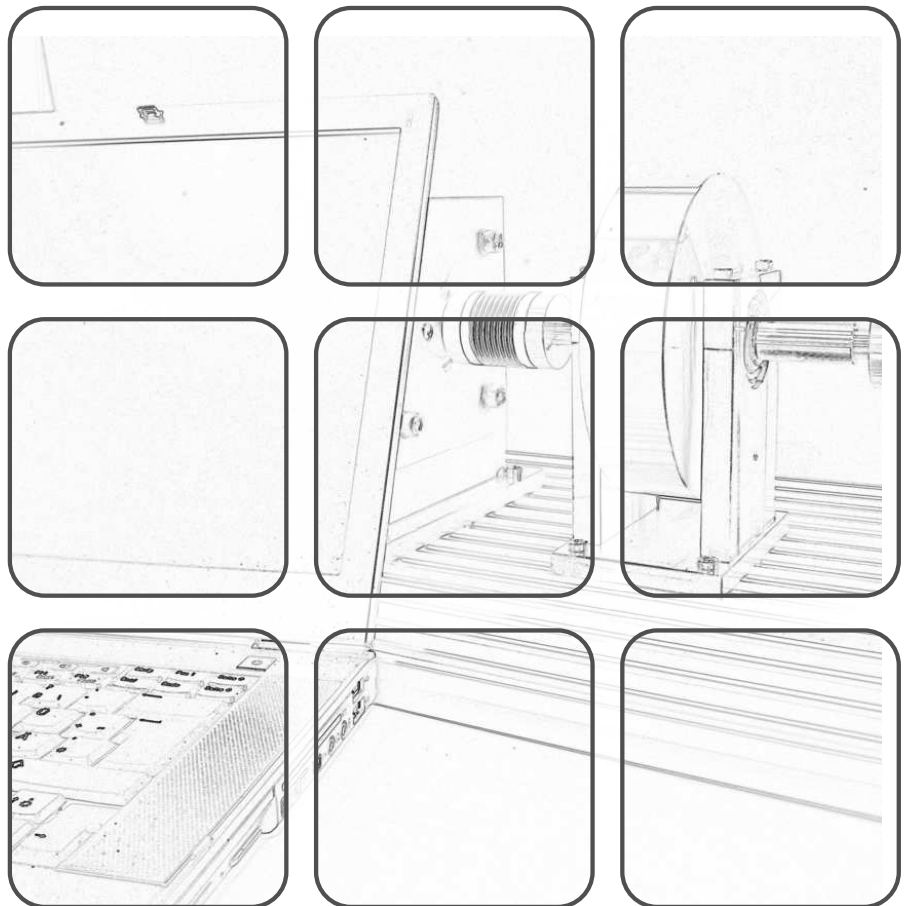
TECHNISCHE
UNIVERSITÄT
WIEN



Laborübung
WS 2020/2021

Univ.-Prof. Dr. techn. Andreas KUGI

REGELUNGSTECHNIK



Regelungstechnik

Laborübung
WS 2020/2021

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: <https://www.acin.tuwien.ac.at>

© Institut für Automatisierungs- und Regelungstechnik, TU Wien

Inhaltsverzeichnis

0	Organisation	1
0.1	Inhalt	1
0.2	Ablauf	1
0.2.1	Vorbereitung	1
0.2.2	Laborübung	1
0.2.3	Anforderungen und Beurteilung	2
0.3	Termine Wintersemester 2020/21	2
0.4	Ansprechperson für organisatorische Belange	3
0.5	Weitere Informationen	3
1	Systemanalyse mit Matlab/Simulink	4
1.1	Matlab	4
1.1.1	Grundlagen in Matlab	6
1.1.2	Control System Toolbox	7
1.2	Simulink	8
1.2.1	Grundlagen von Simulink	8
1.2.2	Implementierung von dynamischen Systemen	10
1.3	Anwendungsbeispiele	16
1.3.1	Elektrisches Netzwerk	16
1.3.2	Datenkompression	19
1.3.3	Rechnen mit Polynomen	20
1.3.4	Integrationsalgorithmen	21
1.4	Literatur	24
2	Reglerentwurf und Simulation	25
2.1	Elektrisches System	25
2.2	Gleichstrommaschine mit Propeller	27
2.3	Reglerentwurf - Einfluss der Abtastzeit T_a	32
2.4	Lead-Lag-Reglerentwurf	34
2.5	Literatur	35
3	Zustandsregler	36
3.1	Zustandsreglerentwurf für die Gleichstrommaschine mit Propeller	36
3.2	Zustandsbeobachter	39
3.3	Praktischer Laborversuch	40
3.3.1	Implementierung des Reglers am Versuchsaufbau	40
3.4	Literatur	42

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.** Trotz der Vorbereitung in Zweier-Gruppen, 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. bei TU.it (<http://www.sss.tuwien.ac.at/sss/>) beziehen. Die bereitgestellten Beispieldateien setzen MATLAB/SIMULINK ab Version R2020b 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 zweistündigen Übungseinheiten werden die von Ihnen ausgearbeiteten Lösungen der Aufgaben besprochen und die zugrunde liegende Theorie diskutiert.

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 Theoriekennnisse

ein.

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

0.3 Termine Wintersemester 2020/21

Die Vorbesprechung zur Lehrveranstaltung findet am 20.10.2020 um 14:00 Uhr als Zoom Web-Meeting statt. Der entsprechende Link ist im TUWEL Kurs zu finden bzw. wird als TISS Nachricht ausgesendet. 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 als Zoom Web-Meeting abgehalten. Datum und Uhrzeit entnehmen Sie bitte nachfolgender Tabelle. Die Uhrzeit richtet sich nach der Gruppenanmeldung. Es sind pro Übungstermin jeweils vier Zeitslots verfügbar.

Übungseinheit	Datum	Zeit
Vorbesprechung	Dienstag, 20.10.2020	14:00 bis 14:30
Übung 1	Donnerstag, 10.12.2020	08:00 bis 10:00
Übung 1	Donnerstag, 10.12.2020	10:00 bis 12:00
Übung 1	Donnerstag, 10.12.2020	13:00 bis 15:00
Übung 1	Donnerstag, 10.12.2020	15:00 bis 17:00
Übung 2	Donnerstag, 14.01.2021	08:00 bis 10:00
Übung 2	Donnerstag, 14.01.2021	10:00 bis 12:00
Übung 2	Donnerstag, 14.01.2021	13:00 bis 15:00
Übung 2	Donnerstag, 14.01.2021	15:00 bis 17:00
Übung 3	Donnerstag, 28.01.2021	08:00 bis 10:00
Übung 3	Donnerstag, 28.01.2021	10:00 bis 12:00
Übung 3	Donnerstag, 28.01.2021	13:00 bis 15:00
Übung 3	Donnerstag, 28.01.2021	15:00 bis 17:00

0.4 Ansprechperson für organisatorische Belange

Bei Fragen oder Anregungen organisatorischer Natur wenden Sie sich bitte an

- Florian Beck <beck@acin.tuwien.ac.at>.

0.5 Weitere Informationen

Aktuelle Informationen zur Lehrveranstaltung sind auf der Instituts-Homepage <https://www.acin.tuwien.ac.at/bachelor/regelungstechnik/> 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.



Eine kostenlose MATLAB-Lizenz wird von der TU-Wien zur Verfügung gestellt. Im Computerlabor des Instituts steht MATLAB/SIMULINK in der Version R2020b zur Verfügung. Informationen zum Installieren von MATLAB finden Sie im als QR-Code dargestellten Link. Dieser Link ist im PDF auch anklickbar.



Studieren Sie als Vorbereitung auf die Übung zumindest folgende Kapitel im Skriptum zur VU Automatisierung (WS 2020/21) [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

- Christopher Pietschnig <pietschnig@acin.tuwien.ac.at> oder
- Jakob Maderthamer <maderthamer@acin.tuwien.ac.at>.

Für organisatorische Fragen wenden Sie sich bitte an

- Florian Beck <beck@acin.tuwien.ac.at>.

Hinweis: Lesen und beachten Sie alle Hinweise. Vermeiden Sie die Symbolic Math Toolbox in der Ausarbeitung der Aufgaben.

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.



Alle MATLAB/SIMULINK Dateien, die zum Bearbeiten dieser Übung benötigt werden, finden Sie in `uebung1.zip` auf der Homepage der Lehrveranstaltung.



1.1.1 Grundlagen in Matlab



Auf der Homepage der Lehrveranstaltung finden Sie die MATLAB Scripts `cds_matlab_intro_part1.m`, `cds_matlab_intro_part2.m` und `cds_matlab_intro_part3.m`. Öffnen Sie diese und führen Sie die enthaltenen Befehle schrittweise aus. Beachten Sie, dass die Funktion `mittelwert.m` aufgerufen wird, welche sich daher im aktuellen Arbeitsverzeichnis befinden muss.



Hinweis: 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.

Hinweis: Machen Sie sich auch mit den in `cds_matlab_intro_part1.m` angeführten `function handles` vertraut. Diese sind für die Zusatzaufgaben in der Übung von Vorteil.

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

$$x_1 + 2x_2 + 4x_3 = 5 \quad (1.1a)$$

$$2x_1 + 2x_2 + x_3 = 4 \quad (1.1b)$$

$$3x_1 + 2x_2 = 2 \quad (1.1c)$$

Schreiben Sie dieses Gleichungssystem in Matrixdarstellung an und bestimmen Sie den Lösungsvektor $\mathbf{x} = [x_1 \ x_2 \ x_3]^T$. 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(Ax - b)` der numerische Fehler der Berechnung bestimmt werden.

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

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

wobei $A = 10$ die Amplitude bezeichnet. Stellen Sie diese Rechteckfunktion für $n = 1, 2, \dots, 100$ 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

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

Stellen Sie die zu dieser Funktion gehörige Fläche $z = f(x, y, t_{\text{konst}})$ 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

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

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

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

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

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

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

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad (1.7a)$$

$$y = \mathbf{c}^T \mathbf{x} \quad (1.7b)$$

mit $\mathbf{x}^T = [x_1, x_2]$ 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 $\Phi(t)$ und bestimmen Sie die allgemeine Lösung von (1.7) für den Fall, dass u durch den Einheitssprung $\sigma(t)$ gegeben ist. Diese Berechnungen sind handschriftlich durchzuführen!

Hinweis: Für die Berechnung der Transitionsmatrix können die Eigenwerte allgemein mit λ_1 und λ_2 bezeichnet werden. Die zuvor berechneten Ausdrücke müssen nicht explizit eingesetzt werden.

2. Schreiben Sie ein m-file, in welchem der Integrationsalgorithmus (1.5) auf das Differentialgleichungssystem (1.7) angewendet wird. Verwenden Sie als Integrationszeitraum $[0, 20]$ s, als Zeitschrittweite (Abtastzeit) $T_a = 0.05$ s und für die Systemparameter $m = 1$ kg, $c = 1$ N s/m, $k = 2$ N/m, $s_0 = 0$ m, $v_0 = 0$ 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. Mithilfe von SIMULINK gestaltet sich das Lösen von Anfangswertproblemen einfach. Die grafische Bedienoberfläche erlaubt die Beschreibung von dynamischen Systemen mithilfe von Blockschaltbildern. Hierbei stellt SIMULINK eine Bibliothek mit vorgefertigten Funktionsblöcken zur Verfügung, welche durch benutzerdefinierte Blöcke erweitert werden können.

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 MATLAB-Script definiert, welches vor der Simulation ausgeführt wird. Damit können die Parameter einfach und an zentraler Stelle geändert werden. Zum Update der Parameter muss das Matlab-Script 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 befinden sich in der Gruppe **User-Defined Functions**. Im Block **Fcn** können sowohl MATLAB Funktionen als auch benutzerdefinierte Funktionen verwendet werden.
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.
5. LTI-Systeme, welche mittels der Control System Toolbox als Übertragungsfunktion oder in Zustandsraumdarstellung erzeugt wurden, können direkt in SIMULINK eingebunden werden (**Control System Toolbox** → **LTI System**).

Simulationseinstellungen können im Menü *Modeling* → *Model Settings* (*Strg + E*) vorgenommen werden. Besonders wesentlich ist dabei die Wahl der Simulationsdauer

und des Integrationsalgorithmus. Ferner können unter *Solver details* 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.

Aufgabe 1.3. Machen Sie sich weiter mit Simulink vertraut.



Auf der Homepage der Lehrveranstaltung finden Sie das Simulink-Modell `cds_simulink_intro_part1.slx`.



Öffnen Sie dieses Modell und starten Sie die Simulation. Achten Sie darauf, zuvor das *Matlab-Script* `init_cds_simulink_intro_part1.m` auszuführen. Dieses *Matlab-Script* erstellt alle für die Simulation notwendigen Variablen im MATLAB-Workspace. Versuchen Sie alle Blöcke zu verstehen und machen Sie gegebenenfalls von der Hilfefunktion Gebrauch.

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 oder auch als MATLAB Function erfolgen. Die beiden Varianten werden im Folgenden kurz erläutert.

Blockschaltbild

Um eine Differentialgleichung höherer Ordnung mit Simulink lösen zu können, muss diese als System von Differentialgleichungen erster Ordnung dargestellt werden. Dieses Vorgehen wird in Beispiel 1.1 dargestellt.

Beispiel 1.1. Das Anfangswertproblem

$$\ddot{y} + \cos^2(y)\dot{y} + ay + u = 0, \quad \ddot{y} = \dot{y} = y = 0, \quad u = \sin(t) \quad (1.8)$$

mit einem Parameter $a = 0.3$ soll mittels einer Integratorkette dargestellt werden. Zuerst wird die Differentialgleichung in Zustandsraumdarstellung $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u)$ übergeführt. Da (1.8) eine Differentialgleichung dritter Ordnung ist, wird ein dreidi-

mensionaler Zustand

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} \quad (1.9)$$

verwendet. Damit ergibt sich ein System von drei Differentialgleichungen erster Ordnung

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ -\cos^2(x_1)x_3 - ax_2 - u \end{bmatrix}, \quad \mathbf{x}(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \quad (1.10)$$

Da man an der Lösung $\mathbf{x}(t)$ interessiert ist, wird (1.10) einmal zeitlich integriert. Dadurch ergibt sich

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \begin{bmatrix} x_1(0) + \int_0^t x_2 \, d\tau \\ x_2(0) + \int_0^t x_3 \, d\tau \\ x_3(0) + \int_0^t -\cos^2(x_1)x_3 - ax_2 - u \, d\tau \end{bmatrix}, \quad \mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (1.11)$$

In (1.11) ist ersichtlich, dass drei Integratoren benötigt werden. Jeder dieser Integratoren kann mittels eines Integratorblocks in Simulink dargestellt werden. Damit kann das Differentialgleichungssystem in Simulink wie in Abbildung 1.1 dargestellt werden.

Hinweis: Die Anfangszustände $(x_{1,0}, x_{2,0}, x_{3,0})$ müssen in den Einstellungen der einzelnen Integratorblöcke definiert werden.

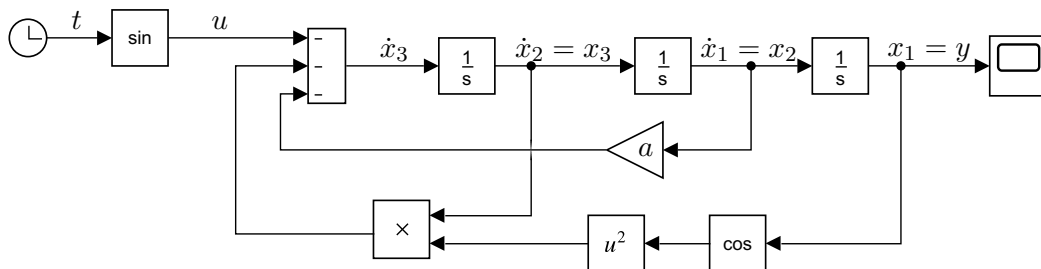


Abbildung 1.1: Implementierung von (1.11) mittels einer Integratorkette.

Aufgabe 1.4. Betrachten Sie für die nachstehenden Teilaufgaben die nichtlineare Differentialgleichung

$$\ddot{x} + \cos(x)^2 \dot{x} + \dot{x} + e^{-x}u = 0. \quad (1.12)$$

1. Ermitteln Sie die numerische Lösung der Differentialgleichung mit dem Zustand x , dem Eingang u und den Anfangsbedingungen $\ddot{x}(0) = 1$, $\dot{x}(0) = -2$ und $x(0) = 3$. 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 Ruhelage $x_R = 0$, $u_R = 0$ und überprüfen Sie die Stabilität des linearen Systems. Diese Berechnungen sind händisch durchzuführen.

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

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad \mathbf{x}(t_0) = \mathbf{x}_0 \quad (1.13a)$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \quad (1.13b)$$

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.

Matlab Function für zeitkontinuierliche Systeme

Die Implementierung einer Differentialgleichung mittels Integratorkette wird bei hochdimensionalen Problemen und steigender mathematischer Komplexität unübersichtlich. Durch das Benutzen von MATLAB Function Blöcken kann dies verhindert werden. Das Benutzen von MATLAB Function Blöcken wird in Beispiel 1.2 erläutert.

Beispiel 1.2. Das Anfangswertproblem (1.8) soll mittels eines MATLAB Function Blocks gelöst werden. Wird das System in Zustandsraumdarstellung $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u)$ übergeführt, so kann mittels Integration

$$\mathbf{x}(t) = \int_0^t \mathbf{f}(\mathbf{x}(\tau), u(\tau)) d\tau + \mathbf{x}_0 \quad (1.14)$$

der Zustand $\mathbf{x}(t)$ bestimmt werden. Die vektorielle Funktion $\mathbf{f}(\mathbf{x}, u)$ wird nun als MATLAB Function in Simulink implementiert. Dazu wird in Simulink ein MATLAB Function Block (User Defined Functions → MATLAB Function) eingefügt. Durch Doppelklick öffnet sich ein Fenster im MATLAB-Editor, in welchem programmiert werden kann. Die Differentialgleichung kann mittels dieses Programmcodes

```
function [xxpunkt] = fcn(u,xx,param)
    % Aufteilen der Parameter (zur besseren Lesbarkeit)
    a = param.a;
```

```

% Aufteilen des Zustandsvektors (zur besseren Lesbarkeit)
x1 = xx(1);
x2 = xx(2);
x3 = xx(3);

% Implementierung von f(x,u)
f1 = x2;
f2 = x3;
f3 = -cos(x1)^2 * x3 - a*x2 -u;

% Zuweisen des Ausgangs des Blocks f(x,u)
ff = [f1;
      f2;
      f3];
end

```

implementiert werden. Hierbei besitzt die Funktion `fcn` drei Übergabeparameter (`xx`, `u`, `param`) und einen Rückgabewert (`xxpunkt`). Ist der MATLAB Function Block im MATLAB Editor geöffnet, so kann man im Reiter `Editor` mit dem Button `Edit Data` den `Ports and Data Manager` öffnen. In diesem kann eingestellt werden ob ein Übergabeparameter als Eingang oder als Parameter behandelt werden soll. Parameter müssen im MATLAB Workspace definiert werden. Es ist empfehlenswert alle benötigten Parameter in einem *Struct Array* zu organisieren. Hierbei ist es wichtig, dass im *Ports and Data Manager* eingestellt wird, dass der Parameter nicht *Tunable* ist.

Die Integration von `xxpunkt` wird nun mittels eines Integratorblocks in Simulink durchgeführt. Dieser Block kann mit vektorwärtigen Signalen umgehen. Man beachte, dass die Dimension des Anfangszustandes des Integratorblocks der Dimension des Zustandes des Systems entsprechen muss.

Der Systemausgang $y = h(\mathbf{x}, u)$ wird nun mittels eines weiteren MATLAB Function Blocks implementiert. Dieser benötigt im Allgemeinen die Systemparameter, den Eingang u sowie den Zustand $\mathbf{x}(t)$. Im gegebenen Beispiel ist dies durch den Programmcode

```

function [y] = fcn(u,xx,param)
% Aufteilen der Parameter (zur besseren Lesbarkeit)
a = param.a;

% Aufteilen des Zustandsvektors (zur besseren Lesbarkeit)
x1 = xx(1);
x2 = xx(2);
x3 = xx(3);

% Implementierung von h(x,u)
y = x1;
end

```

möglich. Die Abbildung 1.2 zeigt die vollständige Implementierung mittels MATLAB Function Blöcken.

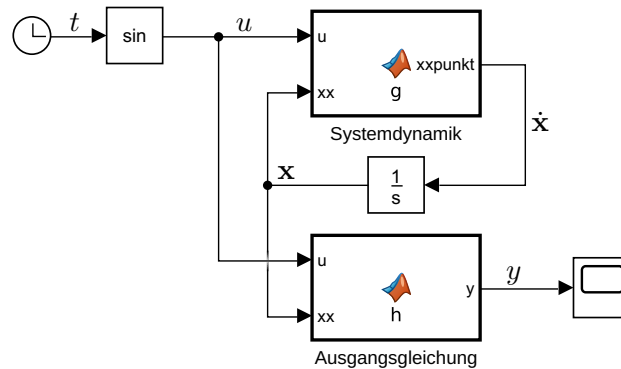


Abbildung 1.2: Implementierung von (1.11) mittels einer MATLAB Function.

Matlab Function für zeitdiskrete Systeme

Auch zeitdiskrete Systeme können mittels MATLAB Function Blöcken implementiert werden. Hierbei gibt es mehrere Möglichkeiten. Die im Folgenden erklärte Methode bietet den Vorteil, dass der resultierende Programmcode direkt in einem Digitalrechner implementiert werden kann. Dadurch können entworfene Regler oder Beobachter mit geringem Zeitaufwand auf industriellen Anlagen oder auf Versuchsaufbauten portiert werden. Im Speziellen ist es auf einfache Weise möglich die MATLAB Function direkt in der Rapid Prototyping System dSPACE zu verwenden.

In Beispiel 1.3 wird erklärt wie ein zeitdiskretes System mittels MATLAB Function Blöcken in Simulink implementiert wird.

Beispiel 1.3. Das Anfangswertproblem (1.8) soll mittels des Euler-vorwärts-Verfahrens diskretisiert werden. Das diskretisierte System soll als zeitdiskretes System mit der Abtastzeit $T_a = 100$ ms simuliert werden. Zuerst wird analog zu Beispiel 1.2 das System in Zustandsraumdarstellung übergeführt und zeitlich integriert (vgl. (1.14)). Das Integral (1.14) wird jedoch mittels des Euler-vorwärts-Verfahrens approximiert. Die sich ergebende Differenzgleichung

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_a \begin{bmatrix} x_{2,k} \\ x_{3,k} \\ -\cos^2(x_{1,k})x_{3,k} - ax_{2,k} - u_k \end{bmatrix}, \quad \mathbf{x}_0 = \mathbf{x}(0) \quad (1.15)$$

kann nun direkt in einer MATLAB Function mit dem Programmcode

```
function y = fcn(u,param)
% Aufteilen der Parameter
a = param.a;
Ta = param.Ta;

% Definition des diskreten Zustandes als persistent Variable
persistent xx;
```

```

% Initialisierung des diskreten Zustandes
if isempty(xx); xx=param.xx0; end

% Aufteilen des Zustandsvektors
x1 = xx(1);
x2 = xx(2);
x3 = xx(3);

% Implementierung von f(x,u)
f1 = x2;
f2 = x3;
f3 = -cos(x1)^2*x3 -a*x2 -u;

% Zusammenfuegen zu einem Vektor
ff = [ f1;
      f2;
      f3 ];

% Approximation mittels Euler-Verfahren
xx = xx + Ta*ff;

% Implementierung von h(x,u)
y = x1;
end

```

implementiert werden. Hierbei wird der Zustand \mathbf{xx} als *persistent* Variable definiert, welche über mehrere Funktionsaufrufe erhalten bleiben. Beim Erstellen dieser Variable ist diese leer (das bedeutet das Gleiche wie die Zuweisung $\mathbf{xx}=[]$). Man beachte, dass im ersten Aufruf des MATLAB Function Blocks die Variable \mathbf{xx} mit dem Anfangswert $\mathbf{xx0}$ initialisiert wird. Wird ein zeitdiskretes System in SIMULINK implementiert, so ist es weiters notwendig die Abtastzeit zu spezifizieren. Dies ist möglich, indem man im MATLAB Function Block mittels *Edit Data* den *Ports and Data Manager* öffnet und die *Update Method* als *Discrete* festlegt, siehe dazu Abbildung 1.3.

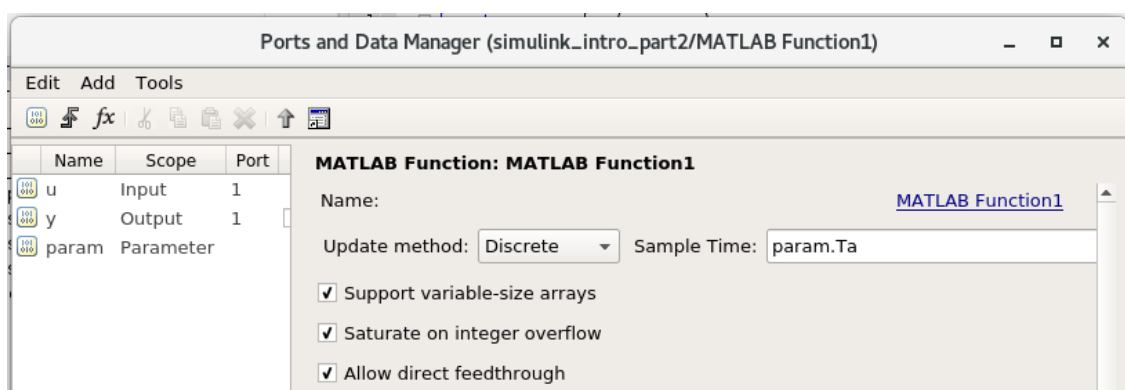


Abbildung 1.3: Einstellungen im *Ports and Data Manager* für zeitdiskrete Systeme.

Aufgabe 1.5. Machen Sie sich mit der Implementierung dynamischer Systeme vertraut.



Auf der Homepage der Lehrveranstaltung finden Sie das Simulink-Modell `simulink_intro_part2.slx`. In diesem Simulinkmodell sind Beispiel 1.1, Beispiel 1.2 und Beispiel 1.3 implementiert.



Öffnen Sie dieses Modell und starten Sie die Simulation. Achten Sie darauf zuvor das MATLAB-Script `init_cds_simulink_intro_part2.m` auszuführen. Dieses MATLAB-Script erstellt alle für die Simulation notwendigen Variablen im MATLAB-Workspace. Versuchen Sie alle Blöcke zu verstehen und machen Sie gegebenenfalls von der Hilfefunktion Gebrauch. Achten Sie besonders auf die Einstellungen welche im *Ports and Data Manager* getroffen wurden.

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 1.4 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.

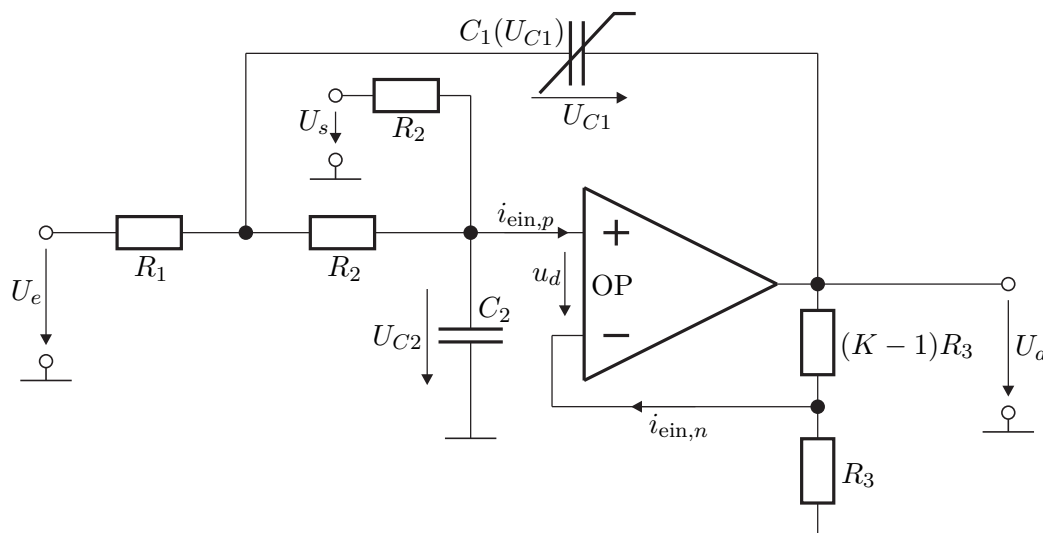


Abbildung 1.4: Elektrisches System.

Aufgabe 1.6 (Modellierung).

- Leiten Sie unter der Annahme eines idealen Operationsverstärkers das mathematische Modell des elektrischen Netzwerks aus Abbildung 1.4 in der Form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, d) \quad (1.16a)$$

$$y = \mathbf{h}(\mathbf{x}, u, d) \quad (1.16b)$$

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

$$C_1(U_{C1}) = \frac{dQ_1}{dU_{C1}}$$

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.

- Linearisieren Sie das System (1.16) für den stationären Eingang $u_R = U_{e,R}$ und den stationären Störeingang $d_R = U_{s,R}$ um eine allgemeine Ruhelage $\mathbf{x}_R = [U_{C1,R} \ U_{C2,R}]^T$. Stellen Sie das linearisierte System in der Form

$$\Delta \dot{\mathbf{x}} = \mathbf{A} \Delta \mathbf{x} + \mathbf{b}_u \Delta u + \mathbf{b}_d \Delta d \quad (1.17a)$$

$$\Delta y = \mathbf{c}^T \Delta \mathbf{x} + d_u \Delta u + d_d \Delta d \quad (1.17b)$$

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.

- Berechnen Sie die Ruhelagen $\mathbf{x}_R = [U_{C1,R} \ U_{C2,R}]^T$ des Systems (1.16) für den stationären Eingang $u_R = U_{e,R}$ und den stationären Störeingang $d_R = U_{s,R}$.



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.



- Bestimmen Sie für das linearisierte System (1.17) 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

$$Q_1(U_{C1}) = U_{C1,ref}(C_{1,ref} - C_{1,\infty}) \arctan\left(\frac{U_{C1}}{U_{C1,ref}}\right) + C_{1,\infty}U_{C1} \quad (1.18)$$

sowie die Ruhelage durch $U_{e,R} = 5 \text{ V}$ und $U_{s,R} = 5 \text{ V}$. Wählen Sie die Parameter $C_{1,ref} = 9 \mu\text{F}$, $U_{C1,ref} = 3.5 \text{ V}$, $C_{1,\infty} = 1 \text{ nF}$, $C_2 = 1 \mu\text{F}$, $R_1 = 600 \Omega$, $R_2 = 3500 \Omega$, $K = 3$.

- Bei der Übertragungsfunktion $G(s)$ aus Punkt 4 handelt es sich um ein Verzögerungsglied 2-ter Ordnung (PT₂) in der Form $G(s) = \frac{V}{1 + 2\xi(sT) + (sT)^2}$. Bestimmen Sie den Verstärkungsfaktor V , den Dämpfungsgrad ξ und die Zeitkonstante T . Führen Sie diese Berechnungen händisch aus.
- Stellen Sie mittels MATLAB die Bode-Diagramme der beiden Übertragungsfunktionen dar und interpretieren Sie diese.

Hinweis: Zur Überprüfung Ihrer Ergebnisse sind an dieser Stelle die numerische Dynamikmatrix und die zugehörigen Eingangsvektoren des Systems (1.17) angegeben

$$\mathbf{A} = \begin{bmatrix} -1985.997 & -5667.357 \\ 285.714 & 285.714 \end{bmatrix}, \quad \mathbf{b}_u = \begin{bmatrix} 1695.363 \\ 0 \end{bmatrix}, \quad \mathbf{b}_d = \begin{bmatrix} 0 \\ 285.714 \end{bmatrix}.$$

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

Aufgabe 1.7 (Implementierung in MATLAB/SIMULINK). Das nichtlineare Modell (1.16) soll nun als MATLAB Function und das linearisierte Modell (1.17) als State-Space-Block implementiert werden.



Verwenden Sie hierzu die Simulationsdatei `Esys.slx` sowie das bereits in Aufgabe 1.6 bearbeitete M-file `Esys_Parameter.m` aus dem zip-Archiv [uebung1.zip](#).



Verwenden Sie weiterhin die Parameter aus Punkt 4 von Aufgabe 1.6.

- Öffnen Sie die Simulationsdatei `Esys.slx` und vervollständigen Sie die MATLAB Function. Unvollständige Einträge sind durch den Kommentar `TODO` gekennzeichnet. Führen Sie anschließend die Simulation `Esys.slx` aus und überzeugen Sie sich von der Lauffähigkeit und Plausibilität der fertiggestellten MATLAB Function.
- Das Simulationsmodell `Esys.slx` enthält bereits eine Grundstruktur für die

Implementierung des linearisierten Modells (1.17) 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 Systems 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.

- 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 $[10^2 \text{ rad/s}, 10^3 \text{ rad/s}, 10^4 \text{ rad/s}]$ und $[0.5 \text{ V}, 1 \text{ V}, 3 \text{ V}]$. 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? Untersuchen Sie die Unterschiede zwischen dem nichtlinearen und dem linearisierten System. Wie erklären sich diese Unterschiede? Dokumentieren und diskutieren Sie ihre Simulationsergebnisse.

1.3.2 Datenkompression

Das in Abbildung 1.5 dargestellte System zur Kompression von Datenströmen wird durch einen Speicher und eineessoreinheit 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 $T_1 = 1 \text{ ms}$ 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 $a = 0.4 \text{ s}^{-1}$ definiert.

Aufgabe 1.8.

Hinweis: Führen Sie die Rechnungen der Punkte 1 und 2 händisch aus.

- 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 $T_a = 1 \mu\text{s}$.
- 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. Betrachten Sie dazu den Speicher von Beginn an als zeitdiskretes System, d. h. beschreiben Sie den Speicher unmittelbar mit einer Differenzgleichung.
- Implementieren Sie das diskrete dynamische System der Datenkompression in MATLAB/SIMULINK als Blockschaltbild. Realisieren Sie dabei den Prozessor als **Zustandsraumdarstellung** in einem LTI-Systems-Block. Achten Sie dabei

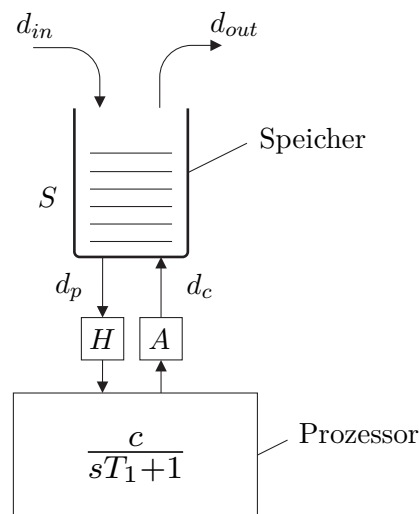


Abbildung 1.5: Datenkompression.

auf eine zeitdiskrete Simulationsumgebung mit der $T_a = 1 \mu\text{s}$.

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 sprunghaft 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 $t = 10 \text{ ms}$ aufschalten.

Hinweis: Beachten Sie die korrekte Wahl des Anfangszustandes.

1.3.3 Rechnen mit Polynomen

Aufgabe 1.9. Gegeben ist das Polynom

$$p_1(s) = s^3 + 8s^2 + 19s + 12 \quad (1.19)$$

und $p_2(s)$ sei das charakteristische Polynom der Matrix

$$\mathbf{A} = \begin{bmatrix} -1 & 2 \\ -2 & -6 \end{bmatrix}. \quad (1.20)$$

Hinweis: Polynome können in MATLAB als Vektoren der absteigend geordneten Polynomkoeffizienten dargestellt werden, d. h. $p(s) = a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0$ wird als Vektor $\mathbf{p} = [a_n, a_{n-1}, \dots, a_1, a_0]^T$ eingegeben. Zur Auswertung eines Polynoms an einer bestimmten Stelle s kann der Befehl `polyval()` verwendet werden.

Hinweis: Zur Bestimmung des charakteristischen Polynoms einer quadratischen Matrix kann der Befehl `poly()` verwendet werden.

Hinweis: Die Multiplikation zweier Polynome entspricht der Faltung ihrer Koeffizientenvektoren. Die (diskrete) Faltungsoperation kann mit dem Befehl `conv()` durchgeführt werden.

1. Berechnen Sie in MATLAB das Polynom $p_3(s) = p_1(s)p_2(s)$.
2. Schreiben Sie eine Funktion mit der Schnittstelle `pd = polydiff(p)`, welche als Argument `p` die Koeffizienten eines beliebigen Polynoms $p(s)$ erhält und als Rückgabewert `pd` die Koeffizienten der ersten Ableitung $dp(s)/ds$ zurückgibt. Vermeiden Sie dabei den Befehl `polyder()`.
3. Schreiben Sie eine Funktion mit der Schnittstelle `s0 = findzero(p, sstart)`, welche als Argument `p` die Koeffizienten eines beliebigen Polynoms $p(s)$ sowie einen Startwert `sstart` erhält. Die Funktion soll mittels des Newton-Verfahrens ausgehend vom Startwert `sstart` eine Nullstelle `s0` des Polynoms $p(s)$ suchen und zurückgeben. Überlegen Sie sich ein geeignetes Abbruchkriterium. Sie können in Ihrem Algorithmus gegebenenfalls die Funktion `polydiff()` verwenden.
4. Testen Sie die Funktion `findzero()` anhand des Polynoms $p_3(s)$. Sie können dazu natürlich $p_3(s)$ zunächst grafisch darstellen.
5. Bestimmen Sie mit dem Befehl `roots()` die Nullstellen von $p_3(s)$.
6. Ist \mathbf{A} eine Hurwitzmatrix?

1.3.4 Integrationsalgorithmen

Beim expliziten Euler-Verfahren handelt es sich um die einfachste Einschrittmethode aus der Klasse der Runge-Kutta-Verfahren. Unter geeigneten Voraussetzungen konvergiert das explizite Euler-Verfahren mit Konvergenzordnung 1, was für viele Anwendungen zu ungenau ist.

Das Verfahren von Heun ist ein weiteres explizites Einschrittverfahren und gehört ebenfalls zur Klasse der Runge-Kutta-Verfahren. Es basiert auf einem Prädiktor- und einem Korrektor-Schritt, d.h. zuerst wird ein Schritt vor-prädiziert und im Anschluss korrigiert. Der Prädiktionsschritt

$$\mathbf{x}_{k+1}^P = \mathbf{x}_k + T_a \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \quad (1.21)$$

wird mithilfe des expliziten Euler-Verfahrens berechnet. Anschließend wird dieser Schritt mittels der Trapezregel

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{2}T_a \left(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{f}(\mathbf{x}_{k+1}^P, \mathbf{u}_{k+1}) \right) \quad (1.22)$$

also

$$\mathbf{x}_{k+1} = \frac{1}{2}\mathbf{x}_k + \frac{1}{2} \left(\mathbf{x}_{k+1}^P + T_a \mathbf{f}(\mathbf{x}_{k+1}^P, \mathbf{u}_{k+1}) \right), \quad (1.23)$$

korrigiert. Unter geeigneten Voraussetzungen konvergiert das Verfahren von Heun mit Konvergenzordnung 2, d.h. der globale Fehler des Verfahrens geht wie T_a^2 gegen Null.

Ein noch genaueres Einschrittverfahren aus der Klasse der expliziten Runge-Kutta-Methoden ist durch das sogenannte klassische Runge-Kutta-Verfahren in der Form

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \quad (1.24a)$$

$$\mathbf{k}_2 = \mathbf{f}\left(\mathbf{x}_k + \frac{1}{2}T_a \mathbf{k}_1, \mathbf{u}_{k+1/2}\right) \quad (1.24b)$$

$$\mathbf{k}_3 = \mathbf{f}\left(\mathbf{x}_k + \frac{1}{2}T_a \mathbf{k}_2, \mathbf{u}_{k+1/2}\right) \quad (1.24c)$$

$$\mathbf{k}_4 = \mathbf{f}(\mathbf{x}_k + T_a \mathbf{k}_3, \mathbf{u}_{k+1}) \quad (1.24d)$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{6}T_a (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \quad (1.24e)$$

gegeben. Hierbei bezeichnet $\mathbf{u}_{k+1/2}$ den Eingangsvektor $\mathbf{u}(t)$ zum Zeitpunkt $t = kT_a + \frac{1}{2}T_a$. Unter entsprechenden Voraussetzungen besitzt das klassische Runge-Kutta-Verfahren Konvergenzordnung 4.

Aufgabe 1.10. In Aufgabe 1.2 haben Sie bereits das explizite Euler-Verfahren für das dynamische System (1.7) verwendet. Implementieren Sie nun für das gleiche dynamische System das Verfahren von Heun und das klassische Runge-Kutta-Verfahren. Die Systemparameter sollen mit $m = 1 \text{ kg}$, $c = 1 \text{ N s/m}$, $k = 2 \text{ N/m}$ unverändert bleiben. Wie in Aufgabe 1.2 soll ein Sprung aufgeschaltet werden. Auch der Integrationszeitraum bleibt mit $[0, 20] \text{ s}$ unverändert. Die Anfangswerte seien nun allerdings durch $s_0 = 1 \text{ m}$ und $v_0 = 0.5 \text{ m/s}$ gegeben.

Vergleichen Sie die numerischen Ergebnisse des expliziten Euler-Verfahrens, des Verfahrens von Heun und des klassischen Runge-Kutta-Verfahrens jeweils mit der in Aufgabe 1.2 berechneten analytischen Lösung. Stellen Sie hierzu den Zeitverlauf der Zustände der numerischen, als auch der exakten Lösung in einer Grafik dar. Erstellen Sie jeweils eine Grafik entsprechend jeder numerischen Methode. Verkleinern Sie die Abtastzeit so lange bis die numerische Lösung des klassischen Runge-Kutta-Verfahrens und die analytische Lösung in der grafischen Darstellung hinreichend gut übereinstimmen. Was gilt bei gleicher Abtastzeit für die numerischen Lösungen des Verfahrens von Heun und des expliziten Euler-Verfahrens?

Aufgabe 1.11. Wir betrachten wieder das dynamische System (1.7) mit den gleichen Systemparametern, den gleichen Anfangswerten und gleichem Integrationszeitraum wie in Aufgabe 1.10. Der einzige Unterschied besteht darin, dass nun anstatt eines

Sprungs eine Sinusschwingung der Form

$$u(t) = \sin(10\pi t/T), \quad t \in [0, 20] \text{ s}, \quad T = 20 \text{ s}$$

aufgeschaltet werden soll. Berechnen Sie die Zeitverläufe der numerischen Lösungen mithilfe des expliziten Euler-Verfahrens, des Verfahrens von Heun und des klassischen Runge-Kutta-Verfahrens.

Die Berechnung der exakten Lösung ist für den oben angegebenen zeitlichen Verlauf der Eingangsgröße recht mühsam. Eine Referenzlösung soll daher mit dem Befehl `lsim` aus der Control System Toolbox berechnet werden. Stellen Sie die Zeitverläufe der numerischen Lösungen denen, welche sie mit dem Befehl `lsim` berechnet haben, gegenüber. Erstellen Sie wieder jeweils eine separate Grafik entsprechend den Ergebnissen jeder numerischen Methode. Beschreiben Sie kurz wie sich die Wahl der Abtastzeit T_a auf die Güte der berechneten Lösungen auswirkt.

Abschließend soll der Begriff einer steifen Differentialgleichung bzw. eines steifen Differentialgleichungssystems $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ motiviert werden. Vereinfacht ausgedrückt handelt es sich bei steifen Differentialgleichungssystemen um solche Systeme, welche sich sehr viel einfacher mit impliziten als mit expliziten Zeitintegrationsmethoden lösen lassen. Es ist im Allgemeinen nicht möglich, den Begriff der Steifigkeit mathematisch eindeutig zu definieren. Typischerweise kann man jedoch sagen, dass bei steifen Differentialgleichungen Stabilitätsanforderung und weniger Genauigkeitsanforderungen die Größe der Zeitschrittweite (Abtastzeit) bestimmen. Oftmals klingen gewisse Komponenten der Lösung eines steifen Differentialgleichungssystems sehr viel schneller ab als andere.

Für lineare Systeme mit konstanten Koeffizienten kann man den sogenannten Steifigkeitsquotienten

$$S = \frac{\max_{j=1, \dots, n} |\Re(\lambda_j)|}{\min_{j=1, \dots, n} |\Re(\lambda_j)|} \quad (1.25)$$

mithilfe der Eigenwerte $\lambda_1, \dots, \lambda_n$ der Dynamikmatrix A definieren. Gewöhnlicherweise gilt für den Steifigkeitskoeffizienten nicht-steifer Differentialgleichungssysteme $S \approx 1$, während der Steifigkeitskoeffizient steifer Systeme sehr große Werte annimmt.

Steife Differentialgleichungssysteme müssen in der Regel mit impliziten Zeitintegrationsmethoden gelöst werden. Das mit Abstand einfachste implizite Einschritt-Verfahren ist das implizite Euler-Verfahren, welches durch die Iterationsvorschrift

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_a \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) \quad (1.26)$$

definiert ist. In Abhängigkeit von \mathbf{f} muss daher in jedem Zeitschritt ein lineares oder nichtlineares Gleichungssystem gelöst werden.

Aufgabe 1.12. Gegenstand von Aufgabe 1.12 ist wieder das dynamische System (1.7). Berechnen Sie den Steifigkeitsquotienten für die Systemparameter aus Aufgabe 1.10, d.h. mit $m = 1 \text{ kg}$, $c = 1 \text{ N s/m}$ und $k = 2 \text{ N/m}$. Wie lautet der Wert des Steifigkeitsquotienten für den Fall $m = 1 \text{ kg}$, $c = 1001 \text{ N s/m}$ und $k = 1000 \text{ N/m}$? Berechnen Sie den Zeitverlauf der Zustände mithilfe des expliziten Euler-Verfahrens für den Integrationszeitraum $[0, 20] \text{ s}$. Wählen Sie die Anfangswerte wie in Aufgabe 1.10. Als externe

Kraft soll wieder ein Sprung aufgeschaltet werden. Vergleichen Sie ihre Lösung mit der exakten Lösung. Wie groß darf die Abtastzeit ($20\text{ s}/T_a$ soll ganzzahlig sein) maximal sein, damit das explizite Euler-Verfahren ein qualitativ richtiges Ergebnis liefert? Stellen Sie den zeitlichen Verlauf der Zustände einmal für den gesamten Integrationszeitraum und einmal für einen kleinen Zeitraum zu Beginn der Simulation dar. Sie können dazu den Befehl `xlim` verwenden (z.B. `xlim([-2, 22])` und `xlim([-0.05, 1.0])`). Vergleichen Sie den zeitlichen Verlauf beider Lösungskomponenten.

Berechnen Sie schließlich den zeitlichen Verlauf der Zustände mithilfe des impliziten Euler-Verfahrens. Wie groß darf die Abtastzeit maximal sein, damit das implizite Euler-Verfahren ein qualitativ richtiges Ergebnis liefert?

1.4 Literatur

- [1.1] A. Kugi, *Skriptum zur VU Automatisierung (WS 2020/2021)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2020. Adresse: <https://www.acin.tuwien.ac.at/bachelor/automatisierung/>.
- [1.2] A. Angermann, M. Beuschel, M. Rau und U. Wohlfarth, *Matlab - Simulink - Stateflow, Grundlagen, Toolboxes, 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 2020/21) [2.1]
 - Kapitel 1 bis Kapitel 6

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

- Philipp Rosenzweig <rosenzweig@acin.tuwien.ac.at> oder
- Florian Beck <beck@acin.tuwien.ac.at>.

Für organisatorische Fragen wenden Sie sich bitte an

- Florian Beck <beck@acin.tuwien.ac.at>.



Alle MATLAB/SIMULINK Dateien, die zum Bearbeiten dieser Übung benötigt werden, finden Sie in `uebung2.zip` auf der Homepage der Lehrveranstaltung.



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 Streckenübertragungsfunktion des linearisierten Systems ist durch

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

mit $V = 1.382$, $\xi = 0.829$ und $T = 0.975 \cdot 10^{-3}$ s gegeben.

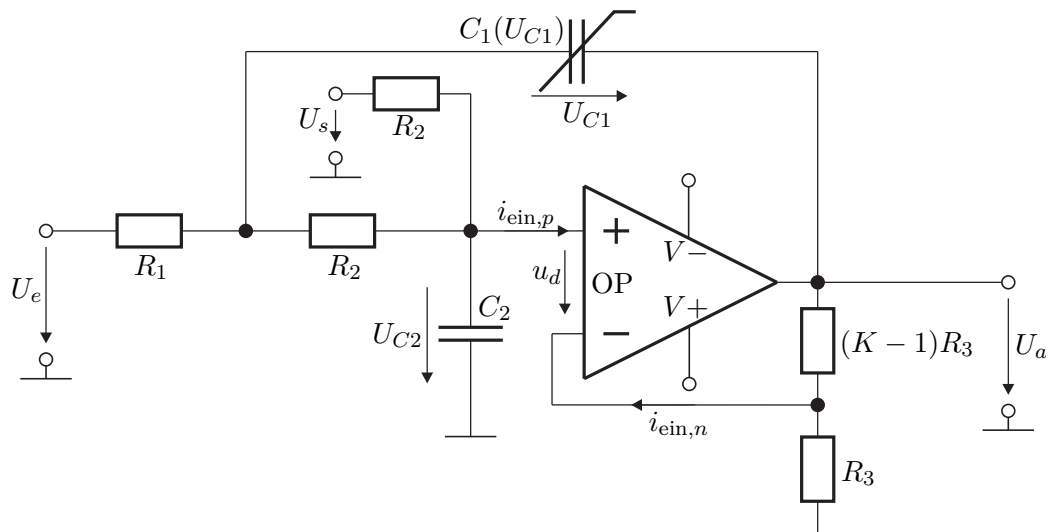


Abbildung 2.1: Elektrisches System.

Aufgabe 2.1 (Reglerentwurf). In dieser Aufgabe sollen zeitkontinuierliche Regler für das elektrische System nach dem Frequenzkennlinienverfahren entworfen werden.

1. Entwerfen Sie in MATLAB einen PI-Regler nach dem Frequenzkennlinienverfahren. Der geschlossene Kreis soll dabei folgende Spezifikationen erfüllen:

bleibende Regelabweichung	$e_{\infty} _{r(t)=\sigma(t)} = 0$
Überschwingen	$\ddot{u} \leq 5\%$
Anstiegszeit	$t_r = 1.5 \text{ ms}$

Begründen Sie warum die Wahl dieser Reglerstruktur für die obigen Anforderungen sinnvoll ist.

2. Ist das Frequenzkennlinienverfahren ein exaktes Entwurfsverfahren, wenn – wie im vorliegenden Fall – die Strecke ein Verzögerungsglied 2-ter Ordnung ist?
3. Im Weiteren soll ein PID-Regler nach dem Frequenzkennlinienverfahren entworfen werden. Der geschlossene Kreis soll dabei folgende Anforderungen erfüllen:

bleibende Regelabweichung	$e_{\infty} _{r(t)=t} = 1 \cdot 10^{-3}$
Überschwingen	$\ddot{u} \leq 5\%$
Anstiegszeit	$t_r = 1.5 \text{ ms}$

Die Zeitkonstante des Integralterms soll $T_I = 0.25 \text{ ms}$ betragen. Eventuell auftretende nichtlineare Gleichungen können Sie z. B. mittels `fsolve` lösen.

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

sowohl für den PI- als auch den PID-Regler. Berechnen Sie die zugehörigen Sprungantworten und die Rampenantworten und plotten Sie diese 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 aufgeschaltet 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 Simulationsmodell 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

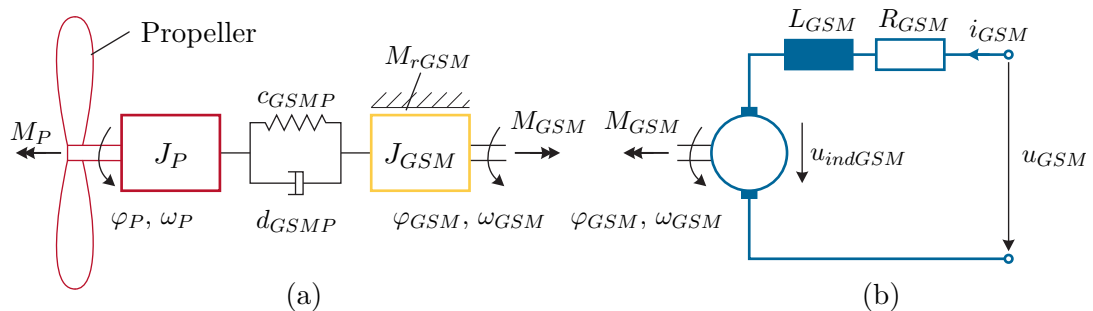


Abbildung 2.2: Gleichstrommaschine mit Propeller, (a) mechanisches Teilsystem, (b) elektrisches Teilsystem.

Abbildung 2.2 zeigt schematisch eine permanenterrregte 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 $\mathbf{q}^T = [\varphi_{GSM} \quad \varphi_P]$. Die zugehörigen Winkelgeschwindigkeiten werden mit $\dot{\mathbf{q}}^T = [\omega_{GSM} \quad \omega_P]$ 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

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

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

$$M_{rGSM} = d_{cGSM} + d_{vGSM}\omega_{GSM} \quad (2.2)$$

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_{GSM} und den Dämpfer d_{GSM} übertragene Moment, berechnet sich zu

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

Das von der (idealen) Gleichstrommaschine erzeugte elektrische Moment ist $M_{GSM} = k_{GSM}i_{GSM}$, wobei k_{GSM} die Ankerkreisinduktivität 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

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

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

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

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

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

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

mit den Zustandsgrößen $\mathbf{x}_m^T = [i_{GSM} \ \varphi_{GSM} \ \omega_{GSM} \ \varphi_P \ \omega_P]$, 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

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

kann daher eine Differentialgleichung eingespart werden. Führen Sie diese Transformation durch, d. h. bestimmen Sie

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

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

$$\Delta \dot{\mathbf{x}} = \mathbf{A} \Delta \mathbf{x} + \mathbf{b}_u \Delta u + \mathbf{b}_d \Delta d \quad (2.8a)$$

$$\Delta y = \mathbf{c}^T \Delta \mathbf{x} \quad (2.8b)$$

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 $u_{GSM,R} = 5.6 \text{ V}$ und $M_{ext,R} = 0 \text{ Nm}$ zu verwenden sind.

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

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

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

$$i_{GSM} = \frac{1}{R_{GSM}} u_{GSM} - \frac{k_{GSM}}{R_{GSM}} \omega_{GSM} \quad (2.10)$$

reduziert.

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

Tabelle 2.1: Parameter des Systems Gleichstrommaschine mit Propeller.

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

$$\dot{\mathbf{x}}_{red} = \mathbf{f}_{red}(\mathbf{x}_{red}, u, d) \quad (2.11a)$$

$$y = h_{red}(\mathbf{x}_{red}) \quad (2.11b)$$

mit dem neuen Zustandsvektor $\mathbf{x}_{red}^T = [\varphi_{GSMP} \quad \omega_{GSM} \quad \omega_P]$, 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 MATLAB Function implementiert werden. Wählen Sie hierzu als Eingänge $u_1 = u_{GSM}$

und $u_2 = M_{ext}$ und als Ausgangsvektoren $\mathbf{y}_1^T = [i_{GSM} \ \varphi_{GSMP} \ \omega_{GSM} \ \omega_P]$ bzw. $\mathbf{y}_2^T = [M_{GSM} \ M_{Kopp}]$. Verwenden Sie als Anfangszustand die Ruhelage für $u_{GSM} = 5.6 \text{ V}$ und $M_{ext} = 0 \text{ Nm}$.

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 Nm und u_{GSM} in V). Vergleichen Sie die Ausgangsgrößen der Systeme und dokumentieren Sie Ihre Ergebnisse.



Verwenden Sie hierzu die Simulationsdatei `Simulation_GSM_out.slx` sowie das M-file aus dem zip-Archiv [uebung2.zip](#).



Hinweis: Beachten Sie bei der Implementierung nachfolgende Punkte:

- Verwenden Sie als Vorlage die Simulationsdatei `Simulation_GSM_out.slx`.
- Alle Systemparameter und Anfangszustände sollen als Parameter von außen an die MATLAB-Funktion übergeben werden. Definieren Sie diese Größen in dem M-file `GSM_Parameter.m` als Teil der Struktur `parGSM`. Führen Sie dieses M-file jeweils vor dem Start der Simulation aus.
- 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 eine Referenzimplementierung als Subsystem in der Simulationsdatei `Simulation_GSM_out.slx` zur Verfügung. Es handelt sich um eine chiffrierte MATLAB-Funktion der Gleichstrommaschine. Das Referenzmodell enthält bereits alle benötigten Parameter mit Ausnahme der Ruhelagen. Die berechnete Ruhelage \mathbf{x}_0 aus der Aufgabe 2.2 (Teilaufgabe 2) ist in der Maske des vollständigen Modells (2.7) zu ergänzen.

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 $u_{GSM,R} = 5.6 \text{ V}$ und $M_{ext,R} = 0 \text{ Nm}$. Der geschlossene Kreis soll folgende Spezifikationen für einen Sollsprung $\Delta r = 20 \text{ rad s}^{-1}$ erfüllen:

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

Als Reglerstruktur wird

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

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

Hinweis: Verwenden Sie als Abtastzeit $T_a = 10 \text{ ms}$. Weiterhin sei erwähnt, dass in MATLAB folgende Prozeduren für die Transformationen zwischen s -, z - und q -Bereich zur Verfügung stehen:

$$\begin{aligned} G(s) \xrightarrow{T_a} G(z) & \quad \text{MATLAB-Befehl: } \text{Gz=c2d(Gs, Ta, 'zoh')} \\ G(z) \rightarrow G^\#(q) & \quad \text{MATLAB-Befehl: } \text{Gq=d2c(Gz, 'tustin')} \\ G^\#(q) \xrightarrow{T_a} G(z) & \quad \text{MATLAB-Befehl: } \text{Gz=c2d(Gq, Ta, 'tustin')} \end{aligned}$$

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 .

^aÜblicherweise wird Messrauschen als bandbeschränktes, weißes Rauschen modelliert.

2.3 Reglerentwurf - Einfluss der Abtastzeit T_a

Gegeben ist die Streckenübertragungsfunktion

$$G(s) = \frac{1.5}{1 + 2 \cdot 0.9 \cdot (s/2) + (s/2)^2}. \quad (2.13)$$

Aufgabe 2.8. Entwerfen Sie für das System (2.13) einen PI-Regler

$$R_1(s) = \frac{V_I(1 + sT_I)}{s} \quad (2.14)$$

mithilfe des Frequenzkennlinienverfahrens in MATLAB so, dass der geschlossene Regelkreis folgende Spezifikationen erfüllt:

bleibende Regelabweichung	$e_\infty _{r(t)=\sigma(t)} = 0$
Anstiegszeit	$t_r = 1.2 \text{ s}$
Überschwingen	$\ddot{u} \leq 10 \%$

Testen Sie in MATLAB, ob die Spezifikationen erfüllt werden.

Hinweis: Mithilfe des MATLAB-Befehls `step` können Sie direkt die Sprungantwort einer Übertragungsfunktion darstellen.

Da Regler heutzutage überwiegend auf Digitalrechnern implementiert werden, ist es notwendig, den Entwurf für die entsprechend abgetastete Strecke $G(z)$ durchzuführen. Dabei soll in weiterer Folge die Bedeutung der Wahl der Abtastzeit T_a untersucht werden.

1. Bestimmen Sie entsprechend den Vorgaben aus dem Skriptum Automatisierung [2.1] eine geeignete Abtastzeit T_a für das System (2.13). Hierbei soll die Abtastzeit T_a so gewählt werden, dass einerseits die dynamischen Vorgänge hinreichend gut erfasst werden und andererseits diese jedoch (im Hinblick auf eine Echtzeit-Implementierung) nicht unnötig klein ist.
2. Bestimmen Sie für die ermittelte Abtastzeit T_a die zu $G(s)$ gehörende q -Übertragungsfunktion $G^\#(q)$ in MATLAB und entwerfen Sie mithilfe des Frequenzkennlinienverfahrens einen Regler $R_2^\#(q)$ im q -Bereich so, dass der geschlossene Regelkreis die Spezifikationen erfüllt.

Implementieren Sie den kontinuierlich entworfenen Regler $R_1(s)$ und den im q -Bereich entworfenen Regler $R_2^\#(q)$ als Abtastsysteme an der abgetasteten Strecke $G(z)$. Stellen Sie die Sprungantwort der beiden geschlossenen Regelkreise mithilfe des MATLAB-Befehls `step` dar.

Erläutern Sie das Ergebnis. Welchen Einfluss hat in diesem Zusammenhang die Wahl der Abtastzeit T_a auf den geschlossenen Regelkreis, bei dem der Regler kontinuierlich entworfen aber als Abtastsystem implementiert wurde? Untersuchen Sie, wie T_a zu wählen ist, damit in diesem Fall trotzdem die Anforderung in Bezug auf das Überschwingen erfüllt wird.

Hinweis: Verwenden Sie gegebenenfalls den MATLAB-Befehl `feedback` zur Berechnung der Übertragungsfunktion des geschlossenen Kreises.

2.4 Lead-Lag-Reglerentwurf

Gegeben ist die Streckenübertragungsfunktion

$$G(s) = \frac{7(1 + \frac{s}{5})}{s(1 + 2 \cdot 0.8 \cdot \frac{s}{4.2} + (\frac{s}{4.2})^2)} . \quad (2.15)$$

Aufgabe 2.9. Entwerfen Sie einen Lead-Lag-Regler

$$R(s) = V R_{Lead}(s) R_{Lag}(s) , \quad (2.16)$$

wobei

$$R_{Lead}(s) = \frac{1 + sT_{Lead}}{1 + s\eta_{Lead}T_{Lead}} \quad 0 < \eta_{Lead} < 1$$

$$R_{Lag}(s) = \frac{1 + sT_{Lag}}{1 + s\eta_{Lag}T_{Lag}} \quad \eta_{Lag} > 1 ,$$

mithilfe des Frequenzkennlinienverfahrens in MATLAB/SIMULINK so, dass der geschlossene Regelkreis folgende Spezifikationen erfüllt:

$$\begin{aligned} \text{bleibende Regelabweichung} \quad e_{\infty}|_{r(t)=t} &= 5\% \\ \text{Anstiegszeit} \quad t_r &= 0.15 \text{ s} \\ \text{Überschwingen} \quad \ddot{u} &= 15\% \end{aligned}$$

Testen Sie in MATLAB/SIMULINK, ob die Spezifikationen erfüllt werden. Begründen Sie, warum in diesem Fall die Wahl eines Lead-Lag-Reglers sinnvoll ist. Zeichnen Sie dazu die Bode-Diagramme der Strecke, des Reglers, des offenen und des geschlossenen Kreises.

2.5 Literatur

- [2.1] A. Kugi, *Skriptum zur VU Automatisierung (WS 2020/2021)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2020. Adresse: <https://www.acin.tuwien.ac.at/bachelor/automatisierung/>.
- [2.2] P. Kokotovic, H. K. Khalil und J. O'Reilly, *Singular Perturbation 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 2020/21) [3.1]
 - Kapitel 7 bis Kapitel 8

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

- Georg Janisch <janisch@acin.tuwien.ac.at> oder
- Alexander Aschauer <aschauer@acin.tuwien.ac.at>.

Für organisatorische Fragen wenden Sie sich bitte an

- Florian Beck <beck@acin.tuwien.ac.at>.



Alle MATLAB/SIMULINK Dateien, die zum Bearbeiten dieser Übung benötigt werden, finden Sie in `uebung3.zip` auf der Homepage der Lehrveranstaltung.



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 $T_a = 10$ 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

$$\Delta u_k = \mathbf{k}^T \Delta \mathbf{x}_{red,k} + g \Delta r_k, \quad (3.1)$$

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

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

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

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 Reglerpa-

parameter \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 \mathbf{dx} und \mathbf{dr} bezeichnen den Zustand $\Delta\mathbf{x}_{red}$ und die Sollgröße Δr und der Blockausgang \mathbf{du} die berechnete Stellgröße Δu . Achten Sie auf eine korrekte Aufschaltung der Ruhelagen *außerhalb* des Blocks. Sämtliche Reglerparameter müssen als Parameter in der Struktur \mathbf{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 \mathbf{sw} dient zum Deaktivieren des Reglers: Gilt $\mathbf{sw}=0$, so muss der Reglerausgang $\mathbf{du}=0$ 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 ($\mathbf{sw}=1$) keine unzulässigen Stellamplituden ausgibt. Verifizieren Sie den Reglerentwurf anschließend am nichtlinearen Modell (2.7) der Gleichstrommaschine.



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 Sollgrößenprung $\Delta r = 20 \text{ rad s}^{-1} \sigma(t - 1)$.
2. Untersuchen Sie das Verhalten der Regler für eine Störung der Form $M_{ext} = -0.5 \text{ N m} \sigma(t - 5)$.
3. Um den Kompensationsregler später am Laborversuch testen zu können, muss der zeitdiskrete Regler $R(z)$ in Proportional-, Integral- sowie Kompensationsteil

aufgespalten werden,

$$R(z) = \left(V_P + \frac{V_I}{z-1} \right) R_{komp}(z). \quad (3.3)$$

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

3.2 Zustandsbeobachter

In der Praxis können oft nicht alle Zustände des Systems gemessen werden oder es muss aus Kostengründen auf Sensoren verzichtet werden. Um einen Zustandsregler für ein System zu entwerfen, bei dem nicht alle Zustände bekannt sind, soll nun ein Zustandsbeobachter entworfen werden.

Aufgabe 3.5. Nehmen Sie an, dass *nur* die Propellerdrehzahl ω_P des Laboraufbaues in Abb. 2.2 gemessen werden kann. Entwerfen Sie nun für die Gleichstrommaschine einen Zustandsbeobachter ausgehend vom zeitdiskreten, linearisierten, reduzierten Modell aus Aufgabe 3.1, um die Zustände $\hat{\mathbf{x}}_{red}^T = [\hat{\phi}_{GSMP} \quad \hat{\omega}_{GSM} \quad \hat{\omega}_P]$ zu schätzen.

1. Testen Sie zuvor das System auf vollständige Beobachtbarkeit.
2. Bestimmen Sie den Rückführvektor $\hat{\mathbf{k}}$ des Korrektors so, dass die Beobachterfehlerdynamik Pole $p_j = \exp(\lambda_e T_a), j = 1 \dots 3$ mit geeignetem $\lambda_e \in \mathbb{R}$ aufweist.

Hinweis: Schlagen Sie im Skriptum Automatisierung [3.1] nach, wie der Beobachterentwurf mit dem Entwurf eines Zustandsreglers im Zusammenhang steht. Wenden Sie den MATLAB-Befehl `acker` entsprechend an. Beachten Sie die vom Skriptum abweichende Vorzeichenkonvention.

Orientieren Sie sich bei der Wahl der Beobachterfehlerdynamik an Ihrer Wahl der Pole des geschlossenen Kreises mit Zustandsregler. In der Regel soll der Beobachtungsfehler schneller abklingen als der Regelfehler.

3. Implementieren Sie den Zustandsbeobachter zusammen mit dem nichtlinearen System und dem PI-Zustandsregler aus Aufgabe 3.3, wobei der Zustandsregler weiterhin die gemessenen Zustände \mathbf{x}_{red} verwenden soll. Vergleichen Sie die geschätzten Zustände $\hat{\mathbf{x}}_{red}$ mit den tatsächlichen Zuständen \mathbf{x}_{red} des Systems mit den Verläufen der Führungsgröße sowie der Störung aus Aufgabe 3.4.

Hinweis: Um sog. algebraic-loops zu vermeiden, verwenden Sie für den Zustandsbeobachter den Block `LTI-System`. Bestimmen Sie dazu den vollständigen Luenberger Beobachter in Zustandsraumdarstellung, wobei Sie \mathbf{C}^T in der Ausgangsgleichung gleich einer Einheitsmatrix passender Dimension wählen.

4. Testen Sie den PI-Zustandsregler mit den geschätzten Zuständen.

Aufgabe 3.6. Bei hohen Drehzahlen kommt es aufgrund des vernachlässigten nicht-linearen Reibanteils zu einer Abweichung der Zustandsschätzung. Weiters können die Zustände bei einer auftretenden Störgröße nur mit großem Fehler geschätzt werden. Um dem entgegenzuwirken, kann zusätzlich das Lastmoment M_{ext} mitgeschätzt werden.

1. Die Störung wird hierfür als unbekannt, aber konstant modelliert. Führen Sie dazu ausgehend vom linearisierten, reduzierten Modell aus Aufgabe 2.3 einen zusätzlichen konstanten Zustand \tilde{M}_{ext} mit $\frac{d}{dt}\tilde{M}_{ext} = 0$ und der Rückwirkung auf das System (entspricht dem Einfluss des Störeinganges d) ein. Zeitdiskretisieren Sie anschließend das resultierende Modell und führen Sie den Zustandsbeobachterentwurf aus Aufgabe 3.5 mit dem Zustandsvektor $\hat{\mathbf{x}}_{red,Last}^T = [\hat{\varphi}_{GSM} \quad \hat{\omega}_{GSM} \quad \hat{\omega}_P \quad \hat{M}_{ext}]$ erneut durch. Wählen Sie dazu einen geeigneten Pol des Beobachtungsfehlers für \hat{M}_{ext} .
2. Implementieren Sie den Zustandsbeobachter inklusive Lastschätzer am nichtlinearen System und diskutieren Sie die erreichte Regelgüte.

3.3 Praktischer Laborversuch

Je nach aktueller COVID-Situation können die entworfenen Regler an einem realen Versuchsaufbau getestet werden. Weitere Informationen über den Ablauf des Laborversuches werden zum Übungstermin verlautbart. Bis dahin ist für dieses Kapitel keine Vorbereitung notwendig.

3.3.1 Implementierung des Reglers am Versuchsaufbau

Implementieren und testen Sie den Kompensationsregler und den PI-Zustandsregler für die Gleichstrommaschine am entsprechenden Laborversuch. Bei diesem Versuchsaufbau wird das Reibmoment des Propellers sowie das zusätzliche externe Moment als Störung des Systems $d = M_{ext}$ mithilfe einer permanenterregten Synchronmaschine aufgebracht. Um einen fehlerfreien Kompiliervorgang auf der entsprechenden Echtzeit-Hardware zu ermöglichen, bereiten Sie die Regler in der folgenden Form auf:

- **Kompensationsregler:** Legen Sie den aufgespaltenen Kompensationsregler aus Aufgabe 3.4 in einer MATLAB-Struktur mit dem Namen `parKompReg` gemäß

$$\begin{aligned} \text{parKompReg.VP} &= V_P \\ \text{parKompReg.VI} &= V_I \\ \text{parKompReg.Rkompz} &= R_{komp}(z) \end{aligned}$$

ab. Dabei ist $R_{komp}(z)$ die z -Übertragungsfunktion des Kompensationsteils und $V_P, V_I \in \mathbb{R}$ sind die Parameter des Proportional- und Integralteils des Kompensationsreglers.

- **PI-Zustandsregler:** Speichern Sie den MATLAB Function Block mit der in Aufgabe 3.3 vorgegebenen Schnittstelle ab. Damit kann der Block direkt auf die Hardware kompiliert werden. Legen Sie darüber hinaus die vom Zustandsregler benötigten Parameter gemäß

$$\text{parZR.kP} = k_P$$
$$\text{parZR.kI} = k_I$$
$$\text{parZR.kx} = \mathbf{k}_x$$
$$\text{parZR.c} = \mathbf{c}$$
$$\text{parZR.Ta} = T_a$$

in der MATLAB-Struktur `parZR` ab.

Speichern Sie die beiden Strukturen `parKompReg` und `parZR` in einem `mat`-File (z.B. MATLAB-Befehl `save`) mit dem Namen `reglerXX.mat`, wobei `XX` für Ihre Gruppennummer steht.

3.4 Literatur

- [3.1] A. Kugi, *Skriptum zur VU Automatisierung (WS 2020/2021)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2020. Adresse: <https://www.acin.tuwien.ac.at/bachelor/automatisierung/>.