TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

D I P L O M A R B E I T

Analyse und Simulation dynamischer Systeme in Simulink mit Web–Interface

Ausgeführt am Institut für

Analysis und Scientific Computing

der Technischen Universität Wien

unter der Anleitung von

Ao.Univ.Prof. Dr. Felix Breitenecker

durch

Andreas Körner

Patrizigasse 7/16,
1210 Wien.

Wien, im Juni 2012 . . . . . .

Zusammenfassung

Ziele dieser Arbeit sind eine mathematische Beschreibung der Simulationsumgebung Simulink und die Darstellung von Modellen in dieser Umgebung, sowie die Einbindung von Simulink auf einem Webserver. Um dies mathematisch korrekt zu gestalten, wird in den ersten beiden Kapiteln eine Einführung in die Graphentheorie gegeben und davon ausgehend eine Definition von Signalflussgraphen gegeben.

Anschließend wird die zeitkontinuierliche und zeitdiskrete Systemtheorie für lineare zeitinvariante Systeme formuliert. Es wird eine Charakterisierung dieser Systeme vorgenommen und die unterschiedlichen Darstellungsformen werden analysiert. Um das Konzept auf nichtlineare Systeme zu erweitern, folgt ein Kapitel über numerische Lösungverfahren von gewöhnlichen Differentialgleichungen.
Es zeigt sich, dass die Systemtheorie zur Modellbeschreibung herangezogen wird, die Simulation hingegen mit der numerische Lösung der Differentialgleichungen im Zeitbereich arbeitet.

Aufbauend auf diesen theoretischen Grundlagen wird die Simulationsumgebung Simulink einschließlich der darin beteiligten Modelle, sowie die Konfiguration der Simulationsparameter, beschrieben.

Den Abschluss der Arbeit bildet ein Abschnitt über die Einbindung von Simulink auf einem Webserver, welcher eine Plattform für interaktive Experimente darstellt. Dort stehen typische Experimente der mathematischen Modellbildung und Simulation aus unterschiedlichsten technisch–naturwissenschaftlichen Bereichen bereit, welche MATLAB und Simulink als Simulationsumgebung verwenden.

Abstract

This thesis is about a mathematical description of the simulation environment Simulink. It starts with the definition and the characterisation of graphs and their extension to signal flow graphs. This mathematical background is necessary to understand the model description in Simulink.

The next part of the thesis consists of the system theory for characterising continuous and discrete linear time invariant systems. This theory combined with signal flow graphs is used to specify these systems. Furthermore the theory of numerical solutions for ordinary differential equations is introduced which is used to consider numerical simulations. These numerical methods are fundamentally to simulate linear and nonlinear systems in Simulink.

Based on the theoretical background the simulation environment Simulink and its models are introduced and the configuration of simulation parameters is studied.

The closing chapter describes the implementation of Simulink at the MATLAB–based MMT–Server which offers a web–interface to make interactive experiments in a natural scientific and technological setting.

Danksagung

An dieser Stelle möchte ich meinen ganz persönlichen Dank einigen Menschen zum Ausdruck bringen, die mir nicht nur im Zuge meiner Diplomarbeit eine große Hilfe waren.

Mein Dank gilt Herrn Ao.Univ.Prof. Felix Breitenecker, der mir die Möglichkeit gegeben hat, dieses Thema als Diplomarbeit zu bearbeiten, sowie meinen Studienkolleginnen und Studienkollegen, die mit mir gemeinsam das Studium bestritten und mir in vielen Situationen geholfen haben. Auch möchte ich meinem Bürokollegen Matthias Rößler für Rede und Antwort bei Grundsatzdiskussionen danken.

Nicht zuletzt ist es mir ein großes Anliegen, meinen Eltern für jegliche Unterstützung während meiner Studienzeit danke zu sagen.

Vor allem möchte ich Caroline danken, sie hat die schwierige Aufgabe der Durchsicht und des Korrekturlesens meiner Arbeit auf sich genommen und ist mir in der Zeit der Fertigstellung liebenswürdig zur Seite gestanden.

Wien, im Juni 2012 Andreas Körner

Man muss viel gelernt haben,
um über das, was man nicht weiß,
fragen zu können.

Jean-Jacques Rousseau

Inhaltsverzeichnis

Kapitel 1

Einleitung

Dieses erste Kapitel soll eine Einleitung und Zielsetzung für die folgende Arbeit darstellen, sowie ein Abriss für das Dokument an und für sich sein. Es werden zuerst die Umstände beschrieben, welche zur Formulierung dieses Diplomarbeitsthemas geführt haben. Ferner wird ein kurzer Überblick über die einzelnen Kapitel, den Aufbau und den Inhalt dieser Arbeit gegeben, sowie deren Zusammenhang hinsichtlich der Darstellung des Themas als Ganzes motiviert.

Ausgangspunkt des Themas ist der sogenannte MMT–Server, wobei MMT für *Mathematics, Modelling and Tools* steht. Der Server stellt ein Webinterface für interaktive Experimente mit MATLAB dar und dient dazu, Grundkonzepte der mathematischen Modellbildung und Simulation in diesem Fachgebiet mit demonstrativen Beispielen zu vermitteln.

MATLAB bietet eine Vielzahl von Toolboxen an, welche auf diverse technisch–naturwissenschaftliche Fachgebiete zugeschnitten sind und so spezielle Werkzeuge für diese Bereiche sind. Eine dieser Toolboxen ist Simulink. Diese unterscheidet sich aber grundsätzlich von MATLAB, da sie eine Oberfläche für grafische Modellbildung anbietet.
Da Simulink in vielen Disziplinen verwendet wurde, hat sich die Anzahl der Toolboxen, welche Simulink selbst erweitern, drastisch erhöht, was dazu führte, dass Simulink als eigenständiger Simulator betrachtet wird. Wohl ist MATLAB hier immer noch im Hintergrund beteiligt, jedoch ist der Ausgangspunkt für die Simulation das Modell in Simulink.

Die Bedeutung von Simulink und von graphisch basierten Simulatoren im Allgemeinen hat dazu geführt, dass auf dem oben beschriebenen MMT–Server auch Simulink als Simulator integriert werden musste. Das Resultat dieser Erweiterung ist Inhalt der vorliegenden Diplomarbeit mit dem Titel „Analyse und Simulation dynamischer Systeme in Simulink mit Web–Interface“.
Darin wird die Simulation dynamischer Systeme in Simulink mathematisch untersucht und analysiert, sowie die Einbindung von Simulink als Simulator mit Webinterface am MMT–Server vorgestellt.

Zum Einstieg in dieses Thema sei die Definition von dynamischen Systemen gegeben. Oft wird die Meinung vertreten, dass ein dynamisches System ein System darstellt, welches durch Differentialgleichungen beschrieben wird. Das ist jedoch nicht korrekt, da ein dynamisches System auch ohne eine Differentialgleichung definiert werden, jedoch unter gegebenen Voraussetzungen mit Differentialgleichungen in Zusammenhang gebracht werden kann.

Definition 1.1. Seien T und $X \neq \emptyset$ Mengen, wobei $T \in \{\mathbb{N}_0, \mathbb{Z}, \mathbb{R}_0^+, \mathbb{R}\}$, und sei $g\colon T \times X \to X$ eine Abbildung, für welche mit $x \in X$ und $t_1, t_2 \in T$ gilt

(a) $g(0, x) = x$,

(b) $g\big(t_1, g(t_2, x)\big) = g(t_1 + t_2, x)$.

Das Trippel (T, X, g) heißt *dynamisches System*, welches für $T \in \{\mathbb{N}_0, \mathbb{Z}\}$ als *zeitdiskret* und für $T \in \{\mathbb{R}_0, \mathbb{R}\}$ als *zeitkontinuierlich* bezeichnet wird. X heißt der *Zustandsraum* und g der *Fluss*. Um im Folgenden Operationen bezüglich t und x besser voneinander unterscheiden zu können, wird die Schreibweise

$$g_t(x) = g(t, x)$$

eingeführt.

Bemerkung 1.2. Die Bezeichnung von X als Zustandsraum legt den Verdacht nahe, dass es sich hier um einen Vektorraum handle, was in technisch–naturwissenschaftlichen Anwendungsbeispielen in der Regel auch zutrifft. Für die Definition eines dynamischen Systems ist diese Forderung jedoch nicht notwendig. Es bedarf lediglich einer Operation $+\colon T \times T \to T$, um die Anforderung in Definition 1.1 (b) an die Abbildung g stellen zu können. Die Bedingung (b) in Definition 1.1 wird als *Habgruppeneigenschaft* bezeichnet.

Satz 1.3. Sei $+$ eine kommutative Verknüpfung auf T. Dann gilt für $t, t_1, t_2 \in T$

(a) $g_0 = \mathrm{id}$,

(b) $g_{t_1+t_2} = g_{t_1} \circ g_{t_2} = g_{t_2} \circ g_{t_1}$,

(c) $g_t^{-1} = g_{-t}$.

Beweis. (a)–(c) gilt wegen

$$g_t\big(g_{-t}(x_0)\big) = g_{-t}\big(g_t(x_0)\big) = g_0(x_0) = \big(g_t^{-1} \circ g_t\big)(x_0) = \big(g_t \circ g_t^{-1}\big)(x_0).$$

□

Ein dynamisches System muss nicht durch Differentialgleichungen beschrieben werden, was im Fall eines zeitdiskreten Systems auch offensichtlich ist. Im zeitkontinuierlichen Fall $T \in \{\mathbb{R}_0^+, \mathbb{R}\}$ kann hingegen schon eine Beziehung bestehen, wie der folgende Absatz zeigt.

Sei $x\colon \mathbb{R} \to \mathbb{R}^n$, $x \in \mathcal{C}^1(\mathbb{R})$ und sei g einmal stetig differenzierbar. Mit $x(0) = x_0$ und dem Zusammenhang $x(t) = g_t(x_0)$ betrachtet man

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}x(t) &= \lim_{h\to 0}\frac{1}{h}\big(g_{t+h}(x_0) - g_t(x_0)\big) = \left(\left(\lim_{h\to 0}\frac{1}{h}(g_h - \mathrm{id})\right) \circ g_t\right)(x_0) = \\ &= \left(\frac{\partial}{\partial t}g_t\bigg|_{t=0} \circ g_t\right)(x_0) = \left(\frac{\partial}{\partial t}g_t\bigg|_{t=0}\right)\big(x(t)\big).\end{aligned}$$

Bei zeitkontinuierlichen dynamischen Systemen mit entsprechenden Voraussetzungen an x und g kann das dynamische System also durch

$$x'(t) = f\big(x(t)\big), \qquad \text{mit} \qquad f\big(x(t)\big) = \left(\frac{\partial}{\partial t}g_t\bigg|_{t=0}\right)\big(x(t)\big),$$

beschrieben werden. Aus diesem Grund wird g als Fluss, genauer als Fluss zu diesem Differentialgleichungssystem, bezeichnet.

Nach dieser Definition von dynamischen Systemen, inklusive der Beschreibung des Zusammenhangs von zeitkontinuierlichen dynamischen Systemen und Differentialgleichungssystemen, soll nun der weitere Aufbau der Arbeit in den folgenden Kapiteln beschrieben werden.

Kapitel 2 und 3 haben zum Ziel, die Grundlagen für die Beschreibung von Modellen in Simulink zu schaffen. In dieser Simulationsumgebung werden Modelle durch Signalflussgraphen beschrieben, welche an anderer Stelle noch mit zusätzlichen Elementen kombiniert werden. Es wird zuerst klassische Graphentheorie betrieben, welche anschließend auf die Theorie eines Signalflussgraphen erweitert wird. In diesem Kapitel 3 wird auch der Begriff des Signalraums, welcher im Wesentlichen mit quadratisch integrierbaren Funktionen zusammenhängt, eingeführt. Weiters wird das Konzept der Abtastung erläutert, welches weiter unten für die zeitdiskreten Modelle und der Implementierung der Simulation auf einem Digitalrechner von großem Interesse ist. Außerdem werden erste Elemente eines Signalflussgraphen vorgestellt, die in Simulink elementare Komponenten darstellen.

Das anschließende Kapitel 4 über lineare zeitinvariante Systeme ist grundlegend für die Beschreibung von Modellen in Simulink. Es wird die Systemtheorie eingeführt und eine Reihe von Beschreibungsmöglichkeiten für lineare zeitinvariante Modelle angegeben. Die vorerst lineare Systemtheorie wird in Kapitel 6 auf nichtlineare Systeme und deren Beschreibung erweitert. Obgleich die lineare Theorie oftmals nicht ausreicht, stellt sie grundlegende Methoden, Werkzeuge und Beschreibungen zur Verfügung, welche später erweitert, ausgebaut und verallgemeinert werden. Es werden sowohl zeitkontinuierliche als auch zeitdiskrete Systeme beschrieben. Mit diesen Konzepten ausgestattet ist es möglich, Modelle in Simulink als Signalflussgraphen aufzufassen. Dies ermöglicht es, alle bekannten Algorithmen der Graphentheorie zur informationstechnischen Modellbeschreibung verwenden zu können.

Eines der zentralsten Elemente eines numerischen Simulators liefert Kapitel 5. Für lineare Systeme bietet die Systemtheorie elegante Beschreibungs– und Berechnungsmethoden. Sowohl lineare als auch nichtlineare zeitkontinuierliche Systeme vereint die Eigenschaft, durch Differentialgleichungssysteme beschrieben werden zu können. Dieses Kapitel zeigt die numerische Behandlung dieser Differentialgleichungssysteme und stellt damit ein Werkzeug dar, welches lineare als auch nichtlineare Systeme handhaben kann. Es werden Lösungsverfahren für Anfangswertprobleme gewöhnlicher Differentialgleichungen vorgestellt, wobei der Fokus auf Methoden liegt, die später in der Simulationsumgebung wiederkehren.
Obgleich die lineare Systemtheorie äußerst elegante Beschreibungsmethoden mit der Laplace–Transformation anbietet, wird in der Simulationsumgebung immer das zugrundeliegende Differentialgleichungssystem simuliert. Die oben erwähnten Beschreibungsformen sind also für die Modellbildung unerlässlich, für die numerische Berechnung der Ergebnisse in der Simulation ist hingegen das numerische Lösen der Differentialgleichungen erforderlich. Dieser Umstand macht es möglich, auch nichtlineare Systeme beschreiben zu können. In einem der Unterkapitel wird speziell ein Blick auf die Lösungsverfahren geworfen, die bei der Simulation in Simulink eine wichtige Rolle spielen.

In Kapitel 6 wird schließlich die Simulationsumgebung Simulink dargestellt. Es werden der Simulator, diverse Simulationsparameter sowie Elemente von Simulink vorgestellt. Zum besseren Verständnis werden einige Beispiele gezeigt, welche die Darstellung diverser Begriffe illustrie-

ren. Zusätzlich werden Systeme vorgestellt, welche in den vorangegangenen Kapiteln bereits theoretisch beleuchtet wurden.

Das inhaltliche Ende stellt Kapitel 7 dar. In diesem wird das Einbinden von Simulink in den MMT–Server erläutert und einige Experimente, welche mit Simulink als Simulator am Server implementiert sind, erklärt. Die Auswahl der Experimente beruht auf der Abdeckung eines breiten technisch–naturwissenschaftlichen Themenbereichs, um die weitverbreitete Anwendung von Simulink in unterschiedlichsten physikalischen Domänen zu unterstreichen.

Den Abschluss der Arbeit bilden einige abschließende Bemerkungen in Kapitel 8. In diesem werden eine kurze Zusammenfassung, einige inhaltliche Anmerkungen, sowie ein Ausblick auf mögliche Fortsetzungen dieser Arbeit hinsichtlich thematischer Inhalte und der Erweiterung des MMT–Servers geboten. Im Anhang befinden sich inhaltliche Ergänzungen, welche an diversen Stellen als Hilfsmittel oder Hilfssätze herangezogen werden.

Kapitel 2

Gerichtete und gewichtete Graphen

Dieses Kapitel behandelt die Graphentheorie, welche im darauffolgenden Kapitel zur Definition von Signalflussgraphen benötigt wird. Diese Signalflussgraphen werden dann schließlich zur Beschreibung von Modellen in Simulink verwendet.

2.1 Grundlegende Definitionen und Sätze

Definition 2.1. Sei $V = V(G)$ die *Knotenmenge* und $E = E(G)$ die *Kantenmenge*. Dann heißt $G = (V, E)$ ein *Graph* und es ist $e \in E(G)$ eine Kante. Die Kante heißt *gerichtet*, wenn diese durch ein geordnetes Paar $e = \langle v_1, v_2 \rangle$ von Knoten $v_1, v_2 \in V(G)$, dem *Anfangsknoten* v_1 und dem *Endknoten* v_2, dargestellt wird. Die Kante heißt *ungerichtet*, wenn diese durch ein ungeordnetes Paar $e = (v_1, v_2)$ von Knoten $v_1, v_2 \in V(G)$ dargestellt wird.

Weiters trifft man die folgenden Begriffsdefinitionen:

(a) Eine Kante der Form $\langle v, v \rangle$ oder (v, v) heißt eine *Schlinge*.

(b) Sind zwei Knoten $v, w \in V(G)$ durch eine Kante e verbunden, so heißen diese *adjazent*. Die Knoten v, w *inzidieren* mit der Kante e, die sie verbindet.

(c) Ein Graph $G = (V, E)$ heißt *schlicht* oder *einfach*, wenn dieser keine Schlingen besitzt.

Es ist grundsätzlich auch möglich, Mehrfachkanten zu betrachten, also dass es mehr als eine Kante zwischen denselben Knoten gibt. Dies ist hier aber wegen der Identifizierung mit Paaren von Knoten nicht möglich und wird später nicht gebraucht.

Definition 2.2. Sei $G = (V, E)$ ein ungerichteter, schlichter Graph. Die Menge der zu $v \in V(G)$ adjazenten Knoten

$$\Gamma(v) = \{w \in V(G) \colon (v, w) \in E(G)\}$$

heißen *Nachbarn* von v. Die Anzahl der Nachbarn von $v \in V(G)$

$$d(v) = |\Gamma(v)|$$

heißt der *Knotengrad* von $v \in V(G)$.

Definition 2.3. Sei $G = (V, E)$ ein gerichteter Graph. Die Elemente von

$$\Gamma^+(v) = \{v \in V(G) \colon \langle v, w\rangle \in E(G)\}$$

heißen *Nachfolger* des Knotens $v \in V(G)$ und die Elemente von

$$\Gamma^-(v) = \{v \in V(G) \colon \langle w, v\rangle \in E(G)\}$$

heißen *Vorgänger* des Knotens $v \in V(G)$. Es bilden

$$\Gamma(v) = \Gamma^+(v) \cup \Gamma^-(v)$$

die Nachbarn von $v \in V(G)$. Die Anzahl $d^+(v) = |\Gamma^+(v)|$ heißt der *Weggrad* von $v \in V(G)$ und die Anzahl $d^-(v) = |\Gamma^-(v)|$ heißt der *Hingrad* von $v \in V(G)$.

Bemerkung 2.4. Bei der Definition für gerichtete Graphen sind Schlingen zugelassen, beim ungerichteten Graphen würde die Zulassung von Schlingen zur doppelten Zählung führen.

Satz 2.5. In einem ungerichteten schlichten Graphen $G = (V, E)$ gilt

$$\sum_{v \in V(G)} d(v) = 2|E(G)|,$$

in einem gerichteten Graphen gilt

$$\sum_{v \in V(G)} d^+(v) = \sum_{v \in V(G)} d^-(v) = |E(G)|.$$

Definition 2.6. Ein Graph $G' = (V', E')$ heißt ein Teilgraph eines Graphen $G = (V, E)$, wenn

$$V' \subseteq V \qquad \text{und} \qquad E' \subseteq E$$

gilt. In diesem Fall notiert man $G' \subseteq G$.

Definition 2.7. Speziell die Kanten betrachtend werden die folgenden Definitionen getroffen:

(a) Sei $G = (V, E)$ ein ungerichteter Graph mit den Kanten $e_1, e_2, \ldots, e_k \in E(G)$. Es heißt (e_k) eine *Kantenfolge*, wenn Knoten $v, v_1, v_2, \ldots, v_{k-1}, w \in V(G)$ existieren mit

$$e_1 = (v, v_1),\ e_2 = (v_1, v_2),\ \ldots,\ e_{k-1} = (v_{k-2}, v_{k-1}),\ e_k = (v_{k-1}, w).$$

Die Anzahl k heißt die *Länge* der Kantenfolge. Eine Kantenfolge der Länge 0 besteht aus keiner Kante und wird als leere Kantenfolge bezeichnet, welche jeden Knoten mit sich selbst verbindet, also eine Schlinge darstellt.

(b) Sei $G = (V, E)$ ein gerichteter Graph mit den Kanten $e_1, e_2, \ldots, e_k \in E(G)$. Es heißt (e_k) eine *Kantenfolge*, wenn für je zwei aufeinanderfolgende Kanten e_k und e_{k+1} mit $1 \leq j < k$ der Endknoten von e_j mit dem Anfangsknoten von e_{j+1} übereinstimmt, d.h. es existieren Knoten $v, v_1, v_2, \ldots, v_{k-1}, w \in V(G)$ mit

$$e_1 = \langle v, v_1\rangle,\ e_2 = \langle v_1, v_2\rangle,\ \ldots,\ e_{k-1} = \langle v_{k-2}, v_{k-1}\rangle,\ e_k = \langle v_{k-1}, w\rangle.$$

Die Anzahl k der Kantenfolge heißt *Länge*.

(c) Eine Kantenfolge (e_k) heißt *Kantenzug*, wenn alle Kanten e_j mit $1 \leq j \leq k$ voneinander verschieden sind.

(d) Eine Kantenfolge (e_k) heißt *Weg*, wenn alle Knoten, die mit den Kanten e_j mit $1 \leq j \leq k$ inzidieren, voneinander verschieden sind.

(e) Eine Kantenfolge (e_k) heißt *geschlossen*, wenn diese einen Knoten $v \in V(G)$ mit sich selbst verbindet.

(f) Eine geschlossene Kantenfolge in einem ungerichteten Graphen heißt *Kreis*, wenn alle Knoten der Kantenfolge mit Ausnahme des Anfangsknotens, der mit dem Endknoten übereinstimmt, voneinander verschieden sind und keine Kante mehrfach durchlaufen wird.

(g) Eine geschlossene Kantenfolge in einem gerichteten Graphen heißt *Zyklus*, wenn alle Knoten dieser Kantenfolge mit Ausnahme des Anfangsknotens, der mit dem Endknoten übereinstimmt, voneinander verschieden sind.

(h) Ein gerichteter Graph G heißt *azyklisch*, wenn er keine Zyklen enthält.

Satz 2.8. Sind in einem Graphen zwei Knoten durch eine Kantenfolge verbunden, so existiert auch mindestens ein Weg, der diese zwei Knoten miteinander verbindet. Existiert eine geschlossene Kantenfolge positiver Länge, so existiert auch ein Kreis bzw. Zyklus positiver Länge.

Definition 2.9. Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ heißen *isomorph*, wenn eine bijektive Abbildung

$$\varphi\colon V(G_1) \to V(G_2)$$

existiert, sodass eine Kante (v, w) bzw. $\langle v, w\rangle$ genau dann in $E(G_1)$ enthalten ist, wenn die Kante $(\varphi(v), \varphi(w))$ bzw. $\langle \varphi(v), \varphi(w)\rangle$ in $E(G_2)$ enthalten ist.

Definition 2.10. Eine Kantenfolge (e_k) eines Graphen G heißt eine *Eulersche Linie*, wenn sie jeden Knoten und genau einmal jede Kante durchläuft.

2.2 Adjazenzmatrix und Inzidenzmatrix eines Graphen

Definition 2.11. Sei $G = (V, E)$ ein Graph mit der Knotenmenge $V(G) = \{v_1, v_2, \ldots, v_n\}$. Die Matrix $A(G) = (a_{ij})_{1 \leq i,j \leq n}$, gegeben durch

$$a_{ij} = \begin{cases} 1 & \text{für } (v_i, v_j) \in E(G) \quad \text{bzw.} \quad \langle v_i, v_j\rangle \in E(G), \\ 0 & \text{sonst,} \end{cases}$$

heißt *Adjazenzmatrix* des Graphen G.

Korollar 2.12. Die Adjazenzmatrix eines ungerichteten Graphen ist symmetrisch und es gilt:

(a) Schlingen eines Graphen haben in den Einträgen der Hauptdiagonale eine 1 zur Folge.

(b) Schlichte Graphen haben eine verschwindende Hauptdiagonale.

(c) Mit Hilfe der Adjazenzmatrix lassen sich die Knotengrade direkt ablesen.

Satz 2.13. Sei $A(G) = (a_{ij})_{1\le i,j\le n}$ die Adjazenzmatrix eines Graphen $G = (V, E)$ mit der Knotenmenge $V(G) = \{v_1, \dots, v_n\}$. Ist G ungerichtet und schlicht, so gilt

$$d(v_i) = \sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} a_{ji}.$$

Ist G gerichtet, so gilt

$$d^+(v_i) = \sum_{j=1}^{n} a_{ij} \qquad \text{und} \qquad d^-(v_i) = \sum_{j=1}^{n} a_{ji}.$$

Definition 2.14. Sei $G = (V, E)$ ein Graph mit Knotenmenge $V(G) = \{v_1, \dots, v_n\}$ und Kantenmenge $E(G) = \{e_1, \dots, e_m\}$.

(a) Ist G ein ungerichteter schlichter Graph, so heißt die Matrix $B(G) = (b_{ij})_{1\le i\le n, 1\le j\le m}$, gegeben durch

$$b_{ij} = \begin{cases} 1 & \text{wenn } v_i \text{ mit } e_j \text{ inzidiert,} \\ 0 & \text{sonst,} \end{cases}$$

Inzidenzmatrix des Graphen G.

(b) Ist G ein gerichteter Graph, so heißt die Matrix $B(G) = (b_{ij})_{1\le i\le n, 1\le j\le m}$, gegeben durch

$$b_{ij} = \begin{cases} 1 & \text{wenn } v_i \text{ Anfangsknoten von } e_j \text{ ist,} \\ -1 & \text{wenn } v_i \text{ Endknoten von } e_j \text{ ist} \\ 0 & \text{sonst} \end{cases}$$

Inzidenzmatrix des Graphen G.

2.3 Zusammenhang von Graphen

Definition 2.15. Sei $G = (V, E)$ ein ungerichteter Graph. G heißt *zusammenhängend*, wenn es zwischen je zwei Knoten $v, w \in V(G)$ einen Weg gibt.

Die maximal zusammenhängenden Teilgraphen eines ungerichteten Graphen G heißen *Zusammenhangskomponenten* von G.

Definition 2.16. Sei $G = (V, E)$ ein gerichteter Graph. G heißt *stark zusammenhängend*, wenn für je zwei Knoten $v, w \in V(G)$ gerichtete Wege von v nach w und w nach v existieren.

Die maximal stark zusammenhängenden Teilgraphen eines gerichteten Graphen G heißen *starke Zusammenhangskomponenten* oder Komponenten des starken Zusammenhangs von G.

Definition 2.17. Sei G ein gerichteter Graph. Der ungerichtete Graph G_u, bestehend aus der selben Knotenmenge $V(G_u) = V(G)$ wie G und der Festlegung

$$(v, w) \in E(G_u) \qquad \Longleftrightarrow \qquad \langle v, w\rangle \in E(G) \quad \text{oder} \quad \langle w, v\rangle \in E(G),$$

heißt der *Schatten* des Graphen G.

Beim Schatten eines gerichteten Graphens werden also gerichtete Kanten durch ungerichtete ersetzt.

Definition 2.18. Ein gerichteter Graph G heißt *schwach zusammenhängend*, wenn der Schatten G_u zusammenhängend ist. Damit sind *schwache Zusammenhangskomponenten* die Zusammenhangskomponenten von G_u.

Definition 2.19. Sei G ein Graph.

(a) Ein schlichter Graph G, der keine Kreise enthält, heißt *Wald*.

(b) Ein Wald G, der zusammenhängend ist, heißt *Baum*.

Korollar 2.20. Die Zusammenhangskomponenten von einem Wald sind Bäume.

Definition 2.21. Sei T ein Baum.

(a) Wird ein Baum von einem ausgezeichneten Knoten $r \in E(T)$ aus betrachtet, so bezeichnet man diesen Knoten als *Wurzel*.

(b) Ein Baum T mit einer ausgezeichneten Wurzel heißt *Wurzelbaum*.

(c) Ein Knoten $b \in V(T)$ eines Wurzelbaumes T mit Knotengrad $d(b) = 1$ heißt *externer Knoten* oder *Blatt*. Alle anderen Knoten $v \in V(T)$ heißen *interne Knoten*.

Definition 2.22. Ein ungerichteter, schlichter Graph mit $n \in \mathbb{N}$ Knoten, bei dem jeder Knoten mit jedem anderen Knoten durch eine Kante verbunden ist, heißt der *vollständige Graph* C_n.

2.4 Gewichtete Graphen

Definition 2.23. Sei $G = (V, E)$ ein Graph und sei $w\colon E \to \mathbb{R}_0^+$. Das Tripel (V, E, w) heißt *gewichteter Graph* oder *bewerteter Graph* und w die *Gewichtsfunktion*.

Definition 2.24. Sei (V, E, w) ein gewichteter Graph und (e_k) eine Kantenfolge des Graphen. Das *Gewicht der Kantenfolge* ist definiert als

$$w\big((e_k)\big) = \sum_{i=1}^{k} w(e_k).$$

Definition 2.25. Sei (V, E, w) ein gewichteter Graph.

(a) Der *Abstand* d zwischen den beiden Knoten $x, y \in V(G)$ ist definiert als

$$d(x, y) = \min_{(e_k)} \{w\big((e_k)\big)\colon (e_k) \text{ ist Kantenfolge mit } e_1 = x \text{ und } e_k = y\}.$$

(b) Für jede Teilmenge $F \subseteq G$ heißt

$$w(F) = \sum_{e \in E(F)} w(e)$$

das *Gewicht* von F.

Für Graphen gibt es graphische Visualisierungen, die meist in Form von Punkten und Linienzügen erfolgen. Liegt ein gewichteter Graph vor, so wird dieses Gewicht an der betreffenden Kante notiert, vergleiche Abbildung 2.1 (a). Eine alternative Darstellung wird in Abbidung 2.1 (b) gezeigt. Hier ist das Gewicht als Eigenschaft der Kante integriert, was weiter unten die Formulierung von Gleichungen bei Signalflussgraphen erlaubt.

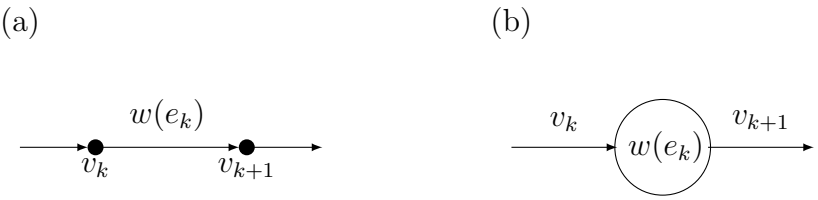

Abbildung 2.1: Visualisierung einer Kante e_k in gerichteten gewichteten Graphen: (a) klassische Darstellung, (b) alternative Darstellung

Kapitel 3

Signalflussgraphen

Der Abschnitt Signalflussgraphen bildet den ersten Teil der Beschreibung des Zusammenhangs zwischen Graphen– und Systemtheorie. Graphentheorie kann dazu herangezogen werden, lineare zeitinvariante Systeme, welche in Abschnitt 4 betrachtet werden, zu beschreiben. Dies stellt die Grundlage der graphischen Modellbeschreibung in Simulink dar.

3.1 Grundlegende Definitionen und Eigenschaften

Definition 3.1. Sei $G_w = (V, E, w)$ ein gerichteter und gewichteter Graph mit der Knotenmenge $V = \{v_1, \ldots, v_k\}$, der Kantenmenge $E = \{e_1, \ldots, e_m\}$ und der Gewichtsfunktion $w\colon E \to \mathbb{R}_0^+$. Ferner sei $X = \{x_1, \ldots, x_k\}$ mit $x_i \in \mathbb{R}$ für $i = 1, \ldots, k$. Die Abbildung

$$\mu\colon V \to X,\ v_i \mapsto x_i$$

ordnet jedem Knoten v_i einen Wert x_i zu. Die Werte $x_j = \mu(v_j),\ x_\ell = \mu(v_\ell)$ der Menge X stehen dabei über die Gewichtsfunktion w gemäß

$$x_\ell = x_j \cdot w(e_j),$$

wenn $e_j = \langle v_j, v_\ell \rangle$ gilt, zueinander in Relation. Enden an einem Knoten v_ℓ mehr als eine gerichtete Kante e_j, also gilt für $r \in \mathbb{N}$

$$e_j = \langle v_j, v_\ell \rangle\,,\ e_{j+1} = \langle v_{j+1}, v_\ell \rangle\,,\ \ldots, e_{j+r} = \langle v_{j+r}, v_\ell \rangle\,,$$

so gilt für x_ℓ die Relation

$$x_\ell = \sum_{s=0}^{r} x_{j+s} \cdot w(e_{j+s}).$$

Es heißt $S = (G_w, \mu)$ ein *Signalflussgraph*.

Für Signalflussgraphen gibt es ebenfalls eine graphische Visualisierung, welche jener der gerichteten und gewichteten Graphen ähnlich ist. In Abbildung 3.1 sind zwei alternative Visualisierungen des Ausschnittes eines Signalflussgraphen dargestellt, sowie der Fall, dass ein Knoten v_ℓ zweifacher Zielknoten ist.

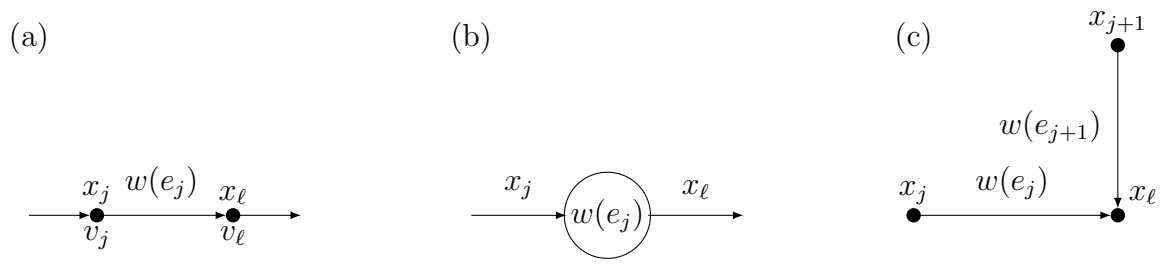

Abbildung 3.1: Visualisierung eines Ausschnittes eines Signalflussgraphen: (a) klassische Darstellung, (b) alternative Darstellung, (c) zweifacher Zielknoten

Der Name Signalflussgraph stammt von der Idee, dass die Werte der Menge $X = \{x_1, \ldots, x_n\}$ Signalwerte repräsentieren, welche durch Gewichtsfunktionen im Laufe des Flusses durch den Graphen manipuliert werden. Dieses Prinzip findet sich bei Simulink wieder.

Ein einfaches Beispiel soll diese Idee illustrieren.

Beispiel 3.2. Betrachtet wird ein Signalflussgraph S mit Knotenmenge $V = \{v_1, v_2, v_3\}$ und Kantenmenge $E = \{e_1, e_2\}$, wobei gilt

$$e_1 = \langle v_1, v_2 \rangle \qquad \text{und} \qquad e_2 = \langle v_3, v_2 \rangle .$$

Für die Gewichtsfunktion w gilt

$$w(e_1) = a_{1,2}, \qquad w(e_2) = a_{3,2}$$

und für die Abbildung μ gilt

$$\mu(v_1) = x_1, \qquad \mu(v_2) = x_2 \quad \text{und} \quad \mu(v_3) = x_3.$$

Durch diese Festlegungen resultiert ein Signalflussgraph, dessen graphische Repräsentation in Abbildung 3.2 zu sehen ist.

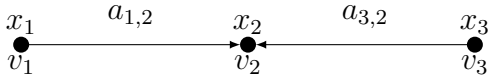

Abbildung 3.2: Visualisierung des Signalflussgraphen aus Beispiel 3.2

Mit den obigen Definitionen der Wirkungen der Gewichtsfunktionen auf die Werte x_i kann dieser Signalflussgraph durch die algebraische Gleichung

$$x_2 = a_{1,2}x_1 + a_{3,2}x_3$$

beschrieben werden. Dabei ist zu erkennen, dass die Werte $a_{i,j}$ die Gewichtsfunktionen vom Knoten i zum Knoten j darstellen und damit eine Relation zwischen den Werten x_i der Menge X hergestellt wird.

3.2 Verallgemeinertes Konzept eines Signalflussgraphen im Signalraum

Die bisher behandelten grundlegenden Eigenschaften eines Signalflussgraphen beschränken sich auf eine Verknüpfung der Werte x_j und x_ℓ durch die Gewichtsfunktion w entlang der Kante e_j des Graphens gemäß

$$x_j = x_\ell \cdot w(e_j).$$

Nun ist in einigen Fällen eine allgemeinere Verknüpfung der beiden Werte x_j und x_k wünschenswert, was eine Verallgemeinerung der Gewichtsfunktion w erfordert. Des Weiteren werden Signalflussgraphen zur Beschreibung dynamischer Systeme herangezogen. Der Graph soll das Verhalten von Signalen beschreiben. Dazu ist es notwendig, das Konzept des Signalflussgraphen von Werten der Menge X auf Elemente des sogenannten Signalraums zu erweitern.

Dieser Abschnitt widmet sich der Erweiterung des grundlegenden Konzepts, um diesen Anforderungen gerecht zu werden.

Definition 3.3. Sei S ein Signalflussgraph. Es bezeichne

$$w_{e_j} \colon X \to X,\ x_j \mapsto x_\ell = w_{e_j}(x_j)$$

die *verallgemeinerte Gewichtsfunktion* des Signalflussgraphen S.

Diese Definition erlaubt es, eine größere Klasse von Gewichtsfunktionen und somit eine größere Anzahl an Realisierungen der Abbildung $x_j \mapsto x_\ell$ zu betrachten. Um die Notation einfacher zu gestalten wird in Zusammenhängen, wo es auf die Kanten der Gewichtsfunktion nicht ankommt, die Verknüpfung als $x_\ell = w(x_j)$ geschrieben.

In Abbildung 3.1(b) wurde bereits eine alternative graphische Darstellung eines Signalflussgraphen gezeigt, welche hier erst ihre volle Bedeutung erlangt. Die verallgemeinerte Gewichtsfunktion gibt dabei an, wie der Wert x_j auf den Wert x_ℓ abgebildet wird. Diese Darstellungsform wird auch als *Eingangs–Ausgangs–Beziehung* oder *Eingangs–Ausgangs–Relation* bezeichnet und ermöglicht weiter unten ein Zusammenführen zweier Betrachtungen.

Definition 3.4. Sei $\Omega \subseteq \mathbb{R}$ und sei $f \colon \Omega \to \mathbb{R}$ eine Funktion.

(a) Eine Funktion f heißt *Signal*, wenn dieses, aufgefasst als Verlauf über die Zeit t, Information einer messbaren physikalischen Größe trägt.

(b) Ein Signal heißt *kausal*, wenn $f(t) = 0$ für alle $t < 0$ gilt. Verallgemeinert auch $f(t) = 0$ für alle $t < t_0$.

(c) Es heißt

$$\mathrm{L}^2(\Omega) = \left\{ f \colon \Omega \to \mathbb{R} \colon \int_\Omega |f(x)|^2 \,\mathrm{d}x < \infty \right\}$$

die *Menge aller quadratisch integrierbaren Funktionen*.

(d) Für $f, g \in \mathrm{L}^2(\Omega)$ und $s \in \mathbb{R}$ ist durch

$$(f+g)(x) = f(x) + g(x) \qquad \text{und} \qquad (s \cdot f)(x) = s \cdot f(x)$$

eine *Addition* $+ \colon \mathrm{L}^2(\Omega) \to \mathrm{L}^2(\Omega)$ und eine *Multiplikation* $\cdot \colon \mathrm{L}^2(\Omega) \to \mathrm{L}^2(\Omega)$ definiert.

Satz 3.5 (Vektorraum der quadratisch integrierbaren Funktionen)**.** Es bildet $(\mathrm{L}^2(\Omega), +, \cdot)$ einen Vektorraum. Durch

$$\langle f, g\rangle_{\mathrm{L}^2} = \int_\Omega f(x)g(x)\,\mathrm{d}x$$

ist ein Skalarprodukt definiert mit der induzierten Norm

$$\|f\|_{\mathrm{L}^2} = \sqrt{\langle f, f\rangle_{\mathrm{L}^2}} = \sqrt{\int_\Omega |f(x)|^2\,\mathrm{d}x}.$$

Der Vektorraum $(\mathrm{L}^2(\Omega), +, \cdot)$ ist bezüglich $\|\cdot\|_{\mathrm{L}^2}$ vollständig und damit ein *Hilbertraum.*

Definition 3.6 (Signalraum)**.** Der Hilbertraum $(\mathrm{L}^2(\Omega), +, \cdot)$ mit dem Skalarprodukt $\langle\cdot,\cdot\rangle_{\mathrm{L}^2}$, dessen Elemente Signale repräsentieren, heißt *Signalraum.* Auch hier wird die Bezeichnung L^2 verwendet, für kausale Signale beispielsweise $\mathrm{L}^2(\mathbb{R}_0^+)$.

Mit dem Begriff des Signalraumes lässt sich nun auch das oben angesprochene Konzept des Signalflussgraphen verallgemeinern. Die Werte aus der Menge X werden nun jeweils durch den Wert eines Signals ersetzt.

Definition 3.7 (Verallgemeinerter Signalflussgraph)**.** Sei $G_w = (V, E, w)$ ein gerichteter und gewichteter Graph mit der Knotenmenge $V = \{v_1, \ldots, v_k\}$, der Kantenmenge $E = \{e_1, \ldots, e_m\}$ und der Gewichtsfunktion $w\colon E \to \mathbb{R}_0^+$. Ferner seien $x_i \in \mathrm{L}^2(I)$ für $i = 1, \ldots, k$ und $I \subseteq \mathbb{R}$. Die Abbildung

$$\mu\colon V \to X,\ v_i \mapsto x_i$$

ordnet jedem Knoten v_i ein Signal x_i zu. Die Werte $x_j = \mu(v_j),\ x_\ell = \mu(v_\ell)$ der Menge X stehen dabei über die verallgemeinerte Gewichtsfunktion w gemäß

$$x_\ell(t) = w_{e_j}\big(x_j(t)\big),$$

wenn $e_j = \langle v_j, v_\ell\rangle$ gilt, zueinander in Relation. Enden an einem Knoten v_ℓ mehr als eine gerichtete Kante e_j, also sei für $r \in \mathbb{N}$

$$e_j = \langle v_j, v_\ell\rangle\,,\ e_{j+1} = \langle v_{j+1}, v_\ell\rangle\,,\ \ldots, e_{j+r} = \langle v_{j+r}, v_\ell\rangle\,,$$

so gilt für x_ℓ die Relation

$$x_\ell = \sum_{s=0}^{r} w_{e_{j+s}}\left(x_{j+s}\right).$$

Es heißt $S = (G_w, \mathrm{L}^2(I), \mu)$ ein *verallgemeinerter Signalflussgraph.*

Bemerkung 3.8. In der Literatur wird nicht zwischen den beiden Definitionen des Signalflussgraphen und des verallgemeinerten Signalflussgraphen unterschieden. In der Darstellung eines Signalflussgraphens ist stets ersichtlich, um welchen der Beiden es sich handelt. Im Folgenden wird als Signalflussgraph immer ein Graph verstanden, welcher der betreffenden Situation angepasst ist.

Abschließend soll der Zusammenhang zwischen den Werten der Menge X und den Signalen aus L^2 hergestellt werden. Auf einem Digitalrechner steht bei einer Simulation nicht das Signal

aus dem Funktionenraum zur Verfügung, sondern eine diskretisierte Variante. Die Beziehung zwischen den beiden Situationen stellt die sogenannte Abtastung zeitkontinuierlicher Signale her.

Definition 3.9. Sei $I = [a, b] \subset \mathbb{R}$ mit $a < b$ ein abgeschlossenes Intervall und sei $x \in \mathrm{L}^2(I)$ ein Signal. Ferner bezeichne $\mathcal{Z} = \{t_1, t_2, \ldots, t_{k-1}, t_k\}$ eine *Zerlegung* des Intervalls I mit $t_1 = a$ und $t_k = b$. Gilt $t_{i+1} - t_i = \Delta t$ für alle $i = 1, \ldots, k$, so heißt die Zerlegung *äquidistant*. Durch die Festlegung

$$x(t_i) = x_i$$

ist eine Abbildung von $\delta_a \colon \mathrm{L}^2(I) \to \mathbb{R}$ definiert, welche als *Abtastung* bezeichnet wird.

Durch die Abtastung wird eine Möglichkeit gegeben, aus der Menge $\mathrm{L}^2(I)$ eine Menge X zu konstruieren, welche den Zusammenhang zwischen einem verallgemeinerten und einem gewöhnlichen Signalflussgraphen erklärt. Man betrachte dazu die Abbildung

$$\delta_a \colon \mathrm{L}^2(I) \to \mathbb{R},\ x \mapsto x(t_i)$$

und konstruiere die Menge X durch

$$X = \big\{x_1 = x(t_1), x_2 = x(t_2) \ldots, x_k = x(t_k)\big\}.$$

In Digitalrechnern stehen nur die abgetasteten Werte eines Signalflussgraphen zur Simulation zur Verfügung.

Die strikte Unterscheidung zwischen einem verallgemeinerten und einem gewöhnlichen Signalflussgraphen ist notwendig, um entsprechende Relationen zwischen den unterschiedlichen Knoten bzw. den Werten oder Signalen, welche den einzelnen Knoten zugeordnet sind, angeben zu können. Bei verallgemeinerten Signalflussgraphen gibt es eine größere Anzahl an Relationen als bei gewöhnlichen Signalflussgraphen. Einen Überblick dazu wird der nächste Abschnitt geben.

3.3 Einige Elemente eines Signalflussgraphen

In diesem Abschnitt werden einige wichtige Elemente eines Signalflussgraphen eingeführt, welche im Kapitel über lineare zeitinvariante Systeme noch genauer motiviert werden.

Eine elementare Operation in einem Signalflussgraphen ist die *Multiplikation*. Dabei wird das Signal x_j an einem Knoten v_j mit einer Konstanten $a \in \mathbb{R}$ multipliziert, das Resultat ist gleich dem Signal x_ℓ an einem Knoten v_ℓ,

$$x_\ell(t) = a \cdot x_j(t).$$

Die graphische Darstellung in klassischer Form und die Darstellung mit Hilfe eines Übertragungsgliedes anstelle der Gewichtsfunktion an der Kante ist in Abbildung 3.3 zu sehen.

Eine weitere wichtige Operation ist die *Addition*. Hierbei ist zu beachten, dass die Addition zweier Signale inhärent in der Definition der verallgemeinerten Gewichtsfunktion inkludiert ist. Die Addition zweier Signale x_j und x_{j+1} zu

$$x_\ell(t) = x_j(t) + x_{j+1}(t)$$

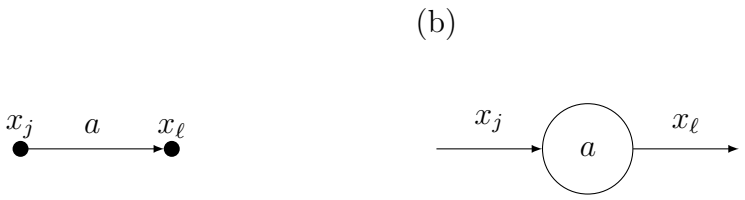

Abbildung 3.3: Visualisierung einer Multiplikation durch einen Signalflussgraphen: (a) klassische Darstellung, (b) alternative Darstellung

ist der Spezialfall, für den die Gewichtsfunktionen der Identität entsprechen, also $w_{e_j}(x_j) = x_j$ und $w_{e_{j+1}}(x_{j+1}) = x_{j+1}$. Damit ergibt sich

$$x_\ell(t) = w_{e_j}\big(x_j(t)\big) + w_{e_{j+1}}\big(x_{j+1}(t)\big) = x_j(t) + x_{j+1}(t).$$

Die graphische Darstellung ist sinngemäß bereits in Abbildung 3.1 (c) ersichtlich.

Die Liste an Relationen zwischen Signalen kann natürlich beliebig lang fortgesetzt werden. Im Folgenden wird ein kleiner Ausschnitt der wichtigsten Relationen gebracht, welche weiter unten erneut aufgegriffen werden.

Eine wesentliche Relation, welche später zur Darstellung von gewöhnlichen Differentialgleichungen herangezogen wird, ist die *Integration*. Dabei wird von kausalen Signalen mit $t_0 = 0$ ausgegangen, deren Verknüpfung im Rahmen eines Signalflussgraphen durch

$$x_\ell(t) = \int_0^t x_j(\tau)\mathrm{d}\tau \tag{3.1}$$

gegeben ist. Darauf aufbauend können auch kompliziertere Strukturen studiert werden.

Definition 3.10 (Faltung)**.** Sei $\Omega \subseteq \mathbb{R}$.

(a) Es heißt

$$\mathrm{L}^1(\Omega) = \left\{ f\colon \Omega \to \mathbb{R}\colon \int_\Omega |f(x)|\,\mathrm{d}x < \infty \right\}$$

die *Menge aller integrierbaren Funktionen.*

(b) Seien $f, g \in \mathrm{L}^1(\Omega)$. Durch

$$(f * g)(t) = \int_\Omega f(\tau)g(t-\tau)\mathrm{d}\tau$$

ist die *Faltung* von f und g definiert.

Die Definition der Faltung an dieser Stelle ist möglichst allgemein gehalten. Selbstverständlich greift sie auch für quadratisch integrierbare Funktionen, welche später betrachtet werden.

Die *Faltung* in einem Signalflussgraphen kann unterschiedlich interpretiert werden. Das Signal x_j am Knoten v_j entlang der Kante $e_j = \langle v_j, v_\ell \rangle$ kann mit einer konstanten Funktion $h \in \mathrm{L}^1(\mathbb{R})$ gemäß

$$x_\ell(t) = (x_j * h)(t) \tag{3.2}$$

gefaltet werden, oder es können auch zwei Signale x_j und x_{j+1} an zwei Knoten v_j und v_{j+1} zu einem Signal x_ℓ gemäß

$$x_\ell(t) = (x_j * x_{j+1})(t)$$

gefaltet werden. In Abbildung 3.4 sind zwei graphische Darstellungen zu finden, welche die Faltung repräsentieren.

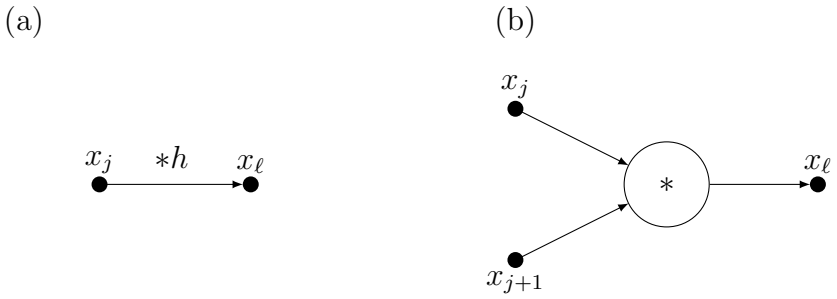

Abbildung 3.4: Visualisierung einer Faltung durch einen Signalflussgraphen: (a) Faltung mit konstanter Funktion, (b) Faltung zweier Signale

Wie aus der oben gegebenen Definition ersichtlich, können Integration und Faltung im Signalflussgraphen nicht durch verallgemeinerte Gewichtsfunktionen beschrieben werden. Die Integration kann noch mit Hilfe eines Integraloperators erfasst werden, die Faltung hingegen ist auch damit nicht mehr beherrschbar. Diese ist eine Verknüpfung von zwei verschiedenen Signalen an verschiedenen Knoten und einer Relation zu einem einzigen Zielknoten. Diese Art von Relation ist in der Definition eines Signalflussgraphen nicht vorgesehen.
Die Lösung für dieses Problem liefert der Abschnitt über lineare zeitinvariante Systeme im anschließenden Kapitel 4. Dort wird eine Beschreibung eingeführt, welche eine geschlossene Darstellung derartiger Relationen in Signalflussgraphen wieder erlaubt.

Kapitel 4

Lineare zeitinvariante Systeme

Dieser Abschnitt widmet sich der Systemtheorie und dem Zusammenhang zur Graphentheorie. Mit diesem Kapitel wird die Grundlage der linearen Modellbeschreibung in Simulink gelegt.

4.1 Grundlegende Definitionen und Eigenschaften

Um die Eigenschaften von *linearen zeitinvarianten Systemen*, *LZI–Systeme*, bzw. *linear time–invariant systems*, *LTI–Systeme*, zu analysieren, wird zuerst eine Definition der Begriffe *Linearität* und *Zeitinvarianz* vorgenommen.

Definition 4.1. Seien V, W Vektorräume über $\mathbb{C}$ und sei $\varphi\colon V \to W$ eine Abbildung. φ heißt $\mathbb{C}$*–linear*, wenn für alle $v_1, v_2 \in V$ und $c_1, c_2 \in \mathbb{C}$ gilt

$$\varphi(c_1 v_1 + c_2 v_2) = c_1 \varphi(v_1) + c_2 \varphi(v_2).$$

Bemerkung 4.2. Sind V, W Vektorräume über $\mathbb{R}$ und $c_1, c_2 \in \mathbb{R}$, so heißt φ $\mathbb{R}$*–linear* oder einfach *linear*.

Die Eingangs–Ausgangs–Relation $\mathbf{S}\colon\ \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ eines *Übertragungssystems*, wie in Abbildung 4.1 gezeigt, stellt einen Zusammenhang zwischen dem Eingangssignal

$$u\colon \Omega \to \mathbb{R},\ t \mapsto u(t),$$

mit $\Omega \subseteq \mathbb{R}$, dem Ausgangssignal y und dem Anfangszustand $x(\tau) \in \mathbb{R}^n$ gemäß

$$y(t) = \mathbf{S}\big(x(\tau), u(t)\big) \tag{4.1}$$

her.

Es ist üblich, das Übertragungssystem mit der Eingangs–Ausgangs–Relation $\mathbf{S}$ zu identifizieren und keine Unterscheidung zwischen System und Beschreibung vorzunehmen.

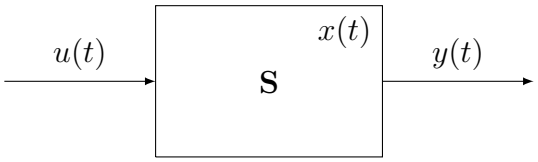

Abbildung 4.1: Blockdiagramm eines allgemeinen Übertragungssystems

Definition 4.3. Es werden die folgenden Begriffsbildungen getroffen:

(a) Ein Zustand $x(\theta) \in \mathbb{R}^n$ heißt *Nullzustand* des Systems $\mathbf{S}$, wenn

$$\mathbf{S}(x(\theta), 0) = 0$$

gilt. Unter Voraussetzung der Existenz kann ohne Beschränkung der Allgemeinheit immer $x(\theta) = 0 \in \mathbb{R}^n$ angenommen werden.

(b) Der Verlauf des Ausgangssignals y bei vorgegebenem Nullzustand $x(\theta)$ und vorgegebenem Eingangssignal u heißt die zu u gehörende *Nullzustandsantwort* y_{0Z} des Systems,

$$y_{0Z}(t) = \mathbf{S}\big(x(\theta), u(t)\big).$$

(c) Der Verlauf des Ausgangssignals y bei verschwindendem Eingangssignal $u(t) \equiv 0$ und allgemeinem Anfangszustand $x(\tau) = (x_1(\tau), \ldots, x_n(\tau))^T$ heißt die *Nulleingangsantwort* y_{0E} des Systems,

$$y_{0E}(t) = \mathbf{S}(x(\tau), 0).$$

Im Allgemeinen ist eine komplette Charakterisierung des Systems anzustreben, welche aus der Zusammensetzung dieser getrennten Antworten besteht.

Definition 4.4. Die Superposition der Nullzustands– und der Nulleingangsantwort

$$y(t) = y_{0Z}(t) + y_{0E}(t)$$

heißt die *vollständige Antwort* des Übertragungssystems.

Mit Hilfe der oben angeführten Definitionen lässt sich nun die *Linearität eines Übertragungssystems* charakterisieren.

Definition 4.5. Ein System $y(t) = \mathbf{S}\big(x(\tau), u(t)\big)$ heißt *linear*, wenn für $c_1, c_2 \in \mathbb{C}$ gilt:

(a) *Nullzustandslinearität.* Die Nullzustandsantwort y_{0Z} ist linear bezüglich des Eingangssignals u.

$$\mathbf{S}\big(0, c_1u_1(t) + c_2u_2(t)\big) = c_1\mathbf{S}\big(0, u_1(t)\big) + c_2\mathbf{S}\big(0, u_2(t)\big)$$

(b) *Nulleingangslinearität.* Die Nulleingangsantwort y_{0E} ist linear bezüglich des Ausgangssignals y.

$$\mathbf{S}\big(c_1x_1(\tau) + c_2x_2(\tau), 0\big) = c_1\mathbf{S}\big(x_1(\tau), 0\big) + c_2\mathbf{S}\big(x_2(\tau), 0\big)$$

(c) Die *vollständige Systemantwort* y ist die Summe aus Nulleingangsantwort und Nullzustandsantwort,

$$\mathbf{S}\big(x(\tau), u(t)\big) = \mathbf{S}\big(0, u(t)\big) + \mathbf{S}\big(x(\tau), 0\big).$$

Ein System, das nicht alle drei Eigenschaften besitzt, heißt *nichtlinear*.

Die zweite Eigenschaft, die zur Charakterisierung von LTI–Systemen benötigt wird, ist die *Zeitinvarianz*. Hierbei genügt es, Signale aus dem Signalraum $\mathrm{L}^2(\mathbb{R})$ zu betrachten.

Definition 4.6. Es sei $x \in \mathrm{L}^2(\mathbb{R})$.

(a) Für alle $\tau \in \mathbb{R}$ wird der *Translationsoperator*

$$\mathbf{T}_\tau \colon \mathrm{L}^2(\mathbb{R}) \to \mathrm{L}^2(\mathbb{R}),\ x \mapsto \mathbf{T}_\tau(x) = x(t-\tau)$$

definiert.

(b) Ein Übertragungssystem $\mathbf{S}$ heißt *zeitinvariant*, wenn für alle u und $x(\theta) = \overline{x}$

$$\mathbf{S}\big(\mathbf{T}_\tau(u), \overline{x}\big) = \mathbf{T}_\tau\big(\mathbf{S}(u, \overline{x})\big), \qquad \forall \tau \in \mathbb{R}$$

gilt.

(c) Übertragungssysteme, welche die Eigenschaft aus (b) nicht erfüllen, heißen *zeitvariant*.

Korollar 4.7. Der Translationsoperator aus Definition 4.6 (a) ist linear.

4.2 Die Laplace–Transformation

Dieser Abschnitt ist der Definition und der Eigenschaften der Laplace–Transformation gewidmet. Diese wird im Folgenden für kausale reellwertige Funktionen definiert.

Definition 4.8. Sei $I = [0, \infty)$ und sei $f \colon I \to \mathbb{R}$.

(a) Eine Funktion f heißt von *exponentieller Ordnung* für $t \to \infty$, wenn Konstanten $s_0 > 0$ und $M > 0$ existieren, sodass mit $T > 0$

$$|f(t)| \leq M \cdot \mathrm{e}^{s_0 t}$$

für alle $t > T$ gilt.

(b) Es bezeichnet $S(I, \mathbb{R})$ die Menge aller Funktionen f von exponentieller Ordnung.

(c) Es bezeichnet $\mathrm{Abb}(\mathbb{C}, \mathbb{C})$ die Menge aller Abbildungen $f \colon \mathbb{C} \to \mathbb{C}$.

(d) Durch

$$\mathcal{L}(f)(s) = \int_0^\infty f(t)\mathrm{e}^{-st}\,\mathrm{d}t$$

wird eine Abbildung $\mathcal{L} \colon S(I, \mathbb{R}) \to \mathrm{Abb}(\mathbb{C}, \mathbb{C})$ definiert, die *Laplace–Transformation*.

Satz 4.9 (Existenz der Laplace–Transformierten)**.** Sei $I = [0, \infty)$ und sei $f\colon I \to \mathbb{R}$. Ist f von exponentieller Ordnung mit den Konstanten $s_0 > 0$ und $M > 0$ und gilt $f \in \mathrm{L}^1(I)$, so existiert in der rechten Halbebene $\mathrm{Re}(s) > s_0$ die Laplace–Transformierte $\mathcal{L}(f)$.

Bemerkung 4.10. Die Bedingung $f \in \mathrm{L}^1(I)$ ist beispielsweise bereits für auf I stückweise stetige Funktionen f erfüllt.

Definition 4.11. Sei $I = [0, \infty)$ und sei $f\colon I \to \mathbb{R}$. Es bezeichnet $\mathcal{S}(I, \mathbb{R})$ die Menge aller integrierbaren Funktionen von exponentieller Ordnung, d.h. es gilt $f \in \mathcal{S}(I, \mathbb{R})$ genau dann, wenn $f \in S(I, \mathbb{R})$ und $f \in \mathrm{L}^1(I)$, also $f \in \big(S(I, \mathbb{R}) \cap \mathrm{L}^1(I)\big)$.

Satz 4.12 (Eigenschaften der Laplace–Transformierten)**.** Seien $x, x_1, x_2 \in \mathcal{S}(I, \mathbb{R})$.

(a) Linearität. Für $c_1, c_2 \in \mathbb{C}$ gilt

$$\mathcal{L}(c_1 x_1 + c_2 x_2)(s) = c_1 \mathcal{L}(x_1)(s) + c_2 \mathcal{L}(x_2)(s).$$

(b) Zeitdehnung. Mit $a \in \mathbb{R}^+$ und dem *Zeitdehnungsoperator*

$$D_a\colon \mathrm{L}^1(I) \to \mathrm{L}^1(I),\ x \mapsto D_a(x) = x(at)$$

gilt

$$\mathcal{L}(D_a x) = \frac{1}{a} \mathcal{L}(x) \left(\frac{s}{a}\right).$$

(c) Zeitverschiebung. Für $\tau \in \mathbb{R}^+$ gilt

$$\mathcal{L}\big(\mathbf{T}_\tau(x)\big) = \mathrm{e}^{-s\tau} \mathcal{L}(x)(s).$$

(d) Zeitdifferentiation. Ist zusätzlich $x \in \mathcal{C}^1(I)$ erfüllt, so gilt

$$\mathcal{L}(x')(s) = s\mathcal{L}(x)(s) - x(0).$$

Dies kann für $x \in \mathcal{C}^n(I)$ leicht verallgemeinert werden zu

$$\mathcal{L}\big(x^{(n)}\big) = s^n \mathcal{L}(x)(s) - s^{n-1} x(0) - s^{n-2} x^{(1)}(0) - \ldots - x^{(n-1)}(0).$$

(e) Zeitintegration. Es gilt

$$\mathcal{L}\left(\int_0^t x(\tau) \mathrm{d}\tau\right)(s) = \frac{1}{s} \mathcal{L}(x)(s).$$

Bemerkung 4.13. An dieser Stelle eine Bemerkung zu den Mengen, auf denen die bisher definierten Operatoren, wie beispielsweise der Translations– und der Dehnungsoperator, definiert sind. In Definition 4.6 (a) ist der Zeitverschiebungsoperator auf $\mathrm{L}^2(\mathbb{R})$ definiert, hingegen der Zeitdehnungsoperator in Satz 4.12 (b) auf $\mathrm{L}^1(I)$.

Ziel ist es, auf eine Theorie im Signalraum hinzuarbeiten, also auf Signale im $\mathrm{L}^2(\mathbb{R})$. Dies ist der Grund, warum an vielen Stellen die Betrachtungen auf diesen Funktionenraum beschränkt sind. Andere Themen, wie die Laplace–Transformation, erlauben aber eine allgemeinere Betrachtung in $\mathrm{L}^1(\mathbb{R})$.

Für eine gemeinsame Theorie muss schlussendlich also $\mathrm{L}^1(\mathbb{R}) \cap \mathrm{L}^2(\mathbb{R})$ betrachtet werden. Hierbei hilft der

Satz 4.14. Die Menge $\mathrm{L}^1(\mathbb{R}) \cap \mathrm{L}^2(\mathbb{R})$ ist bezüglich der durch die L^2–Norm induzierten Metrik dicht in $\mathrm{L}^2(\mathbb{R})$.

Mit dieser Tatsache kann die Laplace–Transformation auf $\mathrm{L}^2(\mathbb{R})$ fortgesetzt werden.

Für die praktische Anwendung der Laplace–Transformation sind oft spezielle Eigenschaften hilfreich, die gewisse Berechnungen erleichtern. Insbesondere werden die folgenden Eigenschaften auch für die Betrachtung von Signalflussgraphen für LTI–Systeme benötigt, wo der Laplace–Bereich eine wichtige Rolle spielen wird.

Satz 4.15. Seien $x, x_1, x_2 \in \mathcal{S}(I, \mathbb{R})$.

(a) Faltungssatz. Es gilt

$$\mathcal{L}(x_1 * x_2)(s) = \mathcal{L}(x_1)(s) \cdot \mathcal{L}(x_2)(s).$$

(b) Endwertsatz. Unter der Voraussetzung, dass der in s auftretende Grenzwert existiert, gilt

$$\lim_{t\to\infty} x(t) = \lim_{s\to 0} s\mathcal{L}(X)(s).$$

(c) Anfangswertsatz. Unter der Voraussetzung, dass der in s auftretende Grenzwert existiert, gilt

$$x(0) = \lim_{s\to\infty} sX(s).$$

Nach dem Studium der Transformation $\mathcal{L}\colon \mathcal{S}(I, \mathbb{R}) \to \mathrm{Abb}(\mathbb{C}, \mathbb{C})$ stellt sich nun die Frage nach einer inversen Transformation, welche eine Abbildung von $\mathrm{Abb}(\mathbb{C}, \mathbb{C}) \to \mathcal{S}(I, \mathbb{R})$ darstellt. Dabei fällt sofort auf, dass nicht jede Funktion $F \in \mathrm{Abb}(\mathbb{C}, \mathbb{C})$ ein Urbild $f \in \mathcal{S}(I, \mathbb{R})$ haben muss.
Die Frage nach der inversen Laplace–Transformation kann nun wie folgt beantwortet werden.

Definition 4.16. Für eine Funktion $X \in \mathrm{Abb}(\mathbb{C}, \mathbb{C})$ heißt jene Funktion $x \in \mathcal{S}(I, \mathbb{R})$ die *inverse Laplace–Transformierte*, für die

$$\mathcal{L}(x) = X$$

gilt.

Satz 4.17 (inverse Laplace–Transformation)**.** Die inverse Laplace–Transformierte $x \in \mathcal{S}(I, \mathbb{R})$ der Laplace–Transformierten $X = \mathcal{L}(x)$ ist gegeben durch das *Bromwich–Integral*

$$x(t) = \mathcal{L}^{-1}(F)(t) = \frac{1}{2\pi\mathrm{i}} \int_\gamma X(s)\mathrm{e}^{st}\mathrm{d}s,$$

mit $\gamma\colon (-\infty, \infty) \to \mathbb{C},\ t \mapsto \sigma + \mathrm{i}t$, und $\sigma > s_0$. Dabei ist s_0 die sogenannte *Konvergenzabszisse* der Funktion $x \in \mathcal{S}(I, \mathbb{R})$ aus Definition 4.8 (a).

Mit der Definition der inversen Laplace–Transformierten kann ein Satz formuliert werden, der ähnlich dem der Zeitverschiebung ist.

Satz 4.18 (Frequenzverschiebung)**.** Sei $x \in \mathcal{S}(I, \mathbb{R})$ und $X = \mathcal{L}(x)$ die Laplace–Transformierte. Für eine beliebige Konstante $\hat{s} \in \mathbb{C}$ gilt

$$\mathcal{L}^{-1}(\mathbf{T}_{\hat{s}} X) = \mathrm{e}^{\hat{s}t} x(t).$$

Bemerkung 4.19. Die praktische Vorgehensweise zur Bestimmung von inversen Laplace–Transformierten ist eine andere. Es ist eine Liste von Korrespondenzen bekannt, welche kausale Signale x und deren Laplace–Transformierte $X = \mathcal{L}(x)$ ausweisen. Lässt sich die Funktion Y als Linearkombination von Funktionen $X_1, \ldots, X_k$ darstellen, welche in den Korrespondenzen oder mit Hilfe von Satz 4.12 und den Korrespondenzen zu finden sind, so kann aufgrund der Linearität der Laplace–Transformation die zugehörige inverse Laplace–Transformierte angegeben werden.
Bei gebrochen rationalen Funktionen X ist oft mit Hilfe der reellen oder komplexen Partialbruchzerlegung eine derartige Linearkombination darstellbar.

Eine Liste an ausgewählten Korrespondenzen ist im Anhang A zu finden.

4.3 Beschreibung von LTI–Systemen

Dieser Abschnitt ist der Beschreibung eines LTI–Systems gewidmet. Diese soll im Zeitbereich, ausgehend von der Eingangs–Ausgangs–Relation

$$y(t) = \mathbf{S}\big(x(t), u(t)\big)$$

mit $y, u \in \mathbb{R}$ und $x \in \mathbb{R}^n$, formalisiert werden und in einer Beschreibung mit Hilfe der Laplace–Transformation münden.

Satz 4.20. Das Übertragungssystem aus Gleichung (4.1) ist genau dann linear und zeitinvariant, wenn es durch eine lineare gewöhnliche Differentialgleichung mit konstanten Koeffizienten der Form

$$y^{(n)}(t) + a_{n-1} y^{(n-1)}(t) + \ldots + a_1 y'(t) + a_0 y(t) = b_m u^{(m)}(t) + b_{m-1} u^{(m-1)}(t) + \ldots + b_1 u'(t) + b_0 u(t),$$

mit $a_i, b_j \in \mathbb{R}$ für $i = 0, \ldots, n-1$ und $j = 0, \ldots, m$, dargestellt werden kann.

Ein weiteres Konzept, um lineare zeitinvariante Systeme beschreiben zu können, ist das Konzept der *Zustandsraumdarstellung*.

Satz 4.21. Das Übertragungssystem aus Gleichung (4.1) ist genau dann linear und zeitinvariant, wenn es sich in die Form

$$\begin{aligned} x'(t) &= A\ x(t) + b\ u(t), \qquad x(0) = x_0 \\ y(t) &= c^T x(t) + d\ u(t), \end{aligned} \tag{4.2}$$

mit $b, c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$ und $u, y, d \in \mathbb{R}$, überführen lässt.

Diese beiden Beschreibungen im Zeitbereich eröffnen gemeinsam mit der Laplace–Transformation eine neue Art der Charakterisierung von LTI–Systemen. Wendet man auf die lineare gewöhnliche Differentialgleichung mit konstanten Koeffizienten aus Satz 4.20 die Laplace–Transformation an, so erhält man unter den Voraussetzungen $y(0) = \ldots = y^{(n)}(0) = 0$ und $u(0) = \ldots = u^{(m)}(0) = 0$

$$s^n\mathcal{L}(y)(s) + a_{n-1}s^{n-1}\mathcal{L}(y)(s) + \ldots + a_1 s\mathcal{L}(y)(s) + a_0\mathcal{L}(y)(s) = \\ = b_m s^m\mathcal{L}(u)(s) + b_{m-1}s^{m-1}\mathcal{L}(u)(s) + \ldots + b_1 s\mathcal{L}(u)(s) + b_0\mathcal{L}(u)(s).$$

Durch Vereinfachung gelangt man zur Darstellung

$$\big(s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0\big)\mathcal{L}(y)(s) = \big(b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0\big)\mathcal{L}(u)(s)$$

und damit zu

$$\frac{\mathcal{L}(y)(s)}{\mathcal{L}(u)(s)} = \frac{b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0}. \tag{4.3}$$

Diese Überlegung motiviert die folgende

Definition 4.22 (Übertragungsfunktion)**.** Es sei $U = \mathcal{L}(u)$ die Laplace–Transformierte des Eingangssignals und $Y = \mathcal{L}(y)$ die Laplace–Transformierte des Ausgangssignals des linearen zeitinvarianten Systems mit dem Zustand $x \in \mathbb{R}^n$ und $x_0 = 0$, dargestellt durch Satz 4.20 bzw. Satz 4.21. Es heißt jene Funktion $G \in \mathrm{Abb}(\mathbb{C}, \mathbb{C})$, welche für alle möglichen Eingangssignale u die Gleichung

$$Y(s) = G(s) \cdot U(s)$$

erfüllt, die *Übertragungsfunktion* des linearen zeitinvarianten Systems.

Ausgehend von Gleichung 4.3 können die Polynome der gebrochen rationalen Funktion faktorisiert geschrieben werden als

$$G(s) = \frac{P(s)}{Q(s)} = b_m \frac{(s-p_1)\cdot(s-p_2)\cdot\ldots\cdot(s-p_m)}{(s-q_1)\cdot(s-q_2)\cdot\ldots\cdot(s-q_m)}. \tag{4.4}$$

Definition 4.23. Mit Gleichung 4.4 wird definiert:

(a) P heißt das *Zählerpolynom* der Übertragungsfunktion G.

(b) Q heißt das *Nennerpolynom* der Übertragungsfunktion G.

(c) Die Nullstellen von Q, also die Polstellen von G, heißen die *Pole der Übertragungsfunktion* G, der Grad von Q heißt die *Ordnung der Übertragungsfunktion.*

Das Konzept der Übertragungsfunktion erlaubt nun, eine weitere Beschreibungsform für LTI–Systeme anzugeben, jene im Laplace–Bereich. Wird das Konzept des Signalflussgraphen um diese Darstellung erweitert, so ist es möglich, LTI–Systeme mit Hilfe von Signalflussgraphen abzubilden. Es ist jedoch dann gelegentlich notwendig, die Beschreibung im Laplace–Bereich miteinzubeziehen.

Um der Tatsache Rechnung zu tragen, dass Übertragungssysteme zumeist aus der Zusammenschaltung von Teilsystemen hervorgehen, wird im Folgenden die Schaltungstopologie von LTI–Systemen, beschrieben durch Übertragungsfunktionen, untersucht. Ziel ist es, Subsysteme zu einem gesamten LTI–System zusammenzufassen.

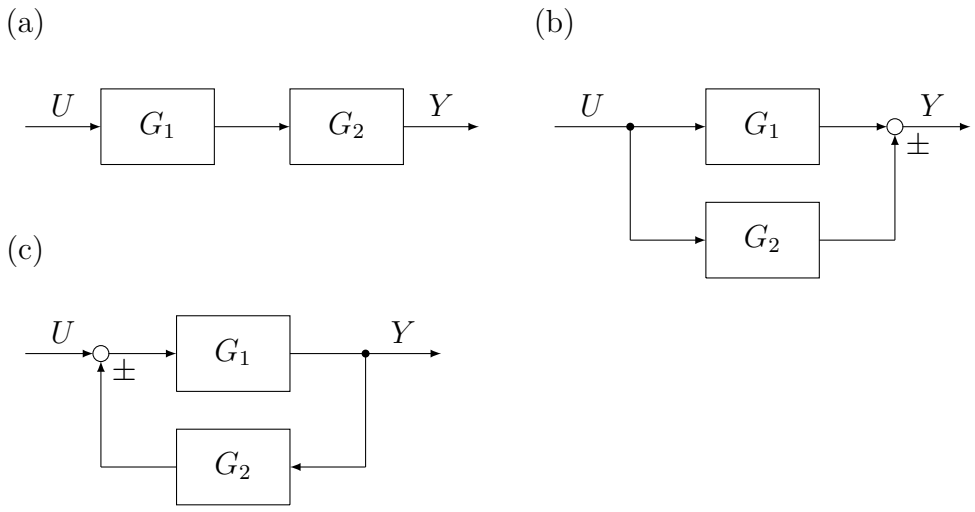

Abbildung 4.2: Schaltungstopologien einiger Blockdiagramme von LTI–Systemen: (a) Reihenschaltung, (b) Parallelschaltung, (c) Rückkopplung

Satz 4.24. Seien G_1 und G_2 Übertragungsfunktionen zweier LTI–Systeme und sei G die Übertragungsfunktion des zusammengefassten LTI–Systems mit Eingangsgröße U und Ausgangsgröße Y, also

$$G(s) = \frac{Y(s)}{U(s)}.$$

(a) Reihenschlatung. Es ist G gegeben durch

$$G(s) = G_1(s) \cdot G_2(s).$$

(b) Parallelschaltung. Es ist G gegeben durch

$$G(s) = G_1(s) \pm G_2(s).$$

(c) Rückkopplung. Es ist G gegeben durch

$$G(s) = \frac{G_1(s)}{1 \mp G_1 \cdot G_2}.$$

An dieser Stelle sei erwähnt, dass die Menge der Übertragungsfunktionen mit Reihen- und Parallelschaltung als Verknüpfungen eine Algebra bilden. Dies hat schaltungstopologische Konsequenzen, sodass gleiche Funktionalitäten durch verschiedene Topologien realisierbar sind.

Satz 4.25 (Algebra der Blockschaltbilder)**.** Die Menge der Übertragungsfunktionen bildet eine Algebra, d.h. für alle Übertragungsfunktionen G_1, G_2, G_3 und alle $s \in \mathbb{C}$ gilt

$$(G_1 + G_2) \cdot G_3 = G_1 \cdot G_3 + G_2 \cdot G_3, \tag{4.5}$$

$$G_1 \cdot (G_2 + G_3) = G_1 \cdot G_2 + G_1 \cdot G_3, \tag{4.6}$$

$$s(G_1 \cdot G_2) = (sG_1) \cdot G_2 = G_1 \cdot (sG_2). \tag{4.7}$$

Diese algebraischen Eigenschaften können als schaltungstopologische Eigenschaften betrachtet werden. Die Gleichungen aus Satz 4.25 werden schaltungstopologisch in Abbildung 4.3 dargestellt.

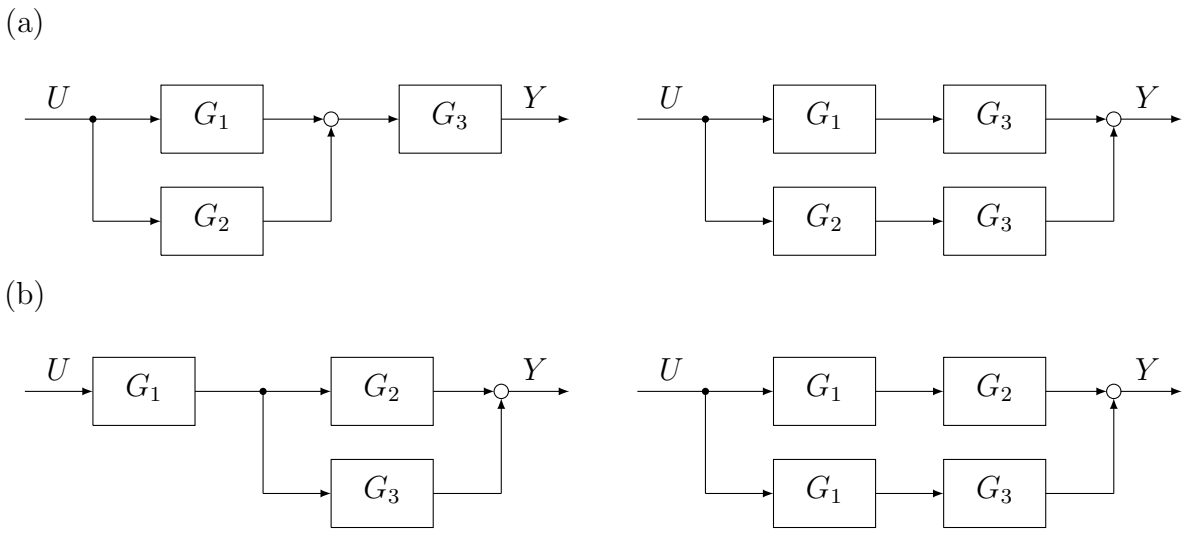

Abbildung 4.3: Algebra der Blockschaltbilder: Schaltungstopologische Bedeutung der Gleichungen (4.5) und (4.6)

4.4 Zusammenhang zur Theorie der Signalflussgraphen

In Kapitel 3.3 wurden einige Elemente des Signalflussgraphen dargestellt und auf die Problematik hingewiesen, dass einige höhere Relationen, wie beispielsweise die Integration, siehe Gleichung (3.1), oder die Faltung, siehe Gleichung (3.2), in verallgemeinerten Signalflussgraphen nicht einfach bzw. überhaupt nicht darstellbar sind. Der Abschnitt über LTI–Systeme stellt neue Werkzeuge zur Verfügung, mit Hilfe derer die Theorie der Signalflussgraphen wieder konsequent verfolgt werden kann.

Wird vorausgesetzt, dass die Signale x_i an den Knoten v_i der Laplace–Transformation unterworfen werden dürfen, also $x_i \in \big(\mathcal{S}(I,\mathbb{R}) \cap \mathrm{L}^2(\mathbb{R})\big) \subseteq \big(\mathrm{L}^1(\mathbb{R}) \cap \mathrm{L}^2(\mathbb{R})\big)$, so sind an den Knoten v_i auch die Laplace–Transformierten $X_i = \mathcal{L}(x_i)$ verfügbar. Mit deren Hilfe können Verknüpfungen nicht als Relation in $t \in I$ beschrieben werden, sondern als verallgemeinerte Gewichtsfunktion in $s \in \mathbb{C}$.

Aus der Integration nach t in Gleichung (3.1) wird unter Anwendung der Laplace–Transformation und Satz 4.12 (e) die Gleichung

$$X_\ell(s) = \frac{1}{s} X_j(s). \tag{4.8}$$

Analoges Vorgehen bei der Faltung in Gleichung (3.2) liefert durch Anwendung der Laplace–Transformation und Satz 4.15 (a) mit $H = \mathcal{L}(h)$ die Gleichung

$$X_\ell(s) = X_j(s) \cdot H(s). \tag{4.9}$$

Die (zeitliche) Integration und die Faltung wird als Blockschaltbild, demnach auch als Signalflussgraph, gemäß Abbildung 4.4 dargestellt.

Mit diesen Möglichkeiten ausgestattet können Signalflussgraphen dazu verwendet werden, algebraische Gleichungen darzustellen, wie es in Beispiel 3.2 gezeigt wurde, und sie können auch

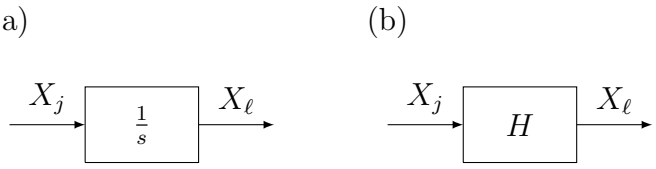

Abbildung 4.4: Blockschaltbilder von Signalflussgraphen im Laplace–Bereich: (a) Zeitintegration, (b) Faltung

dazu herangezogen werden, um Differentialgleichungen darzustellen, wie es das folgende Beispiel zeigt.

Beispiel 4.26. Es wird eine lineare Differentialgleichung 1. Ordnung mit konstanten Koeffizienten betrachtet, welche das Verhalten des Eingangssignals u und des Ausgangssignals y beschreibt, gemäß

$$a_1 y(t) + a_2 y'(t) = a_3 u(t),$$

mit $a_1, a_2, a_3 \in \mathbb{R} \setminus \{0\}$ und $y(0) = 0$. Eine simple Umformung zu

$$y'(t) = \frac{a_3}{a_2} u(t) - \frac{a_1}{a_2} y(t)$$

ergibt die topologische Struktur des Blockschaltbildes, wie in Abbildung 4.5 dargestellt.

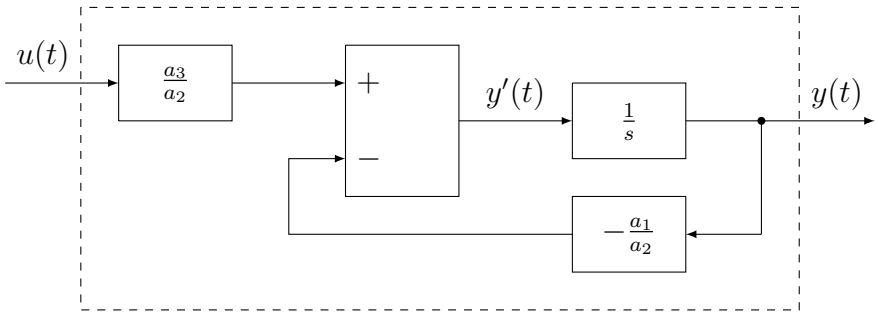

Abbildung 4.5: Blockschaltbild der Differentialgleichung aus Beispiel 4.26

Es kann aber auch die gesamte Differentialgleichung durch eine Übertragungsfunktion beschrieben werden. Dazu wird wie bei der Herleitung der Übertragungsfunktion vorgegangen. Die Differentialgleichung wird der Laplace–Transformation unterworfen

$$a_1 \mathcal{L}(y)(s) + a_2 s \mathcal{L}(y)(s) = a_3 \mathcal{L}(u)(s)$$

und mit $\mathcal{L}(u) = U$ sowie $\mathcal{L}(y) = Y$ erhält man

$$G(s) = \frac{Y(s)}{U(s)} = \frac{a_3}{a_1 + a_2 s}.$$

Dieses Beispiel zeigt anschaulich die äquivalente Betrachtung im Zeit– und Laplace–Bereich, sowie die Beschreibung durch gewöhnliche Differentialgleichungen und Übertragungsfunktionen.

Bemerkung 4.27. Es wird im Folgenden nicht mehr zwischen Blockschaltbildern (von LTI–Systemen) und Signalflussgraphen unterschieden, es sei denn an Stellen, wo strukturelle Unterschiede zu erkennen sind.

4.5 Die z–Transformation

Definition 4.28. Es sei $(f_n)_{n\in\mathbb{N}}$ eine Folge.

(a) Erfüllt die Folge $(f_n)_{n\in\mathbb{N}}$ die Ungleichung

$$|f_n| \leq M\gamma^n, \qquad \forall n \in \mathbb{N}$$

und für passende $\gamma, M > 0$, so heißt (f_n) von *exponentieller Ordnung*.

(b) Es bezeichnet $S(\mathbb{N}, \mathbb{R})$ die Menge aller Folgen von exponentieller Ordnung, analog zu Definition 4.8.

(c) Durch

$$\mathcal{Z}(f_n)(z) = \sum_{n=0}^{\infty} f_n z^{-n}$$

wird eine Abbildung $\mathcal{Z}\colon S(\mathbb{N}, \mathbb{R}) \to \mathrm{Abb}(\mathbb{C}, \mathbb{C})$ definiert, die *z–Transformation*.

(d) Es heißt das Gebiet $C_\gamma = \{z \in \mathbb{C}\colon |z| > \gamma\}$ der *Existenzbereich* von $\mathcal{Z}(f_n)$.

Bemerkung 4.29. Die Schreibweise der z–Transformierten $\mathcal{Z}(f_n)$ ist nicht ganz korrekt, denn $\mathcal{Z}$ wirkt auf eine Folge (f_n), also sollte $\mathcal{Z}\big((f_n)\big)$ geschrieben werden. Um die Notation aber lesbar zu halten, wird auf ein Klammernpaar verzichtet.

Satz 4.30 (Existenz der z–Transformierten)**.** Die z–Transformierte existiert für alle $z \in C_\gamma$, d.h. die Summe $\sum_{n\in\mathbb{N}} f_n z^{-n}$ ist für alle $z \in C_\gamma$ absolut konvergent.

Bemerkung 4.31 (Zusammenhang zwischen Laplace– und z–Transformation)**.** Oft wird die z–Transformation als diskrete Laplace–Transformation interpretiert. Dazu betrachte man die Funktion

$$\tilde{f}(t) = \sum_{n=0}^{\infty} f(nT)\delta(t - nT)$$

für ein $T > 0$ und $\delta\colon \mathbb{R} \to \mathbb{R} \cup \{\infty\}$, definiert durch

$$\delta(t) = \begin{cases} \infty & \text{für } t = 0, \\ 0 & \text{sonst,} \end{cases}$$

und der Forderung $\int_{\mathbb{R}} \delta(t)\,\mathrm{d}t = 1$. Wird auf $\tilde{f}$ die Laplace–Transformation angewendet, so erhält man

$$\mathcal{L}(\tilde{f})(s) = \int_0^\infty \sum_{n=0}^{\infty} f(nT)\delta(t - nT)\mathrm{e}^{-st}\,\mathrm{d}t = \sum_{n=0}^{\infty} f(nT)\mathrm{e}^{-snT}.$$

Mit $f_n = f(nT)$ und $z = \mathrm{e}^{sT}$ erhält man die Darstellung der z–Transformation.

Hierbei sei zu bemerken, dass die mathematisch korrekte Darstellung dieses Sachverhalts über die Theorie der Distributionen erfolgen müsste, die hier gewählte aber für diese Bemerkung genüge.

Für die inverse z-Transformierte bedient man sich des Laurent–Reihen–Entwicklungssatzes aus der Funktionentheorie.

Korollar 4.32. Eine Funktion $f\colon \mathbb{C} \to \mathbb{C}$, die auf einem Gebiet $C_\gamma = \{z \in \mathbb{C}\colon |z| > \gamma\}$ holomorph ist, ist in diesem Gebiet eindeutig in eine absolut konvergente Laurent–Reihe

$$f(z) = \sum_{n\in\mathbb{Z}} f_n\, z^{-n} \qquad \text{mit} \qquad f_n = \frac{1}{2\pi\mathrm{i}} \int_{\partial K_r(0)} f(z)\, z^{n-1}\,\mathrm{d}z$$

um $z_0 = 0$ entwickelbar, für $\gamma < r < \infty$.

Analog zur Laplace–Transformation gilt der folgende

Satz 4.33 (Eigenschaften der z–Transformation)**.** Seien $(x_n), (x_{1,n}), (x_{2,n}) \in S(\mathbb{N}, \mathbb{R})$.

(a) Linearität. Für $c_1, c_2 \in \mathbb{C}$ gilt

$$\mathcal{Z}\left(c_1 x_{1,n} + c_2 x_{2,n}\right)(z) = c_1 \mathcal{Z}(x_{1,n})(z) + c_2 \mathcal{Z}(x_{2,n})(z).$$

(b) Verschiebungssatz. Für $k \in \mathbb{N}$ gilt

$$\mathcal{Z}(x_{n\pm k})(z) = z^{\pm k}\left(\mathcal{Z}(x_n)(z) + \sum_{j=1}^{k} x_{\pm k} z^{\mp k}\right).$$

(c) Dämpfungssatz. Für $c \in \mathbb{C}\setminus\{0\}$ gilt

$$\mathcal{Z}(c^n x_n)(z) = \mathcal{Z}(x_n)\left(\frac{z}{c}\right).$$

(d) Differenzenbildung I. Es gilt

$$\mathcal{Z}(x_{n+1} - x_n)(z) = (z-1)\mathcal{Z}(x_n)(z) - z x_0.$$

(e) Differenzenbildung II. Es gilt

$$\mathcal{Z}(x_n - x_{n-1})(z) = \frac{z-1}{z}\mathcal{Z}(x_n)(z).$$

(f) Folge der Partialsummen. Es gilt

$$\mathcal{Z}\left(\sum_{j=0}^{n} x_j\right)(z) = \frac{z}{z-1}\mathcal{Z}(x_n)(z).$$

(g) Faltungssatz. Mit der Definition der *diskreten Faltung* durch

$$(x_{1,n}) * (x_{2,n}) = \left(\sum_{k=0}^{\infty} x_{1,k} x_{2,n-k}\right)$$

gilt

$$\mathcal{Z}\big((x_{1,n}) * (x_{2,n})\big)(z) = \mathcal{Z}(x_{1,n})(z) \cdot \mathcal{Z}(x_{2,n})(z).$$

Abschließend sei an dieser Stelle noch ein Zusammenhang zur Laplace–Transformation thematisiert. Sei $f \in \mathcal{S}(I, \mathbb{R})$ und (f_n) die Folge der Abtastwerte, gegeben durch $f_n = f(nT)$ für ein $T > 0$, und sei $F = \mathcal{L}(f)$ gegeben. Dann berechnet sich die z–Transformierte gemäß

$$\mathcal{Z}(f_n)(z) = \mathcal{Z}\Big(\big(\mathcal{L}^{-1}(F(s))\big|_{t=nT}\big) \Big)(z)$$

und kann so auf die Laplace–Transformierte zurückgeführt werden. Die für die z–Transformation angelegte Korrespondenztabelle ist ebenfalls in Anhang A zu finden.

4.6 Abtastsysteme

Dieser Abschnitt widmet sich den Abtastsystemen, also linearen zeitinvarianten Systemen, welche in der Domäne der abgetasteten Signale betrachtet werden. Die Abtastung wurde in Definition 3.9 eingeführt und spielt hier eine wesentliche Rolle.

Abtastsysteme sind eng mit zeitkontinuierlichen Systemen verbunden. Aus diesem Grund müssen Schnittstellen zwischen dem zeitkontinuierlichen und dem zeitdiskreten Bereich hergestellt werden. Dazu wird zunächst der Folgenraum in Anlehnung an den Signalraum aus Kapitel 3.2 definiert.

Definition 4.34. Es sei $(x_n)_{n\in\mathbb{N}}$ eine Folge.

(a) Die Menge

$$\ell^p = \left\{ (x_n) \colon \sum_{n=1}^{\infty} |x_n|^p < \infty \right\}$$

für $0 < p < \infty$ heißt die *Menge der beschränkten Folgen.*

(b) Für $(x_n), (y_n) \in \ell^p$ und $s \in \mathbb{R}$ ist durch

$$(x_n) + (y_n) = (x_n + y_n) \qquad \text{und} \qquad s \cdot (x_n) = (s \cdot x_n)$$

eine *Addition* $+ \colon \ell^p \to \ell^p$ und eine *Multiplikation mit Skalaren* $\cdot \colon \ell^p \to \ell^p$ definiert.

Satz 4.35 (Folgenraum)**.** Mit Definition 4.34 gilt:

(a) Für $0 < p < \infty$ definiert

$$\|(x_n)\|_{\ell^p} = \left(\sum_{n=1}^{\infty} |x_n| p \right)^{\frac{1}{p}}$$

eine *Norm* auf ℓ^p. Damit ist ℓ^p ein normierter Vektorraum.

(b) Für $p = 2$ wird durch

$$\langle (x_n), (y_n) \rangle_{\ell^2} = \sum_{n=1}^{\infty} x_n y_n$$

ein *Skalarprodukt* auf ℓ^2 definiert.

(c) Der ℓ^2 ist bezüglich des Skalarproduktes aus (b) vollständig und damit ein *Hilbertraum*.

Da die zeitdiskreten und zeitkontinuierlichen Systeme zusammenspielen, wie auch am Ende des letzten Abschnittes die Laplace– mit der z–Transformation in Zusammenhang gebracht wurde, soll dies in den folgenden Definitionen dargestellt werden.

Definition 4.36. Ein *Abtaster* ist jenes Übertragungssystem, welches die *Abtastung* gemäß Definition 3.9 mit äquidistanter Zerlegung, der sogenannten *Abtastzeit* $T = \Delta t$, durchführt. Er stellt eine Abbildung gemäß $\mathbf{A}\colon \mathrm{L}^1(I) \to \ell^1$ dar. Das Blockdiagramm ist in Abbildung 4.6 zu sehen.

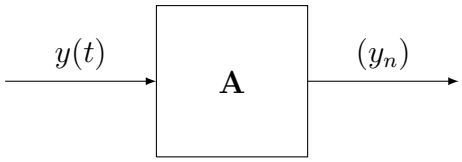

Abbildung 4.6: Blockdiagramm eines Abtasters $\mathbf{A}\colon \mathrm{L}^1(I) \to \ell^1$

Die Definition 4.36 wurde so formuliert, dass Laplace– und z–Transformation zusammenpassen. Laplace–transformierbare Signale sind aus $\mathrm{L}^1(I)$ und z–transformierbare Folgen aus ℓ^1, denn nur betragsbeschränkte Folgenglieder können die Bedingung aus 4.28 (a) erfüllen.

Analog muss auch eine umgekehrte Verbindung zwischen dem zeitdiskreten und zeitkontinuierlichen Bereich definiert werden.

Definition 4.37. Ein *Halteglied* ist jenes Übertragungssystem, welches aus einer Folge (u_n) ein zeitkontinuierliches Signal $u(t)$ gemäß

$$u(t) = u_i \qquad \text{für} \qquad t \in (t_i, t_{i+1}]$$

konstruiert und damit eine Abbildung $\mathbf{H}\colon \ell^1 \to \mathrm{L}^1(I)$ darstellt. Das Blockdiagramm ist in Abbildung 4.7 zu sehen.

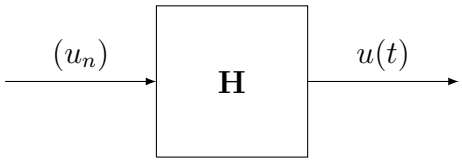

Abbildung 4.7: Blockdiagramm eines Halteglieds $\mathbf{H}\colon \ell^1 \to \mathrm{L}^1(I)$

Das Halteglied nach Definition 4.37 hat eine Treppenfunktion als Ausgangssignal u, wobei der Wert auf $(t_i, t_{i+1}]$ konstant u_i ist.

Um Abtast– und Halteglied gemeinsam verwenden zu können, müssen beide mit gleicher Zerlegung, also gleicher Abtastzeit, arbeiten und zu einem gemeinsamen Zeitpunkt synchronisiert sein. Üblich handelt es sich bei dem Intervall I um eines der Form $I = [0, T_I]$ für ein $T_I > 0$. In Simulationsumgebungen wird T_I durch das Ende der Simulationszeit festgelegt.

Mit diesen Vorbereitungen kann nun an die Definition eines Abtastsystems herangegangen werden. Es soll eine diskrete Umgebung angenommen werden, das System an und für sich ist aber zeitkontinuierlich. Der Sachverhalt ist in Abbildung 4.8 dargestellt.

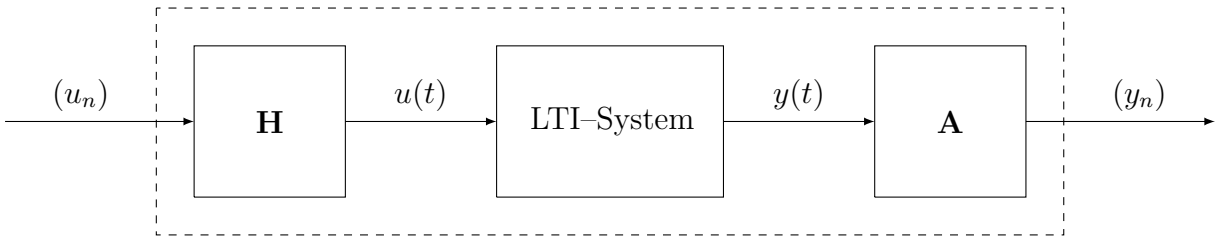

Abbildung 4.8: Blockdiagramm eines Abtastsystems

Definition 4.38 (Abtastsystem)**.** Wird einem LTI–System durch ein Halteglied die Eingangsgröße $u(t) = \mathbf{H}(u_n)$ zugeführt und die Ausgangsgröße y des LTI–Systems mit einem Abtaster diskretisiert zu $(y_n) = \mathbf{A}(y)$, so heißt das gesamte System mit Eingangsgröße (u_n) und Ausgangsgröße (y_n) ein *Abtastsystem*.

Analog zum zeitkontinuierlichen Fall können eine Zustandsraumdarstellung und eine Übertragungsfunktion im z–Bereich definiert werden.

Definition 4.39. Die *Zustandsraumdarstellung* des zu (4.2) gehörenden Abtastsystems ist gegeben durch

$$\begin{aligned} x_{k+1} &= \Phi x_k + \lambda u_k \\ y_k &= c^T x_k + d u_k, \end{aligned} \tag{4.10}$$

mit $x_k, \lambda, c \in \mathbb{R}^n$, $\Phi \in \mathbb{R}^{n\times n}$ und $u_k, y_k, d \in \mathbb{R}$. Die Anfangsbedingung x_0 aus Gleichung (4.2) gilt auch hier.

Analog zur Laplace–Transformation lässt sich nun auch eine Übertragungsfunktion im z–Bereich, ausgehend von der Zustandsraumdarstellung eines Abtastsystems, formulieren.

Definition 4.40 (z–Übertragungsfunktion)**.** Es sei $U = \mathcal{Z}(u_n)$ die z–Transformierte der Eingangsfolge (u_n) und $Y = \mathcal{Z}(y_n)$ die z–Transformierte der Ausgangsfolge (y_n) des linearen zeitinvarianten Abtastsystems mit dem Zustand $x \in \mathbb{R}^n$ und $x_0 = 0$, dargestellt durch (4.10). Die Funktion $G \in \mathrm{Abb}(\mathbb{C}, \mathbb{C})$, welche für alle möglichen Eingangsfolgen u_n die Gleichung

$$Y(z) = G(z) \cdot U(z)$$

erfüllt, heißt die *z–Übertragungsfunktion* des linearen zeitinvarianten Abtastsystems.

Satz 4.41. Die z–Übertragungsfunktion des linearen zeitinvarianten Abtastsystems aus Gleichung (4.10) berechnet sich zu

$$G(z) = \frac{\mathcal{Z}(y_n)}{\mathcal{Z}(u_n)} = c^T (zE - \Phi)^{-1} \lambda + d.$$

Wie bei der Laplace– und z–Transformation besteht auch hier ein Zusammenhang zwischen den Übertragungsfunktionen.

Satz 4.42. Sei $G(s)$ die Übertragungsfunktion eines kontinuierlichen linearen zeitinvarianten Systems, so lautet die z–Übertragungsfunktion des zugehörigen Abtastsystems

$$G(z) = \frac{z-1}{z} \mathcal{Z} \left(\mathcal{L}^{-1} \left(\frac{G(s)}{s} \right) \Big|_{t=nT} \right).$$

Mit diesen Methoden und Darstellungsformen werden zeitdiskrete lineare zeitinvariante Systeme vollständig charakterisiert. Diese Beschreibungen werden auch in Simulink zur Darstellung der Modelle verwendet.

Kapitel 5

Numerische Lösung gewöhnlicher Differentialgleichungen

Der letzte Abschnitt zeigt, dass lineare Systeme durch Differentialgleichungen oder durch Übertragungsfunktionen beschrieben werden können. Mathematisch sind die beiden Beschreibungen äquivalent, hinsichtlich der mathematischen Systemsimulation muss den Differentialgleichungen jedoch besondere Bedeutung zukommen. Die Systeme werden auf Basis der Differentialgleichungen numerisch simuliert. Dies berücksichtigt auch die Behandlung nichtlinearer Systeme. Das sind jene Systeme, welche nicht durch eine lineare gewöhnliche Differentialgleichung mit konstanten Koeffizienten gegeben sind, sondern durch beliebige Formen gewöhnlicher Differentialgleichungen.

Im Folgenden wird das Anfangswertproblem

$$\begin{aligned} x' &= f(t,x) \\ x(0) &= x_0 \end{aligned} \tag{5.1}$$

mit $x\colon [0,T] \to \mathbb{R}^n$, $\mathcal{D} \subset [0,T] \times \mathbb{R}^n$ und $f\colon \mathcal{D} \to \mathbb{R}^n$ betrachtet.

Bevor die Konstruktion einer numerischen Lösung des Anfangswertproblems betrachtet wird, ist es sinnvoll, die Frage nach ihrer grundsätzlichen Existenz zu stellen. Antwort darauf liefert der folgende

Satz 5.1 (Picard[1]–Lindelöf[2])**.** Sei $\mathcal{D} := \{(t,x) \in \mathbb{R} \times \mathbb{R}^n \colon 0 \leq t \leq a, \|x - x_0\| \leq b\}$ mit $a, b \in \mathbb{R}^+$, $f\colon \mathcal{D} \to \mathbb{R}^n$ eine beschränkte und stetige Funktion, die

$$\|f(t,x)\| \leq M$$

auf $\mathcal{D}$ erfüllt und Lipschitz–stetig bezüglich x auf $\mathcal{D}$ mit der Lipschitz–Konstante L ist.
Dann existiert genau eine Lösung x des Anfangswertproblems (5.1) für $0 \leq t \leq T := \min\left(a, \frac{b}{M}\right)$.

[1] Charles Émile Picard (1856 – 1941), französischer Mathematiker
[2] Ernst Leonard Lindelöf (1870 – 1946), finnischer Mathematiker

5.1 Lineare Einschrittverfahren

5.1.1 Methode von Euler und Basiskonzepte

Als erste Methode zur numerischen Lösung von gewöhnlichen Differentialgleichungen wird jene von Euler vorgestellt. Anhand dieser können einige Basiskonzepte wie Konvergenz und Stabilität definiert und erläutert werden.

Betrachtet wird das *skalare* Anfangswertproblem

$$\begin{aligned} x' &= f(t,x) \\ x(t_0) &= x_0 \end{aligned} \tag{5.2}$$

mit $x\colon [0,T] \to \mathbb{R}$ und $f\colon D \to \mathbb{R}$, wobei $D \subset [0,T] \times \mathbb{R}$ ist.

Die Idee der *expliziten Euler–Methode* ist das Ersetzen der Ableitung von x durch den Differenzenquotienten

$$x'(t) \approx \frac{x(t_i) - x(t_{i-1})}{h} \tag{5.3}$$

mit der Schrittweite $h = t_i - t_{i-1}$, wobei h nicht konstant sein muss, also auch $h_i = t_i - t_{i-1}$ möglich ist. Dabei ist $(t_i)_{i=1,2,\dots}$ die Folge von diskretisierten Werten aus $[0,T]$. Die diskreten Approximationen der Werte $x(t_i)$ bzw. $x(t_{i-1})$ werden mit ν_i bzw. ν_{i-1} bezeichnet. Das Ersetzen der Ableitung durch den Differenzenquotienten der Approximationen ν_i liefert nach Einsetzen in das Anfangswertproblem die Gleichung

$$\frac{\nu_i - \nu_{i-1}}{h} = f(t_{i-1}, \nu_{i-1}),$$

welche durch Umformung eine rekursive Formel zur Berechnung der Werte ν_i ergibt.

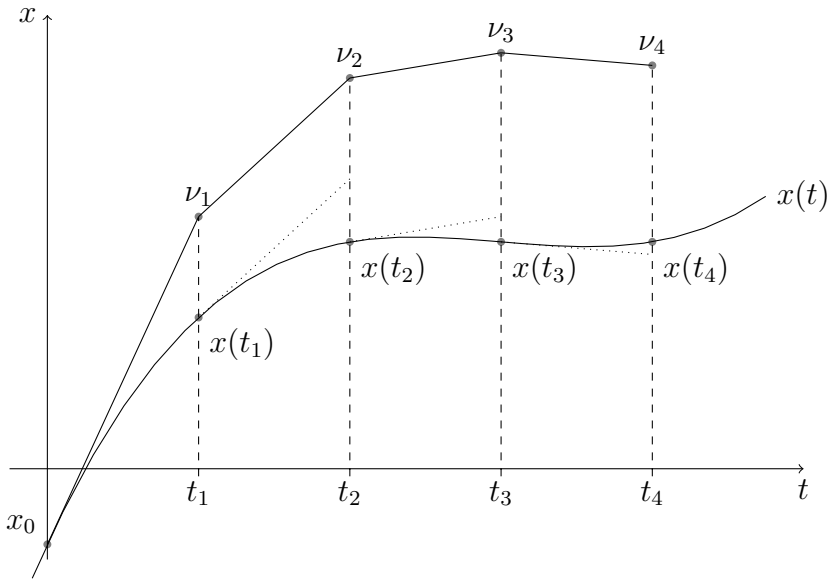

Abbildung 5.1: Zur expliziten Methode von Euler

Definition 5.2. Die *explizite Euler–Methode* zur numerischen Lösung des Anfangswertproblems (5.2) ist durch die Rekursion

$$\nu_i = \nu_{i-1} + hf(t_{i-1}, \nu_{i-1}) \tag{5.4}$$

mit $i = 1, 2 \ldots$ gegeben.

Definition 5.3. Der *lokale Diskretisierungsfehler* ℓ_i ist definiert als

$$\ell_i = \frac{x(t_i) - x(t_{i-1})}{h} - f(t_{i-1}, x(t_{i-1})). \tag{5.5}$$

Bemerkung 5.4. Wegen $f(t_{i-1}, x(t_{i-1})) = x'(t_{i-1})$ ist die geometrische Deutung möglich. $h\ell_i$ ist die Differenz zwischen der exakten Lösung $x(t_i)$ und dem Resultat nach einer Iteration der Euler–Methode, gestartet bei $x(t_{i-1})$. In Abbildung 5.2 ist dies für das skalare Anfangswertproblem (5.2) gezeigt.

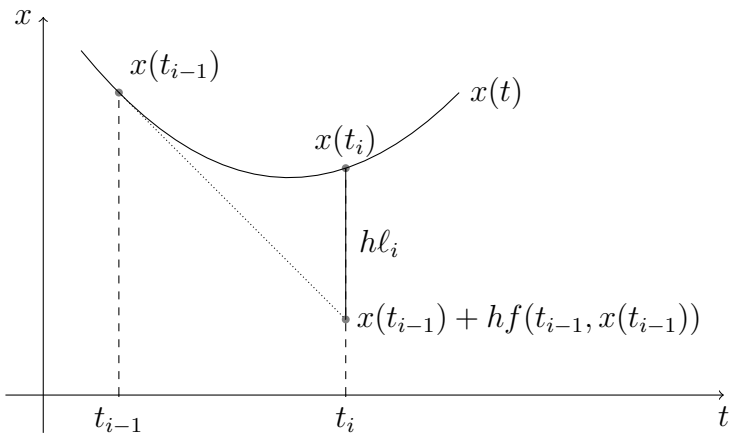

Abbildung 5.2: Lokaler Diskretisierungsfehler der expliziten Euler–Methode

Definition 5.5. Die Euler–Methode ist *konsistent*, das bedeutet, es gilt

$$\|\ell_i\| \to 0 \quad \text{für} \quad h \to 0.$$

Sei x zweimal stetig differenzierbar, so kann der lokale Diskretisierungsfehler mit Hilfe der Taylor–Reihe als

$$\ell_i = \frac{x(t_i) - x(t_{i-1})}{h} - x'(t_{i-1}) = h \int_0^1 x''(t_{i-1} + \theta h)(1 - \theta) \mathrm{d}\theta$$

dargestellt werden. Aus dieser Darstellung folgt, dass

$$\|\ell_i\| = O(h)$$

gilt.

Definition 5.6. Wenn für die Norm des lokalen Diskretisierungsfehlers gilt, dass

$$\|\ell_i\| = O(h^p),$$

so heißt $p \in \mathbb{N}$ die *Konsistenzordnung*.

Bemerkung 5.7. Die explizite Euler–Methode hat eine Konsistenzordnung von $p = 1$. Genauer kann die Norm des Diskretisierungsfehlers der expliziten Euler–Methode durch

$$\|\ell_i\| \leq \frac{M_2 h}{2}$$

eingeschränkt werden, wobei M_2 eine Schranke für $\|x''\|$ darstellt.

Definition 5.8. Der *globale Diskretisierungsfehler* ε_i ist definiert als die Differenz zwischen der diskreten Approximation ν_i und dem Wert der exakten Lösung $x(t_i)$.

$$\varepsilon_i = \nu_i - x(t_i) \tag{5.6}$$

Definition 5.9. Eine Diskretisierungsmethode heißt *konvergent*, wenn die Norm des globalen Diskretisierungsfehlers für $h \to 0$ verschwindet, also wenn gilt

$$\|\varepsilon_i\| \to 0 \quad \text{für} \quad h \to 0.$$

Gilt

$$\|\varepsilon_i\| = O(h^p),$$

so heißt p die *Konvergenzordnung*.

Bemerkung 5.10. Konsistenz einer Methode zieht nicht zwangsläufig Stabilität der Methode nach sich.

Definition 5.11. Betrachtet man zwei „parallele“ Schritte einer Einschritt–Diskretisierungsmethode

$$(t_{i-1}, \nu_{i-1}) \mapsto (t_i, \nu_i)$$
$$(t_{i-1}, \widetilde{\nu}_{i-1}) \mapsto (t_i, \widetilde{\nu}_i)$$

mit der Schrittweite h. Die Methode heißt *stabil*, wenn

$$\|\nu_i - \widetilde{\nu}_i\| \leq (1 + Sh) \, \|\nu_{i-1} - \widetilde{\nu}_{i-1}\|$$

gleichmäßig erfüllt ist für $h \leq h_0$ und einer von h unabhängigen Konstanten S.

Die Stabilität für die explizite Euler–Methode ist gewährleistet, da diese die Ungleichung

$$\|\nu_i - \widetilde{\nu}_i\| \leq (1 + Lh) \, \|\nu_{i-1} - \widetilde{\nu}_{i-1}\|$$

erfüllt, wobei L die Lipschitz–Konstante für die Funktion f darstellt. Gemeinsam mit der Konsistenz führt dies zu

Satz 5.12 (Konvergenz der expliziten Euler–Methode)**.** Sei $x \in C^2([0, T])$ und sei weiters $M_2 := \sup_{t \in [0,T]} x''(t)$. Der globale Fehler $\varepsilon_i = \nu_i - x(t_i)$ der expliziten Euler–Methode mit Schrittweite h und dem Anfangswert $\nu_0 = x_0 + \varepsilon_0$ erfüllt

$$\|\varepsilon_i\| \leq \mathrm{e}^{Lt_i} \, \|\varepsilon_0\| + \frac{\mathrm{e}^{Lt_i} - 1}{L} \frac{M_2 h}{2},$$

d.h. die explizite Euler–Methode ist konvergent von der Ordnung $p = 1$.

Beweis. Sei

$$\nu_{i-1} \mapsto \nu_i := \nu_{i-1} + h \cdot f(t_{i-1}, \nu_{i-1})$$

und

$$x(t_i) \mapsto \widetilde{\nu}_i := x(t_{i-1}) + h \cdot f(t_{i-1}, x(t_{i-1}))$$

ein „paralleler“ Schritt, gestartet von $x(t_{i-1})$. Es gilt wegen der Stabilität

$$\|\nu_i - \widetilde{\nu}_i\| \leq (1 + Lh) \|\nu_{i-1} - x(t_{i-1})\|$$

und aufgrund der Definition $\widetilde{\nu}_i - x(t_i) = h\ell_i$, wobei ℓ_i den lokalen Fehler bezeichnet. Es gilt

$$\begin{aligned} \|\varepsilon_i\| = \|\nu_i - x(t_i)\| &\leq \|\nu_i - \widetilde{\nu}_i\| + \|\widetilde{\nu}_i - x(t_i)\| \leq \\ &\leq (1 + Lh) \| \underbrace{\nu_{i-1} - x(t_{i-1})}_{=\varepsilon_{i-1}} \| + h \|\ell_i\| \leq \\ &\leq (1 + Lh) \|\varepsilon_{i-1}\| + \frac{M_2 h^2}{2}. \end{aligned}$$

Die Folge $\xi_i := \|\varepsilon_i\|$ erfüllt die Voraussetzungen des Lemmas von Gronwall B.2 mit $\omega = Lh$ und $\delta = \frac{M_2 h}{2}$. Damit folgt

$$\|\varepsilon_i\| \leq \mathrm{e}^{L\nu h} \|\varepsilon_0\| + \frac{\mathrm{e}^{L\nu h} - 1}{Lh} \frac{M_2 h}{2} = \mathrm{e}^{Lt_i} \|\varepsilon_0\| + \frac{\mathrm{e}^{Lt_i} - 1}{Lh} \frac{M_2 h}{2}.$$

□

Bemerkung 5.13. Es sei festgehalten:

(a) Der Term $\mathrm{e}^{Lt_i} \|\varepsilon_0\|$ kann als der „verstärkte Anfangsfehler“ interpretiert werden.

(b) Es gilt: Aus Konsistenz und Stabilität folgt die Konvergenz einer Diskretisierungsmethode.

5.1.2 Allgemeine explizite Einschrittverfahren

Definition 5.14. Ein *explizites Einschrittverfahren* für die Lösung eines Anfangswertproblems der Form (5.2) ist eine Diskretisierung der Form

$$\begin{aligned} \nu_0 &= y_0 \\ \frac{\nu_i - \nu_{i-1}}{h} &= \varphi(t_{i-1}, \nu_{i-1}; h_i) \qquad i = 1, 2, \ldots, \end{aligned} \tag{5.7}$$

wobei $h_i = t_{i+1} - t_i$ gilt.

Bemerkung 5.15. Die Funktion φ wird als *Inkrementfunktion* bezeichnet. Für $\varphi = f$ erhält man die Euler–Methode.

Die Definition eines expliziten Einschrittverfahrens, laut (5.7), erlaubt es, eine Rekursion für $\nu_{i-1} \mapsto \nu_i$ gemäß

$$\nu_i = \nu_{i-1} + h \cdot \varphi(t_{i-1}, \nu_{i-1})$$

anzugeben. Hier wird h als konstant angenommen und daher in der Inkrementfunktion φ nicht angegeben.

Die Wahl der Inkrementfunktion φ erlaubt es, höhere Konvergenzordnungen $p > 1$ zu realisieren. Die Konstruktion brauchbarer Inkrementfunktionen basiert auf der Approximation des Integrals in

$$x(t_i) = x(t_{i-1}) + \int_{t_{i-1}}^{t_i} f\big(t, x(t)\big)\mathrm{d}t.$$

Die Euler–Methode arbeitet mit der einfachsten aller Approximationen, der Approximation durch Rechtecksflächen gemäß

$$\int_{t_{i-1}}^{t_i} f\big(t, x(t)\big)\mathrm{d}t \approx h \cdot f\big(t_{i-1}, x(t_{i-1})\big).$$

Diese Approximation lässt sich verbessern, wenn mehr Information zur Verfügung steht, wie beispielsweise eine größere Anzahl an Funktionsauswertungen im Intervall $[t_{i-1}, t_i]$. Diese Funktionswerte sind unbekannt, können aber geschätzt[3] werden und so zu einer verbesserten Approximation führen.

Beispiel 5.16. Ein einfaches Beispiel dazu ist die Verbesserung des Verfahrens von Euler mit dem Zwischenwert

$$X_2 = \nu_{i-1} + \frac{h}{2} f(t_{i-1}, \nu_{i-1}).$$

Dieser Zwischenwert X_2 liefert gemeinsam mit der Euler–Methode die Rekursion

$$\nu_i = \nu_{i-1} + h \cdot f\big(t_{i-1} + \frac{h}{2}, X_2\big),$$

das sogenannte *Verfahren von Heun*.

Definition 5.17. Sei x die exakte Lösung des Anfangswertproblems (5.2). Eingesetzt in (5.7) wird der *lokale Diskretisierungsfehler* allgemeiner Einschrittverfahren definiert durch

$$\ell_i := \frac{x(t_i) - x(t_{i-1})}{h} - \varphi\big(t_{i-1}, x(t_{i-1}); h\big). \tag{5.8}$$

Die geometrische Interpretation ist dieselbe wie bei der expliziten Euler–Methode. $h\ell_i$ ist die Differenz aus dem exakten Lösungswert $x(t_i)$ und dem Ergebnis der Diskretisierung mit dem Startwert $x(t_{i-1})$.

Beispiel 5.18. Für das Verfahren von Heun gilt $p = 2$, d.h. $\ell_i = O(h^2)$.

Satz 5.19 (Konvergenz von Einschrittverfahren)**.** Betrachtet wird eine gewöhnliche Differentialgleichung mit der zugehörigen Lösung x, die hinreichend glatt[4] ist. Das explizite Einschrittverfahren (5.7) ist konsistent von der Ordnung p, es gilt

$$\|\ell_i\| \leq C \cdot h^p$$

[3] vorausgesagt (engl. predict)

[4] Die Existenz der notwendigen Anzahl von Ableitungen von x wird vorausgesetzt.

und das Einschrittverfahren ist stabil mit der Stabilitätskonstante S für $h \leq h_0$. Der globale Fehler $\varepsilon_i = \nu_i - x(t_i)$ des Einschrittverfahrens (mit Anfangswert $\nu_0 = y_0 + \varepsilon_0$) erfüllt

$$\|\varepsilon_i\| \leq \mathrm{e}^{St_i}\|\varepsilon_0\| + \frac{\mathrm{e}^{St_i} - 1}{S} C h^p \tag{5.9}$$

für die Schrittweite $h \leq h_0$. Das Einschrittverfahren ist konvergent von der Ordnung p, d.h. $\varepsilon_0 = O(h^p)$.

5.1.3 Explizite Einschrittverfahren nach Runge–Kutta

Es wird ein Einschrittverfahren der Form $\nu_{i-1} \mapsto \nu_i$ betrachtet. *Runge*[5]*–Kutta*[6]–Verfahren verwenden $s \in \mathbb{N}$ rekursiv definierte Zwischenapproximationen X_j an den Zwischenstellen $\tau_j := t_{j-1} + c_j \cdot h$ für $j = 1, \ldots, s$ mit $c_0 = 0$ und $0 < c_j < 1$. Daraus resultiert das Schema

$$\begin{aligned}
X_1 &= \nu_{i-1}, \\
X_2 &= \nu_{i-1} + h a_{21} f(\tau_1, X_1), \\
X_3 &= \nu_{i-1} + h\big(a_{31} f(\tau_1, X_1) + a_{32} f(\tau_2, X_2)\big), \\
&\vdots \\
X_s &= \nu_{i-1} + h\big(a_{s1} f(\tau_1, X_1) + a_{s2} f(\tau_2, X_2) + \ldots + a_{s,s-1} f(\tau_{s-1}, X_{s-1})\big), \\
\nu_i &= \nu_{i-1} + h \underbrace{\big(b_1 f(\tau_1, X_1) + b_2 f(\tau_2, X_2) + \ldots + b_{s-1} f(\tau_{s-1}, X_{s-1}) + b_s f(\tau_s, X_s)\big)}_{=\varphi(t_{i-1}, \nu_{i-1}; h)}.
\end{aligned} \tag{5.10}$$

Bemerkung 5.20. Man beachte, dass die Werte X_j im Allgemeinen von i abhängen.

Definition 5.21. Ein Einschrittverfahren, welches nach dem Schema (5.10) operiert, mit den Zwischenapproximationen X_j und den Zwischenstellen τ_j, wird als *Runge–Kutta–Einschrittverfahren* der *Ordnung* s bezeichnet.

Das Einschrittverfahren wird durch eine Koeffiziententabelle der Form

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

dargestellt, mit der Matrix

$$A = \begin{pmatrix} 0 & \ldots & & \\ a_{21} & 0 & \ldots & \\ a_{31} & a_{32} & 0 & \ldots \\ \vdots & \vdots & \ddots & 0 \\ a_{s1} & a_{s2} & \ldots & a_{s,s-1} \end{pmatrix} \in \mathbb{R}^{s \times (s-1)},$$

[5]Carl David Tolmé Runge (1856 – 1927), deutscher Mathematiker
[6]Martin Wilhelm Kutta (1867 – 1944), deutscher Mathematiker

und den Vektoren

$$c = (0, c_2, c_3, \ldots, x_{s-1}, c_s)^T,\ b = (b_1, b_2, \ldots, b_{s-1}, b_s)^T \in \mathbb{R}^s\,.$$

Die Koeffiziententabelle hat die Gestalt

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \ldots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \ldots & b_{s-1} & b_s \end{array}$$

und wird als *Butcher*[7] *–Tableau* oder *Butcher–Array* bezeichnet.

Beispiel 5.22. In den folgenden Beispielen werden einige Butcher–Tableaus für typische Verfahren angegeben.

(a) Das Butcher–Tableau der *Euler–Methode* ($p = 1$) ist sehr einfach strukturiert:

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

(b) Das Butcher–Tableau des *Verfahrens von Heun* ($p = 2$) ergibt sich zu:

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

(c) Das *klassische 4–Schrittverfahren von Kutta (1901)* ist ein Schema 4. Ordnung der Form:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Die Einträge b_i des Vektors b korrespondieren mit dem Quadraturverfahren *pulcherrima*.

[7] John Charles Butcher (geb. 1933), neuseeländischer Mathematiker

Praktische Aspekte

Bei praktischen Berechnungen werden zwei Integrationsschemata parallel implementiert. Man spricht von einem *Paar* von Integrationsverfahren. Das exaktere Verfahren wird dabei als Fehlerschätzer für das weniger exakte Verfahren verwendet mit dem Ziel der Fehler– und Schrittweitensteuerung.
Bei einer Runge–Kutta–Implementierung verwendet man ein beliebiges Paar von Runge–Kutta–Schemata mit unterschiedlichen Ordnungen p und $\overline{p}$; für gewöhnlich gilt $\overline{p} = p + 1$. Um den Aufwand an Funktionsauswertungen zu minimieren, verwendet man Schemata, welche die selben Stellen an Zwischenauswertungen verwenden und sich nur in den Gewichtungen b_i und $\overline{b}_i$ unterscheiden. So eine Kombination von Runge–Kutta–Schemata wird als *embedded Runge–Kutta*–Methode oder *Runge–Kutta–Fehlberg*[8]–Methode bezeichnet.

5.1.4 Implizite Einschrittverfahren nach Runge–Kutta

Eingangs sei das grundsätzlich andere Verhalten der *impliziten Methode von Euler* dargestellt, welche durch eine implizite Diskretisierung definiert ist.

Definition 5.23. Die *implizite Methode von Euler* ist definiert durch

$$\frac{\nu_i - \nu_{i-1}}{h} = f(t_i, \nu_i), \qquad i = 1, 2, \ldots \tag{5.11}$$

Diese Methode ist implizit, da die Lösung eines i.A. nichtlinearen Gleichungssystems für jeden Schritt der Rekursion notwendig ist. Implizite Schemata sind die einzigen Iterationsverfahren, um numerische Lösungen für *steife Differentialgleichungen* zu erhalten.

Andere implizite Einschrittverfahren sind jene, die der Klasse der *impliziten Runge–Kutta Einschrittverfahren* angehören. Diese sind eine Verallgemeinerung der expliziten Runge–Kutta–Methode.

Definition 5.24. Ein *implizites s–Schritt Runge–Kutta–Einschrittverfahren* mit den Zwischenapproximationen X_j an den Stellen $\tau_j := t_{i-1} + c_j \cdot h$ mit $j = 1, \ldots, s$ wird definiert durch das folgende $s \times n$ System von nichtlinearen Gleichungen.

$$\begin{aligned}
X_1 &= \nu_{i-1} + h\big(a_{11} f(\tau_1, X_1) + a_{12} f(\tau_2, X_2) + \ldots + a_{1s} f(\tau_s, X_s)\big) \\
X_2 &= \nu_{i-1} + h\big(a_{21} f(\tau_1, X_1) + a_{22} f(\tau_2, X_2) + \ldots + a_{2s} f(\tau_s, X_s)\big) \\
&\vdots \\
X_s &= \nu_{i-1} + h\big(a_{s1} f(\tau_1, X_1) + a_{s2} f(\tau_2, X_2) + \ldots + a_{ss} f(\tau_s, X_s)\big) \\
\nu_i &= \nu_{i-1} + h\big(b_1 f(\tau_1, X_1) + b_2 f(\tau_2, X_2) + \ldots + b_s f(\tau_s, X_s)\big)
\end{aligned} \tag{5.12}$$

Zuerst müssen die Zwischenapproximationen X_j ermittelt, anschließend die ν_i unter Verwendung einer Integrationsformel mit Knoten c_j und Gewichten b_j bestimmt werden.
Auch für die impliziten Runge–Kutta–Verfahren wird ein Butcher–Tableau zur Repräsentation angegeben.

[8] Erwin Fehlberg (geb. 1908), deutscher Mathematiker

Die Struktur ist analog zu jenen der expliziten Runge–Kutta–Verfahren mit dem Unterschied, dass die Matrix A im Allgemeinen voll besetzt ist.

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

Beispiel 5.25. Im Folgenden werden zwei typische Beispiele anhand des Butcher–Tableaus definiert.

(a) Die *implizite Euler–Methode* wird durch das folgende Butcher–Tableau charakterisiert:

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

(b) Die *implizite 2–Schritt Trapezregel* ist definiert durch das Tableau:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Definition 5.26. Wählt man die c_j als die Nullstellen der Legendre–Polynome auf dem Intervall $[0, 1]$ vom Grad s, so führt die angemessene Wahl der Gewichte b_j (und damit auch der a_{ij}) zur *Gauß–Kollokation*.

Bemerkung 5.27. Die Gauß–Kollokation für den Fall $x' = f(t, x) = f(t)$ entspricht der Gauß–Quadratur.

Beispiel 5.28. Ein einfaches Beispiel zur Gauß–Kollokation ist durch das folgende Butcher–Tableau definiert.

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

Dieses Einschrittverfahren wird als *implizite Mittelpunktsregel* bezeichnet und wird beschrieben durch die Rekursion

$$\nu_i = \nu_{i-1} + h \cdot f\big(t_{i-1} + \tfrac{h}{2}, \tfrac{1}{2}(\nu_{i-1} + \nu_i)\big). \tag{5.13}$$

5.1.5 Stabilität und Konvergenz von Einschrittverfahren

Definition 5.29. Für beliebige Einschrittverfahren definiert man die *Stabilitätsfunktion* Φ als eine Funktion, für welche die Diskretisierung des skalaren Problems

$$x' = \lambda x$$

geschrieben werden kann als

$$\nu_i = \Phi(h\lambda) \cdot \nu_{i-1}. \tag{5.14}$$

Beispiel 5.30. Einige Beispiele zur Stabilitätsfunktion:

(a) Für *explizite Runge–Kutta–Verfahren* ist die Stabilitätsfunktion Φ immer ein Polynom.

(b) Die *implizite Euler–Methode* hat eine rationale Stabilitätsfunktion

$$\Phi(z) = \frac{1}{1-z}.$$

Betrachtet man das steife Problem

$$x' = \lambda x, \qquad \lambda \ll 0,$$

so spiegelt sich das robuste Verhalten der impliziten Euler–Methode wieder in

$$0 < \Phi(h\lambda) = \frac{1}{1-h\lambda} < 1,$$

für alle $h > 0$ und $\lambda < 0$. Ebenso gilt $\forall \lambda \in \mathbb{C}$ mit $\operatorname{Re}\lambda < 0$

$$|\Phi(h\lambda)| < 1.$$

Dies ist von Bedeutung, da das skalare Problem aus Definition 5.29 komplexe Eigenwerte aufweisen kann.

Definition 5.31. Das *Stabilitätsgebiet* eines Einschrittverfahrens ist definiert als

$$S_\Phi := \{z \in \mathbb{C} : |\Phi(z)| \leq 1\}. \tag{5.15}$$

Ein Einschrittverfahren heißt *A–stabil*, wenn für die Stabilitätsfunktion Φ

$$|\Phi(z)| \leq 1 \quad \forall z \in \mathbb{C}, \text{ mit } \operatorname{Re} z < 0 \tag{5.16}$$

gilt, d.h. $\{z \in \mathbb{C} : \operatorname{Re} z \leq 0\} \subset S_\Phi$.

Das Einschrittverfahren heißt *strikt A–stabil* oder auch *L–stabil*, wenn zusätzlich

$$|\Phi(z)| \to 0 \qquad \text{für} \qquad \operatorname{Re} z \to -\infty \tag{5.17}$$

gilt. Wenn $|\Phi(z)| \leq 1$ (wenigstens) für alle $z \in \mathbb{R}^-$ gilt, dann heißt die Methode *steif stabil*.

Beispiel 5.32. Einige Beispiele zum Thema A–Stabilität:

(a) Die explizite Euler–Methode ist strikt A–stabil.

(b) Man betrachte die *implizite Trapezregel*, welche definiert ist durch

$$\nu_i = \nu_{i-1} + \frac{h}{2}\big(f(t_{i-1}, \nu_{i-1}) + f(t_i, \nu_i)\big).$$

Die zugehörige Stabilitätsfunktion lautet

$$\Phi(z) = \frac{1 + \frac{z}{2}}{1 - \frac{z}{2}}.$$

Diese Methode ist A–stabil, aber nicht strikt A–stabil.

(c) Eine explizite Methode mit einer polynomiellen Stabilitätsfunktion Φ kann nicht A–stabil sein, denn der Stabilitätsbereich $S := \{z \in \mathbb{C} : |\Phi(z)| \leq 1\}$ ist immer eine beschränkte Teilmenge von $\mathbb{C}$.

Satz 5.33. Ein Einschrittverfahren mit der Stabilitätsfunktion Φ ist genau dann A–stabil, wenn

$$\operatorname{Re} z \leq 0 \qquad \Longrightarrow \qquad |\Phi(z)| \leq 1$$

$\forall z \in \mathbb{C}$ gilt.

Der Beweis folgt direkt aus der Definition.

A–Stabilität ist ein sehr brauchbares Konzept für die Untersuchung auf Stabilität von Diskretisierungsschemata für steife Probleme. Es wird mit der skalaren Modellgleichung

$$x' = \lambda x$$

gearbeitet, bzw. wenn Systeme mit konstanten Koeffizienten vorliegen, ist die Modellgleichung

$$x' = Ax$$

mit $A = X\Lambda X^{-1}$, wobei Λ eine Diagonalmatrix mit den Eigenwerten λ_l ist.

Ein modernerer Zugang und auch ein wesentlich mächtigeres Konzept für steife Probleme führt auf die B–Stabilität. Man betrachte dazu eine Klasse von gewöhnlichen Differentialgleichungen, deren rechte Seiten die einseitige Lipschitz–Bedingung erfüllen.

Definition 5.34. Eine Funktion $f\colon D \to \mathbb{R}^n$ erfüllt die *einseitige Lipschitz–Bedingung* bezüglich x auf D, wenn

$$\langle x - \widetilde{x}, f(t,x) - f(t,\widetilde{x})\rangle \leq m\|x - \widetilde{x}\|_2^2 \tag{5.18}$$

für alle $(t,x),(t,\widetilde{x}) \in D$ gilt[9]. Dabei ist m die *einseitige Lipschitz–Konstante* von f bezüglich x auf D.

Bemerkung 5.35. Die einseitige Lipschitz–Bedingung schwächt die Lipschitz–Bedingung ab. Dies zeigt der Zusammenhang

$$\langle x - \widetilde{x}, f(t,x) - f(t,\widetilde{x})\rangle \leq \|x - \widetilde{x}\|_2 \|f(t,x) - f(t,\widetilde{x})\|_2 \leq L\|x - \widetilde{x}\|_2^2.$$

[9] $\langle .,. \rangle$ bezeichnet dabei das euklidische Skalarprodukt auf $\mathbb{R}^n$ und es gilt $\|.\|_2^2 = \langle .,. \rangle$.

Definition 5.36. Man betrachte zwei parallele Schritte der Diskretisierung eines Einschrittverfahrens mit Schrittweite h

$$(t_{i-1}, \nu_{i-1}) \mapsto (t_i, \nu_i), \tag{5.19}$$
$$(t_{i-1}, \widetilde{\nu}_{i-1}) \mapsto (t_i, \widetilde{\nu}_i), \tag{5.20}$$

eines Anfangswertproblems mit einseitiger Lipschitz–Konstante m der Funktion f. Die Methode heißt *B–stabil*, wenn

$$\|\nu_i - \widetilde{\nu}_i\|_2 \leq \Phi(hm)\|\nu_{i-1} - \widetilde{\nu}_{i-1}\|_2 \tag{5.21}$$

gilt für eine glatte Stabilitätsfunktion Φ, welche $\Phi(0) = 1$ erfüllt.

Bemerkung 5.37. Die Bedingung $\Phi(0) = 1$ aus Definition 5.36 kann unter Umständen eine Restriktion der Schrittweite im Sinne $h \leq h_0$ zur Folge haben.

Satz 5.38 (B–Stabilität der impliziten Euler–Methode)**.** Sei h die Schrittweite der Methode aus Definition 5.23. Die implizite Euler–Methode ist B–stabil mit der Stabilitätsfunktion

$$\Phi(hm) = \frac{1}{1 - hm}. \tag{5.22}$$

Beweis. Mit Definition 5.23 der impliziten Euler–Methode und Definition 5.36 erhält man

$$\nu_i - \widetilde{\nu}_i = \nu_{i-1} - \widetilde{\nu}_{i-1} + h\big(f(t_i, \nu_i) - f(t_i, \widetilde{\nu}_i)\big).$$

Bildet man das innere Produkt dieses Ausdrucks mit sich selbst

$$\langle \nu_i - \widetilde{\nu}_i, \nu_i - \widetilde{\nu}_i \rangle = \|\nu_i - \widetilde{\nu}_i\|_2^2,$$

so erhält man mit der Ungleichung von Cauchy–Schwarz und der einseitigen Lipschitz–Stetigkeit der Abbildung f die Abschätzung

$$\begin{aligned}\|\nu_i - \widetilde{\nu}_i\|_2^2 =& \langle \nu_{i-1} + \widetilde{\nu}_{i-1} \rangle + h \langle \nu_i - \widetilde{\nu}_i, f(t_i, \nu_i) - f(t_i, \widetilde{\nu}_i) \rangle \\ \leq& \|\nu_{i-1} - \widetilde{\nu}_{i-1}\|_2 \|\nu_i - \widetilde{\nu}_i\|_2 + hm\|\nu_i - \widetilde{\nu}_i\|_2^2.\end{aligned}$$

Aus der Umformung

$$(1 - hm)\|\nu_i - \widetilde{\nu}_i\|_2^2 \leq \|\nu_{i-1} - \widetilde{\nu}_{i-1}\|_2 \|\nu_i - \widetilde{\nu}_i\|_2$$

und $\|\nu_i - \widetilde{\nu}_i\|_2 \neq 0$ folgt unmittelbar die B–Stabilität. □

Bemerkung 5.39. Es sei angemerkt, dass hier für $m > 0$ die Einschränkung $hm \leq h_0 m < 1$ verlangt werden muss. Die implizite Euler–Gleichung wäre sonst nicht lösbar.

Definition 5.40. Die *implizite Euler–Methode* (5.23) heißt *B–konvergent* von der Ordnung $p = 1$, wenn Sie den folgenden Satz erfüllt.

Satz 5.41 (B–Konvergenz der impliziten Euler–Methode)**.** Sei f einseitig Lipschitz–stetig mit der einseitigen Lipschitz–Konstante m, $x \in C^2[0, T]$ und $M_2 := \sup_{t \in [0,T]} \|x''(t)\|_2$.

Für $hm \leq h_0 m < 1$ genügt der globale Fehler $\varepsilon_i = \nu_i - x(t_i)$ der impliziten Euler–Methode (5.23) mit Schrittweite h und Anfangswert $\nu_0 = x_0 + \varepsilon_0$ der Ungleichung

$$\|\varepsilon_i\|_2 \leq \mathrm{e}^{\overline{m}t_i}\|\varepsilon_0\|_2 + \frac{\mathrm{e}^{\overline{m}t_i - 1}}{\overline{m}} \frac{1}{1 - h_0 m} \frac{M_2 h}{2} \tag{5.23}$$

mit

$$\overline{m} := \frac{m}{1 - hm} \leq \frac{m}{1 - h_0 m}.$$

Beweis. Zu Beginn wird gezeigt, dass die Konsistenzordnung der impliziten Euler–Methode $p = 1$ ist. Dazu betrachte man den lokalen Fehler ℓ_i, der mit dem Taylorschen Lehrsatz entwickelt wird.

$$\ell_i = \frac{x(t_i) - x(t_{i-1})}{h} - x'(t_i) = -h \int_0^1 x''(t_i - \theta h)(1 - \theta) \mathrm{d}\theta$$

Weiters gilt

$$\|\ell_i\|_2 \leq \frac{M_2 h}{2}.$$

Zunächst betrachte man einen impliziten Euler–Schritt

$$\nu_{i-1} \mapsto \nu_i := \nu_{i-1} + h \cdot f(t_i, \nu_i)$$

und einen von $x(t_{i-1})$ ausgehenden parallelen Schritt

$$x(t_{i-1}) \mapsto \widetilde{\nu}_i := x(t_{i-1}) + h \cdot f(t_i, \widetilde{\nu}_i).$$

Wegen der B–Stabilität aus Satz 5.38 gilt

$$\|\nu_i - \widetilde{\nu}_i\|_2 \leq \frac{1}{1 - hm} \, \|\nu_{i-1} - x(t_{i-1})\|_2.$$

Mit der Definition von $\widetilde{\nu}_i$ gilt

$$\begin{aligned} \widetilde{\nu}_i - x(t_i) &= x(t_{i-1}) - x(t_i) + h \cdot f(t_i, \widetilde{\nu}_i) \\ &= x(t_{i-1}) - x(t_i) + h \cdot x'(t_i) + h \left(f(t_i, \widetilde{\nu}_i) - f\big(t_i, x(t_i)\big) \right) \\ &= -h\ell_i + h \left(f(t_i, \nu_i) - f\big(t_i, x(t_i)\big) \right). \end{aligned}$$

Analog zum Beweis in Satz 5.38 wird das Skalarprodukt des Ausdrucks mit sich selbst gebildet und die Anwendung der einseitigen Lipschitz–Stetigkeit führt auf

$$\|\widetilde{\nu}_i - x(t_i)\|_2 \leq \frac{1}{1 - hm} \|h\ell_i\|_2 \leq \frac{1}{1 - hm} \frac{M_2 h^2}{2}.$$

Folglich gilt

$$\begin{aligned} \|\varepsilon_i\|_2 = &\|\nu_i - x(t_i)\|_2 \leq \|\nu_i - \widetilde{\nu}_i\|_2 + \|\widetilde{\nu}_i - x(t_i)\|_2 \\ \leq &\frac{1}{1 - hm} \left(\|\varepsilon_{i-1}\|_2 + \frac{M_2 h^2}{2} \right). \end{aligned}$$

Deswegen erfüllt $\xi_i := \|\varepsilon_i\|_2$ die Voraussetzungen der Einschritt–Version des diskreten Lemmas von Gronwall laut Satz B.2 mit $\omega = \frac{hm}{1 - hm}$ und $\delta = \frac{1}{1 - h_0 m} \frac{M_2 h^2}{2}$. Damit folgt die Behauptung des Satzes. □

5.1.6 Schrittweitensteuerung und Fehlerschätzung

Der Rechenaufwand für ein Verfahren zur Lösung von Differentialgleichungen ist erheblich von der Wahl der Schrittweite abhängig. Um ein Verfahren effizient zu halten ist es vernünftig, die Schrittweite variabel zu wählen. Im Fall von konstanter Schrittweite würde diese ansonsten von der „ungünstigsten“ Region beeinflusst werden.

Aus Kapitel 5.1.2 ist bekannt, dass der lokale Diskretisierungsfehler ℓ_i ein Maß für die Abweichung der numerischen Approximation von der exakten Lösung ist. Genauer beschreibt $h\ell_i$ diese Abweichung. Dies zeigt, dass die Schrittweite direkt die Abweichung beeinflusst. Hier muss nun ein Kompromiss gefunden werden zwischen geringer Schrittweite, um die Abweichung gering zu halten, und einer hinreichend großen Schrittweite, um den Rechenaufwand gering zu halten. Die Schrittweite soll adaptiv gesteuert werden, wozu eine Fehlerschätzung notwendig ist. Hierzu gibt es die zwei folgenden grundlegenden Ideen:

(a) Es werden für eine Schrittweite h zwei Verfahren, häufig Runge–Kutta, der Ordnung n und $n+1$, berechnet.

(b) Es wird eine Rekursion $\nu_i \mapsto \nu_{i+1}$ mit Schrittweite h und eine Rekursion $\nu_i \mapsto \widetilde{\nu}_{i+1}$ mit $\frac{h}{2}$ berechnet.

Die beiden resultierenden numerischen Approximationen werden jeweils verwendet, um zu ermitteln, ob die Schrittweite verkleinert, gleich belassen oder vergrößert werden kann.

Um ein Maß für den Fehler zu erhalten, muss zuerst ein *Schätzer des lokalen Fehlers* ermittelt werden, um mit diesem anschließend eine Schrittweitensteuerung umsetzen zu können. Dazu betrachtet man zwei unterschiedliche Einschrittverfahren, welche durch die Inkrementfunktionen φ und $\widetilde{\varphi}$ symbolisiert werden, der Form

$$\nu_i = \nu_{i-1} + h_i \varphi(t_i, \nu_i; h_i), \tag{5.24a}$$

$$\widetilde{\nu}_i = \nu_{i-1} + \widetilde{h}_i \widetilde{\varphi}(t_i, \tilde{\nu}_i; h_i). \tag{5.24b}$$

Hierbei können h_i und $\widetilde{h}_i$ zwei unterschiedliche Schrittweiten darstellen. Es sind nun zwei Möglichkeiten denkbar:

(a) Es gilt $0 < \widetilde{h}_i < h_i$ und $\widetilde{\varphi} = \varphi$.

(b) Es gilt $\widetilde{h}_i = h_i$ und $\widetilde{\varphi}$ besitzt eine höhere Konsistenzordnung als φ.

In beiden Fällen kann $\widetilde{\nu}_i$ als die genauere Approximation der Lösung angenommen werden. Ein möglicher Schätzer, welcher oft verwendet wird, lautet dann

$$\eta_i = \|\widetilde{\nu}_i - \nu_i\|. \tag{5.25}$$

Weiters wird zur Hilfe ein Anfangswertproblem der Form

$$\begin{aligned} x'(t) &= f(t, x(t)), \qquad t \geq t_{i-1} \\ x(t_{i-1}) &= \nu_{i-1} \end{aligned} \tag{5.26}$$

formuliert und für dieses die beiden lokalen Diskretisierungsfehler ℓ_i und $\widetilde{\ell}_i$ für die zwei expliziten Einschrittverfahren φ und $\widetilde{\varphi}$ mit gleicher Schrittweite $\widetilde{h}_i = h_i$ angegeben.

$$\ell_i = \frac{x(t_i) - x(t_{i-1})}{h_i} - \varphi(t_{i-1}, x(t_{i-1}); h_i) \tag{5.27a}$$

$$\widetilde{\ell}_i = \frac{x(t_i) - x(t_{i-1})}{h_i} - \widetilde{\varphi}(t_{i-1}, x(t_{i-1}); h_i) \tag{5.27b}$$

Der folgende Satz ermöglicht eine Aussage über den Schätzer η_i.

Satz 5.42. Seien ℓ_i und $\widetilde{\ell}_i$ gemäß (5.27) die beiden lokalen Diskretisierungsfehler der beiden expliziten Einschrittverfahren φ und $\widetilde{\varphi}$ aus (5.24) des Anfangswertproblems (5.26) mit gleicher Schrittweite $\widetilde{h}_i = h_i$ und

$$\theta := \frac{\|\widetilde{\ell}_i\|}{\|\ell_i\|}. \tag{5.28}$$

Dann gilt für den Schätzer η_i aus (5.25)

$$(1-\theta)h_i\|\ell_i\| \leq \eta_i \leq (1+\theta)h_i\|\ell_i\|.$$

Beweis. Es gilt

$$\begin{aligned} x(t_i) &= \nu_{i-1} + h_i\varphi(ti-1, \nu_{i-1}; h_i) + h_i\ell_i \\ \nu_i &= \nu_{i-1} + h_i\varphi(ti-1, \nu_{i-1}; h_i) \end{aligned}$$

und daher

$$x(t_i) - \nu_i = h_i\ell_i.$$

Analog gilt

$$x(t_i) - \widetilde{\nu}_i = h_i\widetilde{\ell}_i.$$

Nun kann der Fehlerschätzer η_i berechnet werden als

$$\eta_i = \|\widetilde{\nu}_i - \nu_i\| = h_i\|\widetilde{\ell}_i - \ell_i\|.$$

Weiters gilt wegen der Dreiecksungleichung

$$(1-\theta)\|\ell_i\| = \|\ell_i\| - \|\widetilde{\ell}_i\| \leq \|\ell_i - \widetilde{\ell}_i\| = \frac{1}{h_i}\eta_i$$

und

$$\frac{1}{h_i}\eta_i = \|\ell_i - \widetilde{\ell}_i\| \leq \|\ell_i\| + \|\widetilde{\ell}_i\| = (1+\eta)\|\ell_i\|.$$

□

Der Schätzer η_i ist also eine umso bessere Approximation von $h_i\|\ell_i\|$, je kleiner θ ist. Hinsichtlich der Schrittweitensteuerung werden nun die folgenden Überlegungen angestellt. Wegen

$$\eta_i = \|\widetilde{\nu}_i - \nu_i\| = h_i\|\widetilde{\ell}_i - \ell_i\|$$

kann unter der Annahme, dass das bessere Vergleichsverfahren $\widetilde{\varphi}$ einen sehr geringen, gegen Null gehenden Fehler $\widetilde{\ell}_i$ aufweist, der Fehlerschätzer als

$$\eta_i \approx h_i\|\ell_i\| \tag{5.29}$$

aufgefasst werden. Dieser soll durch eine vorgegebene Toleranz `tol` beschränkt werden

$$\eta_i \leq \texttt{tol}.$$

Einschrittverfahren mit einer Konsistenzordnung p haben gemäß der Definition die Eigenschaft $\|\ell_i\| = O(h^p)$. Dies erlaubt wegen (5.29) für den Fehlerschätzer η_i den Ansatz

$$\eta_i = c(t_i)h_i^{p+1} + O(h^{p+2})$$

mit einer von h_i unabhängigen Funktion c. Sei h_i^* jene Schrittweite, bei der die vorgegebene Toleranz erreicht wird, also für den zugehörigen Schätzer

$$\eta_i^* = \texttt{tol} = c(t_i)(h_i^*)^{p+1} + O\big((h_i^*)^{p+2}\big)$$

gilt. Mit den Gleichungen für η_i und $\eta_i^* = \texttt{tol}$ folgt

$$\frac{\texttt{tol}}{\eta_i} = \left(\frac{h_i^*}{h_i}\right)(1 + O(h_i)).$$

Eine Strategie zur Schrittweitensteuerung ist, den Term $O(h_i)$ zu vernachlässigen und die obige Gleichung nach h_i^* aufzulösen. Um die Vernachlässigung zu berücksichtigen, wird ein Sicherheitsfaktor $\chi < 1$ vorgesehen und für die neue Schrittweite gilt

$$h_i^* = \chi \left(\frac{\eta_i}{\texttt{tol}}\right)^{m+1} h_i. \tag{5.30}$$

5.2 Lineare Mehrschrittverfahren

Einschrittverfahren basieren auf der Rekursion $\nu_{i-1} \mapsto \nu_i$, welche keine Information über früher berechnete Werte verwendet. Nachteile von Einschrittverfahren beim Versuch der Realisierung höherer Ordnungen sind die zusätzlichen Auswertungen der Funktion f in jedem Schritt, wenn diese Auswertungen aufwändig und zeitintensiv sind.

Mehrschrittverfahren werden so konstruiert, dass die Auswertungen von f und ν aus vorangegangenen Berechnungen effizient verwendet werden.

Definition 5.43. Mit $f_i := f(t_i, \nu_i)$ heißt die Diskretisierung einer Lösung eines Anfangswertproblems der Form

$$\frac{1}{h}\left(\alpha_0\nu_{i-k} + \ldots + \alpha_{k-1}\nu_{i-1} + \alpha_k\nu_i\right) = \beta_0 f_{i-k} + \ldots + \beta_{k-1}f_{i-1} + \beta_k f_i \tag{5.31}$$

mit $k \in \mathbb{N}$ ein k–Schritt *lineares Mehrschrittverfahren*. Linear bezieht sich darauf, dass ν_j und f_j nur linear in Gleichung (5.31) auftreten. Jeder Schritt der Rekursion $(\nu_{i-k}, \ldots, \nu_{i-1}) \mapsto \nu_i$ erfordert somit nur eine einzige neue Auswertung von f.

Die Gleichung kann als

$$\frac{1}{h}\sum_{m=0}^{k} \alpha_m \nu_{i-k+m} = \sum_{m=0}^{k} \beta_m f_{i-k+m} \tag{5.32}$$

geschrieben werden, wobei

$$\sigma(t_{i-k}, \nu_{i-k}, \ldots, t_i, \nu_i) := \sum_{m=0}^{k} \beta_m f_{i-k+m} \tag{5.33}$$

als *Inkrementfunktion* bezeichnet wird. Für Mehrschrittverfahren wird $\alpha_0 + \ldots + \alpha_k = 0$, mit $\alpha_k \neq 0$, gefordert, und die Koeffizienten β_k entstehen durch ein geeignet gewähltes Quadraturverfahren. Ein k–Schritt Mehrschrittverfahren benötigt $k-1$ Startwerte ν_j mit $j = 1, \ldots, k-1$, welche durch eine geeignete Methode, wie beispielsweise ein Runge–Kutta–Verfahren, berechnet werden müssen.

Mit der Abbildung $\nu_{i+j} \mapsto z^j$ und $f_{i+j} \mapsto z^j$ für ein festes $i \in \mathbb{N}$ und ein beliebiges $j \in \mathbb{Z}$ gelangt man von Gleichung (5.32) auf

$$\frac{1}{h} \sum_{m=0}^{k} \alpha_m z^{m-k} = \sum_{m=0}^{k} \beta_m z^{m-k}$$

und für $z \neq 0$ erhält man die Gleichung

$$\sum_{m=0}^{k} \alpha_m z^m = h \sum_{m=0}^{k} \beta_m z^m. \tag{5.34}$$

Definition 5.44. Die Polynome

$$\rho(z) := \sum_{m=0}^{k} \alpha_m z^m \qquad \text{und} \qquad \sigma(z) := \sum_{m=0}^{k} \beta_m z^m$$

heißen die *charakteristischen Polynome* des Mehrschrittverfahrens. Das Mehrschrittverfahren wird durch das Paar (ρ, σ) eindeutig definiert.

Definition 5.45. Ein lineares Mehrschrittverfahren heißt *explizit*, wenn $\beta_k = 0$ gilt, anderfalls *implizit*.

Beispiel 5.46. Das 2–Schritt–Verfahren definiert durch

$$\frac{1}{2h} (\nu_i - \nu_{i-2}) = f(t_{i-1}, \nu_{i-1}), \qquad i = 1, 2, \ldots,$$

heißt *explizite Mittelpunktsregel*. Zur Berechnung von ν_1 kann beispielsweise ein expliziter Euler–Schritt benutzt werden. Es gilt

$$\rho(z) = \frac{1}{2}(z^2 - 1), \qquad \sigma(z) = z,$$

und damit

$$\alpha_0 = -\frac{1}{2}, \quad \alpha_1 = 0, \quad \alpha_2 = \frac{1}{2}, \quad \beta_0 = 0, \quad \beta_1 = 1, \quad \beta_2 = 0.$$

5.2.1 Stabilität und Konvergenz von Mehrschrittverfahren

Die Stabilität von Einschrittverfahren wird anhand der Modellgleichung

$$x' = \lambda x$$

untersucht. Auch für Mehrschrittverfahren ist diese von Interesse und führt zur

Definition 5.47. Für $\lambda \in \mathbb{C}$ heißt das Anfangswertproblem

$$x'(t) = \lambda x(t), \quad t \geq 0, \qquad \text{mit} \qquad x(0) = 1$$

das *Testproblem von Dahlquist*[10].

Die Lösung des Testproblems von Dahlquist für lineare Mehrschrittverfahren lautet

$$\sum_{m=0}^{k} \alpha_m \nu_{i-k+m} = h\lambda \sum_{m=0}^{k} \beta_m \nu_{i-k+m}$$

oder äquivalent

$$\sum_{m=0}^{k} (\alpha_m - h\lambda\beta_m)\nu_{i-k+m} = 0. \tag{5.35}$$

Definition 5.48. Mit

$$\mathrm{p}_\Phi(z, h\lambda) := \sum_{m=0}^{k} (\alpha_m - h\lambda\beta_m) z^m = \rho(z) - h\lambda\sigma(z) \tag{5.36}$$

heißt p_Φ das *Stabilitätspolynom* des linearen Mehrschrittverfahrens (ρ, σ).

Satz 5.49. Sei $\mu \in \mathbb{C}$ eine einfache Nullstelle des Stabilitätspolynoms p_Φ eines linearen Mehrschrittverfahrens (ρ, σ). Dann ist $\nu_i = \mu^i$ eine Lösung der Gleichung (5.35). Ist $\mu \in \mathbb{C}$ eine mehrfache Nullstelle des Stabilitätspolynoms p_Φ mit Vielfachheit größer gleich 2, so ist $\nu_i = i\mu^i$ eine Lösung der Gleichung (5.35).

Beweis. Fallunterscheidung anhand der Vielfachheit von μ:

(a) Sei die Vielfachheit von μ gleich 1. $\nu_i = \mu^i$ ist Lösung von (5.35) genau dann, wenn

$$0 = \sum_{m=0}^{k} (\alpha_m - h\lambda\beta_m)\mu^{i-k+m} = \mu^{i-k} \sum_{m=0}^{k} (\alpha_m - h\lambda\beta_m)\mu^m = \mu^{i-k}\, \mathrm{p}_\Phi(\mu; h\lambda)$$

gilt. Für $\mu = 0$ ist dies stets erfüllt, für $\mu \neq 0$ muss μ eine Nullstelle von p_Φ sein. Umgekehrt müssen Nullstellen von p_Φ Lösungen der Differenzengleichung (5.35) gemäß des Ansatzes $\nu_i = \mu^i$ liefern.

[10] Germund Dahlquist (1925 – 2005), schwedischer Mathematiker

(b) Sei die Vielfachheit von μ größer 1. Es gilt (mindestens) $\mathrm{p}_\Phi(\mu; h\lambda) = \mathrm{p}'_\Phi(\mu; h\lambda) = 0$ und weiters gilt

$$\mathrm{p}'_\Phi(z, h\lambda) \sum_{m=0}^{k} (\alpha_m - h\lambda\beta_m) m z^{m-1}.$$

$\nu_i = i\mu^i$ ist Lösung von (5.35) genau dann, wenn gilt

$$\begin{aligned} 0 &= \sum_{m=0}^{k} (\alpha_m - h\lambda\beta_m)(i-k+m)\mu^{i-k+m} = \\ &= (i-k)\mu^{i-k} \sum_{m=0}^{k} (\alpha_m - h\lambda\beta_m)\mu^m + \mu^{i-k+1} \sum_{m=0}^{k} (\alpha_m - h\lambda\beta_m) m \mu^{m-1} = \\ &= (i-k)\mu^{i-k}\, \mathrm{p}_\Phi(\mu; h\lambda) + \mu^{i-k+1}\, \mathrm{p}'_\Phi(\mu; h\lambda). \end{aligned}$$

Für $\mu = 0$ ist dies stets erfüllt, für $\mu \neq 0$ muss μ eine Nullstelle von p_Φ und p'_Φ sein, also eine mehrfache Nullstelle von p_Φ. Umgekehrt müssen Nullstellen von p_Φ und p'_Φ, also mehrfache Nullstellen von p_Φ, Lösungen der Differenzengleichung (5.35) gemäß des Ansatzes $\nu_i = i\mu^i$ liefern.

□

Definition 5.50. Sei (ρ, σ) ein lineares Mehrschrittverfahren.

(a) (ρ, σ) heißt *stabil* für $h\lambda \in \mathbb{C}$, wenn für jede (mehrfache) Nullstelle μ des Stabilitätspolynoms p_Φ

$$|\mu| < 1$$

und für jede *einfache* Nullstelle μ

$$|\mu| = 1$$

gilt.

(b) Das *Stabilitätsgebiet* ist definiert durch

$$S_\Phi := \{z \in \mathbb{C} \setminus \{0\} : (\rho, \sigma) \text{ ist stabil für } z\}. \tag{5.37}$$

(c) (ρ, σ) heißt *A–stabil* (oder *absolut stabil*), wenn

$$\{z \in \mathbb{C} \setminus \{0\} : \mathrm{Re}(z) \leq 0\} \subset S_\Phi$$

gilt.

In vielen Fällen ist A–stabil eine zu restriktive Forderung, daher wird eine weitere Definition gegeben.

Definition 5.51. Ein konvergentes lineares Mehrschrittverfahren (ρ, σ) heißt *A(α)–stabil* mit $0 < \alpha \leq \frac{\pi}{2}$, wenn

$$S_\alpha := \{z \in \mathbb{C} : |\pi - arg\ z| < \alpha\} \subset S_\Phi$$

gilt. (ρ, σ) heißt A(0)–stabil, wenn es A(α)–stabil ist für ein hinreichend kleines $\alpha > 0$.

Bemerkung 5.52. Einige Bemerkungen im Zusammenhang mit A(α)–Stabilität:

(a) A($\frac{\pi}{2}$)–stabil ist äquivalent zu A–stabil.

(b) Bei A(0)–stabilen Verfahren ist mindestens $S_0 = \mathbb{R}^-$ im Stabilitätsgebiet enthalten.

(c) Die A(α)–Stabilität fordert, dass für $w \in S_\alpha$ das Stabilitätspolynom

$$p_\Phi(z, w) = \rho(z) - w\sigma(z)$$

nur Nullstellen $\mu \in \mathbb{C}$ besitzt, für die $|\mu| \leq 1$ gilt.

(d) $\mathrm{p}_\Phi(\mu_w, w) = 0$ ist äquivalent zu $\frac{\rho(\mu_w)}{w} = \sigma(\mu_w)$. p_Φ ist am Einheitskreis beschränkt und daher gilt weiters

$$0 = \lim_{\mathrm{Re}\, w \to -\infty} \left| \frac{\rho(\mu_w)}{w} \right| = \lim_{\mathrm{Re}\, w \to -\infty} \sigma(\mu_w).$$

Für $\mathrm{Re}\, w \to -\infty$ entsprechen also die Nullstellen von p_Φ jenen von σ, welche nur im Nullpunkt liegen dürfen. Das zweite charakteristische Polynom σ muss demnach die Form $\sigma(z) = \beta_0 z^k$ haben. Dies leisten gerade die BDF–Verfahren, siehe Kapitel 5.2.3.

Analog zu Einschrittverfahren wird der lokale Diskretisierungsfehler als Abweichung von der analytischen Lösung x des gegebenen Anfangswertproblems definiert, unter Berücksichtigung des Mehrschrittverfahrens (5.31).

Definition 5.53. Der *lokale Diskretisierungsfehler* des Mehrschrittverfahrens (5.31) ist definiert als

$$\begin{aligned} \ell_i &= \frac{1}{h} \sum_{m=0}^{k} \left(\alpha_k x(t_{i-k+m}) - h\beta_m f\left(t_{i-k+m}, x(t_{i-k+m})\right) \right) = \\ &= \frac{1}{h} \sum_{m=0}^{k} \alpha_m x(t_{i-k+m}) - h\beta_m x'(t_{i-k+m}). \end{aligned} \tag{5.38}$$

Der lokale Diskretisierungsfehler erlaubt die Untersuchung der Konsistenz des Mehrschrittverfahrens. *Konsistenz der Ordnung p* heißt, dass $\ell_i = O(h^p)$ gilt. Um Bedingungen für die Konsistenz von Mehrschrittverfahren (ρ, σ) zu entwickeln, wird x und x' um t_{i-k} in einer Taylor–Reihe entwickelt

$$x(t_{i-k+m}) = x(t_{i-k}) + mhx'(t_{i-k}) + O(h^2), \qquad x'(t_{i-k+m}) = x'(t_{i-k}) + O(h).$$

Mit (5.38) und unter Verwendung der charakteristischen Polynome folgt

$$\begin{aligned} \ell_i &= \frac{1}{h} x(t_{i-k}) \underbrace{\sum_{m=0}^{k} \alpha_m}_{=0} + x'(t_{i-k}) \underbrace{\sum_{m=0}^{k} (m\alpha_m - \beta_m)}_{\overset{!}{=}0} + O(h) = \\ &= \frac{1}{h} x(t_{i-k})\rho(1) + x'(t_{i-k})(N'(1) - \sigma(1)) + O(h). \end{aligned}$$

Aus diesen Überlegungen folgt

Satz 5.54. Ein lineares Mehrschrittverfahren (ρ, σ) ist genau dann konsistent von der Ordnung $p = 1$, wenn

$$\rho(1) = 0 \qquad \text{und} \qquad \rho'(1) = \sigma(1)$$

gilt.

Definition 5.55. Ein lineares Mehrschrittverfahren erfüllt die *Wurzelbedingung*, wenn alle Wurzeln ξ_i des ersten charakteristischen Polynoms ρ die Bedingung $|\xi_i| \leq 1$ und alle mehrfachen Wurzeln die Bedingung $|\xi_i| < 1$ erfüllen.

Bemerkung 5.56. Für konsistente Mehrschrittverfahren ist $\xi = 1$ aufgrund der Bedingung $\rho(1) = 0$ immer eine Wurzel, die sogenannte *Hauptwurzel*.

Definition 5.57. Ein lineares Mehrschrittverfahren heißt *D–stabil*, auch *asymptotisch stabil* oder *nullstabil*, wenn das erste charakteristische Polynom ρ die Wurzelbedingung erfüllt.

Definition 5.58. Ein k–Schritt Mehrschrittverfahren heißt *konvergent*, wenn für ein Anfangswertproblem der Form (5.2) mit Lipschitz–stetigem f aus der Konvergenz der Startwerte

$$\lim_{h \to 0} \max_{0 \leq i \leq k} \|x_0 - \nu_i\| = 0$$

auch die Konvergenz für alle beschränkten Zeitintervalle $[t_0, t_0 + T]$

$$\lim_{h \to 0} \max_{t_0 \leq t_i \leq t_0 + T} \|x(t_i) - \nu_i\| = 0$$

folgt.

Satz 5.59. Ein lineares Mehrschrittverfahren ist genau dann konvergent, wenn es D–stabil und konsistent ist. Die Konvergenzordnung entspricht dann der Konsistenzordnung.

5.2.2 Adams Schema

Ein wichtiger Spezialfall in der Klasse der linearen Mehrschrittverfahren ist durch die Wahl der Koeffizienten zu

$$\alpha_k = 1, \quad \alpha_{k-1} = -1, \quad \text{und} \quad \alpha_m = 0 \quad \forall m \notin \{k, k-1\}$$

und der Diskretisierung der Ableitung x' durch den Differenzenquotienten analog (5.3) gegeben. Das lineare Mehrschrittverfahren hat unter diesen Voraussetzungen die Gestalt

$$\frac{1}{h}(\nu_i - \nu_{i-1}) = \beta_0 f_{i-k} + \ldots + \beta_{k-1} f_{i-1} + \beta_k f_i. \tag{5.39}$$

Zur Bestimmung der Koeffizienten β_j kann die Überlegung anhand der abstrakten exakten Lösung der Differentialgleichung angestellt werden. Betrachtet man

$$x(t_i) = x(t_{i-1}) + \int_{t_{i-1}}^{t_i} f(t, x(t)) \mathrm{d}t \tag{5.40}$$

und ersetzt den Integranden durch ein Polynom, welches an den Punkten $(t_{i-k+m}, x(t_{i-k+m}))$ interpoliert wird, dann kann das Integral als Linearkombination der vorhergehenden Werte $f(t_{i-k+m}, x(t_{i-k+m}))$ mit den Koeffizienten β_m angeschrieben werden.

Ohne Beweis sei an dieser Stelle erwähnt, dass die Ordnung der Konsistenz um 1 größer ist als die Ordnung des Interpolationspolynoms.

Beispiel 5.60. Im Folgenden werden zwei Beispiele von Adams–Schemata angegeben:

(a) Ein Interpolationspolynom vom Grad $k-1$ an den Stellen $t_{i-k}, t_{i-k+1}, \ldots, t_{i-1}$ führt auf das Schema

$$\nu_i = \nu_{i-1} + h(\beta_0 f_{i-k} + \ldots + \beta_{k-1} f_{i-1}). \tag{5.41}$$

Hier gilt $\beta_k = 0$ und das Schema heißt die *explizite k–Schritt Adams–Bashforth Formel* der Konsistenzordnung $p = k$.

(b) Ein Interpolationspolynom vom Grad k an den Stellen $t_{i-k}, t_{i-k+1}, \ldots, t_{i-1}$ führt auf das Schema

$$\nu_i = \nu_{i-1} + h(\beta_0 f_{i-k} + \ldots + \beta_k f_i). \tag{5.42}$$

Hier gilt $\beta_k \neq 0$ und das Schema heißt die *implizite k–Schritt Adams–Multon Formel* der Konsistenzordnung $p = k+1$.

Die Koeffizienten β_m können elementar bestimmt werden und finden sich in Tabelle 5.1.

	expl. Adams–Bashforth–Verf.					impl. Adams–Moulton–Verf.				
k	β_0	β_1	β_2	β_3	β_4	β_0	β_1	β_2	β_3	β_4
1	1	0				$\frac{1}{2}$	$\frac{1}{2}$			
2	$-\frac{1}{2}$	$\frac{3}{2}$	0			$-\frac{1}{12}$	$\frac{8}{12}$	$\frac{5}{12}$		
3	$\frac{5}{12}$	$-\frac{16}{12}$	$\frac{23}{12}$	0		$\frac{1}{24}$	$-\frac{5}{24}$	$\frac{19}{24}$	$\frac{9}{24}$	
4	$-\frac{9}{24}$	$\frac{37}{24}$	$-\frac{59}{24}$	$\frac{55}{24}$	0	$-\frac{19}{720}$	$\frac{106}{720}$	$-\frac{264}{720}$	$\frac{646}{720}$	$\frac{251}{720}$

Tabelle 5.1: Auflistung der Koeffizienten des Adams–Bashforth– und des Adams–Moulton–Verfahrens

Das charakteristische Polynom der Adams–Schemata lautet

$$\rho(z) = z^k - z^{k-1} = z^{k-1}(z-1).$$

Wegen $\xi_k = 1$ und $\xi_j = 0$ für $j = 0, \ldots, k-1$ sind alle Adams–Schemata D–stabil.

Bemerkung 5.61. Einige abschließende Bemerkungen:

(a) Das Adams–Moulton Schema ist für $k > 1$ nicht A–stabil.

(b) Das Einschritt–Adams–Multon Schema ist äquivalent zur impliziten Trapezregel, diese ist A–stabil.

5.2.3 Backward Differentiation Formula

Backward–Differentiation–Formula (BDF) Verfahren sind eine Klasse von impliziten linearen k–Schritt Methoden. Diese Methoden sind Verallgemeinerungen der impliziten Euler–Methode ($k = 1$) unter Verwendung des Differenzenquotienten k–ter Ordnung, welcher eine Approximation k–ter Ordnung der Ableitung $x'(t_i)$ ist. BDF–Verfahren sind geeignet für das Lösen von steifen Problemen.

Die Konstruktion von BDF–Verfahren basiert auf einer völlig anderen Idee. Unter Annahme eines Interpolationspolynoms wird dieses der Differentiation unterworfen und dann die Koeffizienten der Problemstellung angepasst. Diese Vorgehensweise unterscheidet sich zu den vorangegangenen Ideen, da diese nicht auf der Integration beruhen, wie beispielsweise bei Adams–Schemata. Diesen Sachverhalt zeigt nun die folgende formale Rechnung.

Es wird ein Interpolationspolynom q durch die Punkte $(t_{i-k}, \nu_{i-k}), \ldots, (t_i, \nu_i)$ konstruiert und gefordert, dass $q(t_i)$ die zugrundeliegende Differentialgleichung erfüllt, also

$$q'(t_i) = f(t_i, q(t_i)) = f(t_i, \nu_i)$$

gilt. In der Darstellung des Interpolationspolynoms nach Lagrange gilt

$$q(t) = \sum_{m=0}^{k} \nu_{i-k+m} \omega_{im}(t),$$

wobei die ω_{im} die Basisfunktionen darstellen, damit gilt

$$q'(t) = \sum_{m=0}^{k} \nu_{i-k+m} \omega'_{im}(t).$$

Unter der Annahme, dass die Schrittweite h konstant ist, folgt insgesamt

$$\frac{1}{h} \sum_{m=0}^{k} \alpha_k \nu_{i-k+m} = \sum_{m=0}^{k} \nu_{i-k+m}.$$

Definition 5.62. Ein *BDF–Verfahren* ist ein implizites lineares k–Schritt–Verfahren, welches durch

$$\frac{1}{h} \sum_{m=0}^{k} \alpha_m \nu_{i-k+m} = f(t_i, \nu_i) \tag{5.43}$$

gegeben ist.

Nach Konstruktion ist das Verfahren konsistent von der Ordnung $p = k$. Die Koeffizienten erhält man wegen

$$\frac{1}{h} \sum_{m=0}^{k} \alpha_m \nu_{i-k+m} = f(t_i, \nu_i) = x'(t_i)$$

durch Ableiten des Interpolationspolynoms.

Die Koeffizienten von Verfahren bis zur Ordnung $k = 6$ sind in Tabelle 5.2 aufgelistet.

Die Untersuchung von BDF–Verfahren auf D–Stabilität gestaltet sich schwierig. Ohne Beweis sei angegeben, dass ein *k–Schritt BDF–Verfahren D–stabil* ist für $k = 1, \ldots, 6$ und instabil für $k > 6$. Daher ist ein BDF–Verfahren der Ordnung k konvergent für $k \leq 6$.

k	α_0	α_1	α_2	α_3	α_4	α_5	α_6
1	-1	1					
2	$\frac{1}{2}$	-2	$\frac{3}{2}$				
3	$-\frac{1}{3}$	$\frac{3}{2}$	-3	$\frac{11}{6}$			
4	$-\frac{1}{4}$	$-\frac{4}{3}$	3	-4	$\frac{25}{12}$		
5	$-\frac{1}{5}$	$\frac{5}{4}$	$-\frac{10}{3}$	5	-5	$\frac{137}{60}$	
6	$\frac{1}{6}$	$-\frac{6}{5}$	$\frac{15}{4}$	$-\frac{20}{3}$	$\frac{15}{2}$	-6	$\frac{147}{60}$

Tabelle 5.2: Auflistung der Koeffizienten von BDF–Verfahren bis zur Ordnung $k = 6$

5.3 Bemerkungen zu den Algorithmen in MATLAB und Simulink

In diesem Abschnitt soll ein Bezug zwischen den theoretisch vorgestellten Lösungsverfahren und den sogenannten ode-Lösern in MATLAB und Simulink hergestellt werden.

Dabei werden nur jene Lösungsalgorithmen betrachtet, welche in Simulink zur Verfügung stehen:

(a) *Fixed–step*–solver:

- ode1 (Euler)
- ode2 (Heun)
- ode3 (Bogacki–Shampine)
- ode4 (Runge–Kutta)
- ode5 (Dormand–Prince)
- ode8 (Dormand–Prince)
- ode14x (extrapolation)
- discrete (no continuous states)

(b) *Variable–step*–solver:

- ode23 (Bogacki–Shampine)
- ode45 (Dormand–Prince)
- ode113 (Adams)
- ode15s (stiff/NDF[11])
- ode23s (stiff/Rosenbrock)
- ode23t (mod. stiff/Trapeziodal)
- ode23tb (stiff/TR-BDF2)
- discrete (no continuous states)

Die Verfahren von *ode1 (Euler)* und *ode2 (Heun)* sind aus Abschnitt 5.1.1 wohlbekannt, sowie das Verfahren nach Runge–Kutta aus Abschnitt 5.1.3. Die Löser ode3 bis ode5 sind s–stufige Runge–Kutta–Verfahren mit einem bestimmten Butcher–Tableau. Das Butcher–Tableau von *ode3 (Bogacki–Shampine)* und *ode4 (Runge–Kutta)* ist in Tabelle 5.3 zu finden, das von *ode5 (Dormand–Prince)* ist beispielsweise in [12] nachzulesen.

Für die Verfahren *ode23 (Bogacki–Shampine)* und *ode45 (Dormand–Prince)* von variabler Schrittweite gilt, dass diese eingebettete Runge–Kutta–Verfahren unterschiedlicher Ordnung sind, wie schon am Ende von Abschnitt 5.1.3 angesprochen wurde.

ode113 (Adams) arbeitet mit einem Adams–Verfahren und die Verfahren vom Typ *ode23* sind BDF–Verfahren bzw. arbeiten nach der Rosenbrock–Methode, welche in Abschnitt 6.2 noch

[11] Numerical–Differentiation–Formulas

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{3}{4} & 0 & \frac{3}{4} & \\ \hline & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} \end{array} \qquad \begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}. \end{array}$$

Tabelle 5.3: Butcher–Tableaus der MATLAB/Simulink ode–Löser ode3 und ode4

näher erläutert wird.

Der Löser *discrete (no continuous states)* ist nur für diskrete Systeme zuständig. Besteht das Modell also ausschließlich aus Abtastsystemen, so kann dieser Löser gewählt werden.

Für genauere Zusammenhänge zu den Modellen und detaillierteren Information zu den Iterationsalgorithmen in Simulink sei auf Kapitel 6.2 verwiesen.

Kapitel 6

Die Simulationsumgebung Simulink

Dieses Kapitel widmet sich der Zusammenführung der mathematischen Grundlagen aus den ersten Kapiteln und der Simulationsumgebung Simulink, sowie der Darstellung der Arbeitsweise dieser Simulationsumgebung. Es werden Unterschiede zur mathematischen Theorie aufgezeigt und die speziellen Notationen und Darstellungsformen der Simulationsumgebung vorgestellt. Die grundlegende Definition der Simulationsumgebung Simulink sei dabei wie folgt.

Definition 6.1 (Simulink, Definition nach [11])**.** *Simulink*® is an environment for multidomain simulation and Model–Based Design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that let you design, simulate, implement, and test a variety of time–varying systems, including communications, controls, signal processing, video processing, and image processing.

6.1 Grundlagen, Begriffe und Definitionen

Ein Modell in Simulink basiert auf dem Prinzip der Eingangs–Ausgangs–Relation eines Übertragungssystems und ist als Signalflussgraph konzeptioniert. Die Funktionsbausteine von Simulink haben einen Eingang und einen Ausgang, wobei dieses Konzept auch erweitert werden kann, beispielsweise durch mehrere Eingänge, oder der Annahme eines vektorwertigen Eingangssignals.
Um das Prinzip mehrerer Eingänge mit der Definition 3.7 eines Signalflussgraphen zu kombinieren, muss dieses adaptiert werden. Diese Vorgehensweise soll anhand des Additionsblocks in Simulink erläutert werden.

Beispiel 6.2 (Additionsblock in Simulink)**.** Aus der Definition eines Signalflussgraphen in Definition 3.1 und entsprechend visualisiert in Abbildung 3.1 ist das Enden mehrerer Kanten an einem Knoten als Summation der entsprechenden (verallgemeinerten) Gewichtsfunktionen zu interpretieren.
In Simulink hingegen wird die Addition als eigener Übertragungsblock dargestellt. Abbildung 6.1 zeigt die Addition von zwei Signalen und die Subtraktion von einem dritten. Es gibt in Simulink zwei Möglichkeiten der Addition, wie in Abbildung 6.1 dargestellt.

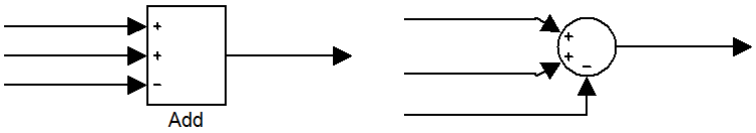

Abbildung 6.1: Darstellung zweier Möglichkeiten der Addition in Simulink

Wird berücksichtigt, dass die Addition nicht in der (verallgemeinerten) Gewichtsfunktion anhand der zulaufenden Kanten an einen Knoten, sondern durch einen eigenen, aus Sicht der Theorie von Signalflussgraphen redundanten, Block dargestellt wird, bleibt die übrige Theorie der Signalflussgraphen unverändert.

Für einige andere Operationen in Simulink gilt Ähnliches. Ohne auf Details einzugehen kann insgesamt gesagt werden, dass Simulink ein Simulator für unterschiedliche Bereiche ist, der Modelle durch Signalflussgraphen beschreibt.

In der Simulationsumgebung Simulink muss noch die Zuführung von Eingangs– und Ausgangssignalen berücksichtigt werden. Der Simulator kann nur ein konkretes Szenario rechnen, d.h. für vorgegebene Einganssignale und definierte Anfangszustände. Dazu müssen sowohl Quellen zur Bereitstellung der Eingangssignale, als auch Senken zur Verarbeitung der Ausgangssignale, für ein Simulationsmodell angegeben werden. Dazu sei Beispiel 6.2 an dieser Stelle fortgesetzt.

Beispiel 6.3. In Abbildung 6.2 wurden die Additionsblöcke aus Beispiel 6.2 mit drei verschiedenden Eingangssignalen versehen und die beiden Ausgangssignale mit Hilfe eines Scopes dargestellt.

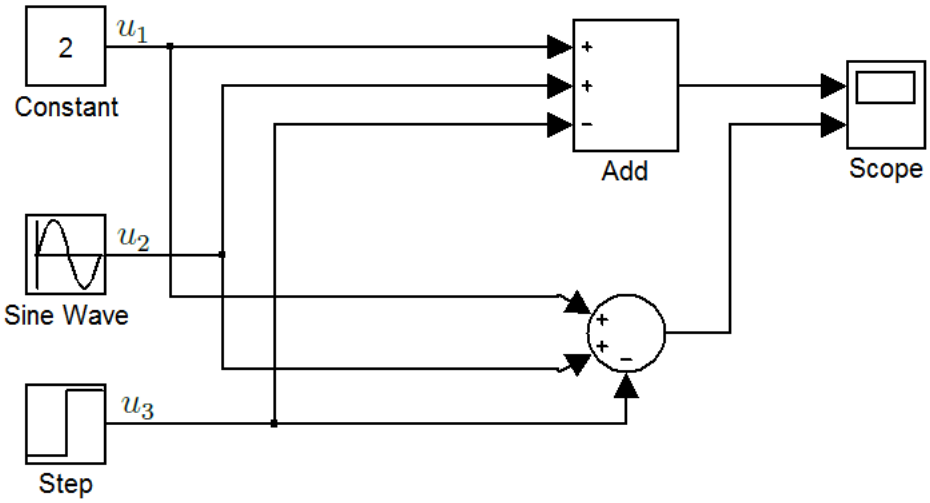

Abbildung 6.2: Darstellung der Addition und Subtraktion dreier Eingangssignale in Simulink

Als Eingangssignale wurden dabei

$$u_1(t) = 2, \qquad u_2(t) = \sin t \quad \text{und} \quad u_3(t) = \begin{cases} 0 & \text{für } t < \pi \\ 1 & \text{für } t \geq \pi \end{cases}$$

gewählt.

Definition 6.4 (Simulink–Modell)**.** Der Begriff *Modell* in Simulink, oder auch ein *Simulink–Modell*, definiert ein Übertragungssystem, welches einen festgelegten Ein– und Ausgang aufweist. Auch der Fall mehrerer Ein– und Ausgänge kann dabei auftreten sowie auch ein vektorwertiges Eingangs– oder Ausgangssignal ist möglich.

Beispiel 6.3 ist so ein System. Es hat drei Eingänge und zwei Ausgänge, vorgegebene Eingangssignale und die Ausgangssignale werden durch ein sogenanntes *Scope* graphisch dargestellt.

Simulink–Modelle können nur mit initialisierten Werten und vorgegebenen Eingangssignalen simuliert werden. Dazu wird das folgende Beispiel betrachtet.

Beispiel 6.5 (LTI–System in Simulink)**.** Es wird noch einmal Beispiel 4.26 herangezogen. Dieses LTI–System kann in Simulink simuliert werden, wobei die Werte $a_1, a_2, a_3 \in \mathbb{R}$ mit $a_2 \neq 0$ vorgegeben sein müssen. Diese können beispielsweise im MATLAB–Arbeitsbereich hinterlegt, so auch durch MATLAB manipuliert und damit der Einfluss dieser Werte auf das Systemverhalten untersucht werden.

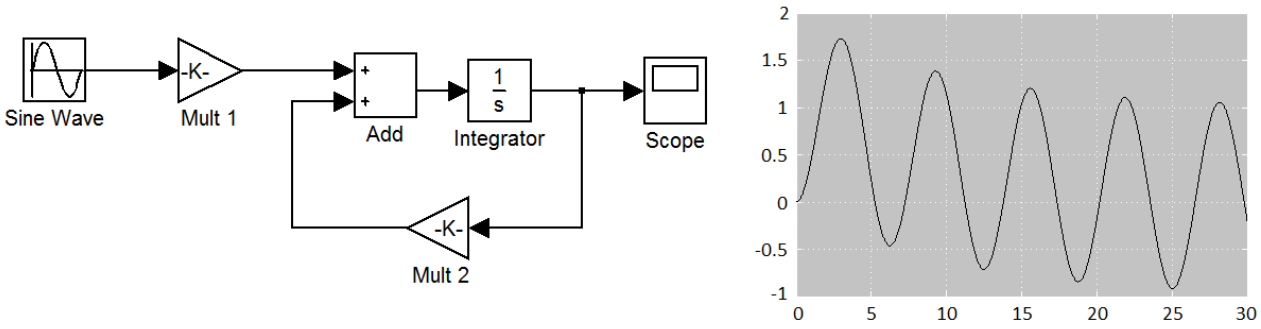

Abbildung 6.3: Das LTI–System aus Beispiel 4.26 als Simulink–Modell implementiert (links) und das Ausgangssignal y dargestellt (rechts)

Der Übertragungsblock in Abbildung 6.3 *Mult 1* führt die Multiplikation mit $\frac{a_3}{a_2}$, der Übertragungsblock *Mult 2* die Multiplikation mit $-\frac{a_1}{a_2}$ aus. Hier gilt $a_1 = \frac{1}{10}$, $a_2 = a_3 = 1$ und das Eingangssignal ist gegeben durch

$$u(t) = \sin t.$$

Zum Nachweis, dass die graphisch dargestellte Lösung richtig ist, kommt man durch Vergleich mit der allgemeinen Lösung der hier zugrundeliegenden gewöhnlichen Differentialgleichung. Die homogene Lösung geht mit dem Ansatz $y_h(t) = c\,\mathrm{e}^{\lambda t}$ aus der Gleichung

$$a_1 + a_2\lambda = 0$$

hervor, eine partikuläre Lösung mit dem Ansatz $y_p(t) = c_1 \sin t + c_2 \cos t$ aus den Gleichungen

$$a_1c_1 - a_2c_2 = a_3, \qquad a_1c_2 + a_2c_1 = 0.$$

Mit dem Anfangswert $y(0) = 0$ erhält man die allgemeine Lösung

$$y(t) = \frac{100}{101}\,\mathrm{e}^{-\frac{t}{10}} + \frac{10}{101}\,\sin t - \frac{100}{101}\,\cos t.$$

In Simulink stehen auch Übertragungsblöcke zur Verfügung, welche es erlauben, eine Übertragungsfunktion durch eine rationale Funktion anzugeben. Dies wurde in Beispiel 4.26 gezeigt und führt zum selben Simulationsergebnis. Weiters ist es möglich, Übertragungsblöcke zu bilden, welche einen Teil der Funktionalität beinhalten, um die Übersicht über ein Simulink–Modell zu bewahren. Die Übertragungsfunktion und der Übertragungsblock, welcher die Funktionalität von Abbildung 6.3 inne hat, ist in Abbildung 6.4 zu sehen.

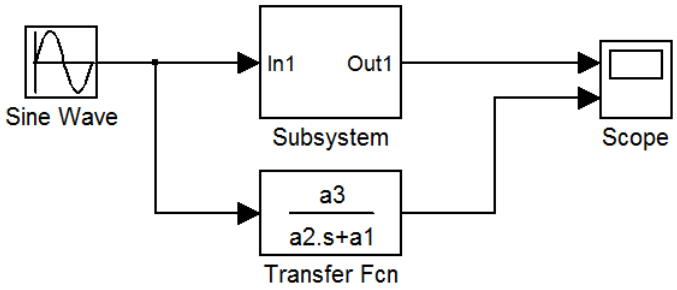

Abbildung 6.4: Das LTI–System aus Abbildung 6.3, einmal durch eine Übertragungsfunktion und einmal als Simulink–Teilmodell beschrieben

Das vorangegangene Beispiel motiviert einige Definitionen. Einerseits konstante Werte in einer Simulation, welche das Verhalten des Ausgangssignals beeinflussen, andererseits Simulink–Teilmodelle, welche Funktionalitäten beinhalten können, um Simulink–Modelle übersichtlich zu gestalten.

Definition 6.6 (Parameter)**.** Eine Variable heißt ein *Parameter*, wenn diese fest aber beliebig gewählt ist. Der Unterschied zu einer Konstanten, welche immer denselben Wert annimmt, ist, dass in einer Simulationsrechnung der Parameter die Lösung qualitativ beeinflusst.

Des Weiteren wurde in Beispiel 6.5 die Möglichkeit vorgestellt, Teile eines Modells durch einen Übertragungsblock zu ersetzen. Dies führt zur

Definition 6.7 (Teilmodelle, Subsysteme)**.** Es heißt ein Teil eines Simulink–Modells ein *Teilmodell* oder *Subsystem*, wenn dieses für sich selbst betrachtet wieder ein Simulink–Modell darstellt.

6.2 Simulationsparameter kontinuierlicher Systeme

In diesem Abschnitt soll gezeigt werden, welche Parameter in Simulink manipulierbar sind und somit bei der Simulation beeinflusst werden können.

Den wesentlichsten Punkt bei der Konfiguration der Simulationsparameter stellt der sogenannte *solver* dar. Dahinter verbirgt sich das numerische Lösungsverfahren für die Differentialgleichungen, welche dem Simulink–Modell zugrunde liegen.
Unter anderem stehen zur numerischen Lösung von gewöhnlichen Differentialgleichungen Lösungsverfahren, wie in Kapitel 5 dargestellt, zur Verfügung.

Simulink bietet dabei einen Teil dieser Verfahren für die Simulation an, wobei folgende Parameter beeinflusst werden können:

- Simulationsdauer
- „solver“–Algorithmus: Numerisches ODE–Lösungsverfahren
- Fixierte oder variable Schrittweite
- Minimale und maximale Schrittweite
- Absolute und relative Toleranz
- „zero crossing detection“

Die der numerischen Lösung von gewöhnlichen Differentialgleichungen zur Verfügung stehenden Algorithmen und die dazugehörigen Anwendungsszenarien sind in Tabelle 6.1 aufgelistet.

Name	Problembeschreibung für das Szenario	Lösungsverfahren
ode45	Differentialgleichungen ohne spezielle Ansprüche	Runge–Kutta
ode23	Differentialgleichungen ohne spezielle Ansprüche	Runge–Kutta
ode113	Differentialgleichungen ohne spezielle Ansprüche	Adams
ode15s	Steife Differentialgleichungen und DAEs	BDF
ode23s	Steife Differentialgleichungen	Rosenbrock
ode23t	Mäßig steife Differentialgleichungen und DAEs	Trapez–Regel
ode23tb	Steife Differentialgleichungen	TR–BDF2

Tabelle 6.1: Auflistung der numerischen Lösungsverfahren für ODEs in Simulink

Tabelle 6.1 beinhaltet unter anderen das Rosenbrock[1]–Verfahren, welches eine Spezialisierung impliziter Runge–Kutta–Verfahren ist. Dazu wird in jedem Schritt eine Newton–Iteration durchgeführt, welche mit Hilfe der Jacobi–Matrix eine Näherungslösung berechnet. Dies erfordert jedoch die Berechnung dieser Matrix in jedem Schritt. Eine Abhilfe stellt das vereinfachte Newton–Verfahren dar, welches die Jacobi–Matrix nur alle n Schritte berechnet. Dies senkt den Berechnungsaufwand wegen der geringeren Anzahl an Auswertungen der Jacobi–Matrix, man verliert jedoch durch diese Vorgehensweise an Konvergenzgeschwindigkeit. Wählt man für die Jacobi–Matrix $J = f_x(\nu_j)$, wird dieses Verfahren als die *Rosenbrock–Methode* bezeichnet.
Das Verfahren TR–BDF2 ist eine Kombination aus der Trapez–Regel im ersten Schritt und einem BDF–Verfahren der Ordnung 2 im zweiten Schritt.
Des Weiteren wird in der Problembeschreibung bei zwei Lösungsverfahren der Begriff DAEs verwendet, welcher für differential–algebraische Gleichung steht. Diese vereint eine gewöhnliche Differentialgleichung mit einer algebraischen Gleichung. Es gilt die folgende

Definition 6.8. Sei $F\colon\ \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R},\ (t, x, y) \mapsto F(t, x, y)$ und

$$F(t, x, x') = 0, \qquad t \geq 0 \quad \text{und} \quad x(0) = x_0 \tag{6.1}$$

ein implizites Anfangswertproblem.

[1]Howard Harry Rosenbrock (1920 – 2010), englischer Elektrotechniker

(a) Ist F_y entlang einer Lösung $x(t)$ bezüglich $x'(t)$ nicht invertierbar, so heißt (6.1) eine *differential–algebraische Gleichung*.

(b) Die kleinste Zahl $k \in \mathbb{N}$, für welche

$$\frac{\mathrm{d}^i}{\mathrm{d}t^i} F(t, x, x') = 0, \qquad i = 0, \ldots, k,$$

gilt, heißt der *Index* der differential–algebraischen Gleichung.

Bemerkung 6.9. Ist die Matrix

$$\frac{\partial}{\partial y} F(t, x, x')$$

regulär, so kann nach dem Satz über implizite Funktionen die Gleichung $F(t, x, x') = 0$ nach x' aufgelöst werden, d.h. die differential–algebraische Gleichung ist ein gewöhnliches Differentialgleichungssystem in impliziter Form.

Eine weitere Konfigurationsmöglichkeit in Simulink ist die *Schrittweite*. *Variable–step Löser* arbeiten mit einer variablen Schrittweite. Ausgehend von einer initialen Schrittweite wird versucht, den Lösungsalgorithmus mit der maximalen Schrittweite durchzuführen. Wenn die Ableitungen der Zustandsgrößen zu groß sind, kann die initiale Schrittweite manuell unterschritten werden. Die zu bestimmenden Größen verhalten sich hierbei wie folgt zueinander: Es wird eine maximale Schrittweite $h_{\max}$, eine minimale Schrittweite $h_{\min}$ und eine initiale Schrittweite h_{init} vorgegeben bzw. von Simulink automatisch gewählt. Soll die maximale Schrittweite von Simulink automatisch bestimmt werden, wird sie mit Hilfe der Simulationszeit berechnet. Ist diese durch das Intervall $[t_1, t_2]$ gegeben, so gilt

$$h_{\max} = \frac{t_2 - t_1}{50}.$$

Des Weiteren werden eine *relative Toleranz* $\mathrm{TOL}_{\mathrm{rel}}$ und eine *absolute Toleranz* $\mathrm{TOL}_{\mathrm{abs}}$ vorgegeben, welche für die Fehlerüberwachung notwendig sind. Die Schrittweite wird durch Überwachung des lokalen Diskretisierungsfehlers ℓ_i in jedem Iterationsschritt geregelt. Sie wird so gewählt, dass

$$\ell_i \leq \max\{\mathrm{TOL}_{\mathrm{rel}} \, |\nu_i|, \mathrm{TOL}_{\mathrm{abs}}\}$$

erfüllt ist. Die Bedingung wird über ein Maximum formuliert, da $\mathrm{TOL}_{\mathrm{rel}} \cdot |\nu_i|$ bei sehr kleinen Werten von $|\nu_i|$ so klein werden könnte, dass die Zustandsgrößen sich nicht mehr ändern dürfen. Wird die absolute Toleranz automatisch von Simulink gewählt, wird diese zum Start der Simulation auf $\mathrm{TOL}_{\mathrm{abs}} = 10^{-6}$ und anschließend auf $\mathrm{TOL}_{\mathrm{abs}} = \mathrm{TOL}_{\mathrm{rel}} \cdot \max(|\nu_i|)$ gesetzt. In Simulink gilt $\mathrm{TOL}_{\mathrm{abs}}$ modellweit. Um jedoch gewissen Genauigkeitsanforderungen gerecht zu werden, bieten einige Blöcke, wie beispielsweise der Integrationsblock, der Übertragungsfunktionsblock u.a., die Möglichkeit, eine absolute Toleranz im jeweiligen Block individuell zu setzen.

Eine weitere wichtige Konfiguration ist die sogenannte *zero–crossing detection*. Darunter werden Unstetigkeiten im Verlauf der Zustandsgröße und gewöhnliche Nulldurchgänge verstanden. Es wird zum Ende jedes Iterationsschritts die Abfrage gestellt, ob ein zero–crossing aufgetreten ist. Wenn dem so ist, dann wird durch Interpolation der Zeitpunkt möglichst genau bestimmt. Das Erkennen eines zero–crossings hängt dabei aber stark von der Schrittweite und damit von den Toleranzen ab.

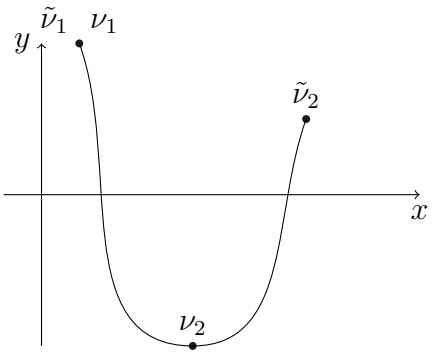

Abbildung 6.5: Zur Erkennung eines zero–crossings in Abhängigkeit der Schrittweite

In Abbildung 6.5 ist dargestellt, wie die Schrittweite Einfluss auf das Erkennen des zero–crossings nimmt. In dem dargestellten Fall wird für die Schrittweite der Iteration ν_j der Nulldurchgang erkannt, im Fall der Iteration $\tilde{\nu}_j$ ist die Schrittweite zu groß und es werden zwei Nulldurchgänge aufgrund der Vorzeichenlage nicht erkannt. Die Aktivierung der zero–crossing Kontrolle ist nur im Fall von variabler Schrittweite möglich, sie kann jedoch auch nur, ähnlich der Vorgabe der absoluten Toleranz, in einzelnen Blöcken aktiviert werden.

In Simulink können auch Löser mit fixer Schrittweite eingestellt werden. Bei diesen entfällt jedoch die Möglichkeit der Fehlerüberwachung und der Erkennung von Unstetigkeiten. Dafür ist die Anzahl der Iterationsschritte bekannt und die Rechenzeit für ein Modell kann somit abgeschätzt werden.

6.3 Lineare und nichtlineare kontinuierliche Systeme

Dieser Abschnitt widmet sich der Modellierung linearer und nichtlinearer dynamischer Systeme in Simulink. Lineare Systeme werden im Wesentlichen so behandelt, wie sie in Kapitel 4 dargestellt wurden. Nichtlineare Systeme werden ebenfalls durch eine Eingangs–Ausgangs–Relation gemäß Gleichung (4.1) dargestellt, welche aber nicht linear ist.

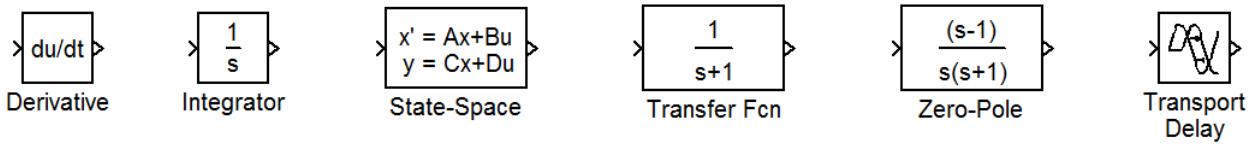

Abbildung 6.6: Eine Auswahl wichtiger linearer Übertragungssysteme

Typische lineare Übertragungssysteme sind in Abbildung 6.6 dargestellt. Aus diesen Blöcken können zeitkontinuierliche lineare Systeme beliebiger Komplexität zusammengesetzt werden.

Der Übertragungsblock der *Ableitung* berechnet die numerische Ableitung

$$\frac{\Delta u}{\Delta t},$$

wobei Δ die Änderung der entsprechenden Größe bezogen auf den vorangegangenen Iterationsschritt bezeichnet. Die Linearisierung des Übertragungsblocks wird durch die Übertragungsfunktion

$$D(s) = \frac{s}{Ns + 1}$$

angegeben, wobei N die sogenannte *Linearisierungszeitkonstante* darstellt, und ein Null–Anfangswert vorausgesetzt wird.

Der *Integrator* hat die Möglichkeit, dass ein Anfangswert verschieden von Null, sowie auch durch einen weiteren externen Eingang am Block vorsehen werden kann. Durch ein weiteres externes Signal, welches durch unterschiedliche Optionen auf fallende oder steigende Flanken eingestellt werden kann, kann der Integrator zurückgesetzt werden. Es besteht die Möglichkeit, das Ausgangssignal zu limitieren, also mit einem Sättigungsbereich zu versehen. Es ist jedoch Vorsicht geboten. Wird beispielsweise das Ausgangssignal ohne zeitliche Verzögerung auf den Rücksetz– oder Anfangswerteingang zurückgeführt, entsteht eine sogenannte *algebraische Schleife*. Diese kann in Simulink nicht berechnet und muss im Modell aufgelöst werden, beispielsweise durch Rückführung der Zustandsgröße anstatt des Ausgangssignals.

Weiters ist der *Zustandsraumblock* verfügbar. Dieser wird durch die Gleichungen

$$\begin{aligned} x'(t) &= A\ x(t) + B\ u(t), \\ y(t) &= C\ x(t) + D\ u(t) \end{aligned} \tag{6.2}$$

mit $x \in \mathbb{R}^n, u \in \mathbb{R}^m, y \in \mathbb{R}^p, A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{p\times n}$ und $D \in \mathbb{R}^{p\times m}$ beschrieben. Der Anfangswert x_0 kann im Block festgelegt werden und für $m = p = 1$ erhält man Gleichung (4.2). In diesem Fall sind vektorwertige Eingangs– und Ausgangssignale angesetzt, die Angabe der Matrizen diktiert dabei die Dimensionen n, m, p. Dieser Fall wird als MIMO[2]–System bezeichnet, Gleichung (4.2) stellt ein SISO[3]–System dar.

Einer der wichtigsten Übertragungsblöcke ist jener, der es erlaubt, direkt *Übertragungsfunktionen* anzugeben. Es sind die Koeffizienten des Zähler– und Nennerpolynoms zu wählen. Die (rationale) Übertragungsfunktion und das damit verbundene Systemverhalten ist vorgegeben. Es können auch Matrizen als Koeffizienten im Zähler vorgegeben werden, d.h. es können vektorwertige Ausgangssignale damit eingestellt, die Eingangsgröße kann aber nur skalar modelliert werden.
Ein verwandtes Übertragungssystem ist das *Zero–Pole System*. Damit kann die Übertragungsfunktion in faktorisierter Darstellung, gemäß Gleichung (4.4), angegeben werden, wobei die Polynome auch Vektoren oder Matrizen für SISO– und MIMO–Systeme sein können. Die Anzahl der Pole muss größer gleich der Anzahl der Nullstellen sein.

Einen technisch sehr wichtigen Übertragungsblock bildet der *Zeitverzögerungsblock*. Dieser gibt das Eingangssignal zeitverschoben um die Verzögerungszeit, oder auch Totzeit T_t, am Ausgang aus. Die Übertragungsfunktion lautet

$$G(s) = \mathrm{e}^{sT_t},$$

[2]Multiple Input Multiple Output
[3]Single Input Single Output

wobei auch eine Padé[4]–Approximation der Ordnung m der Form

$$\frac{p_0 + p_1(sT_t) + p_2(sT_t)^2 + \ldots + p_m(sT_t)^m}{q_0 + q_1(sT_t) + q_2(sT_t)^2 + \ldots + q_m(sT_t)^m}$$

einstellbar ist. Dies ist bei der Linearisierung des Übertragungsglieds von Bedeutung.

Nichtlineare Systeme können ebenfalls durch eine Eingangs–Ausgangs–Relation beschrieben werden. In Anlehnung an Gleichung (6.2) kann nun ein nichtlineares System mit zwei Abbildungen, $f\colon\ \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ und $g\colon\ \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ für $m, n, p \in \mathbb{N}$, in der Form

$$\begin{aligned} x'(t) &= f\left(t, x(t), u(t)\right) \\ y(t) &= g\left(t, x(t), u(t)\right) \end{aligned} \tag{6.3}$$

beschrieben werden. In dieser Beschreibung sind vektorwertige Ein– und Ausgänge möglich und bei Vorliegen eines linearen zeitinvarianten Systems erhält man die Darstellung aus Gleichung (6.2).

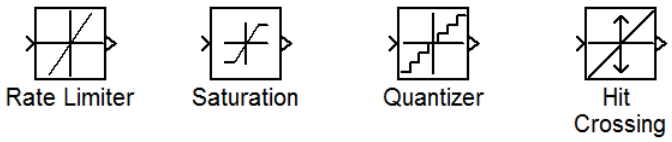

Abbildung 6.7: Eine Auswahl wichtiger nichtlinearer Übertragungssysteme

Abbildung 6.7 zeigt eine Auswahl wichtiger nichtlinearer Übertragungssysteme. Die Verwendung eines dieser Übertragungssysteme hat zur Folge, dass das gesamte System nichtlinear ist.

Der Block *Rate Limiter* begrenzt die erste Ableitung des Eingangssignals durch eine Konstante. Die Begrenzung wird vorzeichensensitiv unterschieden. Sei $s_r > 0$ und $s_f < 0$ so gilt für das Eingangssignal u die Ungleichung

$$s_f \leq \frac{\mathrm{d}}{\mathrm{d}t} u(t) \leq s_r.$$

Es heißt s_r die *rising slew rate* und s_f die *falling slew rate*.

Der *Saturation* Block hat, gleich wie der Rate Limiter, die Aufgabe, einen Wert zu begrenzen. In diesem Fall ist es der Wert des Maximums des Eingangssignals. Für $M, m \in \mathbb{R}$ und für ein Eingangssignal u gilt für das Ausgangssignal

$$y(t) = \begin{cases} M & \text{für } u(t) \geq M \\ u(t) & \text{für } m < u(t) < M \\ m & \text{für } u(t) \leq m \end{cases} \cdot$$

Ein weiterer nichtlinearer Übertragungsblock ist der *Quantiser*. Dieser hat eine vorgegebene Quantisierungsbreite von $q \in \mathbb{R}^+$, mit der das Ausgangssignal eine Stufenfunktion darstellt. Das Ausgangssignal wird gemäß

$$y(\tau) = q \cdot \mathrm{rd}\left(\frac{u(\tau)}{q}\right)$$

[4]Henri Eugène Padé (1863 – 1953), französischer Mathematiker

berechnet, wobei rd: $\mathbb{R} \to \mathbb{Z}$ die mathematische Rundungsfunktion beschreibt. Resultat für $\mathrm{rd}(x)$ ist also jene Zahl $z \in \mathbb{Z}$, für welche $|z - x|$ minimal wird.

Der letzte der ausgewählten nichtlinearen Systeme ist der Block *Hit Crossing*. Dieser Block ermittelt den Zeitpunkt, für den das Eingangssignal u einen Wert c erreicht, d.h. es wird jenes τ ermittelt, für welches $u(\tau) = c$ gilt. Eine zusätzliche Konfigurationsmöglichkeit ist die Bestimmung der Richtung, d.h. ob für stetiges u und ein $\varepsilon > 0$ die Ungleichung

$$u(\tau - \varepsilon) < u(\tau) < u(\tau + \varepsilon)$$

oder

$$u(\tau - \varepsilon) > u(\tau) > u(\tau + \varepsilon)$$

gilt. Es können beide oder nur jeweils eine von beiden Ungleichungen ausgewählt werden. Typischerweise ist das Ausgangssignal eine Bboolesche Variable.

Abschließend in diesem Abschnitt soll noch ein Blick auf eigens definierte Funktionsblöcke geworfen werden. Diese erlauben es, Modelle zu konstruieren, welche entweder eine mathematische Funktion repräsentieren, oder eine MATLAB–Funktion zur Ausführung bringen. Diese beiden Funktionsblöcke entscheiden die Eigenschaft des Gesamtsystems, linear oder nichtlinear zu sein. Der mathematische Funktionsblock bildet für ein Eingangssignal u das Ausgangssignal

$$y(t) = f\left(u(t)\right)$$

mit einer im Block durch einen Term zu definierenden Funktion $f\colon \mathbb{R}^m \to \mathbb{R}^p$. Der Block der MATLAB–Funktion erhält ebenfalls den Vektor $u \in \mathbb{R}^m$. Dieser kann neben mathematischen Funktionen auch MATLAB–Funktionen auf diesen anwenden und dann als Ausgangssignal ausgeben. Die letzte Möglichkeit bildet die sogenannten S–Funktion. Diese erlaubt es, ein ganzes MATLAB–Skript in einen Block einzubinden.

6.4 Algebraische Schleifen

Wie schon beim Integratorblock hinsichtlich des Anfangswerteingangs dargestellt, tritt in Simulink eine sogenannte algebraische Schleife auf, wenn Blöcke einen direkten Durchgriff haben. Bei diesen Blöcken ist zum gleichen (diskreten) Zeitpunkt das Ausgangssignal vom Eingangssignal abhängig. Blöcke, welche diese Eigenschaft aufweisen, sind jene der Addition, Multiplikation mit einer Konstanten, Multiplikation, Zustandsraum mit einer Durchgriffmatrix $D \neq 0$, Integrator bezüglich des Anfangswerteingangs, Übertragungsfunktion und Zero–Pole mit gleichem Grad des Zähler– und Nennerpolynoms.
Wird nun der Ausgang eines solchen Blocks direkt, oder über andere Blöcke mit direktem Durchgriff, auf seinen Eingang zurückgeführt, so entsteht die *algebraische Schleife*. Der Wert des Eingangssignals hängt dann zum gemeinsamen Zeitpunkt vom Ausgangssignal ab, welches wiederum vom Eingangssignal abhängt.

Beispiel 6.10 (Algebraische Schleifen)**.** Es werden zwei Beispiele zu algebraischen Schleifen betrachtet:

(a) Betrachtet wird ein Integrator mit auf den Anfangswerteingang rückgeführtem Ausgang, wie in Abbildung 6.8 dargestellt.

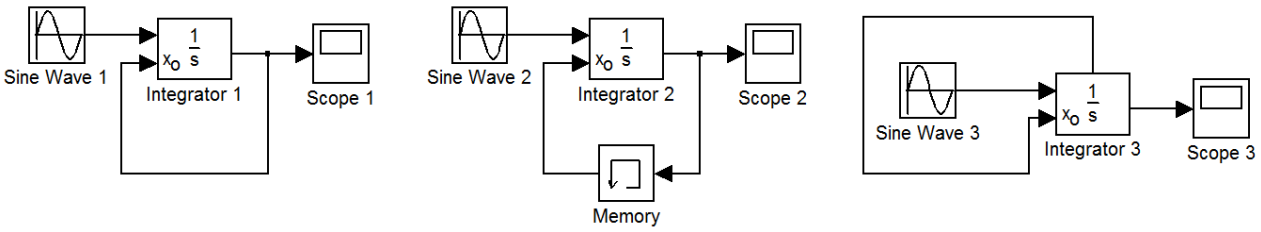

Abbildung 6.8: Algebraische Schleife bei Rückführung an einem Integrationsblock und ihre beiden Abhilfen

Bei Modell 1 wird eine Rückmeldung im MATLAB–Kommandofenster ausgegeben, Modell 2 umgeht mit Hilfe eines Speicherblocks, welcher eine Verzögerung um einen Iterationschritt zur Folge hat, das Problem der Schleife. In Modell 3 wird statt des Ausgangs die Zustandsgröße verwendet, die an sich ident ist, jedoch intern zu einem anderen Zeitpunkt bestimmt wird und somit keine algebraische Schleife auftritt.

(b) Ein anderes, noch einfacheres Beispiel, an dem die Konsequenzen einer algebraischen Schleife gut zu untersuchen sind, ist die Gleichung

$$x = a\,\mathrm{e}^{x} + b,$$

für $a, b > 0$. Das dazugehörige Simulink–Modell dieser Gleichung ist in Abbildung 6.9 zu sehen.

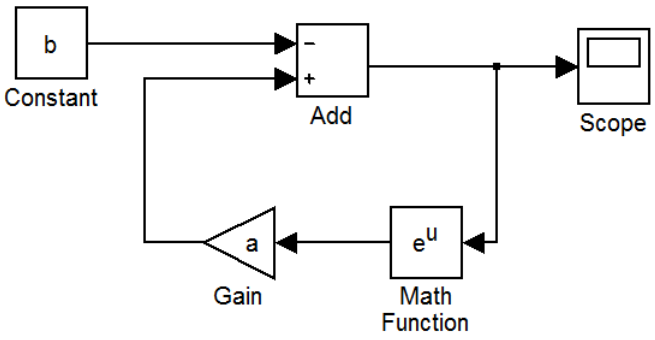

Abbildung 6.9: Algebraische Schleife beim Lösen einer Gleichung in Simulink

Für $a = b = 1$ ist die Lösung einfach zu $x = 0$ zu berechnen, für beispielsweise $a = 3$ und $b = 2$ bricht Simulink die Berechnung wegen der algebraischen Schleife ab.

Simulink versucht die algebraischen Schleifen mit 200 Iterationen zu lösen. Wird dies nicht erreicht, kommt es zum Abbruch der Iteration und zur Ausgabe einer Fehlermeldung. Das Beispiel 6.10 (b) soll die Empfindlichkeit von Parametern, welche hier für die Konvergenz innerhalb der 200 Iterationen verantwortlich ist, unterstreichen. Aus diesem Grund ist es anzustreben, keine algebraischen Schleifen in Simulink–Modellen auftreten zu lassen, da die Konvergenz der Iteration nicht gewährleistet werden kann.
Eine Möglichkeit, algebraische Schleifen im Modell auf eine andere Art zu lösen, bietet der Block der *algebraischen Konstante*. Dieser zwingt den Eingang, gegeben als Funktion $f(z)$, zu Null und gibt den Wert z am Ausgang aus. Im Modell muss die Ausgangsgröße die Eingangsgröße durch eine beliebig beschaffene Rückführung beeinflussen.

6.5 Abtastsysteme

Die letzten beiden Unterkapitel in Kapitel 4 haben sich der Beschreibung von Abtastsystemen gewidmet, welche in Simulink zur Verfügung stehen.

Ein zentraler Simulationsparameter von Abtastsystemen ist die *Abtastzeit* T. Diese muss in Simulink in jedem Block eines Abtastsystems angegeben werden. All diese Blöcke haben intern einen Abtaster nach dem Eingang und ein Halteglied vor dem Ausgang, um eine Schnittstelle zu der a priori kontinuierlichen Simulink–Umgebung bereitzustellen, siehe dazu Abbildung 6.10.

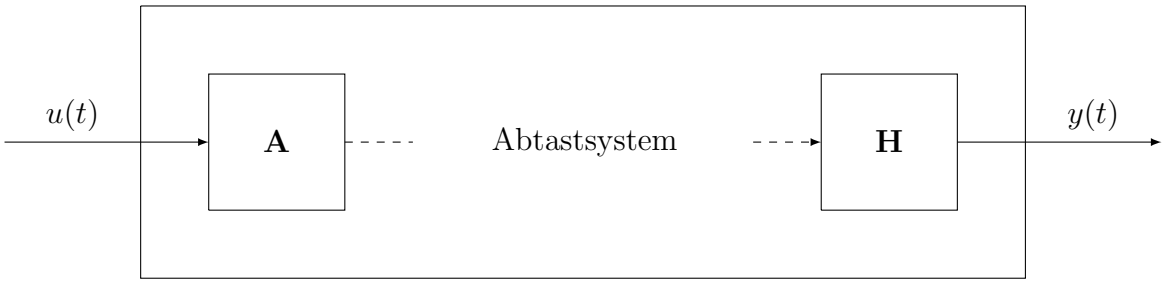

Abbildung 6.10: Blockdiagramm eines Abtastsystems in Simulink

Bei der Abtastzeit T sind zwei Besonderheiten zu beachten:

- Offset: Wird die Abtastzeit als Vektor der Form $(T, t_{\text{offset}})^T$ angegeben, so wird der erste Eintrag als Abtastzeit und der zweite Eintrag als Offset interpretiert, d.h. die Abtastung erfolgt zu den Zeitpunkten

$$t_k = k \cdot T + t_{\text{offset}}, \qquad k \in \mathbb{N}_0 \,.$$

- Vererbung: Die Abtastzeit und der Offset können an nachfolgende Blöcke vererbt werden, indem die Abtastzeit dort nicht gesetzt wird. Dazu muss lediglich der Wert -1 angegeben werden. Bei Synchronisationsproblemen ist dies ein wesentlicher Punkt.

In Abbildung 6.11 sind einige Vertreter von Abtastsystemen dargestellt.

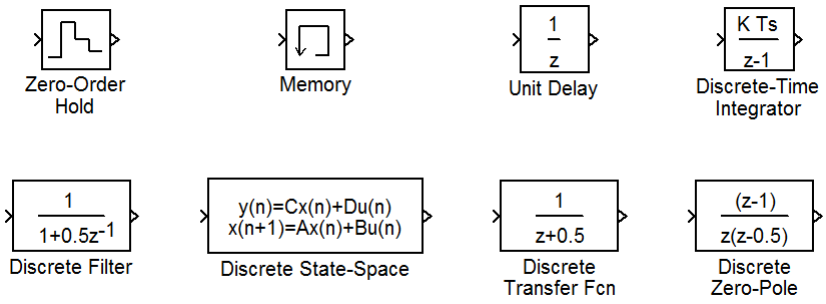

Abbildung 6.11: Eine Auswahl wichtiger Abtastsysteme in Simulink

Der erste Block ist das Halteglied, welches in Simulink als *Zero Order Hold* bezeichnet wird. Mit dieser Bezeichnung wird zum Ausdruck gebracht, dass das System die Folgenwerte durch eine konstante Funktion zwischen den Abtastzeitpunkten darstellt. First Order Hold würde bedeuten, dass die Abtastwerte durch affine Funktionen verbunden werden.

Als zweiter Block ist der *Speicherblock* dargestellt, welcher in Beispiel 6.10 aufgetreten ist. Dieser verzögert das Eingangsdatum um einen Abtastzeitschritt.

Der Block *Unit Delay* hat im Wesentlichen die selbe Wirkung auf ein Eingangssignal wie der Speicherblock mit dem Unterschied, dass die Abtastzeit vorgegeben werden kann.

Unter dem Block des *diskreten Integrators* versteht man eine Akkumulation des Eingangssignals mit Verstärkungsfaktor und anzugebender Zeitkonstante, also im Wesentlichen eine Summation. Die Namensgebung soll das ähnliche Verhalten zum zeitkontinuierlichen Integrator wiederspiegeln.

Der *diskrete Filter* wird durch eine z-Übertragungsfunktion etwas anderer Bauart angegeben. Im Fall des diskreten Filters mit endlicher Impulsantwort, FIR – Finite Impulse Response, wird die Übertragungsfunktion als

$$G(z^{-1}) = \frac{b(z^{-1})}{a(z^{-1})} = \frac{b_1 + b_2 z^{-1} + \ldots + b_{n+1} z^{-n}}{a_1 + a_2 z^{-1} + \ldots + a_{m+1} z^{-m}}$$

mit $m \geq n$ angegeben. Filter mit unendlicher Impulsantwort, IIR – Infinite Impulse Response, unterscheiden sich von Filtern mit endlicher Impulsantwort dadurch, dass bei diesen FIR–Filtern die Koeffizienten $a_2, \ldots, a_{m+1} = 0$ sind.

Wie aus Abschnitt 4.6 bekannt, ist auch für Abtastsysteme eine Zustandsraumdarstellung möglich. In Simulink wird diese repräsentiert durch den *diskreten Zustandsraumblock*. Die Zustandsraumdarstellung von Abtastsystemen wird durch

$$\begin{aligned} x_{k+1} &= \Phi\ x_k + \Lambda\ u_k, \\ y_k &= C\ x_k + D\ u_k, \end{aligned} \tag{6.4}$$

mit $x_k \in \mathbb{R}^n, u_k \in \mathbb{R}^m, y_k \in \mathbb{R}^p, \Phi \in \mathbb{R}^{n \times n}, \Lambda \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^{p \times n}, D \in \mathbb{R}^{p \times m}$ beschrieben.

Die *diskrete Übertragungsfunktion* und der Block *Discrete Zero–Pole* stellen die zeitdiskreten Gegenstücke zum kontinuierlichen Fall dar. Als Übertragungsfunktion ist hier die z–Übertragungsfunktion gemeint. Die übrigen Funktionalitäten sind gleich dem kontinuierlichen Fall. Es ist auch wieder die Verwendung von Matrizen möglich, welche die Anzahl der Ein– und Ausgänge vorgeben.

Abschließend soll ein Beispiel den Vergleich zwischen zeitkontinuierlichen und zeitdiskreten Modellen in Simulink darstellen.

Beispiel 6.11. Es wird ein Integrationsblock und ein sogenannter diskreter Integrationsblock wie in Abbildung 6.12 betrachtet. Das Beispiel soll die Funktionsweise eines zeitkontinuierlichen und eines zeitdiskreten Integrators zeigen und vergleichen.

Als Eingangssignal u wurde ein Dreieck gewählt, wie es im unteren rechten Bild der Abbildung 6.12 zu sehen ist. Der erste zeitdiskrete Integrator hat eine Abtastzeit von $T_1 = 0.2$, der zweite eine von $T_2 = 0.5$ eingestellt. Diese spiegeln sich durch das entsprechende Halteglied am Ausgang der Blöcke dadurch wieder, dass die Treppen am Ausgang genau von dieser Länge sind. Man erkennt gut, dass der zeitdiskrete und der zeitkontinuierliche Integrator dieselbe Aufgabe, das zeitliche Integrieren bzw. Akkumulieren oder Aufsummieren, erfüllen. Zeitkontinuierlich kann die Funktion durch

$$y_1(t) = \int_0^t u(\tau) \mathrm{d}\tau$$

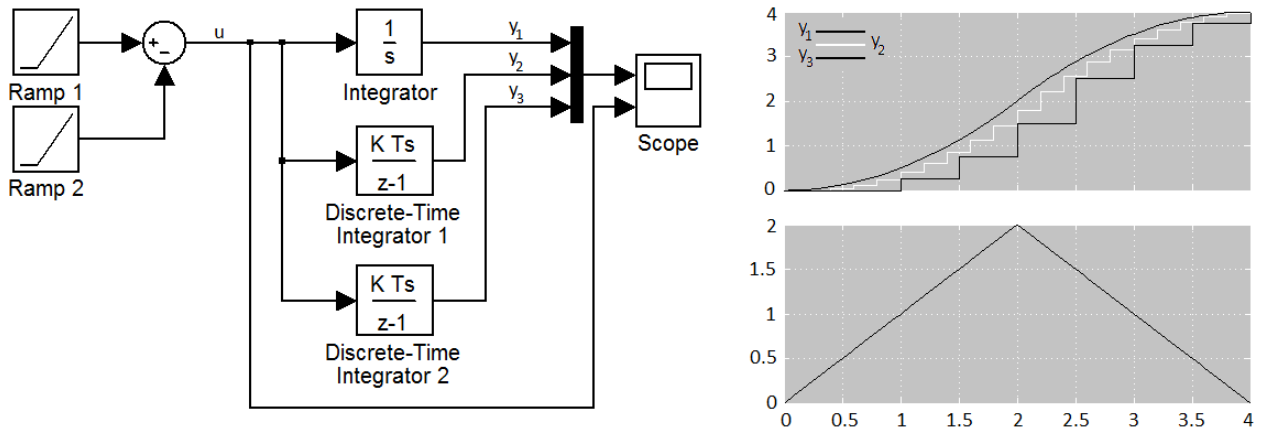

Abbildung 6.12: Gegenüberstellung eines zeitkontinuierlichen und eines zeitdiskreten Integrators für zwei verschiedene Abtastzeiten

beschrieben werden. Für die zeitdiskrete Integration gilt die Beziehung

$$y_j(t) = K_{j-1} \cdot T_{j-1} \cdot \mathbf{H}\left(\sum_{i=1}^{\ell} u_i\right) \quad \text{für} \quad j = 2, 3$$

mit $K_m, T_m > 0$ Parametern des diskreten Integrationsblocks $m = 1, 2$. Dabei ist $\ell \in \mathbb{N}$ jener Index in der Zerlegung $\mathcal{Z} = \{t_1, \ldots, t_\ell, \ldots, t_k\}$ mit $t_1 = 0$, welcher durch den Zeitpunkt t impliziert wird.

Bemerkung 6.12 (Quasi–zeitkontinuierliche Simulation in Simulink)**.** Motiviert durch Beispiel 6.11 muss angemerkt werden, dass das Integral im (sogenannten) zeitkontinuierlichen Bereich ebenfalls durch eine numerische Iteration bestimmt wird, beispielsweise durch die explizite Methode von Euler, wie in Kapitel 5.1.1 dargestellt.
Vergleicht man das Iterationsverfahren mit der Gleichung für den zeitdiskreten Integrator, erkennt man, dass diese sich nur geringfügig unterscheiden. Für den sogenannten zeitkontinuierlichen Block wird die Abtastzeit für die interne numerische Iteration durch den solver–Algorithmus festgelegt, im zeitdiskreten Bereich wird die Abtastzeit manuell vorgegeben. Strukturell ist zwischen den Modellen aber nur wenig Unterschied.
Der zeitkontinuierliche Bereich sollte hinsichtlich dieses Sachverhaltes vielmehr als *quasi–zeitkontinuierlich* bezeichnet werden.

6.6 Simulationsparameter von Abtast– und hybriden Systemen

Der letzte Abschnitt widmet sich den Simulationsparametern von zeitdiskreten sowie hybriden Systemen. *Hybride Systeme* sind solche, die sowohl zeitkontinuierliche als auch zeitdiskrete Systeme beinhalten. Da zeitdiskrete Systeme mit Abtastzeiten arbeiten, also Werte nur zu Zeitpunkten gemäß einer Zerlegung

$$\mathcal{Z} = \{t_1, \ldots, t_k\}$$

verfügbar sind, muss hier bei der Simulation einiges beachtet werden. Weiters muss der Fall unterschiedlicher Abtastzeiten bei Abtast– und bei hybriden Systemen diskutiert werden.

zu Beginn werden Abtastsysteme alleine betrachtet und die Simulationsparameter hinsichtlich der Simulation mit fester und variabler Schrittweite analysiert. Anschließend wird die Betrachtung auf hybride Systeme in Bezug auf diese Simulationsattribute erweitert.

Abtastsystem mit einer einzigen Abtastzeit. Bei der *Simulation mit fester Schrittweite* sind eine Vielzahl von *Fixed–step*–Lösern als auch der Löser *discrete* (no continuous) verfügbar. Wird die Wahl der Schrittweite bei einem Fixed–step–Löser automatisch eingestellt, so wird diese entsprechend der Abtastzeit gewählt. Bei Simulation mit kleineren Schrittweiten muss die Wahl demzufolge auf eine Abtastzeit fallen, welche ein Vielfaches der Schrittweite der im Abtastsystem vorhandenen Blöcke ist.

Für *Simulationen mit variabler Schrittweite* gilt, dass eine Vielzahl von *Variable–step*–Lösern als auch der Löser *discrete* (no continuous) verfügbar sind. Wird die maximale Schrittweite automatisch eingestellt, dann resultiert eine feste, der Abtastzeit entsprechende, Schrittweite. Bei Werten für die maximale Schrittweite, welche die zulässige Abtastzeit unter- oder überschreiten, wird diese automatisch auf die Abtastzeit gesetzt.

Abtastsystem mit mehreren Abtastzeiten. Hier muss bei *Simulation mit fester Schrittweite* und der Auswahl eines *Fixed–step*–Lösers darauf geachtet werden, dass die Abtastzeiten aller beteiligten Blöcke und Subsysteme auf dem Gitter, welches die numerische Iteration erzeugt, zu liegen kommen. Dies gilt auch für Schrittweiten, die durch einen Offset verschoben werden. Es sind auch hier eine Vielzahl an *Fixed–step*–Lösern als auch der Löser *discrete* (no continuous) verfügbar. Bei der automatischen Einstellung der Iterationschrittweite wird diese auf den Wert des größten gemeinsamen Teilers aller auftretenden Abtastzeiten gesetzt, der sogenannten *fundamentalen Abtastzeit*.

Bei der *Simulation* von Abtastsystemen *mit fester Schrittweite* muss auf die Übergänge zwischen Blöcken unterschiedlicher Abtastzeiten geachtet werden. Die Eigenschaft *Tasking mode for periodic sample times* ist bei *Single Tasking* darauf eingestellt, dass die verschiedenen Abtastzeiten ignoriert werden. Die Blöcke werden zu jedem Abtastschritt unabhängig voneinander gerechnet. Hingegen bei *Multi Tasking* können sich die Berechnungen, abhängig von ihren Prioritäten, gegenseitig unterbrechen. Die höhere Priorität hat dabei im Allgemeinen die kürzere Abtastzeit. Dies ist ein heikler Prozess, da bei unsachgemäßer Verwendung Datenverlust die Folge sein kann. Werden in einem Simulink–Modell auch asynchrone Ereignisse simuliert, beispielsweise modelliert durch bedingt ausgeführte Subsysteme, können die Übergänge nicht mehr automatisch behandelt werden. Abhilfe hierfür schafft unter anderem ein *Rate Transmission Block*, der Teilsysteme mit unterschiedlichen Raten behandelt. Die automatische Einstellung der periodischen Abtastung hat bei gleichen Raten im System Single Tasking und bei unterschiedlichen Raten Multi Tasking zur Folge.

Die *Simulation mit variabler Schrittweite* erlaubt die Auswahl einer Vielzahl von *Variable–step*–Lösern als auch den Löser *discrete* (no continuous). Eine automatische Einstellung wählt Iterationsschrittweiten aus, welche es ermöglicht, alle Blöcke mit den jeweiligen Abtastzeiten zu berechnen.

Allgemeines zur Simulation hybrider Systeme. Für hybride Systeme ist der Löser *discrete* (no continuous) natürlich nicht mehr zu verwenden. Es empfiehlt sich, die Löser *ode23* und *ode45*, beide Runge–Kutta–Verfahren variabler Ordnung, zu verwenden. Ausdrücklich ungeeignet sind die Löser *ode15s* und *ode113*, da die diskreten Blöcke unstetige Änderungen in den Signalverläufen zur Folge haben können und dafür diese Löser nicht geeignet sind.

Hybrides System mit einer einzigen Abtastzeit. Bei der *Simulation mit fester Schrittweite* ergibt die automatische Einstellung eine Iteration mit Schrittweite entsprechend der Abtastzeit, die in diesem Fall bei allen Blöcken gleich ist. Sonst muss die Schrittweite so gewählt werden, dass die Abtastzeiten ein Vielfaches der Schrittweite ergeben. Bei der *Simulation mit variabler Schrittweite* wird die maximale Schrittweite auf die Abtastzeit gesetzt, sollte diese zu groß gewählt werden.

Hybrides System mit mehreren Abtastzeiten. Für dieses Szenario sind die Aussagen für Abtastsysteme mit mehreren Abtastzeiten und jene über hybride Systeme mit einer einzigen Schrittweite miteinander zu verknüpfen.

Beispiel 6.11 hat bereits ein einfaches hybrides System mit unterschiedlichen Abtastzeiten dargestellt.

Kapitel 7

Simulink am MMT–Server

Dieses Kapitel soll zum Abschluss und zur Abrundung der Arbeit die Einbindung von Simulink am sogenannten MMT–Server, MMT für *Mathematics, Modelling and Tools*, zeigen. Es wird zuerst die Oberfläche, und damit die Schnittstelle zu MATLAB, gezeigt und dann die Erweiterung mit Simulink. Abschließend sollen einige Simulink–Beispiele, die am Server zur Verfügung stehen, vorgestellt werden.

7.1 Der MMT–Server

Der MMT–Webserver bietet Studierenden die Möglichkeit interaktiv mathematische Simulationen zu betrachten und somit das Verhalten von Modellen und deren Realisierungen in diversen Simulatoren zu untersuchen. Die einzelnen Experimente beleuchten immer nur einen kleinen Ausschnitt, um schrittweise das große Gesamtmodell verstehen und interpretieren zu lernen.

Den Beginn dieser interaktive Lern– und Lehrplattform machte 2006 der MATLAB–Webserver, der in der MATLAB–Version 2006 zur Verfügung stand. Studierende hatten dadurch die Möglichkeit, MATLAB online zu verwenden. Die Einstellung des Servers durch The Mathworks® führte zur Entwicklung eines Nachfolgerkonzepts. Dieses Webinterface ermöglicht nun die Interaktion mit MATLAB. Auf diesem anfangs rudimentären System wurden einige typische Beispiele der Modellbildung und Simulation implementiert und für Online–Experimente ausgelegt.

Von der MATLAB–Webserver–Idee getrieben, beschränkte sich das System in seiner Funktionalität anfangs auf MATLAB als Simulator. Es wurde mit der Zeit offensichtlich, dass auch andere Simulationsumgebungen auf diesem System implementiert werden sollten um eine möglichst breite Palette an verschiedenen Simulatoren anbieten zu können. Aufgrund der Nähe zu MATLAB war die erste Idee, Simulink als graphische Simulationsumgebung einzubinden.

Im Jahr 2011 wurde das Interface für den MMT–Server neu gestaltet. Zu diesem Anlass wurde auch die Integration von Simulink in Angriff genommen. Das Webinterface des MMT–Servers ist in Abbildung 7.1 zu sehen.

Die linke Seite des Webinterfaces bietet einen Verzweigungsbaum, der die unterschiedlichen Experimente hierarchisch aufgebaut darstellt. Die untersten Verzweigungen stellen einzelne Expe-

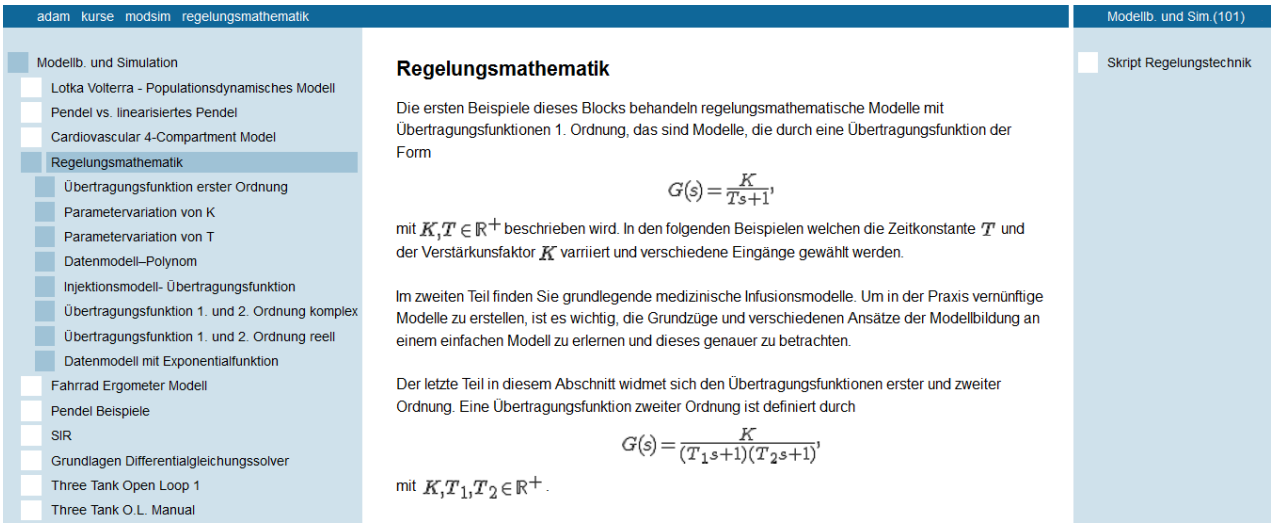

Abbildung 7.1: Webinterface eines Experiments am MMT–Webserver

rimente dar, die Ebene darüber das Modell allgemein.
In der mittleren Spalte sind die Interaktionsmöglichkeiten beheimatet, also Eingabe von Parameterwerten, Auswahl von Funktionen sowie die graphische Ausgabe und Rückgabe von Simulationswerten.
Die rechte Spalte beinhaltet Dateien mit MATLAB–Files, welche das Nachlesen des Codes ermöglicht, mit welchem die Simulation implementiert ist. Zur Verdeutlichung werden die folgenden Beispiele gebracht.

Beispiel 7.1. In diesem Beispiel werden Verzögerungsglieder erster und zweiter Ordnung betrachtet. Dabei sollen der Einfluss von Parametern auf das Verhalten der Übertragungsglieder sowie die Ausgangssignale für verschiedene Eingangssignale untersucht werden.

Gegeben sei ein Verzögerungsglied erster Ordnung durch

$$G_1(s) = \frac{K}{Ts+1}$$

mit $K, T > 0$. Dabei ist K der Verstärkungsfaktor und T die Zeitkonstante des Verzögerungsglieds. Aufgabe ist es nun, Experimente bei festen Werten von K und T und verschiedenen Eingangssignalen anzustellen. Als Eingangssignal liegen dabei die folgenden Funktionen vor:

- Heaviside[1] Sprungfunktion[2]
- Rechteckimpuls
- Treppenfunktion mit kompaktem Träger
- Periodische Rechtecksfunktion
- Rampenfunktion
- Sägezahnimpuls
- Sinusfunktion
- Gedämpfte Sinusfunktion

Ziel ist es, die Auswirkung des Verzögerungsglieds erster Ordnung auf verschiedene Eingangssignale zu untersuchen. Weiters soll der Einfluss der Parameter K und T durch wiederholte Simulationsrechnungen beobachtet werden.

[1] Oliver Heaviside (1850 – 1925), britischer Mathematiker und Physiker
[2] Siehe Definition A.1.

Sei weiters ein Verzögerungsglied zweiter Ordnung betrachtet, welches gegeben ist durch

$$G_2(s) = \frac{K}{(T_1 s + 1)(T_2 s + 1)},$$

mit $K, T_1, T_2 > 0$. Hier ist der Einfluss der Polstellen sowohl des Verzögerungsglieds erster als auch des Verzögerungsglieds zweiter Ordnung zu untersuchen.

Für all diese Aufgaben ist jeweils auch das MATLAB–Programm der zugehörigen MATLAB–Simulation als Datei verfügbar. In dieser Datei sind jedoch nicht nur klassische MATLAB–Befehle enthalten, sondern auch Befehle, welche die Schnittstelle betreffen.

Für dieses Beispiel lautet der spezifische Code für die Schnittstelle

```
K  = str2double(instruct.var1);
T  = str2double(instruct.var2);
number  = str2double(instruct.var3);
tend  = str2double(instruct.var4);
dt  = str2double(instruct.var5);
```

Hier werden die Eingaben aus den Textfeldern des Webinterfaces in MATLAB–Variablen konvertiert. Ausgehend von diesen Variablen wird dann die Simulation in MATLAB weiterbehandelt. Die Übertragungsfunktion des Verzögerungsglieds erster Ordnung wird beispielsweise durch

```
num = [ K ];
den = [ T 1 ];
sys = tf ( num, den );
```

generiert. Hier schließt im Programmtext ein Absatz über die Fallunterscheidung hinsichtlich des Eingangssignals mit Hilfe einer `switch`–Anweisung an. Abschließend gibt es einen Programmabschnitt für die Aufbereitung der graphischen Ausgabe.

Dieses einfache Beispiel soll die Struktur des Webinterfaces verdeutlichen. Ein fortgeschritteneres Beispiel ist das Folgende.

Beispiel 7.2. Hier werden die unterschiedlichen Modelle eines Pendels behandelt. Es gibt mehrere Varianten, ein Pendel zu simulieren, wobei immer die entsprechenden Voraussetzungen und Vernachlässigungen beachtet werden müssen. Um die unterschiedlichen Modelle zu verstehen und zu vergleichen, welches Modell welche Stärken und Schwächen aufweist, stehen die folgenden Varianten zur Verfügung:

- Mathematisches Pendel
- Linearisiertes Pendel
- Fadenpendel mit Anschlag
- Doppelpendel
- Gleitendes Pendel

Das *mathematische Pendel* wird durch die Differentialgleichung

$$m\,l\,\ddot{\varphi} + l\,d\,\dot{\varphi} = -m\,g\,\sin(\varphi) \tag{7.1}$$

für die Masse $m > 0$, die Fadenlänge $l > 0$, die Erdbeschleunigung $g > 0$ und die Dämpfungskonstante $d > 0$ beschrieben. Dabei ist $\varphi\colon (-\frac{\pi}{2}, \frac{\pi}{2}) \to \mathbb{R}$ die Funktion des Winkels mit dem Startwinkel $\varphi(0)$ und der Startwinkelgeschwindigkeit $\dot{\varphi}(0)$.

Für das *linearisierte Pendel* wird die rechte Seite der Gleichung 7.1 betrachtet und die Sinusfunktion linearisiert, was für kleine Werte von φ gerechtfertigt ist. Die beschreibende Differentialgleichung lautet dann

$$m\,l\,\ddot{\varphi} + l\,d\,\dot{\varphi} + m\,g\,\varphi = 0.$$

Die Aufgabe besteht nun darin, das linearisierte Modell mit dem nichtlinearen Modell für Experimente mit kleinen Werten von φ und mit großen Werten von φ zu betrachten. Dabei ist zu beobachten, dass das lineare Modell für große Werte von φ falsche Ergebnisse im Vergleich zum nichtlinearen Modell liefert.

Im Hinblick auf MATLAB werden diese Modelle mit ode–Löser behandelt. Der hierfür interessante Abschnitt im MATLAB–Code für den Vergleich von linearen mit nichtlinearen Pendeln ist der Folgende.

```
options = odeset('AbsTol',0.0001)
[t,phi] = ode45(@example_pendel1_gleichung,[tstart tend], [phi10; phi20],
                options)
[tlin,philin] = ode45(@example_pendel2_gleichung_lin,[tstart tend],
                      [phi10; phi20], options)
for j = 1:length(t)
    if phi(j,1) > 2*pi
        phi(j,1) = mod(phi(j,1),2*pi);
    else
        phi(j,1) = phi(j,1);
    end
end
```

Hier ist die Verwendung der `ode`–Funktion jeweils für die lineare und nichtlineare Differentialgleichung `example_pendel1_gleichung` und `example_pendel2_gleichung_lin` zu sehen.

Diese ausgesuchten Beispiele und die Tatsache, dass Simulink auch durch MATLAB gesteuert werden kann, bzw. dass ein Simulink–Modell mit MATLAB–Befehlen erzeugt werden kann, legt die Möglichkeit nahe, dass auch Simulink–Beispiele in den MMT–Server integriert werden können.

7.2 Integration von Simulink

Ein Simulink–Modell ist nach Definition 6.4 ein graphisches Blockdiagramm, welches im Sinne der System– und Signalflussgraphentheorie erklärt ist. Aus Sicht der Informationstechnik ist das Simulink–Modell aber ein syntaktisches MATLAB–Konstrukt.

Simulink ist eine graphisch orientierte Simulationsumgebung. Das Modell wird als Signalflussgraph visualisiert, welchem ein durch MATLAB ausgeführter Code zugrundeliegt. Dieser Um-

stand zeigt, dass Simulink eine Toolbox von MATLAB ist, oft aber als eigenständiger Simulator betrachtet wird.

Betrachtet man das Simulink–Modell aus Abbildung 7.2, so entspricht diesem graphischen Modell eine Codesequenz, welche das Simulink–Modell mit allen Böcken und allen Verbindungen als MATLAB–Programm beschreibt.

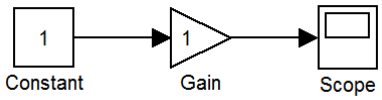

Abbildung 7.2: Einfaches graphisches Modell in Simulink

Der grundlegende Aufbau des zu Abbildung 7.2 gehörenden MATLAB–Codes lautet wie folgt.

```
Model {
  Name   "simple"
  [...]
  BlockParameterDefaults {
    Block {
      BlockType       Constant
      Value       "1"
      VectorParams1D       on
      SamplingMode       "Sample based"
  [...]
      SampleTime       "inf"
      FramePeriod       "inf"
    }
  [...]
}
```

`[...]` deuten an, dass an dieser Stelle Code nicht angegeben wurde. Unter anderem wird hier die gesamte Parametrisierung und Einstellung von Attributen des Simulink–Modells vorgenommen. Um eine Größenordnung der benötigten Code–Zeilen zu vermitteln, sei die Anzahl an Zeilen für das sehr simple Beispiel aus Abbildung 7.2 mit 730 angegeben.

Im Zuge der Neugestaltung des Webinterfaces im Jahr 2011 wurde eine Barriere entfernt, womit die Lauffähigkeit von Simulink am MMT–Server gewährleistet werden konnte. Zuvor durften die MATLAB–Programme auf dem Server nur aus einer einzigen MATLAB–Funktion bestehen. Dies machte esunmöglich, Simulink auszuführen, da ein Simulink-Modell notwendigerweise durch eine MATLAB-Funktion gesteuert werden muss. Nachdem die Möglichkeit zur Ausführung mehrere MATLAB–Funktionen hinzugekommen ist, konnten Simulink–Modelle am MMT–Server auszuführt werden. Wie genau ein Simulink–Modell am MMT–Server für Experimente zur Verfügung steht, soll das folgende Beispiel zeigen.

Beispiel 7.3. Es wird erneut ein Pendel–Experiment mit einem Webinterface aus Abbildung 7.3 betrachtet. Die Oberfläche des Webinterfaces gleicht einem MATLAB–basierten Experiment. An dieser Stelle wird jedoch ein Simulink–Modell zur Simulation herangezogen.

In der mittleren Spalte sind die Parameterwerte einzugeben, die graphische Ausgabe erfolgt

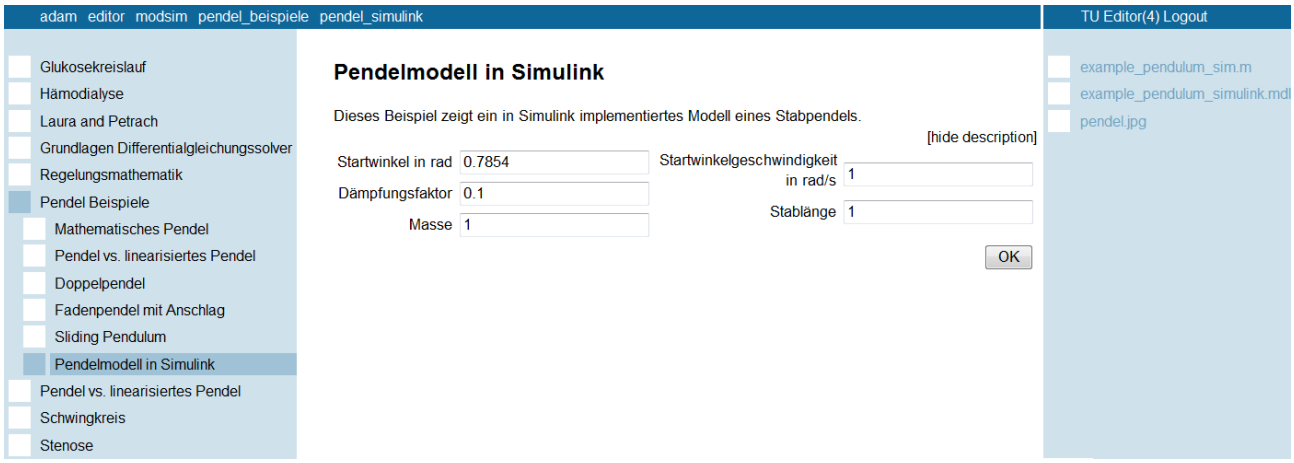

Abbildung 7.3: Webinterface eines Simulink–basierten Experiments mit einem Pendel

ebenfalls in dieser Spalte. Rechts findet sich eine MATLAB–Datei, welche die Parameter aus dem Webinterface verwaltet und die dort hinterlegte Simulink–Datei ausführt. Nachdem diese Datei der Code des Simulink–Modells ist, ist auch eine Graphikdatei hinzugefügt, welche ein Bild des dort hinterlegten Simulink–Modells zeigt. Wird die Simulink–Datei heruntergeladen und mit MATLAB geöffnet, so öffnet sich ein Simulink–Modell im Modell–Explorer, wie in Abbildung 7.4 zu sehen ist.

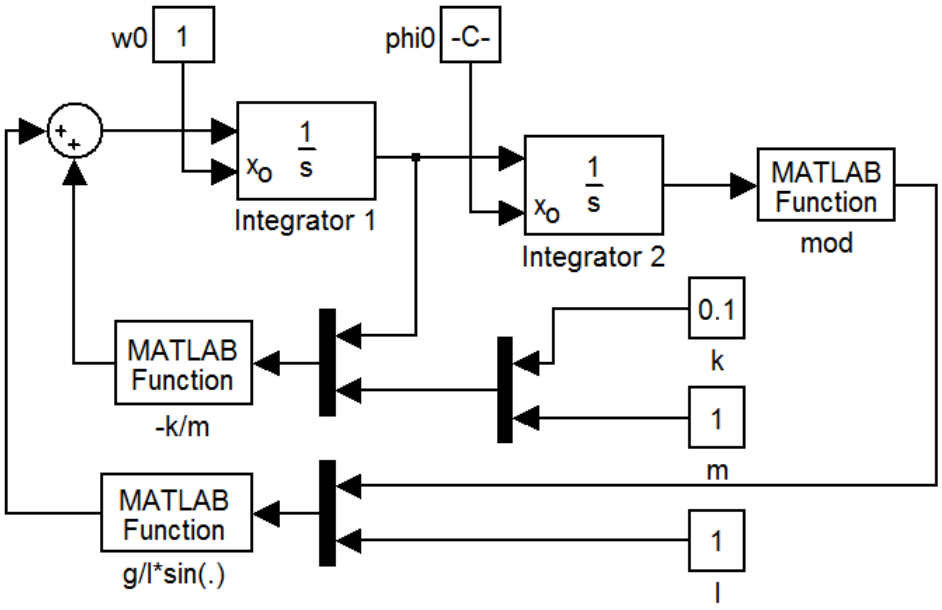

Abbildung 7.4: Simulink–Modell eines mathematischen Pendels

Dies eröffnet auch die Möglichkeit, ein Simulink–Modell mit einem in MATLAB implementierten Modell hinsichtlich diverser Anforderungen vergleichen zu können.

7.3 Ausgewählte Simulink–Beispiele

Dieser Abschnitt soll eine Auswahl an Beispielen zeigen, welche mit Simulink–Modellen arbeiten. Dabei handelt es sich um Beispiele aus unterschiedlichen technisch–naturwissenschaftlichen Bereichen, die sowohl zeitdiskrete als auch (quasi–) zeitkontinuierliche Systeme beinhalten.

Das erste Beispiel aus dem Bereich der klassischen Mechanik war das mathematische Pendel aus Beispiel 7.3. Weitere Beispiele aus unterschiedlichen Disziplinen sind die Folgenden.

Beispiel 7.4. Hier soll die Fähigkeit zum Transport einer Substanz oder eines Markers in der menschlichen Niere untersucht werden. Dazu wird ein Modell mit zwei Kompartimenten angenommen, wie in Abbildung 7.5 dargestellt.

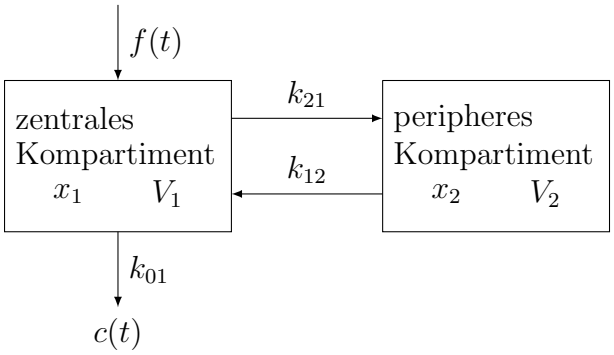

Abbildung 7.5: Kompartiment–Modell des Transports einer Substanz in der Niere

x_1 und x_2 geben die Menge der Substanz oder des Markers in dem jeweiligen Kompartiment an, deren Ableitungen die Änderungsraten der Mengen. Die Parameter k_{12} und k_{21} beschreiben den Austausch der Substanzen zwischen den Kompartimenten und k_{01} den Austausch zur Umgebung. f beschreibt die Injektion der Substanz oder des Markers in das Modell. Das Kompartiment–Modell wird durch das Differentialgleichungssystem

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = f(t) - (k_{01} + k_{21})x_1 + k_{12}x_2$$
$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = k_{21}x_1 - k_{12}x_2$$

und durch

$$c(t) = \frac{x_1(t)}{V_1}$$

beschrieben. Die Injektion wird durch

$$f(t) = \frac{D}{\tau}, \quad \text{für} \quad 0 \leq t \leq \tau$$

modelliert, wobei $D > 0$ die Menge der insgesamt injizierten Substanz ist.

Das oben beschrieben Differentialgleichungsmodell wird für dieses Experiment als Simulink–Modell implementiert, welches in Abbildung 7.6 zu sehen ist.

Im Experiment kann der Einfluss der Parameter k_{01}, k_{12} und k_{21}, des Volumens V_1, der Substanzmenge D und der Injektionszeit $\tau > 0$ untersucht werden.

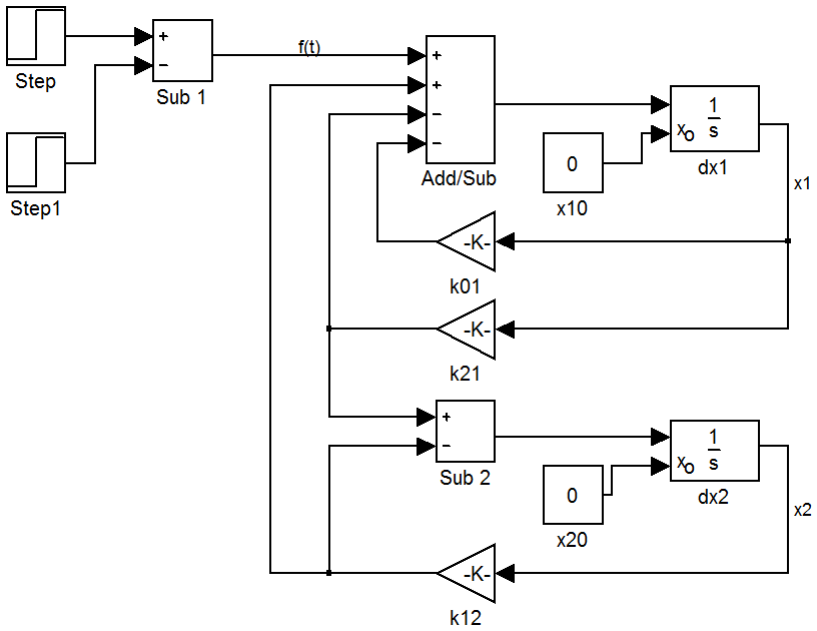

Abbildung 7.6: Simulink–System zum Kompartiment–Modell des Transports einer Substanz in der Niere

Beispiel 7.5. Dieses Beispiel ist aus dem Bereich der Übertragungstechnik, wo Störungen die empfangenen Daten verfälschen. Das System ist dargestellt durch das Simulink–Modell in Abbildung 7.7.

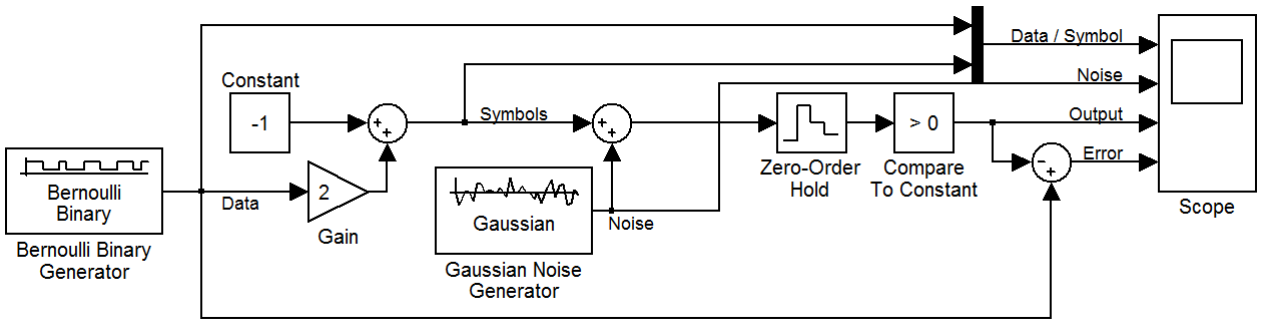

Abbildung 7.7: Simulink–Modell einer binären Datenübertragung

Es werden binäre Daten $s \in \{s_0 = 0, s_1 = 1\}$, hier in diesem Beispiel zufällig nach einer Bernoulli–Verteilung, generiert. Diese Daten werden für die Übertragung auf antipodale Symbole abgebildet, gemäß der Abbildungsvorschrift

$$s_0 \mapsto -1, \qquad s_1 \mapsto 1.$$

Bei allen auftretenden Systemen handelt es sich um Abtastsysteme mit einer Abtastzeit von $T = 0.5$. Auf dem Übertragungskanal wird additiv Rauschen überlagert. Die Werte z der Rauschquelle sind über eine Abtastperiode konstant und, mit gegebenem Mittelwert μ und gegebener Varianz σ^2, normalverteilt, $z \sim \mathcal{N}(\mu, \sigma^2)$. In diesem Beispiel gilt $\mu = 0$ und $\sigma^2 = 1$, also handelt es sich um die Standard–Normalverteilung $\mathcal{N}(0, 1)$. Das Summensignal

$$y = x + z$$

mit $x \in \{-1, 1\}$ wird abgetastet und mit dem Wert Null verglichen. Lediglich das Vorzeichen entscheidet, welchem Datum $\tilde{y} \in \{s_0, s_1\}$ der abgetastete Wert zugewiesen wird. Für die hier angegebenen Werte sind die Signalverläufe in Abbildung 7.8 dargestellt.

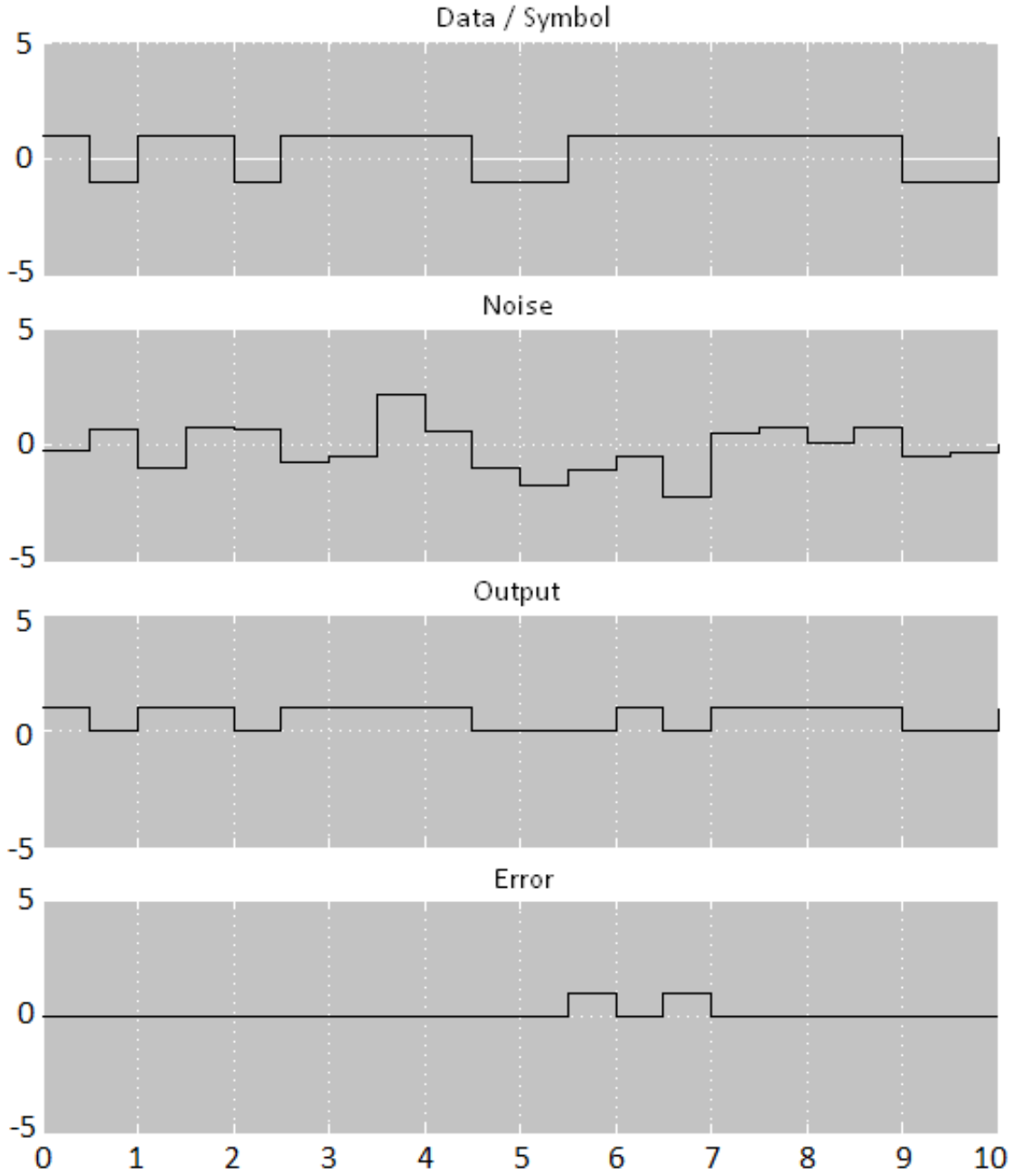

Abbildung 7.8: Signalverläufe der binären Datenübertragung aus Abbildung 7.7

In diesem Experiment soll der Einfluss der Abtastzeit T, des Mittelwerts μ und der Varianz σ^2 auf die Datenübertragung studiert werden.

Kapitel 8

Abschließende Bemerkungen

Dieser Abschnitt ist abschließenden Bemerkungen gewidmet. Auf der einen Seite soll hier auf einige mathematische Gegebenheiten hingewiesen werden, die genauer untersucht werden können. Auf der anderen Seite soll ein kleiner Ausblick hinsichtlich graphischer Modellbildung gemacht werden. Auch einige Anmerkungen zur Experimentierplattform MMT–Server sollen hier angeführt werden.

Die erste Bemerkung bezieht sich auf die Abtastung aus Definition 3.9. Hierbei wird argumentiert, dass ein Signalflussgraph an den Knoten nicht die Signale $x \in \mathrm{L}^2(\Omega)$, sondern Werte der Menge X zur Verfügung hat. Diese Werte aus X sind die Abtastwerte $x_i = x(t_i)$. In der Simulationsumgebung Simulink werden später die ode–Löser vorgestellt, welche die numerischen Lösungen der Differentialgleichungen in einem (quasi–) zeitkontinuierlichen System durch eine Iteration berechnen. Diese gehen aber von kontinuierlichen Größen aus, welche erst durch das Iterationsverfahren des ode–Lösers zu diskreten Werten werden. Hier gilt dieselbe Aussage wie in Bemerkung 6.12. Die Werte an den Knoten sind, wegen der a priori diskreten Natur von Simulink, diskrete Werte, welche durch den ode–Löser noch einmal diskretisiert werden. Grundsätzlich stimmt die Aussage über die diskrete Natur der Werte an den Knoten des Signalflussgraphen, man muss nur die quasi–kontinuierliche Struktur von Simulink mitberücksichtigen, um den Sachverhalt richtig einordnen zu können.

An dieser Stelle ist es vernünftig, eine Anmerkung zur Auffassung von Simulink als Signalflussgraph zu machen. Diese Definition ist richtig, jedoch muss darauf hingewiesen werden, dass einige Relationen nicht als verallgemeinerte Gewichtsfunktion im Zeitbereich geschrieben werden können. Eine Abhilfe stellt dabei die Möglichkeit dar, bei linearen Systemen den Laplace–Bereich miteinzubeziehen. Dabei muss hinzugefügt werden, dass an jedem Knoten das Signal und die korrespondierende Laplace–Transformierte vorliegen. So können einige Verknüpfungen im Laplace–Bereich als verallgemeinerte Gewichtsfunktionen geschrieben werden. Bei nichtlinearen Systemen kann die Beschreibung mit Hilfe des numerischen Lösungsalgorithmus erfolgen, da dieser im Allgemeinen eine Iteration darstellt, und so die Verknüpfungen durch Differenzengleichungen im zeitdiskreten Bereich beschrieben werden können.

Weiters folgt eine Ergänzung zu Bemerkung 4.31 über den Zusammenhang zwischen Laplace– und z–Transformation. Es wird dort eine Funktion $\tilde{f}$ angegeben, auf welche die Laplace–Transformation formal angewandt wird. An dieser Funktion ist wiederum δ beteiligt, dessen Definition in Bemerkung 4.31 nachzulesen ist. Außerdem wird die Forderung $\int_{\mathbb{R}} \delta(t)\,\mathrm{d}t = 1$ beigefügt,

was in der klassischen Analysis Probleme aufwirft. Es wird hier ein Riemann–Integral über eine Funktion, welche fast überall identisch Null ist, gebildet. Dies ist ein Hinweis darauf, dass hier die klassische Theorie versagt. In der Systemtheorie ist dieser unsaubere Zugang durchaus üblich und auch ausreichend. Mit der schon beschriebenen Tatsache, dass nur abgetastete Realisierungen der Signale in Simulink zur Verfügung stehen. Dabei folgt man der Konvention, dass die Abtastung von δ die Folge (δ_n) aus Kapitel A.2 liefert. Das Integral degeneriert damit zu einer Forderung an die Summe über (δ_n) gemäß

$$\sum_{n=0}^{\infty} \delta_n = 1.$$

Für die mathematisch richtige Formulierung der Integralbedingung an δ muss die Theorie über Distributionen herangezogen und die Systemtheorie hinsichtlich δ mit Hilfe dieser Theorie formuliert werden.

In Abbildung 4.8 ist ein Abtastsystem dargestellt, wie es in Kapitel 4.6 untersucht wird. Die Abtastsysteme, welche in Simulink modelliert werden, arbeiten etwas anders. Die Simulink–Abtastsysteme haben eine (quasi–) zeitkontinuierliche Umgebung und bei Verwendung eines Abtastsystems hat dieses am Eingang einen Abtaster und am Ausgang ein Halteglied (0. Ordnung). Die Strategie in Simulink ist, eine (quasi–) kontinuierliche Umgebung anzunehmen und die Abtastsysteme mit Abtast– und Haltegliedern daran anzupassen. Ein einfaches Argument dafür ist, dass die Signalquellen und –senken in dieser (quasi–) kontinuierlichen Umgebung vorhanden sind und die Abtastsysteme darin gewisse Funktionalitäten übernehmen.

Im Abschnitt über Schrittweitensteuerung und Fehlerschätzung von Kapitel 5 sind nur allgemeine Konzepte vorgestellt. Wie die Konzepte in MATLAB und Simulink aussehen, ist nicht im Detail bekannt, daher werden in dieser Arbeit diesbezüglich auch keine Aussagen getroffen.

Kapitel 6 ist eine Einführung in Simulink. In diesem Kapitel werden Zusammenhänge zwischen den theoretischen Kapiteln zu Beginn der Arbeit hergestellt und typische Elemente aus Simulink vorgestellt. Es wurden bei weitem nicht alle Toolboxen, welche in Simulink zur Verfügung stehen, betrachtet, und es wurden auch nicht alle Feinheiten der Simulation in Simulink ausgearbeitet.
Eine dieser Feinheiten betrifft die Simulation von hybriden Systemen, welche differential–algebraische Gleichungen repräsentieren. In diesen Systemen kommen sogenannte Ereignisse vor, welche (unstetig) die Modellstruktur, beispielsweise in Abhängigkeit eines Parameters oder eines Zustandes, verändern. Diese Systeme bieten viele Möglichkeiten, einen Simulator bis an die Grenze der Leistungsfähigkeit gehen zu lassen, und sind durchaus einer tieferen Betrachtungen würdig.

Einen ersten Gedanken für einen Ausblick stellt der folgende Aspekt dar.
Eine weitere Toolbox in Simulink ist Simscape, welche die sogenannte *physikalische Modellbildung* erlaubt. In derartigen Simulationsumgebungen werden nicht mehr die Gleichungen betrachtet und ausgehend von diesen ein Modell konstruiert, es werden standardisierte physikalische Komponenten, mit dem Ziel, eine Funktionalität zu realisieren, verbunden. Die Verbindung erfolgt dabei nicht, wie in Simulink, über gerichtete Signalflussverbindungen. Es werden ungerichtete Größen, deren Produkt proportional zur Energie ist, herangezogen. Die Modellbildung basiert hier auf sogenannten *Leistungsgraphen* und nicht, wie bei Simulink, auf Signalflussgraphen.

Auch für den MMT–Server ist die Integration von Simscape interessant, denn diese Toolbox benötigt Simulink als Umgebung zur Simulation. Es können also die Simulink–Signale, welche gerichtete Signalflüsse sind, mit der Simscape–Umgebung verkoppelt werden. Es wäre interessant, für den Server diese Simulationsumgebung integrieren zu können, um zum Vergleich einen weiteren Zugang zur Implementierung von Modellen einzubringen. Auf diese Weise wäre es beispielsweise möglich, die Implementierungen der unterschiedlichen Modelle für das mathematische Pendel in MATLAB, Simulink und Simscape qualitativ und quantitativ vergleichen zu können.

Das Gebiet der Modellbildung und Simulation ist sehr ausgedehnt und hat große Überschneidungen mit vielen Gebieten des technische–naturwissenschaftlichen Bereichs. Dies ist mit ein Grund dafür, warum Simulink als Simulator basierend auf einem Signalflussgraphen konzipiert ist, da dieser Simulator zu Beginn stark in der Regelungstechnik eingesetzt wurde und diese durch die Systemtheorie mit dem Umgang mit diesen Systemen vertraut ist. Im Bereich physikalischer Modellbildung ist es ebenfalls so. Viele technische Systeme sind zu komplex, um durch einen geschlossenen Satz von Gleichungen beschrieben werden zu können. Damit ist dieser komponentenorientierte Zugang aus dem Bedarf, große Systeme beschreiben zu können, gewachsen.

Diese Arbeit, gemeinsam mit dem abschließenden Bemerkungen über mögliche Fortsetzungen und detailliertere Betrachtungen, soll die herausragende Bedeutung von mathematischer Modellbildung und Simulation unterstreichen. Kenntnis über Simulatoren, deren Arbeitsweisen und Funktionalitäten sowie der Methoden im Hintergrund, sind damit im Umfeld technisch–natruwissenschafltlicher Frage– und Problemstellungen von besonderem Interesse.

Anhang A

Korrespondenzen ausgewählter Funktionaltransformationen

Der Begriff *Korrespondenz* bedeutet wörtlich übersetzt *gegenseitige Beantwortung*. Im Fall der Systemtheorie ist es die Angabe einer Tabelle, in welcher einige typische Urbilder und deren Bilder unter einer gewissen Funktionaltransformation aufgelistet sind. Hier wird die Tabelle für die Laplace– und z– Transformation angeführt.
Auf diese Weise können die Transformierten der Urbilder einfach abgelesen werden. Umgekehrt kann eine bekannte Transformierte gemäß Korrespondenz umgeformt werden, sodass ihr Urbild der Tabelle zu entnehmen ist.

A.1 Die Laplace–Transformation

Im Folgenden gilt $n \in \mathbb{N}$ und $a, b \in \mathbb{R} \setminus \{0\}$.

$f(t)$	$\mathcal{L}(f)(s)$	$f(t)$	$\mathcal{L}(f)(s)$
1	$\frac{1}{s}$	$\sin(bt)$	$\frac{1}{s^2+1}$
t^n	$\frac{n!}{s^{n+1}}$	$\cos(bt)$	$\frac{s}{s^2+1}$
e^{at}	$\frac{1}{s-a}$	$\mathrm{e}^{at}\ \sin(bt)$	$\frac{b}{(s-a)^2+b^2}$
$t^n\,\mathrm{e}^{at}$	$\frac{n!}{(s-a)^{n+1}}$	$\mathrm{e}^{at}\ \cos(bt)$	$\frac{s-a}{(s-a)^2+b^2}$

Tabelle A.1: Auflistung ausgewählter Korrespondenzen der Laplace–Transformation

Die Korrespondeztabelle A.1 bezieht sich wegen der Definition der Laplace–Transformierten auf kausale Signale, also jene, die $f(t) = 0$ für alle $t < 0$ erfüllen. Um solche Signale mit Hilfe elementarer Funktionen angeben zu können, verwendet man häufig die Heaviside–Funktion.

Definition A.1. Die Funktion $\varepsilon\colon \mathbb{R} \to \mathbb{R}$, definiert durch

$$\varepsilon(t) = \begin{cases} 0 & \text{für } t < 0, \\ 1 & \text{für } t \geq 0, \end{cases}$$

heißt die *Heaviside–Funktion*.

Ein beliebiges Signal h wird durch

$$f(t) = h(t) \cdot \varepsilon(t)$$

zu einem kausalen Signal f. Dementsprechend ist die erste Korrespondenz in Tabelle A.1 die Transformation der Heaviside–Funktion

$$\mathcal{L}(1)(s) = \mathcal{L}(\varepsilon)(s) = \frac{1}{s}.$$

A.2 Die z–Transformation

Im Folgenden gilt $n \in \mathbb{N}$, $a \in \mathbb{R} \setminus \{0\}$ und es sei $(\delta_n)_{n\in\mathbb{N}_0}$ die *Dirac*[1]*–Folge*, gegeben durch

$$\delta_n = \begin{cases} 1 & \text{für } n = 0 \\ 0 & \text{sonst.} \end{cases}$$

x_n	$\mathcal{Z}(x_n)(z)$	Existenzbereich
δ_n	1	$\forall z \in \mathbb{C}$
1	$\dfrac{z}{z-1}$	$\lvert z\rvert > 1$
a^n	$\dfrac{z}{z-a}$	$\lvert z\rvert > \lvert a\rvert$
n	$\dfrac{z}{(z-1)^2}$	$\lvert z\rvert > 1$
$\sin(an)$	$\dfrac{z \sin a}{z^2 - 2z\cos a + 1}$	$\lvert z\rvert > 1$
$\cos(an)$	$\dfrac{z(z - \cos a)}{z^2 - 2z\cos a + 1}$	$\lvert z\rvert > 1$

Tabelle A.2: Auflistung ausgewählter Korrespondenzen der z–Transformation

[1] Paul Adrien Maurice Dirac (1902 – 1984), britischer Physiker

Anhang B

Die Sätze von Gronwall

Satz B.1 (Lemma von Gronwall[1])**.** Sei $v\colon [0,T] \to \mathbb{R}$ und es gilt

$$\begin{aligned} v'(t) &\le \omega v(t) + \delta, \qquad t \in [0,T] \\ v(0) &\le \delta_0 \end{aligned} \tag{B.1}$$

mit $\delta_0, \delta \in \mathbb{R}_0^+$. Dann folgt

$$v(t) \le e^{\omega t}\delta_0 + \frac{e^{\omega t}-1}{\omega}\delta, \qquad t \in [0,T]. \tag{B.2}$$

Beweis. Die erste Gleichung aus (B.1) multipliziert mit $e^{-\omega t}$ führt auf

$$\left(e^{-\omega t}v(t)\right)' \le e^{-\omega t}\delta.$$

Integration und Multiplikation mit $e^{\omega t}$ liefert die Behauptung. □

Satz B.2 (Diskretes Lemma von Gronwall, Einschritt–Version)**.** Sei $(\xi_i)_{i=1,2,\dots}$ eine Folge nichtnegativer reeller Zahlen, welche

$$\begin{aligned} \xi_0 &\le \delta_0, \\ \xi_i &\le (1+\omega)\xi_{i-1} + \delta, \quad i = 1,2,\dots, \end{aligned} \tag{B.3}$$

mit $\omega, \delta_0, \delta \ge 0$ erfüllen. Dann gilt

$$\xi_i \le e^{i\omega}\delta_0 + \frac{e^{i\omega}}{\omega}\delta, \qquad \forall i. \tag{B.4}$$

Beweis. Die Rekursion

$$\begin{aligned} \xi_1 &\le (1+\omega)\delta_0 + \delta \\ \xi_2 &\le (1+\omega)\xi_1 + \delta \le (1+\omega)^2\delta_0 + (1+\omega)\delta + \delta \\ &\vdots \end{aligned}$$

[1] Thomas Hakon Grönwall (1877–1932), schwedischer Mathematiker

führt auf

$$\begin{aligned}\xi_i &\leq (1+\omega)^i\delta_0 + \big(1 + (1+\omega) + \ldots + (1+\omega)^{i-1}\big)\delta \\ &= (1+\omega)^i\delta_0 + \frac{(1+\omega)^i - 1}{\omega}\delta \\ &\leq e^{i\omega}\delta_0 + \frac{e^{i\omega} - 1}{\omega}\delta\end{aligned}$$

für $1 + \omega \leq e^{\omega}$ und $\omega > 0$. □

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Angermann A., Beuschel M., Rau M., Wohlfarth U., *MATLAB® – Simulink® – Stateflow®*, 6. Auflage, Oldenbourg, 2009

[2] Kugi A., *Automatisierung*, Skriptum, Technische Universität Wien, 2011

[3] Kugi A., *Regelungssysteme*, Skriptum, Technische Universität Wien, 2011

[4] Braack M., *Numerik für Differentialgleichungen*, Skriptum, Christian–Albrechts–Universität zu Kiel, 2011

[5] Auzinger W., *Einführung in die Numerik der Differentialgleichungen*, Skriptum, Technische Universität Wien, 2007

[6] Drmota M., *Diskrete Mathematik*, Skriptum, Technische Universität Wien

[7] Prechtl A., *Signale und Systeme*, Skriptum, Technische Universität Wien, 2002

[8] Fulmek M., Krattenthaler C., *Diskrete Mathematik*, Skriptum, Universität Wien, 2008

[9] Desch G., *Fourieranalysis*, Skriptum, Universität Graz

[10] Schnellenberg B., Jung B., *Die* L^1 *und* L^2*–Theorie der Fourier–Analysis*, Seminarbericht Harmonische Analysis, Eidgenössische Technische Hochschule Zürich, 2007

[11] *http://www.mathworks.com/products/simulink/*, April 2012

[12] Engeln-Müllges G., Niederdrenk K., Wodicka R., *Numerik-Algorithmen*, 10. Auflage, Springer-Verlag, 2011

Index