Universität-Gesamthochschule Paderborn
Fachbereich Mathematik / Informatik

Wissensbasierte Modellbildung und Simulation von hydraulischen Schaltkreisen

Diplomarbeit von

Michael Suermann

vorgelegt bei
Herrn Prof. Dr. Hans Kleine Büning

31. Mai 1994

Hiermit versichere ich, daß ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Inhaltsverzeichnis

1 Einführung **1**

1.1 Motivation und Überblick 1

1.2 Dynamisches Verhalten eines hydraulischen Schaltkreises 5

2 Konzeption der „Dynamik“ im System ^{art}*deco* **9**

2.1 Module der Modellbildung und Simulation 9

2.2 Simulationsablauf 12

3 Mathematische Grundlagen **15**

3.1 Problemformulierung 15

3.2 Problemklassen 17

3.3 Numerische Problemlösung 22

3.3.1 Prinzipielles Vorgehen 23

3.3.2 Beschreibung der Verfahren 24

3.3.3 Automatische Schrittweitensteuerung 27

3.3.4 Verfahrensauswahl 32

4 Wissensbasierte Modellbildung **37**

4.1 Problembeschreibung 37

4.2 Bestimmung der Modelltiefe von Komponenten 40

4.2.1 Grundidee, Prinzipielle Vorgehensweise 40

4.2.2 Einsatz von heuristischem Wissen 43

4.2.3 Wissensrepräsentation, Inferenzmechanismus 48

4.2.4 Der Algorithmus 49

4.3 Bestimmung des Informationsflusses 51

5 Wissensbasierte Simulation **55**

5.1 Grundidee, prinzipielle Vorgehensweise 55

5.2 Einsatz von heuristischem Wissen 57

5.2.1 Voraussetzungen, Basiswissen 57

5.2.2 Festlegung der Fehlerschranken 57

5.2.3 Bestimmung der Steifheit und Fehlergewichte 58
5.2.4 Quasi-optimale Verfahrensauswahl 59
5.3 Wissensrepräsentation, Inferenzmechanismus 60
5.4 Der Algorithmus 61

6 Perspektiven 65

A Die technische Beschreibungssprache 67
A.1 Definition der Syntax 67
A.2 Eine Auswahl dynamischer Verhaltensbeschreibungen 68
A.2.1 Gleichgangzylinder 70
A.2.2 4/3 Wege-Proportionalventil 79
A.2.3 Pumpe 90
A.2.4 Tank 92
A.2.5 Ventilverstärker 93
A.2.6 Numeric Control Einheit 95
A.2.7 Hydraulische Leitung 95
A.2.8 Elektrische Leitung 97

B Die Regelmenge 99
B.1 Definition der Syntax 99
B.2 Regeln zur Modelltiefenbestimmung 99
B.3 Regeln zur Simulationssteuerung 101

C Runge-Kutta-Verfahren 107
C.1 Explizite Verfahren 107
C.2 Implizite Verfahren 110

Literaturverzeichnis 117

Abbildungsverzeichnis

1.1 Eingangsgrößen der dynamischen Schaltkreissimulation 6
1.2 Ausgangsgrößen der dynamischen Schaltkreissimulation 7
2.1 Module der wissensbasierten Modellbildung und Simulation 10
3.1 Lösungen eines instabilen Anfangswertproblems 19
3.2 Steifes Anfangswertproblem mit explizitem Verfahren gerechnet 21
3.3 Steifes Anfangswertproblem mit implizitem Verfahren gerechnet 21
3.4 Klassen von lösbaren Anfangswertproblemen 22
3.5 Fehlerwechselwirkung der numerischen Lösungsberechnung 28
3.6 Fehlerfortpflanzung und numerische Stabilität 32
4.1 Modelltiefenabstufung der dynamischen Verhaltensbeschreibungen . . . 37
4.2 Differentialgleichungssysteme eines Gleichgangzylinders 38
4.3 Vollständige Verhaltensbeschreibung eines Gleichgangzylinders 39
4.4 Reduzierte Verhaltensbeschreibung eines Gleichgangzylinders 40
4.5 Statisch und dynamisch zu modellierende Komponenten 41
4.6 Sukzessive Modelltiefenzuordnung 42
4.7 Einfache Schaltkreisstrukturen 45
4.8 Regel zur Bestimmung der Abschneidekonstante β_{cut} 48
4.9 Initiale Verhaltensbeschreibung des Gleichgangzylinders 51
4.10 Teilbestimmte Verhaltensbeschreibung des Gleichgangzylinders 52
5.1 Funktionsdiagramm eines statischen Zustandsverlaufs 55
5.2 Regel zur Bestimmung des optimalen, impliziten Verfahrens 61

Tabellenverzeichnis

4.1 Wertetabelle der Abschneidekonstanten β_{cut} 46
4.2 Wertetabelle der Bewertungsfaktoren 47
5.1 Wertetabelle der Fehlerschranken $\varepsilon_{abs}, \varepsilon_{rel}$ 58
5.2 Verfahrensstufe und -ordnung der verwendeten Runge-Kutta-Verfahren 60
A.1 Äquivalente Bezeichnungen und ihre Bedeutung 69

C.1 Koeffizientenschema Runge-Kutta-Einbettungsformel 2./3. Ordnung . . 108
C.2 Koeffizientenschema Prince-Dormand-Einbettungsformel 4./5. Ordnung 108
C.3 Koeffizientenschema Prince-Dormand-Einbettungsformel 7./8. Ordnung 109
C.4 Koeffizientenschema Verner-Einbettungsformel 5./6. Ordnung 110
C.5 Koeffizientenschema Gauß-Legendre-Formel 4. Ordnung 111
C.6 Koeffizientenschema Gauß-Legendre-Formel 6. Ordnung 111
C.7 Koeffizientenschema Gauß-Legendre-Formel 10. Ordnung 112
C.8 Koeffizientenschema Gauß-Legendre-Formel 12. Ordnung 112
C.9 Koeffizientenschema Gauß-Legendre-Formel 16. Ordnung 113
C.10 Koeffizientenschema Gauß-Legendre-Formel 18. Ordnung 114
C.11 Koeffizientenschema Gauß-Legendre-Formel 22. Ordnung 115
C.12 Koeffizientenschema Gauß-Legendre-Formel 24. Ordnung 116

Nomenklatur

Notationen

$\mathbb{N}$	$\{1, 2, \ldots\}$
$\mathbb{Z}$	$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}^n$	$\underbrace{\mathbb{R} \times \mathbb{R} \times \ldots \times \mathbb{R}}_{\text{n-mal}}$
$\mathbb{C}$	Menge der komplexen Zahlen
$\Re(c)$	Realteil von $c \in \mathbb{C}$
$\Im(c)$	Imaginärteil von $c \in \mathbb{C}$
$\boldsymbol{z}$	Vektor
$\boldsymbol{A}$	Matrix
$\boldsymbol{J} = \frac{\partial \boldsymbol{f}(t, \boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)}$	Jacobi-Matrix
$\dot{z}(t)$	$\frac{\partial z(t)}{\partial t}$
$z(t_s)$	Funktionswert zum diskreten Zeitpunkt t_s
z_s	Näherungswert des exakten Funktionswertes $z(t_s)$
$\overline{z}(t)$	lokale Ausgangsgröße einer Komponente
$\underline{z}(t)$	lokale Eingangsgröße einer Komponente

Funktionen

$$\| \boldsymbol{x} \| = \max\{|x_1|, |x_2|, \ldots, |x_n|\} \text{ für } \boldsymbol{x} \in \mathbb{R}^n$$

$$\lceil x \rceil = \min\{z \in \mathbb{Z} \,|\, z \geq x \in \mathbb{R}\}$$

$$\lfloor x \rfloor = \max\{z \in \mathbb{Z} \,|\, z \leq x \in \mathbb{R}\}$$

$$\text{round}(x) = \begin{cases} \lceil x \rceil & \text{für } |x - \lfloor x \rfloor| \geq \frac{1}{2} \\ \lfloor x \rfloor & \text{sonst} \end{cases}$$

$$\text{sign}(x) = \begin{cases} 1 & \text{für } x > 0 \\ 0 & \text{für } x = 0 \\ -1 & \text{für } x < 0 \end{cases}$$

$$\text{sg}(x) = \begin{cases} x & \text{für } x > 0 \\ 0 & \text{sonst} \end{cases}$$

$$\text{sn}(x) = \begin{cases} 1 & \text{für } x = 0 \\ 0 & \text{sonst} \end{cases}$$

Formelzeichen

α	Stützstelle der Funktion $\boldsymbol{f}(t, \boldsymbol{z}(t))$
β	Gewicht der Funktion $\boldsymbol{z}(t)$
β_{cut}	Abschneidekonstante
$\mathbb{CC}$	Menge von verbundenen Komponenten
δ	mittlerer, geschätzter Iterationsfehler
Δ	mittlere, praktische Differenz zweier Iterationen
e_l	lokale Fehlerordnung eines Runge-Kutta-Verfahrens
e_g	globale Fehlerordnung bzw. Ordnung eines Runge-Kutta-Verfahrens
$\boldsymbol{\varepsilon}_l$	lokaler Verfahrensfehler, Diskretisierungsfehler
$\boldsymbol{\varepsilon}_g$	globaler Verfahrensfehler, Diskretisierungsfehler
ε	Fehler, Fehlerschranke
ε_{abs}	absoluter Fehler
ε_{rel}	relativer Fehler
γ	Gewicht der Funktion $\boldsymbol{f}(t, \boldsymbol{z}(t))$
h	Schrittweite
h_{adapt}	Schrittweitenadaptionsfaktor
K	Verstärkungsfaktor einer Komponente
K_{max}	maximaler Verstärkungsfaktor
K_{val}	Bewertungsfaktor der Verstärkung
$\boldsymbol{k}$	Steigungszwischenwert eines Runge-Kutta-Verfahrens
κ	Steifheitsgrad eines Differentialgleichungssystems
L	Lipschitz-Konstante
λ	Eigenwert
m	maximale Stufe eines Runge-Kutta-Verfahrens
$M(\lambda h)$	diskrete Übertragungsfunktion eines Runge-Kutta-Verfahrens
MD	Modelltiefe (Model<u>d</u>epth) einer Komponente
$\overline{MD}_{rel}$	durchschnittlich gewählte, relative Modelltiefe
MD_{val}	Bewertungsfaktor der durchschnittlich gewählten, relativen Modelltiefe
$\mathbb{MD}$	Menge der zur Verfügung stehenden Modelltiefen einer Komponente
N	Systemordnung einer Komponente
N_{max}	maximale Systemordnung
N_{val}	Bewertungsfaktor der Systemordnung

NL	Nicht-Linearität (Non-Linearity) einer Komponente
NL_{max}	maximale Nicht-Linearität
NL_{val}	Bewertungsfaktor der Nicht-Linearität
ν	Eigenfrequenz einer Komponente
ν_{min}	minimale Eigenfrequenz
ν_{cut}	Abschneidefrequenz
ν_{val}	Bewertungsfaktor der Eigenfrequenz
$\mathbf{\Phi}(t, \boldsymbol{z})$	Verfahrensfunktion
SI	Simulationszweck (Simulation-Intention)
$\mathbb{SI}$	Menge der zur Verfügung stehenden Simulationszwecke
σ	Gewicht der Fehlergröße
SO	Suchordnung bzgl. der dynamisch anspruchsvollsten Komponente (Search-Order)
t	Zeit
$\boldsymbol{z}$	Zustandsvektor

Technische Größen [1]

F_i	[N]	Kraft
i_{a_i}	[m A]	Ansteuerstrom
M_i	[N m]	Drehmoment
Q_i	$[\frac{\mathrm{l}}{\mathrm{min}}]$	Fluß
p_i	[bar]	Druck
U_i	[m V]	Spannung
x_{0i}	[mm]	Position
x_{1i}	$[\frac{\mathrm{m}}{\mathrm{s}}]$	Geschwindigkeit
x_{2i}	$[\frac{\mathrm{m}}{\mathrm{s}^2}]$	Beschleunigung
ω_{0i}	[rad]	Winkelposition
ω_{1i}	$[\frac{\mathrm{rad}}{\mathrm{s}}]$	Winkelgeschwindigkeit
ω_{2i}	$[\frac{\mathrm{rad}}{\mathrm{s}^2}]$	Winkelbeschleunigung

[1] Der Index i einer Größe bezeichnet das Gate einer Komponente, an dem diese anliegt.

1 Einführung

Die Modellbildung und Simulation von hydraulischen Schaltkreisen stellt einen wesentlichen Teilaspekt bei der rechnergestützten Inbetriebnahme hydraulischer Anlagen dar. Die Aufgabe, automatisch eine adäquate Modellbeschreibung eines hydraulischen Schaltkreises zu generieren und anschließend das dynamische, d.h. zeitveränderliche Verhalten des Systems, basierend auf diesem Modell, zu simulieren, erweist sich selbst für moderne, leistungsfähige Rechner als sehr zeitaufwendig. Da keine Einflußnahme auf die Problemkomplexität besteht, kann eine Reduzierung des Rechenaufwandes, ohne schwerwiegende Qualitätseinbußen beim Simulationsergebnis hinnehmen zu müssen, nur durch eine „intelligente“ Modellbildung und Simulationsberechnung erreicht werden. Diesen *neuen* Ansatz in der Simulationstechnik verfolgt die vorliegende Arbeit. Sie stellt die Konzeption und den Einsatz der wissensbasierten Methoden vor, mit denen im Softwaresystem ^{art}*deco* versucht wird, eine am Vorgehen des Hydraulik-Experten orientierte, dynamische Simulation zu berechnen.

1.1 Motivation und Überblick

Die Inbetriebnahme einer hydraulischen Anlage, die aus mechanischen, hydraulischen und elektronischen Komponenten bestehen kann, setzt sich im wesentlichen aus der Konfigurationsprüfung und der Diagnose zusammen.

In der Konstruktionsphase eines hydraulischen Schaltkreises spielt die Diagnose zunächst eine untergeordnete Rolle, gewinnt aber bei der Analyse unter- oder überbestimmter Anlagenteile sowie dem Auffinden defekter Bauteile innerhalb des Schaltkreises an Bedeutung.

Unterbestimmte Anlagenteile treten im allgemeinen immer dann auf, wenn die Kundenspezifikation des Schaltkreises unvollständig ist, d.h. wenn sie zu wenig Information enthält, um alle Druck-, Flußgrößen etc. in der Anlage zu bestimmen. In diesem Fall versucht die Diagnose, die Komponenten im Schaltkreis ausfindig zu machen, bei denen zusätzliche Vorgaben am ehesten zu einer vollständigen Berechnung des Schaltkreises führen.

Überbestimmte Anlagenteile folgen entsprechend aus einer widersprüchlichen Kundenspezifikation. Sie liegt vor, wenn die Berechnung der Schaltkreisgrößen Unstimmigkeiten mit den Sollwerten ergeben hat. Beispielsweise kann ein vorgeschriebener Druckwert an einer Komponente die Einstellung eines erforderlichen Flußwertes an einem anderen Bauteil im Schaltkreis verhindern. Die Diagnose sollte dann Aufschluß darüber geben, welche Anforderungen zum Widerspruch geführt haben und wie dieser aufzulösen ist.

Die Konfigurationsprüfung eines hydraulischen Schaltkreises umfaßt eine vollständige, technische Funktionsprüfung des hydraulischen Schaltkreises und besteht aus einer Schnittstellenprüfung, der statischen Funktionsprüfung sowie der dynamischen Funktionsprüfung (vgl. [Lem91]).

Die Schnittstellenprüfung eines hydraulischen Schaltkreises garantiert im wesentlichen die Kompatibilität der einzelnen Komponenten untereinander, d.h. sie sorgt dafür, daß die Schnittstellentypen verbundener Komponenten und ihre Anschlußmaße übereinstimmen. Darüberhinaus überwacht sie die Einhaltung der Nenngrößen der Komponenten. Beispielsweise würde die Verbindung einer hydraulischen Leitung mit einer elektrischen Komponente, die im allgemeinen nur über elektrische Anschlüsse verfügt, von der Schnittstellenprüfung erst gar nicht zugelassen. Genauso verhält sich der Fall, wenn der hydraulische Anschluß eines Gleichgangzylinders mit einer hydraulischen Leitung verbunden werden soll, die einen anderen Leitungsquerschnitt als den vom Anschluß des Zylinders geforderten besitzt. Unzulässige Wertvorgaben für die Zustandsgrößen einer Komponente wie z.B. einen über der Nenngröße liegenden Betriebsdruck werden ebenfalls von der Schnittstellenprüfung solange nicht akzeptiert, bis eine gültige Wertzuweisung erfolgt.

An die erfolgreiche Schnittstellenprüfung schließt sich die statische Funktionsprüfung eines hydraulischen Schaltkreises an. In dieser Prüfungsphase wird untersucht, ob sich die gemäß der Kundenspezifikation vorgegebenen, stationären Werte der Zustandsgrößen in der Anlage einstellen und der Schaltkreis somit den Anforderungen im statischen Sinne genügt. Aus Anwendersicht ergeben sich prinzipiell zwei Möglichkeiten einen Schaltkreis zu spezifizieren. Interessiert den Anwender primär, daß bestimmte Werte von gewissen Ausgangsgrößen des Schaltkreises eingehalten werden, sind die Sollwerte für diese Größen von ihm vorzugeben, und das Softwaresystem versucht dann aufgrund dieser Informationen die zugehörigen Werte der Eingangsgrößen zu bestimmen. Stehen hingegen die Eingangsgrößen im Vordergrund des Interesses, werden die aus den Wertvorgaben resultierenden Werte der Ausgangsgrößen des Schaltkreises berechnet. Beide Vorgehensweisen treten gleichberechtigt in der Praxis auf. Als Beispiel diene die Auslegung einer hydraulischen Hebebühne. Fordert der Kunde eine gewisse Mindesthebelast und ist die Geometrie des verwendeten Zylinders und des Steuerventils festgelegt, kann aufgrund dieser Angaben z.B. auf den von der Pumpe zu liefernden Betriebsdruck geschlossen und die Pumpe entsprechend den Anforderungen ausgelegt werden. Soll hingegen die Hebebühne in eine bereits bestehende Anlage integriert werden, so daß die Betriebsdaten der Pumpe vorgegeben sind, kann bei bekannter Bauform der verwendeten Komponenten die maximale Hebelast des Zylinders ermittelt werden.

Die statische Funktionsprüfung eines hydraulischen Schaltkreises basiert auf der Modellierung der Komponenten in Form von nichtlinearen Gleichungssystemen, die die Funktionalität der einzelnen Komponenten wiedergeben. Die Funktionalität einer Komponente ist durch die im Gleichungssystem beschriebenen Beziehungen zwischen den Parametern und den lokalen Zustandsgrößen der Komponenten festgelegt. Sie werden als von der Zeit unabhängige Variablen aufgefaßt, deren Druck-, Volumenstrom-, Kraft-, etc. werte den stationären Systemzustand des Schaltkreises zu einem diskreten Zeitpunkt repräsentieren. Die Topologie des Schaltkreises wird durch das in [KlBü93] vorgestellte Bausteinmodell beschrieben, das zwischen Bausteinen, Anschlüssen, Konnektoren, Quellen und Senken unterscheidet. Das Modell setzt die lokalen Verhaltensbeschreibungen der Bauteile untereinander in Beziehung, so daß eine Gesamtbeschreibung der Wirkzusammenhänge des technischen Systems entsteht. Faßt man die lokalen Verhaltensbeschreibungen der Komponenten als Constraints auf, bildet das Baustein-

modell einen hydraulischen Schaltkreis auf ein Constraint-Netz ab. Sind die Parameter sämtlicher Komponenten des hydraulischen Schaltkreises bekannt und die Werte einiger Zustandsgrößen als Kundenanforderungen vorgegeben, kann mittels lokaler Propagierung versucht werden, die restlichen Zustandsgrößen der nichtlinearen Gleichungssysteme zu bestimmen, d.h. eine erfüllende Belegung des Constraint-Netzes zu finden. In den meisten Fällen führt die isolierte Betrachtung der nichtlinearen Gleichungssysteme jedoch nicht zum gewünschten Erfolg, so daß man auf zusätzliches Wissen angewiesen ist, um das Constraint-Netz vollständig berechnen zu können. Dieses Wissen verbirgt sich in der Struktur des Schaltkreises. Eine Strukturanalyse des hydraulischen Schaltkreises macht dieses Wissen für die lokale Propagierung nutzbar, so daß die Lösungsmethode, falls eine Lösung existiert, dann zum Erfolg führt (vgl. [Hoff93]).

Die dynamische Funktionsprüfung setzt eine erfolgreiche, statische Funktionsprüfung voraus und soll u.a. letzten Aufschluß darüber geben, ob sich die von der statischen Funktionsprüfung vorhergesagten, stationären Werte im Rahmen gewisser, vorgegebener Toleranzen auch bei einer dynamischen Simulation des hydraulischen Schaltkreises einstellen. Neben dem Vergleich von stationären mit dynamisch berechneten Werten untersucht die dynamische Funktionsprüfung aber auch zeitabhängige Randbedingungen bezüglich der Zustandsgrößen, also ob z.B. eine Zustandsgröße ihren Sollwert innerhalb einer vorgeschriebenen Zeit erreicht. Anforderungen dieser Art konnten bisher im statischen Modell des hydraulischen Schaltkreises nicht formuliert und überprüft werden. Bezogen auf das vorangehende Beispiel der Hebebühne, ist es mit den aus der Statik bekannten Daten für die Pumpe möglich, das Verhalten z.B. des Zylinders über das interessierende Zeitintervall zu simulieren und das Ergebnis anschließend mit den Werten aus der Statik zu vergleichen. Treten in der Anlage z.B. aufgrund der großen Distanz zwischen Pumpe und Zylinder unvorhergesehene Druckverluste in den hydraulischen Leitungen auf, kann es sein, daß der benötigte Druck im Zylinder nicht erreicht wird, um die vom Kunden geforderte Mindesthebelast zu garantieren. In diesem Fall hat die dynamische Funktionsprüfung entgegen der statischen eine falsche Auslegung der Pumpe aufgedeckt, und die Anlage muß mit anderen Parametern erneut berechnet werden.

Die Basis der dynamischen Funktionsprüfung bildet die Simulation des hydraulischen Schaltkreises. Da in der Regel nur wenige Anlagenteile des Schaltkreises hohen, dynamischen Anforderungen unterliegen und die Simulation der Gesamtanlage im allgemeinen einen zu hohen Rechenaufwand erfordert, wird man sich auf die Simulation der dynamisch wesentlichen Anlagenteile beschränken. Für diese Anlagenteile gilt es ein adäquates, dynamisches Modell aufzustellen und anhand diesem Modell die Simulation durchzuführen. Die mathematische Beschreibung des dynamischen Modells formuliert ein globales Anfangswertproblem, das sich aus den nichtlinearen Differentialgleichungssystemen der lokalen Komponentenverhaltensbeschreibungen zusammensetzt und mit numerischen Verfahren gelöst wird. Mit zunehmender Detailgetreue des Modells nimmt jedoch der Rechenaufwand zur Lösung des Anfangswertproblems zu. Ziel ist es daher, ein dynamisches Modell des zu prüfenden Anlagenteils zu entwerfen, daß zum einen die an die Simulation gestellten Ansprüche des Anwenders erfüllt und zum anderen alle daran geknüpften, wesentlichen, dynamischen Eigenschaften des Teilschaltkreises wiedergibt und sich dennoch bezüglich des Rechenaufwandes in einem vertretbaren Rah-

men bewegt. Neben der optimalen Modellbildung spielt aber auch die richtige Wahl eines geeigneten, numerischen Verfahrens zur Lösung von Anfangswertproblemen eine wichtige Rolle um korrekte Simulationsergebnisse und kurze Rechenzeiten zu erhalten.

Bleibt die Modellbildung und die Verfahrensauswahl dem Anwender überlassen, wird er im allgemeinen mit dieser Aufgabe überfordert sein. Um dennoch auch unerfahrenen Anwendern, also Hydraulik-Laien, die Möglichkeit zu geben, problemlos eine dynamische Simulation eines hydraulischen Schaltkreises durchzuführen, werden wissensbasierte Techniken herangezogen, die das mangelnde Detailwissen des Anwenders ausgleichen und eine Automatisierung der Simulationsberechnung erlauben.

Die vorliegende Diplomarbeit stellt die Grundlagen und die Grundkonzeption vor nach der wissensbasiert die dynamische Modellbildung und die numerische Verfahrensauswahl für die Simulationsberechnung erfolgt. Basierend auf diesen Konzepten kann dann prototypisch die dynamische Schaltkreissimulation in das Softwaresystem ^{art}*deco* integriert werden, so daß dieses Softwaresystem in Zukunft neben der derzeit schon realisierten Schnittstellenprüfung und statischen Funktionsprüfung auch eine dynamische Funktionsprüfung von hydraulischen Schaltkreisen erlaubt.

Um dem Leser eine konkrete Vorstellung von der dynamischen Simulation eines hydraulischen Schaltkreises zu geben, soll das im anschließenden Abschnitt ausführlich vorgestellte Beispiel einer dynamischen Teilschaltkreissimulation dienen.

Der Abschnitt 2 erläutert den prinzipiellen Aufbau und die Funktionsweise der wissensbasierten Modellbildung und Simulation innerhalb des Softwaresystems ^{art}*deco* und beschreibt den möglichen Ablauf einer Simulationsberechnung mit dem System aus der Anwenderperspektive.

Im Abschnitt 3 wird zunächst das Problem der Simulationsberechnung formalisiert und die verschiedenen, relevanten Problemklassen von Anfangswertproblemen diskutiert, um dann die verwendeten, numerischen Verfahren ausführlich vorzustellen. Die Darstellung der Verfahren umfaßt hierbei auch die Schrittweiten- und Verfahrenssteuerung, auf die dann im Abschnitt 5 beim Algorithmusentwurf zurückgegriffen wird. Der Abschnitt 3 vermittelt daher nicht nur die mathematischen Fakten, sondern versucht auch die Grundlagen, soweit es der Rahmen dieser Diplomarbeit gestattet, zu motivieren, so daß die folgenden Abschnitte für den Leser transparent bleiben. Eine allumfassende Behandlung der Thematik erscheint hier jedoch nicht möglich, so daß der interessierte Leser an entsprechenden Stellen auf die im Anhang aufgeführte Literatur verwiesen wird.

Die Konzepte der wissensbasierten Modellbildung findet der Leser im Abschnitt 4. Dieser Abschnitt führt zuerst allgemein in die Problematik ein, beschreibt dann die Grundidee und prinzipielle Vorgehensweise und stellt hiernach im einzelnen die Einflußfaktoren und ihr Zusammenwirken bei der wissensbasierten Modellbildung vor. Die Gesamtkonzeption mündet abschließend in der algorithmischen Formulierung der Modelltiefenzuordnung, wobei Implementierungsdetails unberücksichtigt bleiben. Die Bestimmung des Informationsflusses wird in dieser Diplomarbeit nur noch angedacht, so daß sich die Darstellung auf die prinzipielle Vorgehensweise beschränkt.

Der Abschnitt 5 behandelt die wissensbasierte Simulationsberechnung und stellt die wesentlichen Kriterien vor, nach denen heuristisch Simulationsparameter belegt werden und eine optimale Verfahrensauswahl erfolgt. Neben dem mathematischen Wissen aus Abschnitt 3 werden hier vor allem die domänenspezifischen Besonderheiten erläutert. In der nachfolgenden, algorithmischen Formulierung der Simulationsberechnung findet sich dann eine Kombination beider Wissensbereiche wieder.

Abschließend wird ein Ausblick auf die noch offenen Punkte gegeben und mögliche Verbesserungsansätze der bisherigen Vorgehensweisen aufgezeigt.

1.2 Dynamisches Verhalten eines hydraulischen Schaltkreises

Im vorangegangenen Abschnitt sind die wesentlichen Aufgaben und Funktionen der statischen und dynamischen Funktionsprüfung eines hydraulischen Schaltkreises vorgestellt worden, so daß der Leser einen ersten Eindruck von der Komplexität dieser Aufgaben gewinnen konnte und einen groben Überblick über die beiden Prüfungsphasen erhalten hat. An dieser Stelle soll nun an einem konkreten Beispiel das Ergebnis einer Simulationsberechnung für den in Abbildung 1.1 gezeigten Anlagenteil dargestellt werden.

Den Ausgangspunkt der dynamischen Simulation bildet die poststatische Situation, d.h. der abgebildete Anlagenteil ist bereits erfolgreich statisch geprüft worden, so daß sämtliche, stationären Zustandswerte im Schaltkreis bekannt sind. In diesem Beispiel interessiert sich der Anwender für das dynamische Verhalten des Gleichgangzylinders, also für die Kolbenbewegung im Zeitintervall von $0 \ldots 10\,s$, wenn an der Kolbenstange die Kräfte $F_1(t)$, $F_2(t)$ anliegen und nach $4\,s$ sprunghaft eine Druckumschaltung am Ventil erfolgt. Für das Simulationsergebnis respektive den Funktionswerten der Kolbenposition und -geschwindigkeit erscheint ihm hierbei eine Genauigkeit von 10^{-3} als ausreichend.

Zum Simulationsanfangszeitpunkt $t_0 = 0\,s$ befindet sich die Kolbenstange des Gleichgangzylinders in der Mittenstellung, sie besitzt keine initiale Geschwindigkeit, und das 4/3 Wege-Proportionalventil steht auf *parallel*, so daß in der linken und rechten Kammer des Gleichgangzylinders zunächst die aus der Statik bekannten Werte des Betriebsdrucks p_0 und des Tankdrucks p_T anliegen. Die an den Gleichgangzylinder angreifenden Kräfte $F_1(t)$ und $F_2(t)$ sind in dem Beispiel vom Anwender so definiert worden, daß die resultierende Kraft $5500\,N$ beträgt und der jeweiligen Bewegungsrichtung des Kolbens entgegen wirkt. Um den am Ventil anliegenden Betriebsdruck nach $4\,s$ umzuschalten, hat der Anwender die in den Diagrammen der Abbildung 1.1 dargestellten Spannungsverläufe vorgegeben. Sie werden im jeweiligen Ventilverstärker des Schaltkreises in proportionale Ströme gewandelt, mit denen dann die Ansteuerung des 4/3 Wege-Proportionalventils erfolgt.

Alle weiteren hier nicht erwähnten System- und Simulationsdaten findet der Leser am Ende des Abschnitts in einer zusammenfassenden Darstellung.

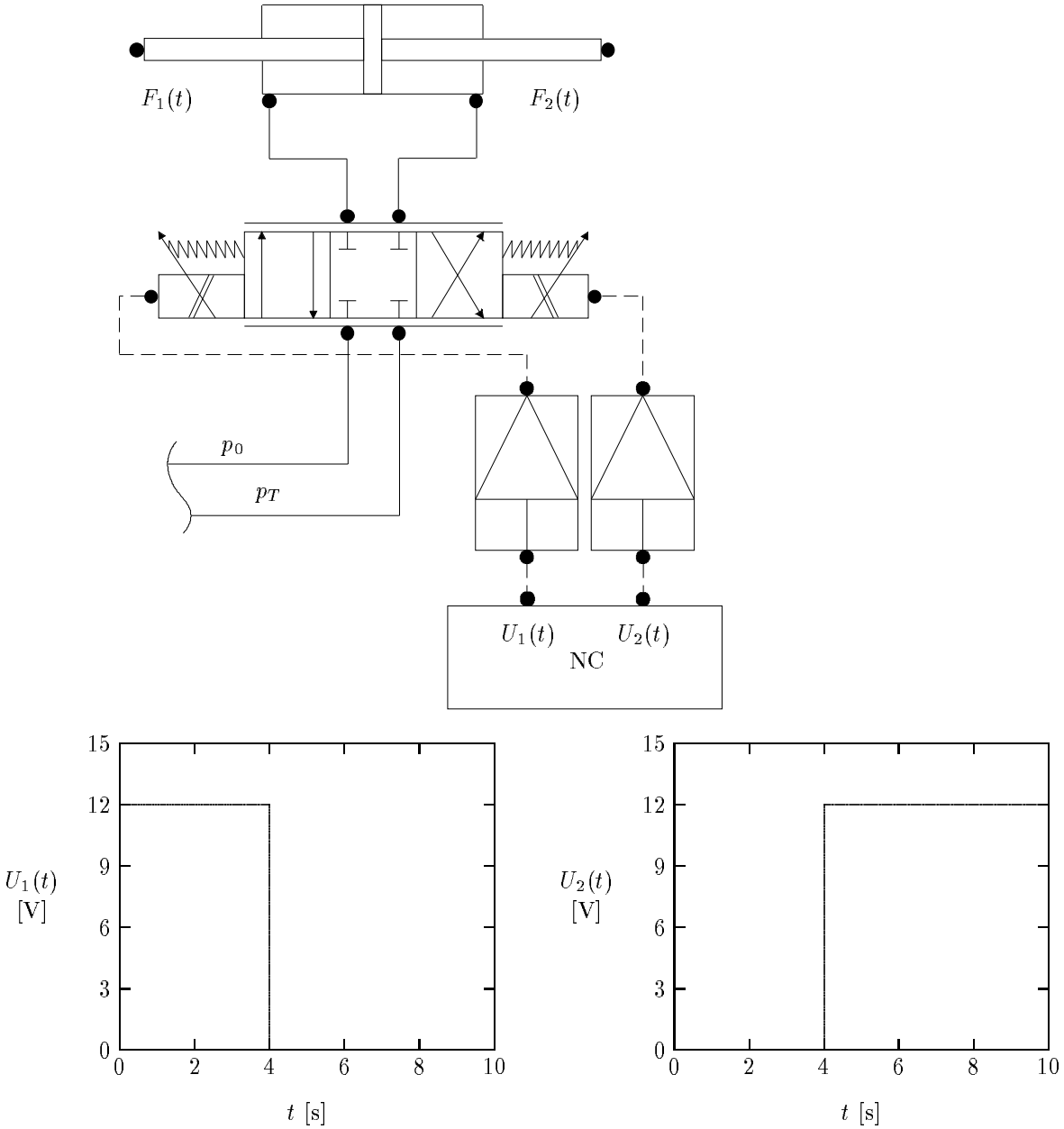


Abbildung 1.1: Eingangsgrößen der dynamischen Schaltkreissimulation

Mit den vom Anwender vorgegebenen Eingangssignalen und den Zustandswerten für den Betriebs- und Tankdruck aus der Statik ist der Anlagenteil bezüglich der Dynamik vollständig bestimmt und somit die Simulation des Teilschaltkreises durchführbar. Aufgrund der niedrigen Genauigkeitsanforderungen reicht hierzu ein einfaches, dynamisches Modell, das lediglich die Massenträgheit des Gleichgangzylinders und des 4/3 Wege-Proportionalventils berücksichtigt, aus. Die hydraulischen Leitungen und restlichen, elektrischen Komponenten des Schaltkreises werden in diesem Modell durch ihr statisches Verhalten beschrieben. Basierend auf der Modellbeschreibung liefert die Simulationsberechnung als Ergebnis die in den Diagrammen der Abbildung 1.2 darge-

stellten Kurven für die Kolbenposition und -geschwindigkeit des Gleichgangzylinders. Die Ausgangsgrößen beziehen sich auf das Gate 2 des Zylinders, d.h. den rechten Anschluß der Kolbenstange, so daß die Kolbenposition für eine ausfahrende Bewegung entsprechend mit einem negativen Vorzeichen behaftet ist. Gemäß dieser Vereinbarung kann aus den Diagrammen die Bewegung der Kolbenstange wie folgt beschrieben werden: Aus der Mittenstellung und der Ruhelage beschleunigt der Kolben unmittelbar auf seine Endgeschwindigkeit, mit der er sich konstant bis zum rechten Anschlag bewegt. In dieser Position verharrt er ca. 1 s lang, um dann wieder sprunghaft mit konstanter Endgeschwindigkeit bis zum linken Anschlag zurückzufahren. Dort angekommen, bleibt er bis zum Simulationsende in Ruhelage.

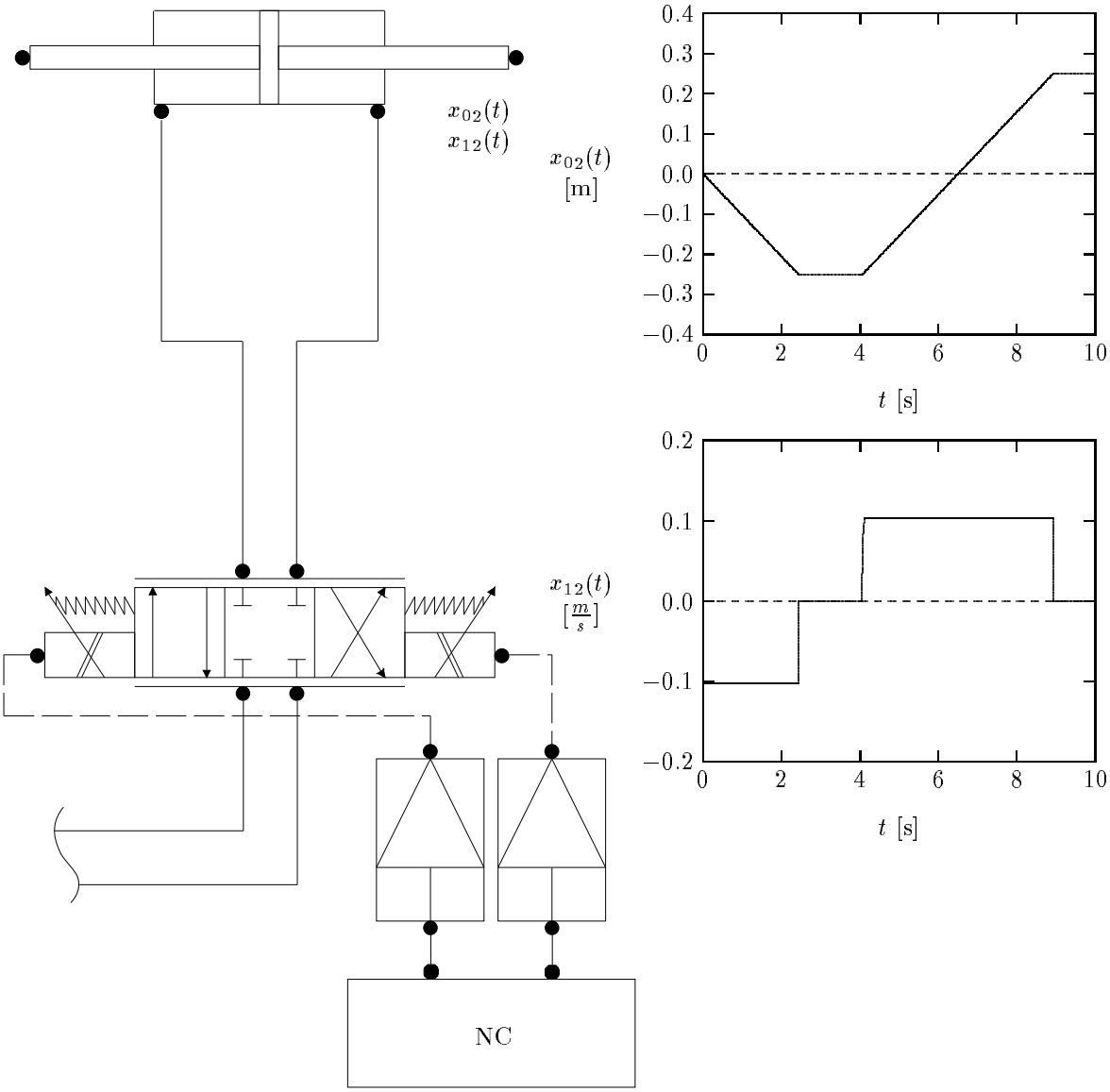


Abbildung 1.2: Ausgangsgrößen der dynamischen Schaltkreissimulation

System- und Simulationsdaten:

- *Eingangssignale*

  Betriebsdruck : $p_0(t) = 70\,\mathrm{bar}$
  Tankdruck : $p_T(t) = 1\,\mathrm{bar}$
  result. Kraft am Gleichgangzylinder : $|\,F_1(t) - F_2(t)\,| = 5500\,\mathrm{N}$
  Steuerspannungen am NC : Siehe Diagramme $U_1(t)$, $U_2(t)$ in Abbildung 1.1!

- *Gleichgangzylinder*

  – *Parameter*

    Hub : $x_{max} = 0.5\,\mathrm{m}$
    Kolbenfläche : $A_k = 900\,\mathrm{mm}^2$
    viskose Reibung : $d = 1500\,\frac{\mathrm{N\,s}}{\mathrm{m}}$
    bewegte Masse : $m = 3.0\,\mathrm{kg}$

  – *Anfangswerte*

    Kolbenposition : $x_{02}(t) = 0.0\,\mathrm{m}$
    Kolbengeschwindigkeit : $x_{12}(t) = 0.0\,\frac{\mathrm{m}}{\mathrm{s}}$

- *4/3-Wege-Proportionalventil*

  – *Parameter*

    Hub : $x_{max} = 0.0028\,\mathrm{m}$
    maximaler Ansteuerstrom : $i_{max} = 1.5\,\mathrm{A}$
    Kraftverstärkung : $K_F = 55\,\frac{\mathrm{N}}{\mathrm{A}}$
    viskose Reibung : $d = 1500\,\frac{\mathrm{N\,s}}{\mathrm{m}}$
    result. Federsteifigkeit : $c = 8100\,\frac{\mathrm{N\,s}}{\mathrm{m}}$
    bewegte Masse : $m = 0.185\,\mathrm{kg}$
    min. hydr. Widerstand : $R_{h_{min}} = 0.1\,\frac{\mathrm{bar\,min}^2}{\mathrm{l}^2}$

  – *Anfangswerte*

    Steuerschieberposition : $x_{02}(t) = -0.0028\,\mathrm{m}$
    Steuerschiebergeschwindigkeit : $x_{12}(t) = 0.0\,\frac{\mathrm{m}}{\mathrm{s}}$

- *Ventilverstärker*

  – *Parameter*

    Proportionalitätsfaktor : $k = 0.125\,\frac{1}{\Omega}$

  – *Anfangswerte*

    Nicht benötigt!

2 Konzeption der „Dynamik“ im System ^{art}*deco*

^{art}*deco* ist ein Softwaresystem, das Ingenieure bei der Inbetriebnahme hydraulischer Anlagen weitestgehend unterstützt. Als herausragendes Merkmal des Systems kann hierbei die graphische Problemformulierung angesehen werden, d.h. der Ingenieur beschreibt das Prüfungsproblem, indem er den hydraulischen Schaltkreis auf einem Zeichentableau konstruiert und anschließend die Anforderungen definiert. Hierzu wählt er einfach die graphischen Objekte aus der Komponentenbibliothek aus und ordnet sie mit Hilfe von Mausoperationen auf dem Zeichentableau an. Durch Ziehen von Verbindungslinien zwischen den Komponentenanschlüssen werden automatisch die entsprechenden Leitungstypen instanziiert. Diese graphischen Benutzeraktionen manipulieren gleichzeitig die sich hinter den Objekten verbergenden Wissensbasen, so daß während des Konstruktionsprozesses das Schaltkreismodell intern vom System aufgebaut wird. Als Anforderungen gibt der Ingenieur in diesem Modell lediglich die Werte einiger Zustandsgrößen vor und startet dann die Propagierung, d.h. die statische Funktionsprüfung (vgl. [KlBü93]). Basierend auf dieser Vorgehensweise wird in diesem Abschnitt die Konzeption der wissensbasierten Modellbildung und Simulation vorgestellt, die in Zukunft auch eine dynamische Funktionsprüfung von hydraulischen Schaltkreisen mit dem System ^{art}*deco* gestattet.

2.1 Module der Modellbildung und Simulation

Die wissensbasierte Modellbildung und Simulation stellt in ^{art}*deco* einen eigenständigen Modul dar. Seinen schematischen Aufbau veranschaulicht die Abbildung 2.1, an der im folgenden die Funktion der einzelnen Submodule und ihr Zusammenwirken näher erläutert werden soll. Die Pfeile in der Abbildung kennzeichnen hierbei die Richtung des Informationsaustausches zwischen den Modulen.

Die Schnittstelle zu ^{art}*deco* bildet in der Darstellung neben der Ein- und Ausgabeverarbeitung der graphischen Oberfläche die technische Wissensbasis, die sowohl modell- als auch regelbasierte Teile in sich vereint. Der modellbasierte Teil enthält die interne Abbildung des hydraulischen Schaltkreises. Diese Abbildung besteht im wesentlichen aus der Strukturbeschreibung sowie den lokalen Beschreibungen der Komponenten. Sie geben Aufschluß über die Parameter eines Bauteils und definieren dessen Funktionalität. Hierbei wird zwischen statischem und dynamischem Verhalten einer Komponente unterschieden. Das statische Verhalten modelliert ein nichtlineares Gleichungssystem, das die gegenseitigen Abhängigkeiten der lokalen Größen und Parameter der Komponenten wiedergibt. Das dynamische Pendant bildet ein entsprechendes, nichtlineares Differentialgleichungssystem. Im Unterschied zu der allgemeingültigen, statischen Komponentenbeschreibung bietet der dynamische Teil der Wissensbasis mehrere Modellierungen ein und derselben Komponente an, aus denen heuristisch eine geeignete Verhaltensbeschreibung gewählt wird. Die Vielzahl an Verhaltensbeschreibungen reflektiert hierbei eine mehr oder weniger genaue, dynamische Modellierung der Komponente.

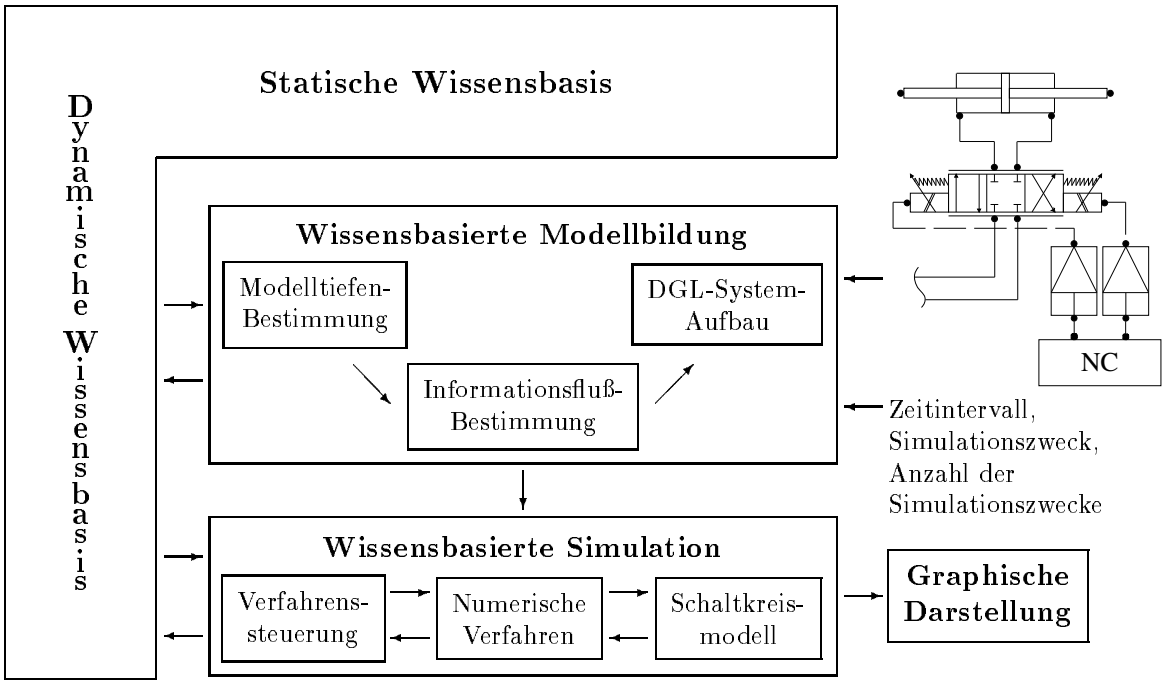


Abbildung 2.1: Module der wissensbasierten Modellbildung und Simulation

Der regelbasierte Teil der Wissensbasis enthält die Fakten und Regeln zur Selektion der dynamischen Verhaltensbeschreibung sowie zur Auswahl des optimalen, numerischen Verfahrens für die Simulationsberechnung. Auf diesen Teil der dynamischen Wissensbasis greift zunächst der Submodul der wissensbasierten Modellbildung zu. Er gliedert sich in weitere drei Submodule, die die Teilprobleme der dynamischen Modellbildung lösen. Die Teilaufgaben umfassen die heuristische Auswahl der dynamischen Modellbeschreibung einer Komponente, die Bestimmung des Informationsflusses innerhalb des Schaltkreismodells sowie die Aufstellung des entsprechenden Differentialgleichungssystems.

Als Eingabe dient dem Modellbildungsmodul der vom Benutzer vorgegebene, dynamisch zu simulierende Teilschaltkreis mit den entsprechenden Anfangsbedingungen, das Simulationszeitintervall, der Simulationszweck sowie die Anzahl der zur Verfügung stehenden Simulationszwecke. Basierend auf diesen Daten initialisiert der Submodul der Modelltiefenbestimmung zunächst die regelbasierte Wissensbasis mit dem Simulationszweck und der Gesamtanzahl der Simulationszwecke, um dann sukzessive die dynamische Verhaltensbeschreibung sämtlicher Komponenten des Schaltkreises heuristisch abzuleiten. Der Simulationszweck bestimmt hierbei die vom Benutzer geforderte Genauigkeit bezüglich der Modellbildung und kann zwischen einer sehr *groben* bis sehr *detaillierten Anlagenauslegung* variieren. Die Gesamtanzahl der Simulationszwecke legt im voraus den maximal möglichen Detaillierungsgrad der Modellbildung fest. Gleichzeitig beeinflußt der Wert die Abstufung zwischen den verschiedenen, dynamischen Verhaltensbeschreibungen einer Komponente.

Ist die Verhaltensbeschreibung sämtlicher Komponenten des Teilschaltkreises be-

stimmt, wird der Informationsfluß, der nicht mit dem Signalfluß zu verwechseln ist, innerhalb des Schaltkreises ermittelt. Der Signalfluß beschreibt die Richtungen der einzelnen Größen im Schaltkreis, wohingegen der Informationsfluß die Größen des lokalen Differentialgleichungssystems zu Ein- oder Ausgangsgrößen bestimmt. Als Eingangsgrößen werden in diesem Kontext die Größen einer Komponente aufgefaßt, deren Werte bekannt sind, d.h. direkt vom Benutzer als Konstante bzw. Funktion vorgegeben sind oder von den mit ihr verbundenen Komponenten bereitgestellt werden. Diese Größen treten ausschließlich auf der rechten Seite des Differentialgleichungssystems auf. Die Ausgangsgrößen folgen als Ergebnis aus der Integration des Differentialgleichungssystems und bilden die linken Seiten der Gleichungen, können aber auch auf der rechten Gleichungsseite auftauchen. Ihre Anfangswerte sind nach einer erfolgreichen, statischen Funktionsprüfung mit den stationären Werten belegt und somit dem System bekannt. Klassifiziert die *Informationsflußbestimmung* eine Größe einer Komponente gleichzeitig als Eingangs- und Ausgangsgröße oder kann sie eine Größe nicht einordnen, ist der Schaltkreis widersprüchlich bzw. unvollständig vom Benutzer spezifiziert worden, und der Anwender muß entweder seine Angaben ergänzen oder korrigieren. Erfolgte jedoch eine widerspruchsfreie und vollständige Informationsflußbestimmung, kann das entsprechende globale Differentialgleichungssystem des Schaltkreises aus den lokalen Systemen zusammengestellt werden. Diese Aufgabe übernimmt der in der Abbildung 2.1 als *DGL-System-Aufbau* bezeichnete Submodul. Er filtert aus den gewählten Verhaltensbeschreibungen der einzelnen Komponenten die erforderlichen Gleichungen zur Beschreibung des Gesamtsystems heraus und unifiziert deren Variablen.

Auf dem von der wissensbasierten Modellbildung generierten Systemmodell setzt der Modul der wissensbasierten Simulation auf. Er führt die Simulationsberechnung des Schaltkreises durch und speichert das Ergebnis zur weiteren Verwendung ab. Diese Aufgabe teilen sich drei Submodule, wobei ein Modul für die Verfahrenssteuerung, einer für die numerische Lösungsberechnung und ein weiterer für die Repräsentation und Verwaltung der Schaltkreisdaten verantwortlich ist.

Als Schaltzentrale kann der Modul der Verfahrenssteuerung angesehen werden, denn von ihm aus werden die zwei anderen Module aufgerufen. Zunächst wählt dieser Modul ein geeignetes, numerisches Verfahren aus der Vielzahl der zur Verfügung stehenden Verfahren aus, um dann die Simulationsberechnung zu starten. Im Laufe der Berechnung wird die Verfahrensauswahl zu bestimmten Zeitpunkten innerhalb des Simulationszeitintervalls immer wieder überprüft. Sollte sich nun ein anderes, numerisches Verfahren als geeigneter erweisen, schaltet die *Verfahrenssteuerung* auf dieses Verfahren um und setzt die Berechnung mit dem neu gewählten Verfahren fort. Ein Kriterium für die Verfahrensauswahl bildet neben den Eigenschaften des Differentialgleichungssystems sowie der an die Lösung gestellten Genauigkeitsanforderung vor allem der Aufwand des Verfahrens. Dieser sollte bezogen auf das aktuelle Problem stets minimal sein, um so möglichst kurze Simulationsrechenzeiten zu erhalten. Auf der anderen Seite darf der Aufwand zur Verfahrensauswahl nicht zu groß werden, damit die durch den Einsatz eines optimalen Verfahrens eingesparte Rechenzeit an dieser Stelle nicht wieder verloren geht. Um dieser Anforderung zu genügen, setzt die *Verfahrenssteuerung* heuristische Techniken ein. Das hierzu erforderliche Wissen ist im regelbasierten Teil der dynamischen Wissensbasis abgelegt. Die notwendigen, initialen Fakten der

Wissensbasis werden während der Simulationsberechnung an den dafür vorgesehenen Simulationszeitpunkten ermittelt.

Der Submodul der numerischen Verfahren enthält die zur Verfügung stehenden Verfahren, auf die die *Verfahrenssteuerung* zurückgreifen kann. Diese Lösungsmethoden lassen sich grob in zwei Klassen unterteilen, die die zu erwartenden Probleme der Domäne abdecken. Jede Klasse umfaßt wiederum mehrere Verfahren, die sich bezüglich ihrer „Genauigkeit“ unterscheiden. Prinzip aller Verfahren ist es, die gesuchte Lösungsfunktion, d.h. sämtliche Zustandsgrößen des Schaltkreises, im Simulationszeitintervall an diskreten Zeitpunkten mittels Näherungswerte darzustellen. Die Qualität der Näherungswerte kann hierbei durch die Wahl eines „genaueren“, numerischen Verfahrens oder durch eine Verkleinerung der Zeitabstände positiv beeinflußt werden.

Für die Speicherung der Näherungswerte ist der in der Abbildung 2.1 als *Schaltkreismodell* bezeichnete Submodul verantwortlich. Er bietet der *Verfahrenssteuerung* Zugriffsdienste auf die in Datenfiles in Form von Wertetabellen abgelegten Simulationswerte an und verwaltet die Zuordnung von Näherungswerten auf Zustandsgrößen.

Der Modul der graphischen Darstellung ist für die graphische Ausgabe des Simulationsergebnisses verantwortlich. Als Datenbasis dienen ihm die vom Schaltkreismodul bereitgestellten Wertetabellen. Diese Tabellen enthalten sämtliche von dem numerischen Verfahren notwendigerweise berechneten Näherungswerte der Lösungsfunktionen. Für eine gute Darstellungsqualität des Funktionsverlaufs ist im allgemeinen jedoch nur eine Teilmenge dieser Näherungswerte erforderlich. Die Teilmenge hängt von der Größe des Zeitintervalls und einen zu bestimmenden Skalierungsfaktor ab. Zur optimalen, graphischen Darstellung werden beide Parameter von der Ausgabetechnik berücksichtigt.

2.2 Simulationsablauf

Dieser Abschnitt skizziert grob den Ablauf einer Simulationsberechnung aus der Anwenderperspektive und soll prinzipiell die möglichen Interaktionen des Systems mit dem Benutzer vorstellen. Den Ausgangspunkt der Simulationsberechnung bildet das graphische Modell eines hydraulischen Schaltkreises. Der Schaltkreis ist bereits erfolgreich einer statischen Funktionsprüfung zum Zeitpunkt t_0 des vorgegebenen Simulationszeitintervalls $I = [t_0, t_e]$ unterzogen worden, so daß die Anfangswerte sämtlicher Zustandsgrößen bekannt sind.

Über die graphische Benutzungsschnittstelle selektiert der Benutzer nun mit der Maus den Anlagenteil, für den er eine dynamische Simulation beabsichtigt. Dann kennzeichnet er im Teilschaltkreis die Größen peripherer Komponenten, deren Werte als Eingangssignale dienen sollen. Hierzu öffnet er die entsprechende Dialogbox einer gewählten Komponente und markiert in der Liste der aufgeführten, lokalen Größen die *input-check box* der Größe, die als Eingangsgröße fungieren soll. Dieses Vorgehen setzt natürlich für zeitveränderliche Signale eine gültige Definition des Funktionsverlaufs voraus. Ist der Verlauf unbekannt, muß der Anwender zusätzlich die Funktion in einem zu öffnenden Texteditorfenster definieren und für die Komponente hinterlegen. Ansonsten wird vereinfachend der stationäre Wert der Größe aus der Statik angenommen. In

ähnlicher Weise wie für die Eingangsgrößen beschrieben, legt der Anwender auch für beliebige Komponenten des Teilschaltkreises die Ausgangsgrößen fest, indem er einfach die entsprechende *output-check box* der gewünschten Ausgangsgröße aktiviert.

Sind alle Ein- und Ausgangsgrößen des Teilschaltkreises definiert, bleibt nur noch der Simulationszweck festzulegen. Der Parameter bestimmt indirekt die Repräsentation der einzelnen Komponenten im dynamischen Gesamtmodell und beeinflußt somit auch die zu erwartende Simulationsrechenzeit. Über ein entsprechendes Pulldownmenü kann der Benutzer aus der Menge der dargestellten, möglichen Simulationszwecke den beabsichtigten Zweck auswählen. Erscheint ihm die Abstufung der dynamischen Modelle zu grob, muß er zuvor den maximalen Detaillierungsgrad erhöhen, d.h. dem System eine größere Gesamtanzahl an Simulationszwecken bereitstellen. Standardmäßig ist dieser Parameter in ^{art}*deco* mit dem kleinsten Wert, also mit 2, vorbesetzt. Zur Auswahl eines anderen Wertes aus der endlichen Menge von Alternativen dient ein entsprechendes zyklisches Menü.

Hat der Benutzer seine Auswahl bezüglich des Simulationszweckes getroffen, startet er den Modellbildungsprozeß durch Selektion des entsprechenden Items aus einem Pulldownmenü. Das System ermittelt nun zunächst für jede Komponente des selektierten Teilschaltkreises eine adäquate Modellbeschreibung, um dann aufgrund der vorgegebenen Ein- und Ausgangsgrößen den Informationsfluß innerhalb des Anlagenteils zu bestimmen. Die einzelnen Phasen der Berechnung werden hierbei jeweils vom System durch Meldungen in einem Statusfenster dokumentiert. Führen die Benutzervorgaben bezüglich der Schaltkreisgrößen zu keinen Widersprüchen und sind sämtliche Größen im Schaltkreis bekannt bzw. können durch die Berechnung des Informationsflusses bestimmt werden, baut das System das Gesamtmodell auf und startet die Simulationsberechnung. Im anderen Fall wird der Benutzer durch entsprechende Fehlermeldungen gewarnt und die Simulation abgebrochen. Nach Fehlerkorrektur kann der Anwender dann erneut die Simulationsberechnung starten.

Nach erfolgter Simulationsberechnung sind dem System die Funktionsverläufe der Ausgangsgrößen bekannt. Ihre Lösungskurve kann sich der Benutzer graphisch darstellen lassen, indem er die gewünschte Größe einer Komponente aus dem entsprechenden, kontextsensitiven Menü des Gates selektiert. Es erscheint dann ein Ausgabefenster, in dem der zeitliche Verlauf der Größe über dem Simulationszeitintervall zu sehen ist.

3 Mathematische Grundlagen

Dieser Abschnitt soll dem Leser die wesentlichen Grundlagen zur Berechnung einer dynamischen Schaltkreissimulation vermitteln. Zunächst wird die mathematische Modellierung eines beliebigen hydraulischen Schaltkreises formal eingeführt. Der Definition folgt eine Diskussion der Problemklassen, die bei der Berechnung der Lösung zu berücksichtigen sind. Abschließend werden dann die im Rahmen dieser Arbeit relevanten, numerischen Verfahren ausführlich vorgestellt.

3.1 Problemformulierung

Ein hydraulischer Schaltkreis ist durch die Parameter der dynamischen Systembeschreibung eindeutig definiert. Eine Änderung dieser zeitlichen Konstanten stellt somit immer eine Systemänderung dar. In diesem Sinn wird z.B. die Kolbenquerschnittsfläche des Gleichgangzylinders A_k als ein Parameter aufgefaßt.

Die Größen des hydraulischen Schaltkreises hingegen beschreiben zu jedem Zeitpunkt den Systemzustand. Eine Modifikation ihrer Werte kann nur über der Zeit erfolgen und überführt das Gesamtsystem in einen neuen Zustand, ändert das System jedoch nicht. Neben den bekannten Eingangsgrößen, die die Eingangssignale darstellen, treten in der Systembeschreibung die Ausgangsgrößen als Unbekannte auf. Ihr zeitlicher Verlauf kann bei einem vollständig definierten System aus der Systembeschreibung und der Kenntnis sämtlicher Eingangsgrößen bestimmt werden.

Die Systembeschreibung, als funktionaler Zusammenhang zwischen Ein- und Ausgangsgrößen, bildet im allgemeinen ein Differentialgleichungssystem n-ter Ordnung mit $n \geq 2$. Ausgehend von einem Zeitpunkt t_0, zu dem sich das Gesamtsystem in einem bekannten Zustand befindet, was die Kenntnis der Werte aller nichtredundanter Größen voraussetzt, liefert die zu berechnende Lösung des Differentialgleichungssystems den zeitlichen Verlauf sämtlicher Ausgangsgrößen bis zu einem vorgegebenen Zeitpunkt t_e. Das Differentialgleichungssystem der Systembeschreibung hängt somit nur von einer Veränderlichen, der Zeit t, ab. Solche Gleichungen bezeichnet man auch als gewöhnliche Differentialgleichungen. Zusammen mit dem Anfangszustand formuliert es ein Anfangswertproblem, das mit numerischen Verfahren gelöst wird. Die Entscheidung für den Einsatz numerischer Verfahren liegt in den domänenspezifischen Differentialgleichungen begründet. Ihre Lösung muß nicht immer in geschlossener Form existieren, so daß numerische Lösungsverfahren unumgänglich sind.

Wie in [For84] gezeigt, läßt sich jede Differentialgleichung n-ter Ordnung eindeutig auf ein System von n Differentialgleichungen 1-ter Ordnung zurückführen, so daß im weiteren nur gewöhnliche Differentialgleichungen 1-ter Ordnung betrachtet werden. Die formale Beschreibung eines Differentialgleichungssystems 1-ter Ordnung zeigt die Definition 3.1. In dieser Definition stellt der Vektor $\boldsymbol{z}(t)$ den zu berechnenden Funktionsverlauf der Ausgangsgrößen über dem Zeitintervall $I = [t_0, t_e]$ dar. Die Systembeschreibung verbirgt sich hinter der Funktion $\boldsymbol{f}$, in die die Eingangsgrößen als Funktionsparameter einfließen. Diese Größen sind jedoch in der mathematischen Notation nicht mehr explizit aufgeführt.

Definition 3.1 (Differentialgleichungssystem 1-ter Ordnung) *Sei $n \in \mathbb{N}$, $I \subset \mathbb{R}$ ein Intervall, $G \subset \mathbb{R} \times \mathbb{R}^n$ ein Gebiet und $\boldsymbol{z} : I \to \mathbb{R}^n$ mit $t \mapsto \boldsymbol{z}(t)$ eine auf dem Intervall I differenzierbare, vektorwertige Funktion, deren Graph in G enthalten ist, d.h. $\{(t, \boldsymbol{z}(t)) \in I \times \mathbb{R}^n\}$. Sei ferner $\boldsymbol{f} : G \to \mathbb{R}^n$ mit $(t, \boldsymbol{z}(t)) \mapsto \boldsymbol{f}(t, \boldsymbol{z}(t))$ eine über dem Gebiet G stetige, vektorwertige Funktion. Dann beschreibt die Gleichung*

$$\dot{\boldsymbol{z}}(t) = \boldsymbol{f}(t, \boldsymbol{z}(t)), \tag{3.1}$$

ein System von n Differentialgleichungen 1-ter Ordnung. Falls für alle $t \in I$ die Gleichung 3.1 gilt, ist die Funktion $\boldsymbol{z}(t)$ eine Lösung des Differentialgleichungssystems 1-ter Ordnung. Ausführlich geschrieben, sieht Gleichung 3.1 wie folgt aus:

$$\begin{pmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \\ \vdots \\ \dot{z}_n(t) \end{pmatrix} = \begin{pmatrix} f_1(t, z_1(t), z_2(t), \ldots, z_n(t)) \\ f_2(t, z_1(t), z_2(t), \ldots, z_n(t)) \\ \vdots \\ f_n(t, z_1(t), z_2(t), \ldots, z_n(t)) \end{pmatrix} \tag{3.2}$$

Basierend auf dieser Definition kann nun ein Anfangswertproblem, wie in Definition 3.2 formuliert, beschrieben werden. Die Anfangsbedingung $\boldsymbol{z}(t_0) = \boldsymbol{z}_0$ entspricht hierbei dem Anfangszustand des hydraulischen Schaltkreises.

Definition 3.2 (Anfangswertproblem) *Sei $n \in \mathbb{N}$, $I \subset \mathbb{R}$ ein Intervall, $G \subset \mathbb{R} \times \mathbb{R}^n$ ein Gebiet und $\boldsymbol{f} : G \to \mathbb{R}^n$ eine differenzierbare, vektorwertige Funktion. Dann beschreibt das Differentialgleichungssystem 1-ter Ordnung zusammen mit der Anfangsbedingung*

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= \boldsymbol{f}(t, \boldsymbol{z}(t)) && \text{für alle } t \in I \text{ und} \\ \boldsymbol{z}(t_0) &= \boldsymbol{z}_0 && \text{für ein } t_0 \in I \end{aligned} \tag{3.3}$$

ein Anfangswertproblem, dessen eindeutige Lösung die Funktion $\boldsymbol{z}$ ist.

Abschließend soll die mathematische Modellierung der dynamischen Verhaltensbeschreibung eines hydraulischen Schaltkreises am Beispiel einer einzelnen Komponente, dem Gleichgangzylinder, demonstriert werden. Das Beispiel beschränkt sich hierbei auf eine vereinfachte Darstellung der Bewegungsgleichung des Kolbens mit den Ausgangsgrößen Position $x_{02}(t)$ und Geschwindigkeit $x_{12}(t)$. Vereinfachend deshalb, weil in der Bewegungsgleichung die Kolbenhubbegrenzung unberücksichtigt bleibt. Die vollständige Verhaltensbeschreibung des Gleichgangzylinders ist dem Anhang unter Abschnitt A.2.1 zu entnehmen. Er enthält auch eine Erläuterung der Parameter. Die Bedeutung der technischen Größen geht aus der Nomenklatur hervor und wird hier als bekannt vorausgesetzt.

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= \begin{pmatrix} \dot{x}_{02}(t) \\ \dot{x}_{12}(t) \end{pmatrix} = \begin{pmatrix} x_{12}(t) \\ -\frac{A_k/10\,(p_3(t)-p_4(t))-F_2(t)+F_1(t)+d\,x_{12}(t)}{m} \end{pmatrix} = \boldsymbol{f}(t, \boldsymbol{z}(t)) \\ \boldsymbol{z}(t_0) &= \begin{pmatrix} x_{02}(t_0) \\ x_{12}(t_0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{aligned} \tag{3.4}$$

Die Anfangsbedingung des Anfangswertproblems 3.4 besagt, daß sich der Kolben des Gleichgangzylinders in Mittenstellung befindet und seine anfängliche Geschwindigkeit

Null beträgt. Ausgehend von diesem Zustand kann in Abhängigkeit der Eingangsgrößen, $p_3(t)$, $p_4(t)$, $F_1(t)$ und $F_2(t)$ sowie aus der Bewegungsgleichung folgenden Funktion $\boldsymbol{f}$ die Kolbenbewegung des Gleichgangzylinders simuliert werden.

3.2 Problemklassen

Im vorangehenden Abschnitt wurde das Problem der Simulationsberechnung formal definiert, ohne jedoch Aussagen über die Existenz und Eindeutigkeit der Lösung eines Anfangswertproblems zu machen. Ein hinreichendes Kriterium für die Existenz und Eindeutigkeit stellt die *Lipschitz-Bedingung* dar. Erfüllt das Differentialgleichungssystem des Anfangswertproblems 3.2 diese Bedingung, können, wie in [For84] gezeigt, aus dem Existenzsatz von Picard-Lindelöf und dem Eindeutigkeitssatz die entsprechenden Aussagen gefolgert werden. Die Definition 3.3 führt den Begriff der Lipschitz-Bedingung formal ein.

Definition 3.3 (Lipschitz-Bedingung) *Sei $n \in \mathbb{N}$, $I = [t_0, t_e] \subset \mathbb{R}$ ein Intervall, $G \subset I \times \mathbb{R}^n$ ein Gebiet und $\boldsymbol{f} : G \to \mathbb{R}^n$ mit $(t, \boldsymbol{z}(t)) \mapsto \boldsymbol{f}(t, \boldsymbol{z}(t))$ eine stetige Funktion, dann genügt $\boldsymbol{f}$ auf G einer Lipschitz-Bedingung, wenn eine Zahl $L \in \mathbb{R}$ existiert, so daß*

$$\| \boldsymbol{f}(t, \boldsymbol{z_1}(t)) - \boldsymbol{f}(t, \boldsymbol{z_2}(t)) \| \leq L \| \boldsymbol{z_1}(t) - \boldsymbol{z_2}(t) \| \quad \forall (t, \boldsymbol{z_1}(t)), (t, \boldsymbol{z_2}(t)) \in G \tag{3.5}$$

gilt. Die Funktion $\boldsymbol{f}$ heißt dann auch lipschitz-beschränkt.

In Bezug auf ein Anfangswertproblem fordert die Lipschitz-Bedingung, anschaulich interpretiert, daß zu jedem Zeitpunkt des betrachteten Zeitintervalls die Änderung der Steigung $\boldsymbol{f}$, d.h. die rechte Seite des Differentialgleichungssystems für zwei beliebige Lösungskurven $\boldsymbol{z}_1$, $\boldsymbol{z}_2$ beschränkt ist. Sie darf somit keine Unendlichkeitsstellen aufweisen, denn nur in diesem Fall stellt die Lösung eine Funktion dar.

Ein Kriterium zur Charakterisierung von lipschitz-beschränkten Anfangswertproblemen liefert der Stabilitätsbegriff, der in Definition 3.4 vorgestellt wird. Die Eigenschaft eines Differentialgleichungssystems *stabil* zu sein definiert die Subklasse der lipschitz-beschränkten Anfangswertprobleme, die sich numerisch berechnen lassen. Deshalb muß vor der Anwendung eines numerischen Verfahrens die Stabilitätsbedingung 3.7 überprft werden.

Definition 3.4 (Stabiles Anfangswertproblem) *Sei $t^* \in I$ ein beliebiger Punkt aus dem Intervall $I \subset \mathbb{R}$ und*

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= \boldsymbol{f}(t, \boldsymbol{z}(t)) && \text{für alle } t \in I \text{ und} \\ \boldsymbol{z}(t_0) &= \boldsymbol{z}_0 && \text{für ein } t_0 \in I \end{aligned} \tag{3.6}$$

ein lipschitz-beschränktes Anfangswertproblem gemäß der Definition 3.2. Das Anfangswertproblem 3.6 ist in t^* stabil, genau dann wenn für alle Eigenwerte $\lambda_i(t^*, z_i(t^*))$ der Jacobi-Matrix des Differentialgleichungssystems die Stabilitätsbedingung

$$\Re(\lambda_i(t^*, z_i(t^*))) < 0 \text{ für } i = 1, 2, \ldots, n \tag{3.7}$$

erfüllt ist. Das Anfangswertproblem heißt stabil, wenn für alle $t^* \in I$ die Ungleichung 3.7 gilt.

Verbal beschrieben, sorgt die Stabilitätsbedingung dafür, daß sich ein geringfügiger Fehler in der Anfangsbedingung, der z.B. aufgrund von numerischen Rundungs- oder Verfahrensfehlern entstanden ist, nicht monoton wachsend fortpflanzt, sondern abklingenden Charakter besitzt. Faßt man das Differentialgleichungssystem des betrachteten Anfangswertproblems zusammen mit der fehlerbehafteten Anfangsbedingung als ein eigenständiges, zweites Anfangswertproblem auf, würde das bedeuten, daß sich der Abstand der beiden Lösungskurven mit zunehmender Zeit verringert.

Ein einfaches Beispiel soll das Versagen der numerischen Verfahren bei der Lösungsberechnung von instabilen, lipschitz-beschränkten Anfangswertproblemen veranschaulichen. In [Schw93] wird eine Klasse von Anfangswertproblemen vorgestellt, deren Lösungsmenge in geschlossener Form darstellbar ist. Sie besitzen die in 3.8 gezeigte Struktur und weisen für $\mu > 0$ instabiles Verhalten auf.

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= \mu\,(z(t) - g(t)) + \dot{g}(t) && \text{für alle } t \in I \subset \mathbb{R} \text{ und} \\ \boldsymbol{z}(t_0) &= z_0 && \text{für ein } t_0 \in I \end{aligned} \tag{3.8}$$

Die analytische Lösung von 3.8 kann leicht über den allgemeinen Lösungsansatz $z(t) = g(t)\,\exp(\mu\,t)$ gewonnen werden. Sie lautet unter Berücksichtigung der Anfangsbedingung

$$z(t) = \mu\,(z_0 - g(t_0))\exp(\mu\,(t - t_0)) + g(t). \tag{3.9}$$

Um die analytische mit der numerischen Lösung direkt vergleichen zu können, werden hier die Parameter μ und $g(t)$ auf die Werte $\mu = 10$ und $g(t) = \frac{t^2}{1+t^2}$ gesetzt, so daß sich aus der allgemeinen Form 3.8 das konkrete Anfangswertproblem

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= 10\,(z(t) - \tfrac{t^2}{1+t^2}) + \tfrac{2\,t}{(1+t^2)^2} && \text{für alle } t \in I \subset \mathbb{R} \\ \boldsymbol{z}(t_0) &= 0 && \text{für ein } t_0 \in I \end{aligned} \tag{3.10}$$

ergibt. Den Verlauf der beiden Lösungskurven über dem Zeitintervall $I = [0, 2.2]$ illustriert die Abbildung 3.1. Die numerisch mit einem expliziten Runge-Kutta-Verfahren 4-ter Ordnung[2] berechnete Lösungskurve driftet in der Umgebung von $t = 1.75$ ins Negative ab. Der Kurvenverlauf ist sowohl vom verwendeten Verfahren als auch von der vorgegebenen Rechengenauigkeit unabhängig. Diese Parameter beeinflußen lediglich den Schnittpunkt mit der Zeitachse.

Eine weitere Subklasse der lipschitz-beschränkten Anfangswertprobleme stellen die sogenannten *steifen* Probleme dar. Unter ihnen kommt den stabilen Anfangswertproblemen bei der numerischen Berechnung eine besondere Bedeutung zu. Ihre aufwandsgerechte Lösung erfordert speziell geeignete Verfahren, denn die Lösungskurven weisen ein stark unterschiedlich oszillierendes Verhalten auf, d.h. die Lösung besteht sowohl aus hochfrequenten als auch aus Anteilen wesentlich niedriger Frequenzen. Nach dem Shannonschen Abtasttheorem ist die Sampling-Rate bzw. die Schrittweite des numerischen Verfahrens durch die höchsten, spektralen Anteile in der Lösungsfunktion bestimmt.

[2] Die Beschreibung der numerischen Verfahren erfolgt später im Abschnitt 3.3.

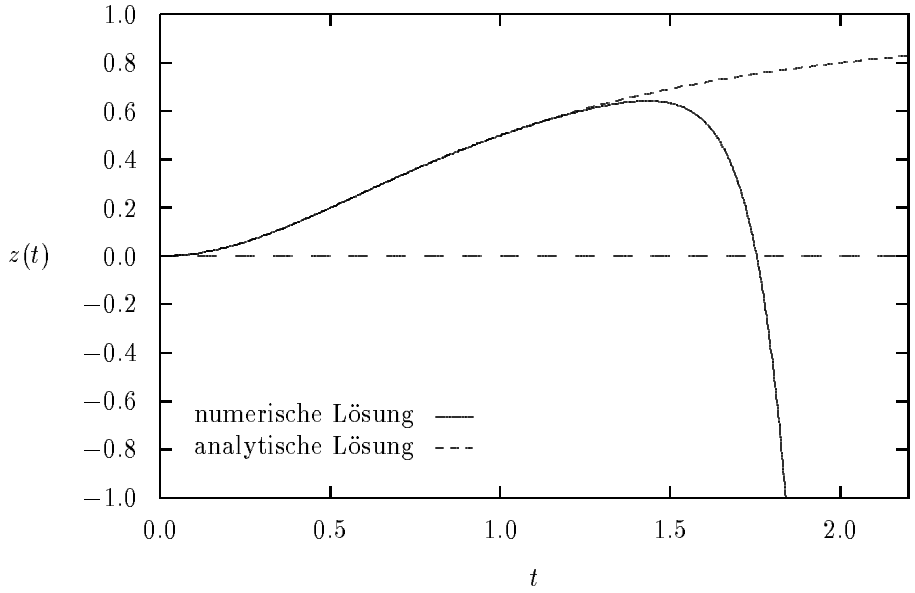


Abbildung 3.1: Lösungen eines instabilen Anfangswertproblems

Sie muß kleiner als die halbe maximal auftretende Schwingungsdauer sein, wobei in der Praxis aufgrund vorgegebener Fehlerschranken und der numerischen Stabilität[3] meist eine wesentlich geringere Schrittweite gewählt wird.

Nicht immer ist jedoch eine sehr genaue Wiedergabe der hochfrequenten Anteile in der Lösungsfunktion gewünscht bzw. erforderlich, so daß mit einer den niedrigen Schwingungsdauern konforme Schrittweite gerechnet werden könnte, vorausgesetzt die Fehler für die hochfrequenten Lösungskurven bleiben hierbei lokal beschränkt. Diese Eigenschaft besitzen die *impliziten* Runge-Kutta-Verfahren, die hier zur Berechnung steifer, lipschitz-beschränkter Anfangswertprobleme herangezogen werden.

Die formale Beschreibung der steifen Anfangswertprobleme findet der Leser in der Definition 3.5. Der Steifheitsgrad κ stellt hierin ein Maß zur Beurteilung der Steifheit eines Systems dar, wobei der Grenzwert, ab wann ein System als steif einzustufen ist, u.a. auch von der betrachteten Domäne abhängt. Die Ermittlung des Steifheitsgrades setzt die Kenntnis der Eigenwerte $\lambda_i(t, z_i(t))$ der Jacobi-Matrix voraus. Ihre Werte können über das betrachtete Zeitintervall stark variieren, so daß immer nur eine lokale Aussage bezüglich der Steifheit eines Anfangswertproblems gemacht werden kann.

Definition 3.5 (Steifes Anfangswertproblem) *Sei das lipschitz-beschränkte Anfangswertproblem*

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= \boldsymbol{f}(t, \boldsymbol{z}(t)) && \textit{für alle } t \in I \textit{ und} \\ \boldsymbol{z}(t_0) &= \boldsymbol{z}_0 && \textit{für ein } t_0 \in I \end{aligned} \tag{3.11}$$

[3]Der Begriff der numerischen Stabilität wird im Abschnitt 3.3.4 noch ausführlich vorgestellt.

gemäß der Definition 3.2 gegeben. Das Anfangswertproblem 3.11 heißt steif, wenn für den Steifheitsgrad κ mit

$$\kappa = \frac{\max\limits_{t \in I} |\, \lambda_i(t, z_i(t)) \,|}{\min\limits_{t \in I} |\, \lambda_i(t, z_i(t)) \,|} \gg 1 \text{ für } i = 1, 2, \ldots, n \tag{3.12}$$

gilt, wobei $\lambda_i(t, z_i(t))$ die Eigenwerte der Jacobi-Matrix des Differentialgleichungssystems bezeichnen.

Das folgende Beispiel soll die Auswirkungen einer falschen Verfahrensauswahl bei der numerischen Berechnung eines steifen Anfangswertproblems verdeutlichen. Als Testaufgabe dient das Anfangswertproblem 3.13, dessen Lösungsfunktion sowohl mit einem speziell geeigneten, d.h impliziten, als auch ungeeigneten, d.h. expliziten Verfahren bestimmt wird.

$$\begin{aligned} \begin{pmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \end{pmatrix} &= \begin{pmatrix} -z_1(t) \\ -100\, z_2(t) \end{pmatrix} \quad \text{für alle } t \in [0,1] \\ \begin{pmatrix} \dot{z}_1(0) \\ \dot{z}_2(0) \end{pmatrix} &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{aligned} \tag{3.13}$$

Als Lösung ergeben sich zwei um den Zeitfaktor 100 unterscheidende, abklingende exp-Funktionen. Da die beiden Differentialgleichungen des Anfangswertproblems nicht miteinander gekoppelt sind, wäre eine getrennte Berechnung der exp-Funktionen möglich. Sie sollen hier aber, wie im allgemeinen Fall und insbesondere für hydraulische Schaltkreise, als gekoppelt angesehen und simultan gelöst werden. Bei Verwendung eines expliziten Verfahrens zur Lösungsberechnung ist die maximale Schrittweite durch die kleinste Zeitkonstante des Systems, also durch die Zeitkonstante $T_2 = \frac{1}{100}$ der zweiten Differentialgleichung, festgelegt. D.h. selbst wenn nur der genaue Verlauf der langsam abklingenden exp-Funktion der ersten Differentialgleichung interessieren würde, müßte diese aus numerischen Stabilitätsgründen mit der vom System vorgegebenen Schrittweite berechnet werden. Die Folge wäre ein vermeidbar überhöhter Rechenaufwand. Auf der anderen Seite hätte eine Vergrößerung der Schrittweite über das erlaubte Maß hinaus ein Ausbrechen des Verfahrens zur Konsequenz. Im Gegensatz hierzu kann bei Verwendung eines impliziten Verfahrens sehrwohl mit einer zur größeren Zeitkonstante $T_1 = 1$ konformen Schrittweite gerechnet werden, ohne daß der hierbei zugelassene Fehler zu einer totalen Verfälschung der Lösung führt. Die schnell abklingende exp-Funktion wird zwar nur sehr ungenau wiedergegeben, jedoch bleibt diese Ungenauigkeit in jedem Schritt lokal beschränkt. Die Abbildung 3.2 zeigt die mit der konstanten Schrittweite $h = 0.03$[4] berechneten Lösungskurven des expliziten Runge-Kutta-Verfahrens 4. Ordnung. Die gewählte Schrittweite liegt außerhalb des numerischen Stabilitätsbereiches des Verfahrens, so daß die Lösungskurve der schnell abklingenden exp-Funktion unmittelbar ausbricht. Sie ist mit dem eigentlichen Verlauf der zu erwartenden Lösung nicht mehr zu vergleichen. Die schwach gedämpfte exp-Funktion wird hingegen korrekt wiedergegeben. Wären die beiden Differentialgleichungen jedoch

[4] Diese Schrittweite wird anstelle von $h = 0.1$ verwendet, um einen größeren Abbildungsmaßstab zu ermöglichen. Auf den qualitativen Verlauf der Lösungskurven hat sie keine Auswirkungen.

miteinander gekoppelt, würde auch ihr Verlauf durch die große Steigung der anderen Lösungskurve verfälscht. Die Abbildung 3.3 stellt die mit der konstanten Schrittweite $h = 0.1$ berechneten Lösungskurven eines impliziten Runge-Kutta-Verfahrens 4. Ordnung dar. Der Effekt der lokalen Fehlerbeschränkung ist für die stark gedämpfte exp-Funktion deutlich zu erkennen. Beide Lösungen werden im Rahmen einer gewissen Genauigkeit korrekt wiedergegeben.

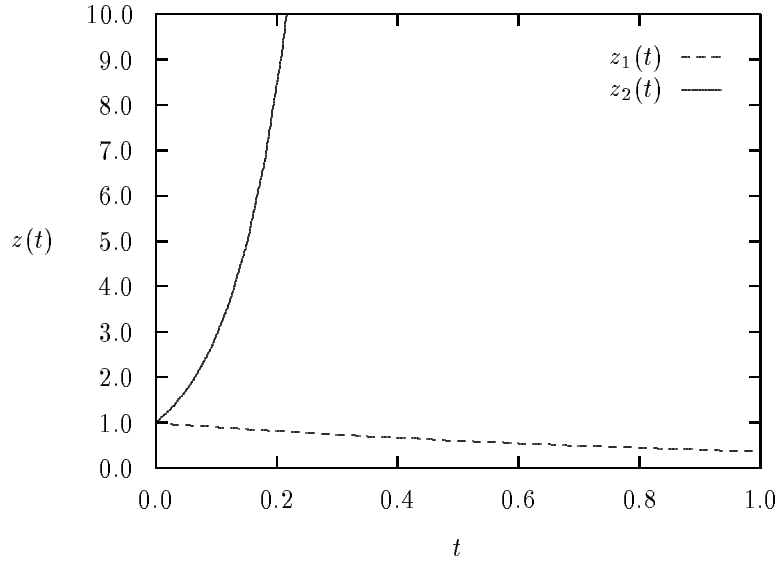


Abbildung 3.2: Steifes Anfangswertproblem mit explizitem Verfahren gerechnet

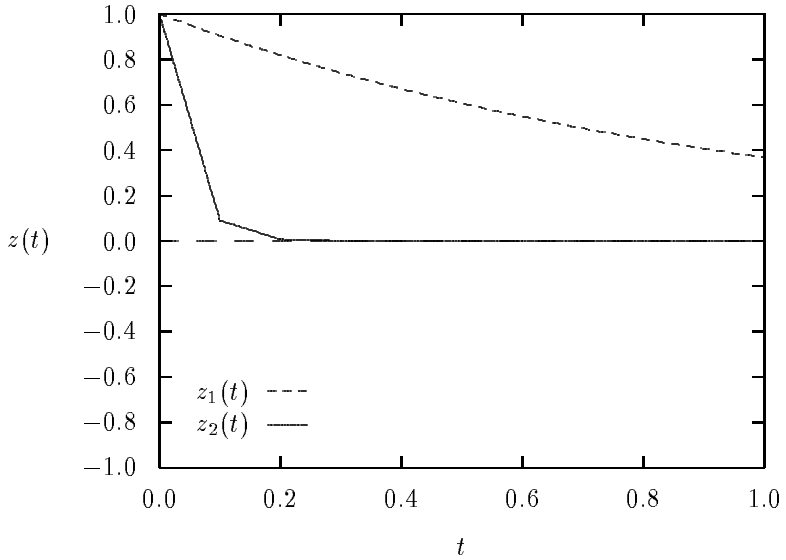


Abbildung 3.3: Steifes Anfangswertproblem mit implizitem Verfahren gerechnet

Die verschiedenen, vorgestellten Problemklassen von Anfangswertproblemen verdeutlichen, daß für eine angemessene, numerische Behandlung eine genaue Kenntnis der Problemklasse erforderlich ist. Die einfachen Beispiele haben gezeigt, daß alle Klassen in der Praxis auftreten können. Insbesondere muß bei der numerischen Simulation von hydraulischen Schaltkreisen verstärkt mit steifen Anfangswertproblemen gerechnet werden, denn diese liegen meistens dann vor, wenn kurze Steuersignale auf langsame, d.h. träge mechanische Komponenten treffen, wie das z.B. für Proportionalventile der Fall sein kann. Instabile Anfangswertprobleme können numerisch nicht gelöst werden und sind in der Praxis wahrscheinlich weniger häufig anzufinden, da im allgemeinen technische Systeme betrachtet werden, die ein gedämpftes Verhalten zeigen. Dennoch kann ein System bei entsprechender Systemparameterauswahl instabiles Verhalten aufweisen. Die Graphik 3.4 soll abschließend die diskutierten Problemklassen lösbarer Anfangswertprobleme nochmals veranschaulichen. Die schraffierten Flächen stellen hierin die Menge von Anfangswertproblemen dar, die auch numerisch gelöst werden können. Die Schnittmenge der steifen mit den stabilen, lipschitz-beschränkten Anfangswertproblemen erfordert eine besondere numerische Behandlung. Ihre Lösung ist mit einem vertretbaren Rechenaufwand nur mit impliziten Methoden[5] möglich.

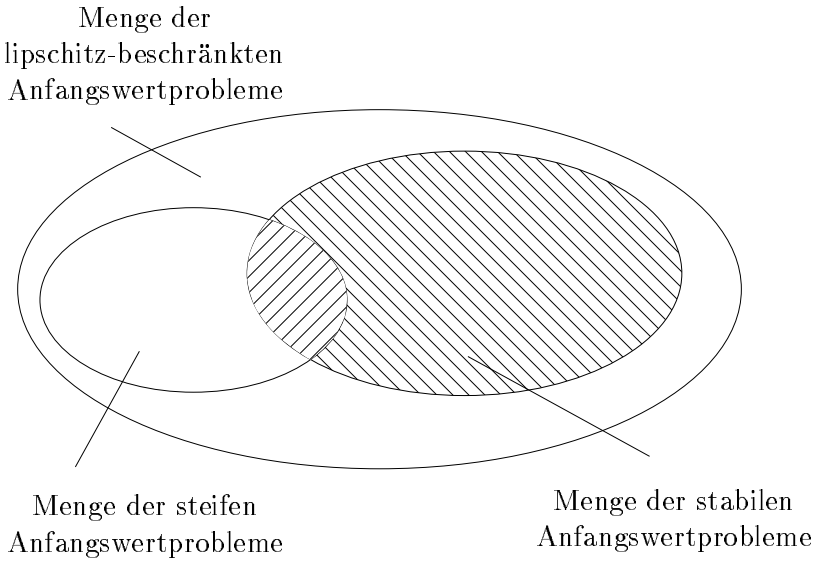


Abbildung 3.4: Klassen von lösbaren Anfangswertproblemen

3.3 Numerische Problemlösung

Nach der mathematischen Problembeschreibung und der Gliederung der lipschitz-beschränkten Anfangswertprobleme in Problemklassen werden nun die Methoden zur Problemlösung, d.h. die numerischen Verfahren zur Lösung von Anfangswertproblemen vorgestellt. Zunächst erfolgt eine Erläuterung des prinzipiellen Vorgehens bei der

[5]Neben den impliziten Methoden gibt es noch weitere Verfahren, die sich zur Lösung steifer Anfangswertprobleme eignen. Sie finden aber hier keine Berücksichtigung.

Lösungsberechnung anhand eines einfachen Verfahrens. Ihr schließt sich eine allgemeine Beschreibung der Runge-Kutta-Verfahren an. Eine Motivation des allgemeinen Runge-Kutta-Ansatzes findet der Leser im Folgeabschnitt. Abschließend werden dann die wesentlichen Aspekte der Schrittweitensteuerung und Verfahrensauswahl beleuchtet.

3.3.1 Prinzipielles Vorgehen

Ausgangspunkt der numerischen Methoden bildet ein lipschitz-beschränktes Anfangswertproblem gemäß der Definition 3.2, für das die Lösungsfunktion im vorgegebenen Zeitintervall $I = [t_0, t_e]$ gesucht ist. Zur Bestimmung dieser Funktion zerlegen die Verfahren das Zeitintervall I in ein Gitter $t_0 < t_1 < t_2 < \ldots < t_n = t_e$ mit der lokalen Schrittweite $h_s = t_{s+1} - t_s$ für $s = 0, 1, \ldots, n-1$ und berechnen an den diskreten Gitterpunkten Näherungswerte $\boldsymbol{z}_s \approx \boldsymbol{z}(t_s)$ der Lösungsfunktion. Die Näherungswerte ergeben sich durch die sukzessive Integration des Differentialgleichungssystems über die durch die Gitterpunkte bestimmten Teilintervalle. Die Gleichung 3.14 zeigt die Integration des Differentialgleichungssystems über ein Teilintervall. Sie kann in die Beziehung 3.15 umgeformt werden.

$$\int_{t_s}^{t_{s+1}} \dot{\boldsymbol{z}}(t)\, dt = \int_{t_s}^{t_{s+1}} \boldsymbol{f}(t, \boldsymbol{z}(t))\, dt \tag{3.14}$$

$$\Rightarrow \boldsymbol{z}(t_{s+1}) = \boldsymbol{z}(t_s) + \int_{t_s}^{t_{s+1}} \boldsymbol{f}(t, \boldsymbol{z}(t))\, dt \tag{3.15}$$

Prinzipiell unterscheiden sich die numerischen Verfahren nur darin, wie sie das Integral in 3.15 näherungsweise berechnen. Sie lassen sich grob in die folgenden drei Klassen gliedern, die kurz charakterisiert werden.

- *Einschrittverfahren:* Sie verwenden, wie ihr Name schon vermuten läßt, zur näherungsweisen Berechnung des Funktionswertes $\boldsymbol{z}(t_{s+1})$ nur einen vorangehenden Funktionswert $\boldsymbol{z}(t_s)$. Zu Beginn der Berechnung ist dieser Wert durch die Anfangsbedingung vorgegeben, so daß die Verfahren unmittelbar ohne zusätzlichen Rechenaufwand auf das Anfangswertproblem anwendbar sind. Eine Steuerung der Integrationsschrittweite läßt sich hierbei leicht realisieren.

- *Mehrschrittverfahren:* Sie verwenden $r + 1$, $r \geq 1$ vorangehende Werte $\boldsymbol{z}_{s-r}$, $\boldsymbol{z}_{s-r+1}, \ldots, \boldsymbol{z}_{s-1}, \boldsymbol{z}_s$ zur Berechnung des Näherungswertes $\boldsymbol{z}_{s+1}$ und erfordern zu Beginn der Integration zunächst die vollständige Berechnung dieses Anlaufstücks. In den Folgeschritten sind dann nur noch zwei bis drei Funktionsauswertungen der rechten Seite des Differentialgleichungssystems notwendig. Dieser im Vergleich zu anderen Verfahren geringere Rechenaufwand geht aber verloren, wenn die Schrittweite gesteuert werden soll bzw. muß, denn dann hat jede Schrittweitenänderung eine Neuberechnung des Anlaufstücks zur Folge.

- *Extrapolationsverfahren:* Sie basieren auf einen zum Integrationsverfahren von Romberg analogen Algorithmus zur Lösung von Anfangswertproblemen und sind selbststartend, d.h. sie benötigen keine Anlaufphase. Wie bei den Einschrittverfahren erlaubt das Rechenschema der Extrapolationsverfahren eine leicht einzubettende Schrittweitensteuerung. Der Rechenaufwand pro Integrationsschritt ist jedoch oft erheblich.

Mehrschritt- und Extrapolationsverfahren zeigen sich in Bezug auf die zu erwartenden Anfangswertprobleme der Domäne weniger gut geeignet als Einschrittverfahren. Gegen die Mehrschrittverfahren spricht vor allem die fehlende Möglichkeit einer effizienten Schrittweitensteuerung, die gerade in Bezug auf größere Anfangswertprobleme erforderlich ist, um die Rechenzeit in einem erträglichen Rahmen zu halten. Die Extrapolationsverfahren spielen ihre Vorteile gegenüber den anderen Verfahrensklassen erst bei großen, komplizierten Differentialgleichungssystemen aus, wenn hohe Genauigkeitsanforderungen zu erfüllen sind. Bei den vielfach linearen Differentialgleichungen der hydraulischen Schaltkreismodellierung und der im allgemeinen niedrigen Genauigkeitsanforderung an den Simulationsdaten sind deshalb die Einschritt- den Extrapolationsverfahren vorzuziehen, so daß im folgenden ausschließlich diese Klasse behandelt wird.

Einfachster Vertreter der Einschrittverfahren ist das Verfahren von *Euler-Cauchy.* Bei dieser Methode wird das Integral in 3.15 näherungsweise mit der Rechteckformel $h_s * \boldsymbol{f}(t_s, \boldsymbol{z}_s)$ berechnet. Da h_s vorgegeben ist und $\boldsymbol{f}(t_s, \boldsymbol{z}_s)$ durch Einsetzen des bekannten Funktionswertes $\boldsymbol{z}_s$ bestimmt werden kann, erhält man aus 3.15 für den zu berechnenden Funktionswert $\boldsymbol{z}_{s+1}$ die Rechenvorschrift:

$$\boldsymbol{z}_{s+1} = \boldsymbol{z}_s + h_s\, \boldsymbol{f}(t_s, \boldsymbol{z}_s) \tag{3.16}$$

Die sukzessive Anwendung der Formel 3.16 liefert sämtliche Näherungswerte an den vorgegebenen Gitterpunkten des Intervalls. Ein Vergleich dieser Werte mit der Taylor-Entwicklung 3.17 des entsprechenden Funktionswertes um den Punkt $\boldsymbol{z}(t_s)$

$$\boldsymbol{z}(t_{s+1}) = \boldsymbol{z}(t_s) + h_s\, \boldsymbol{f}(t_s, \boldsymbol{z}(t_s)) + \frac{h_s^2}{2}\, \boldsymbol{f}'(t_s, \boldsymbol{z}(t_s)) + \ldots \tag{3.17}$$

verdeutlicht, daß die so gefundenen Näherungswerte den Funktionswert nur bis zum linearen Glied wiedergeben. Für die Praxis scheint dieses Verfahren somit weniger geeignet, da die erreichbare Genauigkeit linear von der Schrittweite abhängt und zur Erzielung guter Approximationen sehr kleine Schrittweiten erforderlich sind. Eine gemeinsame Darstellung von numerischen Verfahren höherer Genauigkeit, die hier zur Simulationsberechnung verwendet werden, erfolgt im nächsten Abschnitt.

3.3.2 Beschreibung der Verfahren

Neben der gemeinsamen Darstellung der in dieser Arbeit relevanten Einschrittverfahren, die in der Literatur auch als *Runge-Kutta-Verfahren* bekannt sind, werden die zur Charakterisierung der Verfahren notwendigen Begriffe eingeführt. Die konkreten

Runge-Kutta-Verfahren lassen sich dann mit den im Anhang C aufgeführten Koeffiziententabellen konstruieren.

Ein Runge-Kutta-Verfahren wird formal über den allgemeinen Runge-Kutta-Ansatz in Definition 3.6 beschrieben. Die Definition zeigt, daß jedes m-stufige Verfahren eindeutig durch seine Verfahrensfunktion $\boldsymbol{\Phi}(t_s, \boldsymbol{z}_s)$ gekennzeichnet ist, die durch die $m(m+2)$ Koeffizienten α_i, β_{ij}, γ_i für $1 \leq i, j \leq m$ festgelegt wird. Wie in [Som67] gezeigt, kann man die Integration von Differentialgleichungen als eine Verallgemeinerung der Quadratur[6] auffassen. Das Prinzip der Quadratur besteht darin, die Funktion $\boldsymbol{f}(t_s, \boldsymbol{z}(t_s))$ im Teilintervall $[t_s, t_{s+1}]$ durch ein Interpolationspolynom $\boldsymbol{\phi}(t_s, \boldsymbol{z}_s)$ zu approximieren und über dieses Polynom anstelle der Funktion zu integrieren. Als Quadraturformel erhält man so die Verfahrensfunktion $\boldsymbol{\Phi}(t_s, \boldsymbol{z}_s)$. Die Koeffizienten α_i bezeichnen hierin die Stützstellen des Interpolationspolynoms $\boldsymbol{\phi}(t_s, \boldsymbol{z}_s)$, die mit den Gewichten γ_i in die Gesamtfläche einfließen. Da der Funktionsverlauf $\boldsymbol{z}(t)$ im Integrationsintervall unbekannt ist, die Berechnung von $\boldsymbol{f}(t_s, \boldsymbol{z}(t_s))$ an den Stützstellen α_i jedoch diese Funktionswerte erfordert, wird die Funktion $\boldsymbol{z}(t)$ ebenfalls über eine Quadraturformel mit den Gewichten β_{ij} bestimmt. Bei Vorgabe der Stützstellen α_i sind an die Gewichte β_{ij} und γ_i bestimmte Bedingungen geknüpft, um Verfahren maximaler Fehlerordnung zu erhalten, auf die hier aber nicht weiter eingegangen wird. Der interessierte Leser findet sie z.B. in [Glas67]. Die Fehlerordnung oder auch Ordnung eines Verfahrens bezeichnet hierbei den höchsten Grad der h-Potenzen in der Taylor-Entwicklung von $\boldsymbol{z}(t_{s+1})$ um den Punkt $\boldsymbol{z}(t_s)$ bis zu dem der Näherungswert $\boldsymbol{z}_{s+1}$ mit der Taylor-Entwicklung übereinstimmt. Eine genauere Begriffsbildung ist der Definition 3.7 zu entnehmen.

Definition 3.6 (Runge-Kutta-Verfahren) *Seien die maximale Stufe $m \in \mathbb{N}$ und die $m(m+2)$ Koeffizienten $\alpha_i, \beta_{ij}, \gamma_i \in \mathbb{R}$ für $1 \leq i, j \leq m$ gegeben, dann definiert die Verfahrensfunktion $\boldsymbol{\Phi}(t_s, \boldsymbol{z}_s)$ in 3.18 bzw. 3.19 ein m-stufiges*

1. *Explizites Runge-Kutta-Verfahren:*

$$\boldsymbol{z}_{s+1} = \boldsymbol{z}_s + h_s \underbrace{\sum_{i=1}^{m} \gamma_i \, \boldsymbol{k}_i}_{\boldsymbol{\Phi}(t_s, \boldsymbol{z}_s)} \tag{3.18}$$

$$mit \quad \boldsymbol{k}_i = \boldsymbol{f}(t_s + \alpha_i \, h_s, \boldsymbol{z}_s + h_s \sum_{j=1}^{i-1} \beta_{ij} \, \boldsymbol{k}_j)$$

2. *Implizites Runge-Kutta-Verfahren:*

$$\boldsymbol{z}_{s+1} = \boldsymbol{z}_s + h_s \underbrace{\sum_{i=1}^{m} \gamma_i \, \boldsymbol{k}_i}_{\boldsymbol{\Phi}(t_s, \boldsymbol{z}_s)} \tag{3.19}$$

$$mit \quad \boldsymbol{k}_i = \boldsymbol{f}(t_s + \alpha_i \, h_s, \boldsymbol{z}_s + h_s \sum_{j=1}^{m} \beta_{ij} \, \boldsymbol{k}_j)$$

[6] Als Quadratur bezeichnet man die Berechnung eines bestimmten oder unbestimmten Integrals.

Verbal beschrieben, besteht der Unterschied zwischen einem expliziten und impliziten Verfahren darin, daß bei der expliziten Formel in die Berechung des i-ten Funktionswertes des Differentialgleichungssystems nur vorangehende Funktionswerte einfließen, bei der impliziten Formel hingegen sämtliche Funktionswerte berücksichtigt werden. Da diese im allgemeinen jedoch noch nicht bekannt sind, führt die implizite Runge-Kutta-Formel 3.19 auf ein nichtlineares Gleichungssystem für die $\boldsymbol{k}_j$-Vektoren, das iterativ gelöst wird. Als Startwerte verwendet man die Vektoren

$$\boldsymbol{k}_1^{(0)}, \boldsymbol{k}_2^{(0)}, \ldots, \boldsymbol{k}_m^{(0)} = \boldsymbol{f}(t_s, \boldsymbol{z}_s). \tag{3.20}$$

Nach $2\,m - 1$ Iterationsschritten wird die lokale Fehlerordnung e_l des Verfahrens erreicht, vorausgesetzt die Schrittweite h_s genügt der Konvergenzbedingung 3.21 aus [Som67].

$$\max_{1 \le i \le m} h_s L_s \sum_{j=1}^{m} |\, b_{ij} \,| < 1 \quad \text{mit} \quad L_s = \max_{t \in [t_s, t_{s+1}]} \left| \frac{\partial f_l(t, \boldsymbol{z}(t))}{\partial z_k(t)} \right|, \; 1 \le k, l \le n \tag{3.21}$$

Die lokale Fehlerordnung e_l eines Verfahrens beschreibt die Definition 3.7, die neben diesem Begriff auch alle anderen, wesentlichen Begriffe formal einführt.

Definition 3.7 (Verfahrensfehler, Fehlerordnung) *Sei durch die Verfahrensfunktion $\boldsymbol{\Phi}\,(t_s, z_s)$ ein beliebiges Runge-Kutta-Verfahren gegeben. Dann gelten bezüglich diesem Verfahren mit $h_{max} = \max\limits_{0 \le s \le n-1} h_s$ und $c > 0$ die folgenden Bezeichnungen:*

1. *Lokaler Verfahrensfehler, Diskretisierungsfehler $\boldsymbol{\varepsilon}_l$:*

$$\boldsymbol{\varepsilon}_{l_{s+1}} = \frac{1}{h_s}\left(\boldsymbol{z}(t_{s+1}) - \boldsymbol{z}(t_s)\right) - \boldsymbol{\Phi}\,(t_s, \boldsymbol{z}_s) \tag{3.22}$$

2. *Lokale Fehlerordnung, Konsistenzordnung e_l:*

$$\max_{0 \le s \le n-1} \|\boldsymbol{\varepsilon}_{l_s}\| \le c\, h^{e_l} = O(h^{e_l}) \quad \textit{mit} \quad h_{max} \to 0 \tag{3.23}$$

3. *Globaler Verfahrensfehler, Diskretisierungsfehler $\boldsymbol{\varepsilon}_g$:*

$$\boldsymbol{\varepsilon}_g = \boldsymbol{z}(t_s) - z_s \tag{3.24}$$

4. *Globale Fehlerordnung, Ordnung e_g:*

$$\max_{0 \le s \le n-1} \|\boldsymbol{\varepsilon}_g\| \le c\, h^{e_g} = O(h^{e_g}) \quad \textit{mit} \quad h_{max} \to 0 \tag{3.25}$$

Die lokale und globale Fehlerordnung eines Einschrittverfahrens stehen über die Gleichung $e_g = e_l + 1$ in Beziehung. Für explizite m-stufige Runge-Kutta-Formeln mit $m \le 4$ können stets Verfahren mit einer lokalen Fehlerordnung $e_l = m + 1$ konstruiert werden. Für $m > 4$ erhält man jedoch höchstens eine lokale Fehlerordnung von $e_l = m$. Ein besseres Verhältnis zwischen maximaler Stufe und lokaler Fehlerordnung des Verfahrens besitzen implizite Runge-Kutta-Formeln, die auf einer Gauß-Legendre Stützstellenverteilung basieren. Diese Formeln erreichen die maximal mögliche, lokale Fehlerordnung von $e_l = 2\,m + 1$. Doch trotz des wesentlich günstigeren Verhältnisses zwischen Verfahrensstufe und lokaler Fehlerordnung benötigen sie aufgrund der iterativen Berechnung der Funktionswerte $\boldsymbol{k}_j$ für $1 \le j \le m$ gegenüber einem vergleichbaren, expliziten Verfahren einen höheren Rechenaufwand.

3.3.3 Automatische Schrittweitensteuerung

Im vorangegangenen Abschnitt wurde zwar der lokale und globale Verfahrensfehler sowie ihre gegenseitige Verknüpfung als auch ihre Abhängigkeit von der Schrittweite h_s beschrieben, jedoch erfolgte keine quantitative Aussage über die Größe dieser Fehler. Da bei der numerischen Simulationsberechnung im allgemeinen die Genauigkeit des Ergebnisses von Interesse ist, soll daher an dieser Stelle neben den Verfahrensfehlern auch auf Rundungsfehler, den Fehlerwechselwirkungen sowie die Fehlerabschätzung näher eingegangen werden. Diese Betrachtungen zeigen, daß bei vorgegebener Genauigkeitsanforderung die Minimierung der Rechenzeit eine Steuerung der Integrationsschrittweite erfordert. Sie wird sowohl für die expliziten als auch impliziten Runge-Kutta-Verfahren vorgestellt, wobei jedoch Implementationsdetails zu Gunsten der Übersichtlichkeit unberücksichtigt bleiben.

Fehlerbetrachtung

Die Güte, d.h. die Genauigkeit der numerischen Simulationsberechnung hängt von der Größe der lokalen Rundungs- und Verfahrensfehler ab. Die Akkumulation dieser Fehler kann bei genügend betragsgroßen Abweichungen zu einer Fehlerfortpflanzung führen, die das Simulationsergebnis total verfälschen. Man wird deshalb bestrebt sein, die einzelnen Fehlereinflüsse möglichst gering zu halten. In Bezug auf die Rundungsfehler bedeutet dies, daß entweder mit einer höheren Stellenanzahl gerechnet werden muß, um einen kleineren, elementaren Rundungsfehler zu erreichen oder die Anzahl der Rechenschritte zu verringern ist, was einer Vergrößerung der Schrittweite gleichkommt. Denn der globale Rundungsfehler nimmt umgekehrt proportional zur Schrittweite h_s ab. Da eine Rechnung mit höherer Stellenanzahl die Simulationsrechenzeit verlängert und primäres Ziel kurze Rechenzeiten sind, bietet sich als Alternative zur Minimierung des Rundungsfehlers nur die Vergrößerung der Integrationsschrittweite an. Gleichzeitig wirkt eine Inkrementierung der Schrittweite negativ auf den globalen Verfahrensfehler ein, denn sie läßt ihn, wie aus der Beziehung 3.25 hervorgeht, proportional zur Schrittweite h_s anwachsen. Die Vergrößerung des globalen Verfahrensfehlers könnte hierbei nur durch die Wahl eines Verfahrens höherer Ordnung ausgeglichen werden, was aber unweigerlich eine Aufwandserhöhung mit sich bringt. Ziel ist es daher, zu einer vorgegebenen Fehlerschranke eine optimale Schrittweite zu bestimmen, so daß der Gesamtfehler, der sich additiv aus dem globalen Rundungs- und Verfahrensfehler zusammensetzt, minimal wird. Die Abbildung 3.5 illustriert diesen Zusammenhang am Beispiel eines expliziten Runge-Kutta-Verfahrens 4-ter Ordnung. Sie zeigt den qualitativen Verlauf der Fehlerkomponenten sowie den daraus resultierenden Gesamtfehler, der an der Stelle h_{opt} sein Minimum besitzt. Die Praxis zeigt aber, daß die theoretisch bestimmte Schrittweite h_{opt} in Bezug auf die damit verbundene Rechenzeit zu gering ausfällt. Hier muß man einen Kompromiß zwischen Genauigkeit und Rechenzeit eingehen. Als Anhaltspunkt für eine praxistaugliche Schrittweite kann hierbei die Beziehung aus [Stum71]

$$h_s \max |\, \boldsymbol{J}(t_s) \,| \approx 10^{-1} \tag{3.26}$$

gelten.

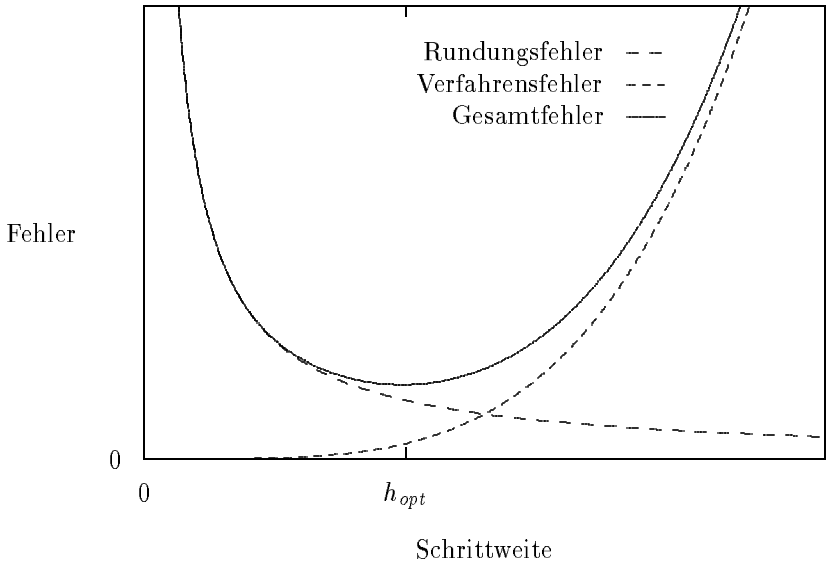


Abbildung 3.5: Fehlerwechselwirkung der numerischen Lösungsberechnung

Im Vergleich zum globalen Verfahrensfehler spielt der globale Rundungsfehler bei 64-Bit breiten Datenworten, was einer 16-stelligen Genauigkeit entspricht, eher eine untergeordnete Rolle. Der Rundungsfehler gewinnt erst bei 32-Bit breiten Datenworten an Bedeutung. Da die heutigen Rechner im allgemeinen noch höhere Genauigkeiten als 16 Stellen erlauben und 64-Bit breite Datenworte quasi einen de facto Standard bilden, bleibt der Rundungsfehler in der Fehlerabschätzung der Schrittweitensteuerung außer Acht. Der Verfahrensfehler kann theoretisch, wie im Satz 7.8 aus [Koe90] beschrieben, über die Lipschitzkonstante abgeschätzt werden. Diese Fehlerabschätzung wächst exponentiell mit der Zeit t, so daß sie in vielen Fällen den tatsächlichen Fehler um ein Vielfaches überschätzt. Sie scheint daher für die Praxis weniger geeignet. Bessere, d.h. genauere Fehlerabschätzungen, die sich ohne großen, zusätzlichen Rechenaufwand realisieren lassen und hier auch zur Steuerung der Integrationsschrittweite eingesetzt werden, stellen die Fehlerabschätzungen in den folgenden Schrittweitensteuerungen dar. Diese Fehlerabschätzungen verwenden einen gemischten Fehlertest, d.h. die vorgegebene Fehlerschranke ε setzt sich gemäß Gleichung 3.27 additiv aus absolutem und relativem Fehler zusammen.

$$\varepsilon = \varepsilon_{abs} + \varepsilon_{rel} \parallel \hat{\boldsymbol{z}}_{s+1} \parallel \tag{3.27}$$

Schrittweitensteuerung für explizite Verfahren

Die Grundidee der Schrittweitensteuerung für explizite Runge-Kutta-Formeln besteht darin, den Näherungswert mit zwei Verfahren unterschiedlicher, globaler Fehlerordnung e_g und $\hat{e}_g$ mit $\hat{e}_g > e_g$ zu berechnen und dann über die Differenz dieser beiden Werte die Fehlergröße abzuschätzen. Im allgemeinen wird man $\hat{e}_g = e_g + 1$ wählen, so daß sich die beiden Werte im direkten Vergleich zur Taylor-Entwicklung des exakten Funktionswertes um ein Glied, nämlich um das $\hat{e}_g$-te unterscheiden. Die Fehlerabschätzung

und die daran geknüpfte Beurteilung der Schrittweite h_s kann dann, wie im Schritt 1b dargestellt, vorgenommen werden. Der genaue Beweis für die Schrittweitensteuerung ist in [Luth87] zu finden. Um den zusätzlichen Rechenaufwand für die Berechnung des zweiten Näherungswertes möglichst in Grenzen zu halten, verwendet man sogenannte Einbettungsformeln. Sie besitzen die Eigenschaft, daß die $\boldsymbol{k}_i$-Werte bis zur Ordnung e_g identisch sind und so nur ein bis zwei zusätzliche Funktionsauswertungen des Differentialgleichungssystems pro Integrationsschritt erforderlich sind. Eine formale Beschreibung der Einbettungsformeln stellt die Definition 3.8 vor.

Definition 3.8 (Einbettungsformel) *Sei eine m- und eine $\widehat{m}$-stufige, explizite Runge-Kutta-Formel der globalen Fehlerordnung e_g bzw. $\hat{e}_g$ mit $\hat{e}_g > e_g$ für $\widehat{m} \geq m$ gegeben. Stimmen die Koeffizienten α_i und β_{ij} für $1 \leq i, j \leq m$ überein, bezeichnet man das Formelpaar 3.28*

$$\begin{aligned} \boldsymbol{z}_{s+1} &= \boldsymbol{z}_s + h_s \underbrace{\sum_{i=1}^{m} \gamma_i \, \boldsymbol{k}_i}_{\boldsymbol{\Phi}(t_s, \boldsymbol{z}_s)} \\ \hat{\boldsymbol{z}}_{s+1} &= \boldsymbol{z}_s + h_s \underbrace{\sum_{i=1}^{\widehat{m}} \hat{\gamma}_i \, \boldsymbol{k}_i}_{\widehat{\boldsymbol{\Phi}}(t_s, \boldsymbol{z}_s)} \\ \textit{mit} \quad \boldsymbol{k}_i &= \boldsymbol{f}(t_s + \alpha_i \, h_s, \boldsymbol{z}_s + h_s \sum_{j=1}^{i-1} \beta_{ij} \, \boldsymbol{k}_j) \end{aligned} \tag{3.28}$$

als Einbettungsformel. Man sagt auch, daß die m-stufige in der $\widehat{m}$-stufigen Runge-Kutta-Formel eingebettet ist.

Sei eine Einbettungsformel gemäß der Definition 3.8 mit den beiden Verfahrensfunktionen $\boldsymbol{\Phi}(t_s, \boldsymbol{z}_s)$ und $\widehat{\boldsymbol{\Phi}}(t_s, \boldsymbol{z}_s)$ sowie eine Anfangsschrittweite h_s und eine Fehlerschranke $\varepsilon > 0$ für die Genauigkeit der Näherungslösung gegeben. Dann wird die Schrittweite h_s nach folgendem Algorithmus gesteuert:

1. Wiederhole

   (a) Berechne die Näherungswerte

   $$\begin{aligned} \boldsymbol{z}_{s+1} &= z_s + h_s \, \boldsymbol{\Phi}(t_s, \boldsymbol{z}_s) \\ \hat{\boldsymbol{z}}_{s+1} &= z_s + h_s \, \widehat{\boldsymbol{\Phi}}(t_s, \boldsymbol{z}_s). \end{aligned}$$

   (b) Bestimme den Schrittweitenadaptionsfaktor

   $$h_{adapt} = \left(\frac{h_s \, \varepsilon}{\| \boldsymbol{z}_{s+1} - \hat{\boldsymbol{z}}_{s+1} \|} \right)^{\frac{1}{e_g}} .$$

(c) Falls $h_{adapt} < 1$ gilt, dann verkleinere Schrittweite

$$h_s = \max\{\frac{1}{2}, h_{adapt}\}\, h_s.$$

solange bis $h_{adapt} \geq 1$ gilt.

2. Akzeptiere $\hat{\boldsymbol{z}}_{s+1}$ als neue Näherung im Gitterpunkt t_{s+1}. Setze $s = s + 1$ und berechne beginnend im Schritt 1 den nächsten Näherungswert an der Stelle $t_{s+1} = t_s + h_s$ mit der neuen, vergrößerten Schrittweite

$$h_s = \min\{2, h_{adapt}\}\, h_s.$$

Schrittweitensteuerung für implizite Verfahren

Der Schrittweitensteuerung der impliziten Runge-Kutta-Verfahren liegt die gleiche Idee zugrunde wie die der expliziten Verfahren. Im Unterschied hierzu existieren jedoch keine eingebetteten, impliziten Formeln, so daß man auf zwei vollständige Formeln der globalen Fehlerordnung e_g und $\hat{e}_g$ angewiesen ist. In Erweiterung zu der Schrittweitensteuerung der expliziten Verfahren bietet die Steuerung der impliziten Verfahren die Möglichkeit, die zugelassene Fehlergröße der einzelnen Differentialgleichungen mittels der Gewichte σ_i für $1 \leq i \leq n$ unterschiedlich festzulegen, so daß die vorgegebene Fehlerschranke ε die mittlere, geforderte Genauigkeit wiederspiegelt. Da die Näherungslösungen der impliziten Runge-Kutta-Verfahren, wie bereits in 3.3.2 beschrieben, iterativ gewonnen werden, muß die Konvergenzbedingung 3.21 erfüllt sein. In Bezug auf die Schrittweitensteuerung ist diese Bedingung als erfüllt anzusehen, wenn für alle Iterationsschritte die mittlere, praktische Differenz zweier Iterationen unterhalb des mittleren, geschätzten Iterationsfehlers liegt. Die Schrittweite h_s hängt somit sowohl von der mittleren, vorgegebenen Fehlerschranke ε als auch von der Konvergenzbedingung 3.21 ab. Entsprechend diesen Anforderungen ist sie gegebenfalls in Schritt 4c bzw. 4d zu korrigieren. Ihr Adaptionsfaktor, der sich im Schritt 4b des Algorithmus wiederfindet, erlaubt eine Beurteilung der Schrittweite schon nach dem ersten Iterationsschritt, so daß sich der zusätzliche Rechenaufwand bei Korrektur der Schrittweite im Rahmen hält. Der ausführliche Beweis für die Schrittweitensteuerung ist in [Som67] dargestellt.

Seien zwei implizite Runge-Kutta-Formeln gemäß der Definition 3.6 mit den beiden Verfahrensfunktionen $\boldsymbol{\Phi}(t_s, \boldsymbol{z}_s)$ und $\hat{\boldsymbol{\Phi}}(t_s, \boldsymbol{z}_s)$ und eine Fehlerschranke $\varepsilon > 0$ für die mittlere Genauigkeit der Näherungslösung gegeben. Dann wird die Schrittweite h_s nach folgendem Algorithmus gesteuert:

1. Bestimme die Schrittweite h_s im Gitterpunkt t_s

$$h_s = \left(\frac{\varepsilon\,(2\,m)!}{\sqrt{\frac{\sum_{i=1}^{n} \sigma_i (\dot{\boldsymbol{f}}(t_s, \boldsymbol{z}(t_s))\, \boldsymbol{J}(t_s)^{2\,m-2})_i^2}{\sum_{i=1}^{n} \sigma_i}}} \right)^{\frac{1}{2\,m}}$$

und ermittle den mittleren, geschätzten Iterationsfehler

$$\delta^{(l)} = \frac{h_s^{(l+2)}}{(l+2)!} \sqrt{\frac{\sum_{i=1}^n \sigma_i (\dot{\boldsymbol{f}}(t_s, \boldsymbol{z}(t_s))\, \boldsymbol{J}(t_s))_i^2}{\sum_{i=1}^n \sigma_i}} \quad \text{für} \quad l = 0, 1, \ldots, 2\,m-1.$$

2. Berechne die Näherungswerte des 0-ten Iterationsschrittes

$$\begin{aligned} \boldsymbol{z}_{s+1}^{(0)} &= z_s + h_s\, \boldsymbol{\Phi}\,(t_s, z_s) \\ \hat{\boldsymbol{z}}_{s+1}^{(0)} &= z_s + h_s\, \widehat{\boldsymbol{\Phi}}\,(t_s, z_s). \end{aligned}$$

3. Setze $l = 1$.

4. Wiederhole solange $l \leq 2\,m - 1$:

   (a) Berechne die Näherungswerte des l-ten Iterationsschrittes

$$\begin{aligned} \boldsymbol{z}_{s+1}^{(l)} &= z_s + h_s\, \boldsymbol{\Phi}\,(t_s, z_s) \\ \hat{\boldsymbol{z}}_{s+1}^{(l)} &= z_s + h_s\, \widehat{\boldsymbol{\Phi}}\,(t_s, z_s). \end{aligned}$$

   (b) Bestimme die mittlere praktische Differenz zweier Iterationen

$$\Delta^{(l)} = \sqrt{\frac{\sum_{i=1}^n \sigma_i (\hat{\boldsymbol{z}}_{s+1}^{(l)} - \hat{\boldsymbol{z}}_{s+1}^{(l-1)})_i^2}{\sum_{i=1}^n \sigma_i}}$$

   und berechne den Schrittweitenadaptionsfaktor

$$h_{adapt}^{(l)} = \sqrt{\frac{\sum_{i=1}^n \sigma_i\, (\hat{\boldsymbol{z}}_{s+1}^{(l)} - \boldsymbol{z}_{s+1}^{(l)})_i^2}{\sum_{i=1}^n \sigma_i}}.$$

   (c) Falls $h_{adapt}^{(l)} \geq \varepsilon$ gilt, dann bestimme die neue Schrittweite

$$h_s = h_s \left(\frac{\varepsilon}{h_{adapt}^{(l)}} \right)^{\frac{1}{2\,m+1}}.$$

   und gehe zu 2.

   (d) Falls $\Delta^{(l)} \geq \delta^{(l-1)}$ gilt, dann bestimme die neue Schrittweite

$$h_s = 0.6\, h_s.$$

   und gehe zu 2.

   (e) Falls $\Delta^{(l)} < \varepsilon$ gilt, dann gehe zu 5 sonst setze $l = l + 1$.

5. Akzeptiere $\hat{\boldsymbol{z}}_{s+1}$ als Näherungslösung im Gitterpunkt t_{s+1}, setze $s = s + 1$ und berechne beginnend im Schritt 1 den nächsten Näherungswert an der Stelle $t_{s+1} = t_s + h_s$.

3.3.4 Verfahrensauswahl

Neben der optimalen Integrationsschrittweite spielt auch die richtige Wahl des Runge-Kutta-Verfahrens im Hinblick auf eine zu minimierende Rechenzeit eine wichtige Rolle. Abgesehen von der Hardware hängt die Rechenzeit besonders von der Größe des Simulationszeitintervalls, dem Differentialgleichungssystem und der an die Lösungsfunktion gestellten Genauigkeitsanforderung ab. Aber auch der Steifheitsgrad des Differentialgleichungssystems kann bei ungeeigneter Verfahrensauswahl die Rechenzeit verlängern und im Falle numerischer Instabilität zu falschen Ergebnissen führen. Aus diesem Grund werden hier Kriterien zur optimalen Verfahrensauswahl vorgestellt, die später dann, als Regeln formuliert, die Steuerung der Simulationsberechnung übernehmen. Zunächst erfolgt jedoch eine informelle Beschreibung des numerischen Stabilitätsbegriffs. Die Klassen der Einschrittverfahren führen hierbei zu zwei unterschiedlichen Stabilitätsbegriffen, die die Wahl der Verfahrensklasse bestimmen, so daß dann nur noch innerhalb einer Klasse nach einem optimalen Verfahren gesucht werden muß.

Numerische Stabilität

Zur Erläuterung des numerischen Stabilitätsbegriffs dient die Abbildung 3.6. Sie stellt die Fehlerfortpflanzung innerhalb eines Integrationsschrittes schematisch dar. Ausgehend vom exakten Anfangswert A mündet die Lösungskurve bei fehlerfreier Rech-

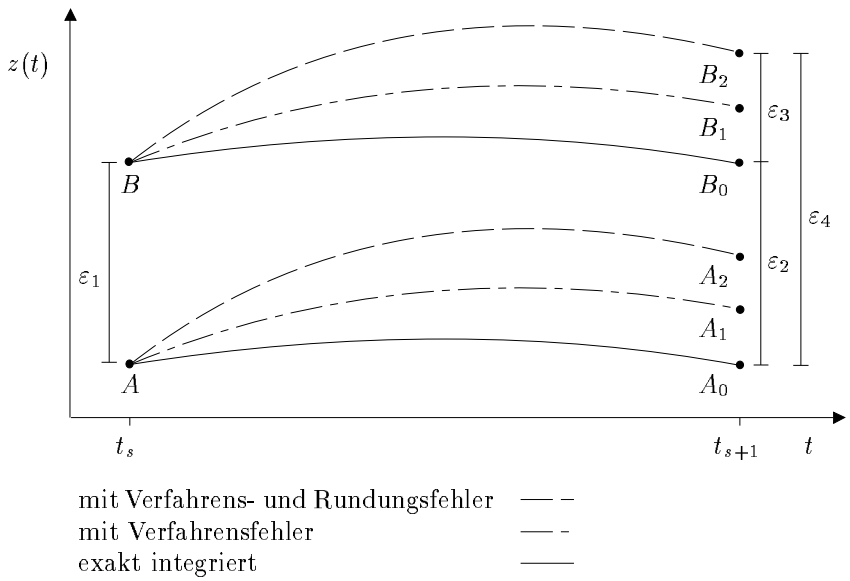


Abbildung 3.6: Fehlerfortpflanzung und numerische Stabilität

nung in den Punkt A_0, bei numerischer Integration in den Punkt A_1 und bei zusätzlicher Berücksichtigung von Rundungsfehlern in den Punkt A_2. Legt man zum Zeitpunkt t_s einen fehlerbehafteten Anfangswert B zugrunde, der die Folge von vorangegangenen Rundungs- und Verfahrensfehlern bei der numerischen Lösungsberechnung sein kann, verläuft die Kurve entsprechend durch den Punkt B_0 bei exakter Rechnung, durch B_1

bei numerischer Integration und durch B_2, wenn neben dem lokalen Verfahrensfehler auch die lokalen Rundungsfehler berücksichtigt werden. Der anfängliche Fehlerbetrag ε_1 fließt somit in die Berechnung des Näherungswertes an der Stelle t_{s+1} als Folgefehler ε_2 ein. Neben dem durch den Verfahrens- und Rundungsfehler verursachten Fehleranteil ε_3 bildet er eine wesentliche Komponente des Gesamtfehlers ε_4 im Gitterpunkt t_{s+1}. Offenbar liefert ein Verfahren zur Lösung von lipschitz-beschränkten Anfangswertproblemen nur dann eine Näherungslösung der exakten Lösung, wenn der Fehler ε_4 über eine Folge von Integrationsschritten betrachtet, beschränkt ist, d.h. wenn für eine Konstante $c \in \mathbb{R}$ gilt:

$$\sum_{s=0}^{n} \varepsilon_{4_s} < |\, c \,| \tag{3.29}$$

Damit die Ungleichung 3.29 Gültigkeit besitzt, muß der Fortpflanzungsfehleranteil ε_{2_s} von ε_{4_s} gegen Null konvergieren. Das Verfahren heißt dann *numerisch stabil.*

Zur Untersuchung der numerischen Stabilität betrachtet man das lipschitz-beschränkte Testanfangswertproblem 3.30

$$\begin{aligned} \dot{z}(t) &= \lambda\, z(t) \quad \text{für } \lambda \in \mathbb{C} \\ z(0) &= 1 \end{aligned} \tag{3.30}$$

mit der bekannten Lösung $z(t) = \exp(\lambda\, t)$. Die Funktionswerte der Lösungskurve an den Gitterpunkten t_s entstehen jeweils durch Multiplikation des vorangehenden Funktionswertes mit dem Faktor $M(\lambda\, h_s) = \exp(\lambda\, h_s)$, also

$$\begin{aligned} z(t_{s+1}) &= z(t_s + h_s) = \exp(\lambda\,(t_s + h_s)) = \exp(\lambda\, h_s)\, \exp(\lambda\, t_s) \\ \Rightarrow z(t_{s+1}) &= M(\lambda\, h_s)\, z(t_s) \quad \text{für } s = 0, 1, \ldots, n-1. \end{aligned} \tag{3.31}$$

Die numerische Lösung des Anfangswertproblems 3.30 führt ebenfalls zu der Beziehung 3.31. Der Multiplikator $M(\lambda\, h_s)$ ist nun aber eine mehr oder weniger genaue Approximation der exp-Funktion im Punkt $\lambda\, h_s$. Die Approximation ergibt sich durch die Anwendung der Verfahrensfunktion $\Phi(t_s, z_s)$ auf 3.30, wenn das Ergebnis in Abhängigkeit von $\lambda\, h_s$ dargestellt wird. Für den stabilen Fall der Differentialgleichung, d.h. $\Re(\lambda) < 0$, repräsentiert die Lösung des Anfangswertproblems 3.30 eine abklingende exp-Funktion. Die Näherungslösung z_s klingt ebenfalls wie $z(t_s)$ ab, genau dann wenn

$$|\, M(\lambda\, h_s) \,| < 1 \tag{3.32}$$

gilt. Diese Bedingung ist nicht zwangsläufig für alle negativen Werte von $\lambda\, h_s$ erfüllt, so daß die Ungleichung das Stabilitätsgebiet des numerischen Verfahrens definiert. Eine graphische Darstellung der Stabilitätsbereiche verschiedener Einschrittverfahren ist z.B. in [Schw93] zu finden. Differenziertere Stabilitätsaussagen bezüglich der hier verwendeten Runge-Kutta-Verfahren ermöglicht die Definition 3.9.

Definition 3.9 (Numerische Stabilität) *Sei ein beliebiges Runge-Kutta-Verfahren gemäß der Definition 3.6 und ein lipschitz-beschränktes Anfangswertproblem nach Definition 3.4 gegeben. Bezeichne $\lambda_i(t, \boldsymbol{z}_i(t))$ mit $\Re(\lambda_i(t, \boldsymbol{z}_i(t))$ für $i = 1, 2, \ldots, n_1$ die Eigenwerte des Differentialgleichungssystems. Dann heißt das Runge-Kutta-Verfahren bezüglich des Anfangswertproblems 3.6*

1. *absolut stabil, falls die Schrittweite h_s für $s = 0, 1, \ldots, n_2 - 1$ so gewählt ist, daß die numerische Stabilitätsbedingung 3.32 gilt.*

2. *A-stabil, falls für beliebige Schrittweiten $h_s > 0$, $s = 0, 1, \ldots, n_2 - 1$ die numerische Stabilitätsbedingung 3.32 gilt.*

Offensichtlich sind explizite Runge-Kutta-Verfahren für lipschitz-beschränkte, stabile Anfangswertprobleme stets absolut stabil. Wie jedoch das Beispiel 3.13 illustriert, sind sie nicht A-stabil. Diese Eigenschaft besitzen hingegen die impliziten Runge-Kutta-Verfahren nach der Gauß-Legendre Stützstellenverteilung, so daß der abklingende Verlauf der Lösungskurve im obengenannten Beispiel korrekt wiedergegeben wird. Einen allgemeinen Beweis dieser Aussage findet der Leser in [Hall76].

Ein erstes Kriterium zur optimalen Verfahrensauswahl liefert somit die Anfangswertproblemklasse. Liegt ein steifes Anfangswertproblem vor und interessieren im wesentlichen nur die schwach abklingenden Komponenten der Lösungsfunktion, ist ein implizites Runge-Kutta-Verfahren zu wählen, denn die A-Stabilität dieser Verfahrensklasse garantiert, daß sämtliche Komponenten der Lösungsfunktion innerhalb einer gewissen Genauigkeit wiedergegeben werden und die Rechnung numerisch stabil bleibt. In allen anderen Fällen sind die expliziten den impliziten Verfahren wegen des geringeren Verfahrensaufwandes vorzuziehen.

Verfahrensauswahl für explizite Verfahren

Ist die Entscheidung für die Verfahrensklasse gefallen, wird unter den zur Verfügung stehenden, expliziten Runge-Kutta-Formeln das optimale Verfahren anhand einer Aufwandsabschätzung bestimmt. Dieses Verfahren benötigt den minimalsten Rechenaufwand bei Einhaltung einer vorgegebenen Fehlerschranke ε. Als Aufwandsmaß dient die Anzahl der Funktionsauswertungen des Differentialgleichungssystems, die zur Erzielung der vorgegebenen Genauigkeit ε über das feste Zeitintervall der Länge 1 benötigt werden. Dieses Maß vernachlässigt den Rechenaufwand des Verfahrens gegenüber dem Aufwand zur Ermittlung der Funktionswerte und erlaubt daher eine objektive Beurteilung unterschiedlicher Verfahren.

Die Menge der hier zur Verfügung stehenden, expliziten Runge-Kutta-Verfahren umfaßt die $m/\widehat{m}$-stufigen Einbettungsformeln 2./3., 4./5., 5./6. und 7/8. Ordnung. Sie benötigen jeweils $\widehat{m}$-Funktionsauswertungen des Differentialgleichungssystems zur Berechnung des Näherungswertes im Gitterpunkt t_{s+1}. Ist die zur Einhaltung der Fehlerschranke ε erforderliche Schrittweite h_s bekannt, läßt sich der Aufwand des Verfahrens, wie im Schritt 2d des Algorithmus dargestellt, berechnen. Diese Schrittweite kann nach [Luth87] über die Fehlerabschätzung der beiden Näherungswerte $\boldsymbol{z}_{s+1}$, $\hat{\boldsymbol{z}}_{s+1}$ gewonnen werden. Sie liefert einen Schrittweitenadaptionsfaktor h_{adapt} mit der die Testschrittweite h_s^* zu multiplizieren ist, um die gesuchte Schrittweite h_s zu erhalten.

Sei eine Menge von $m/\widehat{m}$-stufigen Einbettungsformeln $e_g/\hat{e}_g$-ter Ordnung und eine Fehlerschranke $\varepsilon > 0$ für die Lösungsfunktion gegeben. Dann liefert der folgende Algorithmus das optimale Verfahren zur Berechnung der Näherungslösung:

1. Initialisiere die Testschrittweite nach Gleichung 3.26 zu

$$h_s^* = \frac{10^{-1}}{\max |\, \boldsymbol{J}(t_s)\, |}.$$

2. Wiederhole für alle $m/\widehat{m}$-stufigen Einbettungsformeln $e_g/\hat{e}_g$-ter Ordnung mit $\widehat{m} \in \{3, 6, 8, 13\}$:

   (a) Berechne die Näherungswerte

$$\begin{aligned} \boldsymbol{z}_{s+1} &= \boldsymbol{z}_s + h_s^* \, \boldsymbol{\Phi}\,(t_s, \boldsymbol{z}_s) \\ \hat{\boldsymbol{z}}_{s+1} &= \boldsymbol{z}_s + h_s^* \, \widehat{\boldsymbol{\Phi}}\,(t_s, \boldsymbol{z}_s). \end{aligned}$$

   (b) Bestimme den Schrittweitenadaptionsfaktor zu

$$h_{adapt}(\varepsilon, e_g) = \left(\frac{h_s^* \, \varepsilon}{\| \, \boldsymbol{z}_{s+1} - \hat{\boldsymbol{z}}_{s+1} \, \|} \right)^{1/e_g}.$$

   (c) Ermittle die im Gitterpunkt t_s zur Einhaltung der Fehlerschranke ε erforderliche Schrittweite

$$h_s(\varepsilon, e_g) = h_s^* \, h_{adapt}(\varepsilon, e_g).$$

   (d) Berechne den Gesamtaufwand zu

$$AW(\varepsilon, \widehat{m}, e_g) = \frac{\widehat{m}}{h_s(\varepsilon, e_g)}.$$

3. Wähle als optimales Verfahren die $m/\widehat{m}_{opt}$-stufige Einbettungsformel $e_g/\hat{e}_g$-ter Ordnung mit

$$\widehat{m}_{opt} = \min_{\widehat{m}} AW(\varepsilon, \widehat{m}, e_g).$$

Verfahrensauswahl für implizite Verfahren

Die Auswahl eines optimalen Verfahrens unter den zur Verfügung stehenden, impliziten, $m/\widehat{m}$-stufigen Runge-Kutta-Verfahren 4./6., 10./12., 16./18. sowie 22./24. Ordnung erfolgt nach dem gleichen Schema wie bei den expliziten Verfahren. Zunächst wird für den Gitterpunkt t_{s+1} eine Integrationsschrittweite h_s bestimmt, so daß der globale Verfahrensfehler unterhalb der vorgegebenen Fehlerschranke ε bleibt, und dann mit diesem Parameter der Aufwand des Verfahrens berechnet. Im Unterschied zur Verfahrensauswahl der expliziten Einbettungsformeln kann die Schrittweitengröße jedoch direkt, d.h. in einem Schritt ermittelt werden. Sie resultiert gemäß [Som67], Gleichung 2.31 und 3.2, aus der Abschätzung des lokalen Diskretisierungsfehlers, wenn man fordert, daß diese Fehlergröße nach $2\,m - 2$ Iterationen gleich der vorgegebenen Fehlerschranke ε ist. Die im Vergleich zur Aufwandsabschätzung der expliziten Verfahren kompliziertere Berechnungsformel der impliziten Verfahren liegt im wesentlichen in der

iterativen Bestimmung der Näherungswerte $\boldsymbol{z}_{s+1}$, $\hat{\boldsymbol{z}}_{s+1}$ begründet. Ihre Berechnung erfordert $2\,m-1$ Iterationsschritte, in denen jeweils m und $m+1$ $\boldsymbol{k}$ Werte ermittelt werden. Zusammen mit der Berechnung des Iterationsstartwertes $\boldsymbol{f}(t_s, \boldsymbol{z}_s)$ und den für die Schrittweitensteuerung benötigten Funktionswerten der Ableitung $\dot{\boldsymbol{f}}(t_s, \boldsymbol{z}(t_s))$ sowie der Jacobi-Matrix $\boldsymbol{J}(t_s)$ ergibt sich so die im Algorithmus aufgeführte Gesamtanzahl

$$2 + n + m\,(2\,m-1) + (m+1)\,(2\,m-1) = 1 + n + 4\,m^2$$

an Funktionsauswertungen[7].

Sei eine Menge von $m/\widehat{m}$-stufigen Runge-Kutta-Formelpaaren $e_g/\hat{e}_g$-ter Ordnung und eine Fehlerschranke $\varepsilon > 0$ für die Lösungsfunktion gegeben. Dann selektiert der folgende Algorithmus das optimale Verfahren zur Berechnung der Näherungslösung:

1. Wiederhole für alle $m/\widehat{m}$-stufigen Runge-Kutta-Formelpaare $e_g/\hat{e}_g$-ter Ordnung mit $m \in \{2, 5, 8, 11\}$:

   (a) Ermittle die im Gitterpunkt t_s zur Einhaltung der Fehlerschranke ε erforderliche Schrittweite

   $$h_s(\varepsilon, m) = \left(\frac{\varepsilon\,(2\,m)!}{\sqrt{\frac{\sum_{i=1}^{n} \sigma_i (\dot{\boldsymbol{f}}(t_s, \boldsymbol{z}(t_s))\,\boldsymbol{J}(t_s)^{2\,m-2})_i^2}{\sum_{i=1}^{n} \sigma_i}}} \right)^{\frac{1}{2\,m}}.$$

   (b) Berechne den Gesamtaufwand zu

   $$AW(\varepsilon, m) = \frac{n + 1 + 4\,m^2}{h_s(\varepsilon, m)}.$$

2. Wähle als optimales Verfahren das $m_{opt}/\widehat{m}$-stufige Runge-Kutta-Formelpaar mit

   $$m_{opt} = \min_m AW(\varepsilon, m).$$

[7] Die Berechnung eines Funktionsvektors wird stets als ein Wert aufgefaßt.

4 Wissensbasierte Modellbildung

Im Abschnitt 2 ist dem Leser u.a. bereits der prinzipielle Aufbau und die Funktionsweise der wissensbasierten Modellbildung vorgestellt worden. An dieser Stelle soll nun weiterführend eine detaillierte Erläuterung aller wesentlichen Teilaspekte bei der heuristischen Modellbildung erfolgen. Beginnend mit einer allgemeinen Problemeinführung wird dann konkret die Vorgehensweise bei der Modelltiefenbestimmung beschrieben und anschließend die Untersuchung des Informationsflusses erläutert. Die Bestimmung des Informationsflusses bleibt in dieser Arbeit jedoch auf eine Darstellung der Grundidee und des Lösungsansatzes beschränkt.

4.1 Problembeschreibung

Das dynamische Verhalten einer Komponente kann, wie in Abschnitt 3.1 dargestellt, durch ein gewöhnliches, nichtlineares Differentialgleichungssystem beschrieben werden. Anders als bei der statischen Modellierung gibt es nicht das *universelle*, dynamische Modell einer Komponente, das alle an die Verhaltensbeschreibung gestellten Anforderungen wie z.B. eine hohe Detailgetreue oder eine niedrige Simulationszeit gleichermaßen erfüllt. Als Maß für die Detailgetreue einer Verhaltensbeschreibung faßt man hierbei den Grad der Berücksichtigung von Einflußfaktoren auf die Komponente und ihre Umsetzung im Modell auf. Sie wird im folgenden auch als *Modelltiefe* bezeichnet. Die Simulationszeit gibt die über ein festes Zeitintervall zur Simulationsberechnung benötigte Zeit wieder. Je nach Modelltiefe wird somit das Verhalten einer Komponente mehr oder weniger genau modelliert, wobei gleichzeitig die Simulationszeit entsprechend zu- oder abnimmt. Eine Auswahl von Komponentenverhaltensbeschreibungen unterschiedlicher Modelltiefen ist im Anhang A.2 zu sehen. Ihr liegt im allgemeinen die folgende Modellabstufung zugrunde:

<u>Modelltiefe 1:</u> Berücksichtigung der Massenträgheit

<u>Modelltiefe 2:</u> Zusätzlich zu Modelltiefe 1, Berücksichtigung des Druckaufbaus

<u>Modelltiefe 3:</u> Zusätzlich zu Modelltiefe 2, Berücksichtigung der Reibungskräfte

<u>Modelltiefe 4:</u> Zusätzlich zu Modelltiefe 3, Berücksichtigung der Leckagen

Abbildung 4.1: Modelltiefenabstufung der dynamischen Verhaltensbeschreibungen

Vergleicht man das im folgenden dargestellte Differentialgleichungssystem des Gleichgangzylinders der Modelltiefe 1 mit dem der Modelltiefe 2 fällt der höhere Rechenaufwand sofort ins Auge, aus den zwei Differentialgleichungen der Modelltiefe 1 sind 4 in der Modelltiefe 2 geworden. Ein mit dem Detaillierungsgrad zunehmender Rechenaufwand muß aber nicht zwingend in einer größeren Anzahl an Differentialgleichungen begründet liegen, sondern kann auch durch einen neuen Satz komplexerer Differentialgleichungen gleicher Anzahl verursacht werden.

Modelltiefe 1: Berücksichtigung der Massenträgheit

$$\begin{aligned}\dot{x}_{02}(t) &= x_{12}(t)\\ \dot{x}_{12}(t) &= \frac{A_k/10\,(p_3(t)-p_4(t)) - F_2(t) + F_1(t) + d\,x_{12}(t)}{m}\end{aligned}$$

Modelltiefe 2: Zusätzlich zu Modelltiefe 1, Berücksichtigung des Druckaufbaus

$$\begin{aligned}\dot{x}_{02}(t) &= x_{12}(t)\\ \dot{x}_{12}(t) &= \frac{A_k/10\,(p_3(t)-p_4(t)) - F_2(t) + F_1(t) + d\,x_{12}(t)}{m}\\ \dot{p}_3(t) &= \frac{16660\,E_{\ddot{O}l}(p_3(t))}{V_0 + 1000\,A_k\,x_{02}(t)}\,(Q_3(t) + 0.06\,A_k\,x_{12}(t))\\ \dot{p}_4(t) &= \frac{16660\,E_{\ddot{O}l}(p_4(t))}{V_0 + 1000\,A_k\,x_{01}(t)}\,(Q_4(t) + 0.06\,A_k\,x_{11}(t))\end{aligned}$$

Abbildung 4.2: Differentialgleichungssysteme eines Gleichgangzylinders

Die Simulation eines hydraulischen Schaltkreises soll dem Ingenieur im allgemeinen letzten Aufschluß über die geforderte Funktionalität des Schaltkreises bezüglich der Dynamik geben. Nicht immer ist hierfür jedoch ein sehr detailliertes Modell erforderlich. Wenn der Ingenieur beispielsweise nur an einer *groben* Anlagenauslegung interessiert ist, macht es keinen Sinn eine sehr genaue Modellbeschreibung zu verwenden, da die überhöhte Modellgenauigkeit voll zu Lasten der Simulationszeit geht. Auf der anderen Seite gibt es aber durchaus Anwendungen, für die eine sehr hohe Modellgenauigkeit eine zwingende Voraussetzung ist, wie z.B. die Untersuchung der Reaktionszeit eines Schaltventils bei schnell wechselnden Steuerimpulsen in einem zeitkritischen Schaltkreis.

Eine *optimale* Modellbildung hängt also im höchsten Maß von der jeweiligen Simulationsintention (-zweck) des Ingenieurs bzw. Anwenders ab, wobei ein Modell als *optimal* anzusehen ist, wenn es den Genauigkeitsanforderungen des vorgegebenen Simulationszweckes genügt und gleichzeitig in seiner Beschreibung möglichst einfach bleibt. Die Aufgabe der automatisierten Modellbildung besteht somit zunächst darin, jeder Komponente in Abhängigkeit vom Simulationszweck eine adäquate Modelltiefe zuzuordnen, wobei die Modelltiefe von Komponente zu Komponente natürlich gemäß der jeweils zur Verfügung stehenden Modelltiefen variieren kann. Ist beispielsweise eine sehr hohe Modellgenauigkeit gefordert und eine Komponente besitzt 4 und eine andere nur 2 Modelltiefen, wird das System in beiden Fällen wahrscheinlich die maximale Modelltiefe also 4 und 2 wählen.

Sind allen Komponenten im betrachteten Schaltkreis Modelltiefen zugeordnet worden, muß der Informationsfluß für den Schaltkreis in Abhängigkeit von den vorgegebenen Ein- und Ausgangsgrößen des Anwenders bestimmt werden. Die Untersuchung des Informationsflusses stellt die vollständige Ein- und Ausgangsgrößendefinition des

Benutzers sicher. Diese Größendefinition ist als vollständig anzusehen, wenn sämtliche Größen der lokalen Differentialgleichungssysteme bestimmt sind bzw. aus den Gleichungen ermittelt werden können. Das Beispiel einer vereinfachten Verhaltensbeschreibung eines Gleichgangzylinders in der Abbildung 4.3 soll die Problematik näher erläutern. In dieser Verhaltensbeschreibung sind die Gleichungen, aufgelöst nach allen Größen, aufgeführt. Im Prinzip stellt dies eine redundante Information dar, die aber in Kauf genommen wird, denn eine automatisierte Umformung der im allgemeinen nichtlinearen Gleichungen nach der gesuchten Größe wäre wesentlich aufwendiger, als aus einer Menge die bereits entsprechend umgeformte Gleichung auszuwählen. Angenommen der Anwender

<u>Modelltiefe 1:</u> Berücksichtigung der Massenträgheit

$$
\begin{aligned}
\overline{\dot{x}_{02}}(t) &= \overline{x_{12}}(t) \\
\overline{\dot{x}_{12}}(t) &= \overline{x_{22}}(t) \\
\overline{x_{01}}(t) &= -\overline{x_{02}}(t) \\
\overline{x_{11}}(t) &= -\overline{x_{12}}(t) \\
\overline{x_{21}}(t) &= -\overline{x_{22}}(t) \\
\overline{x_{22}}(t) &= \frac{A_k/10\,(\underline{p_3}(t) - \underline{p_4}(t)) - \underline{F_2}(t) + \underline{F_1}(t) + d\,\overline{x_{12}}(t)}{m} \\
\overline{x_{11}}(t) &= \frac{\overline{Q_4}(t)}{0.06\,A_k} \\
\overline{x_{12}}(t) &= \frac{\overline{Q_3}(t)}{0.06\,A_k} \\
\overline{Q_3}(t) &= 0.06\,A_k\,\overline{x_{12}}(t) \\
\overline{Q_4}(t) &= 0.06\,A_k\,\overline{x_{11}}(t) \\
\underline{F_1}(t) &= \underline{F_2}(t) - \frac{A_k}{10}\,(\underline{p_3}(t) - \underline{p_4}(t)) \\
\underline{F_2}(t) &= \underline{F_1}(t) + \frac{A_k}{10}\,(\underline{p_3}(t) - \underline{p_4}(t)) \\
\underline{p_3}(t) &= \underline{p_4}(t) + \frac{10}{A_k}\,(\underline{F_2}(t) - \underline{F_1}(t)) \\
\underline{p_4}(t) &= \underline{p_3}(t) - \frac{10}{A_k}\,(\underline{F_2}(t) - \underline{F_1}(t))
\end{aligned}
$$

Abbildung 4.3: Vollständige Verhaltensbeschreibung eines Gleichgangzylinders

hätte für den Gleichgangzylinder die beiden Kräfte $\underline{F_1}(t)$, $\underline{F_2}(t)$ als Eingangsgrößen und die Kolbenposition $\overline{x_{02}}(t)$, sowie Geschwindigkeit $\overline{x_{12}}(t)$ und Beschleunigung $\overline{x_{22}}(t)$ als Ausgangsgrößen definiert, erkennbar an der jeweiligen Über- bzw. Unterstreichung, und der Gleichgangzylinder sei Teil eines Schaltkreises, aus dem die Druckgrößen $\underline{p_3}(t)$, $\underline{p_4}(t)$ hervorgehen, dann reduziert die lokale Untersuchung der Informationsflußbestimmung das Differentialgleichungssystem zu dem in der Abbildung 4.4 dargestellten System. Dieses System enthält keine redundanten Gleichungen mehr und jede lokale Größe des Gleichgangzylinders ist hierin entweder als Eingangsgröße bekannt oder geht eindeu-

tig als Ausgangsgröße aus einer Gleichung hervor, wobei die auf der rechten Seite der Gleichungen auftretenden Ausgangsgrößen zunächst durch die Anfangsbedingung des Anfangswertproblems definiert sind und anschließend aus der numerischen Lösung des Differentialgleichungssystems folgen. So fortfahrend, sind sämtliche lokalen Verhaltensbeschreibungen der Komponenten im betrachteten Schaltkreis zu überprüfen, wobei die Klassifizierung der lokalen Größen als Ein- oder Ausgangsgröße im Schaltkreis entsprechend weitergereicht wird. Das Ergebnis dieser Untersuchung repräsentiert dann den Informationsfluß im Schaltkreis. Kann er vollständig bestimmt werden, läßt sich anschließend aus den lokalen Differentialgleichungssystemen das Gesamtsystem aufstellen und mit dem bekannten Anfangsbedingungen das entsprechende Anfangswertproblem numerisch lösen.

Modelltiefe 1: Berücksichtigung der Massenträgheit

$$
\begin{aligned}
\overline{\dot{x}_{02}}(t) &= \overline{x_{12}}(t) \\
\overline{\dot{x}_{12}}(t) &= \overline{x_{22}}(t) \\
\overline{x_{01}}(t) &= -\overline{x_{02}}(t) \\
\overline{x_{11}}(t) &= -\overline{x_{12}}(t) \\
\overline{x_{21}}(t) &= -\overline{x_{22}}(t) \\
\overline{x_{22}}(t) &= \frac{A_k/10\,(\underline{p_3}(t) - \underline{p_4}(t)) - \underline{F_2}(t) + \underline{F_1}(t) + d\,\overline{x_{12}}(t)}{m} \\
\overline{Q_3}(t) &= 0.06\,A_k\,\overline{x_{12}}(t) \\
\overline{Q_4}(t) &= 0.06\,A_k\,\overline{x_{11}}(t)
\end{aligned}
$$

Abbildung 4.4: Reduzierte Verhaltensbeschreibung eines Gleichgangzylinders

4.2 Bestimmung der Modelltiefe von Komponenten

Um dem Leser einen allgemeinen Überblick über die heuristische Modelltiefenbestimmung zu geben, wird zunächst die Grundidee und prinzipielle Vorgehensweise erläutert. Dieser Darstellung schließt sich eine Beschreibung der einzelnen Einflußgrößen und ihrer Bedeutung in der Heuristik an. Ihre Repräsentation und den Inferenzmechanismus stellt der Unterabschnitt 4.2.3 vor. Abschließend wird dann das heuristische Vorgehen geschlossen als Algorithmus formuliert.

4.2.1 Grundidee, Prinzipielle Vorgehensweise

Zunächst sucht man im gegebenen Schaltkreis die Komponente, deren dynamisches Verhalten am wahrscheinlichsten und stärksten von ihrem statischen Verhalten abweicht, also erwartungsgemäß ein maximal zeitverzögerndes Verhalten aufweist. Diese Komponente stellt die höchsten Ansprüche bezüglich der Modellierungsgenauigkeit, denn das

ihr zugewiesene Modell muß in der Lage sein, alle geforderten, dynamischen Details wiederzugeben. Die Komponente kann über ihre Eigenfrequenz ν identifiziert werden, wobei diejenige mit der minimalsten Eigenfrequenz ν_{min} am ehesten ein Schwingungs- und damit auch ein zeitverzögerndes Verhalten zeigt. Die Klassifizierung der hydraulischen Komponenten in *Arbeits-*, *Steuer-*, *Versorgungs-* und *Verbindungselemente* definiert hierbei die Suchordnung und spiegelt gleichzeitig auch die dynamische Gewichtung der einzelnen Komponentenklassen wieder, d.h. den Arbeitselementen kommt aufgrund ihrer im allgemeinen größeren Massenträgheit auch die größte, den Verbindungselementen entsprechend die geringste dynamische Bedeutung zu.

Um die Simulationszeit in einem vernünftigen Rahmen zu halten, ist dann aufgrund des im Schaltkreis auftretenden Eigenfrequenzspektrums zu entscheiden, welche Komponenten in ihrem zeitverzögernden Verhalten vernachlässigbar sind, also einfach durch ihr statisches Modell beschrieben werden können. Diese Grenzeigenfrequenz, im folgenden auch als *Abschneidefrequenz* ν_{cut} bezeichnet, hängt hauptsächlich von der Simulationsintention des Anwenders ab, so daß für sie kein allgemeingültiger Wert existiert. Ihre Bestimmung orientiert sich an Erfahrungswerten aus der Hydraulik, m.a.W. der Wert wird entsprechend heuristisch vorbesetzt. Auf diese Weise entstehen zwei unterschiedlich zu wertende Mengen von Komponenten im Schaltkreis. Die eine Menge enthält alle Komponenten, die eine größere Eigenfrequenz als die Abschneidefrequenz besitzen und somit durch ihr statisches Modell beschrieben werden und die andere Menge die restlichen Komponenten des Schaltkreises, denen noch ein dynamisches Modell zugeordnet werden muß. Die Abbildung 4.5 veranschaulicht diesen beschriebenen Zusammenhang.

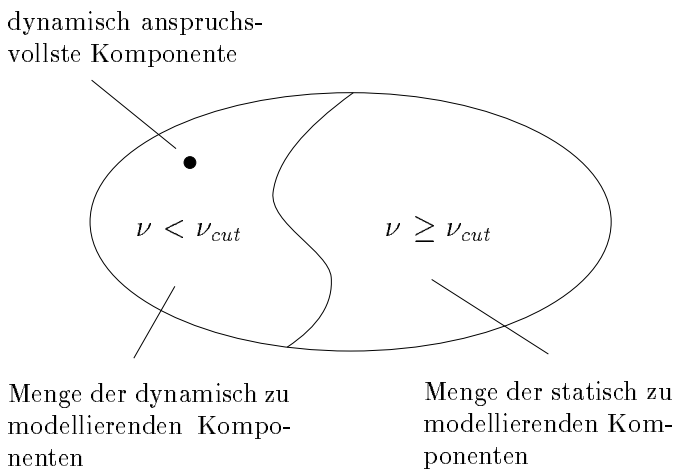


Abbildung 4.5: Statisch und dynamisch zu modellierende Komponenten

Im folgenden ist demnach nur noch die Menge der dynamisch zu modellierenden Komponenten zu berücksichtigen. Aus dieser Menge wird man beginnend mit der dynamisch anspruchslosten bis hin zur dynamisch anspruchsvollsten Komponente sukzessive den einzelnen Komponenten in der Reihenfolge abnehmender Eigenfrequenzen heuristisch Modelltiefen zuordnen. Anhand der Abbildung 4.6 soll dieses Vorgehen näher erläutert werden. Sie zeigt schematisch die Komponenten (Knoten) eines Schaltkreises,

die dynamisch zu modellieren sind und die Reihenfolge in der die Knoten bis zum 4-ten Schritt besucht, d.h. den Komponenten eine Modelltiefe zugeordnet wird. Beginnend mit der dynamisch anspruchslosten Komponente (Knoten 1) bestimmt man alle mit ihr verbundenen Komponenten, denen noch keine Modelltiefe zugeordnet worden ist, also die Knoten 2 und 4. Als nächstes ist dann unter diesen Knoten derjenige mit der größten Eigenfrequenz zu besuchen, in diesem Fall der mit 2 markierte Knoten. Dieser Knoten stellt nun die dynamisch anspruchsloste Komponente im restlichen Schaltkreis dar und bildet den Ausgangspunkt des nächsten Zuordnungsschrittes, in dem aus den noch nicht betrachteten Knoten 3, 4 wiederum die Komponente mit der größten Eigenfrequenz gewählt wird, also der Knoten 3. So fortfahrend gelangt man schließlich zum dynamisch anspruchsvollsten Knoten n, nach dessen Bearbeitung der Schaltkreis dann bezüglich der Modelltiefenzuordnung vollständig bestimmt ist.

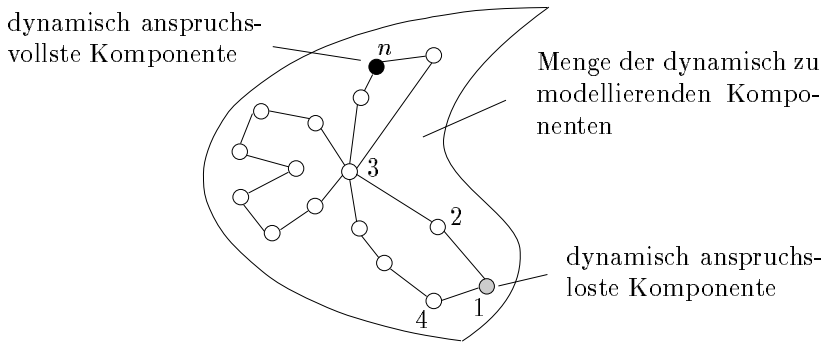


Abbildung 4.6: Sukzessive Modelltiefenzuordnung

Die Idee dieser Vorgehensweise beruht auf der Überlegung, daß die Zuordnung einer adäquaten Modelltiefe bei der dynamisch anspruchslosten Komponente weniger kritisch ist als bei allen übrigen Komponenten. Ihr Wert hängt im wesentlichen nur von der Anzahl der zur Verfügung stehenden Modelltiefen und der Simulationsintention des Anwenders ab. Auf dem Weg zur Komponente mit der niedrigsten Eigenfrequenz fließen dann die zuvor verteilten Modelltiefen als ein Kriterium neben anderen bei der aktuellen Modelltiefenauswahl mit ein. Die Zuordnung wird somit von Komponente zu Komponente immer differenzierter. Gleichzeitig versucht dieses Vorgehen große Sprünge in den Modellstufen entgegenzuwirken, um eine möglichst ausgewogene Modelltiefenverteilung im Schaltkreis zu erhalten.

Die oben beschriebene Vorgehensweise bei der Modelltiefenbestimmung legt die Vermutung nahe, daß für jeden Knoten ein Ableitungsprozeß stattfindet, der als Ergebnis die Modelltiefe der Komponente liefert. Diese Methode hätte jedoch den Nachteil, daß der Aufwand der heuristischen Modelltiefenbestimmung bezogen auf den Gesamtaufwand proportional mit der Größe des Schaltkreises, d.h. der Anzahl der Komponenten ansteigt. Sinnvoller ist es daher, den Inferenzmechanismus einmal vor Beginn des Graphendurchlaufes durchzuführen und alle nötigen Fakten abzuleiten, die später in die lokale Modelltiefenzuordnung einfließen. Nach diesem Paradigma ist hier auch die Modelltiefenbestimmung realisiert. Noch bevor die Abschneidefrequenz ermittelt wird, erfolgt der Ableitungsprozeß, der neben dem zur Bestimmung der Abschneidefrequenz

notwendigen Wissen auch alle anderen Fakten bereitstellt. Im wesentlichen handelt es sich hierbei um Wertbelegungen von Bewertungsfaktoren, die gemeinsam mit gewissen Komponentenparametern, sogenannten *Einflußfaktoren*, die Basis für die Bewertung einer Komponente hinsichtlich der ihr zuzuordnenden Modelltiefe bilden. Die eigentliche Modelltiefenberechnung erfordert dann für jede Komponente nur noch einen Funktionsaufruf, der als Ergebnis die entsprechende Modelltiefe liefert. Eine ausführliche Darlegung dieser heuristischen Modelltiefenbestimmung findet der Leser im anschließenden Unterabschnitt.

4.2.2 Einsatz von heuristischem Wissen

Bei der Modelltiefenbestimmung wird sowohl auf regelbasiertes als auch auf modellbasiertes Wissen zurückgegriffen. Erst die Kombination beider Wissensbereiche erlaubt eine am Hydraulik-Experten orientierte Modellbildung, wobei das modellbasierte Wissen neben den Benutzervorgaben einen Teil der Basisfakten bildet. Gestützt auf diese Fakten wird dann heuristisch jeder Komponente des betrachteten Schaltkreises eine Modelltiefe zugewiesen und so das Gesamtmodell konzipiert. Im folgenden soll zunächst das fallspezifische Wissen der Heuristik vorgestellt werden ohne hierbei zwischen regel- und modellbasiertem Wissen zu unterscheiden. Dieser Darstellung schließt sich dann eine Erläuterung sämtlicher Teilaspekte des heuristischen Vorgehens an.

Basiswissen, Einflußfaktoren

- *Simulationszweck SI, Menge der Simulationszwecke* $\mathbb{SI}$

  Der Simulationszweck stellt ein Maß für die vom Anwender geforderte Untersuchungs- und damit Modellierungsgenauigkeit dar und ist durch die Menge der aktuell zur Verfügung stehenden Simulationszwecke $\mathbb{SI}$ beschränkt, d.h. es gilt stets $SI \in \mathbb{SI}$. Diese Menge kann ebenfalls vom Anwender aus einer endlichen Anzahl von Alternativen gewählt werden, ist aber im allgemeinen vom Softwaresystem vordefiniert. Derzeit sind 6 Alternativen nämlich $\mathbb{SI}_2, \mathbb{SI}_3, \ldots, \mathbb{SI}_7$ vorgesehen, wobei der jeweilige Index die Kardinalität der Menge anzeigt. Die Kardinalität bestimmt die Auflösung der Simulationszweckabstufung, die entsprechend mit steigendem Wert zunimmt. Ihr Wert geht sowohl bei der Bestimmung der Abschneidefrequenz als auch bei der Festlegung der Bewertungsfaktoren ein und hat somit wesentlichen Einfluß auf die Heuristik.

- *Suchordnung SO:*

  Die Suchordnung 4.1 basiert auf der bereits erwähnten Klassifizierung der Komponenten. Sie spiegelt die Gewichtung der hydraulischen Elemente hinsichtlich ihrer dynamischen Relevanz bei der Suche nach der dynamisch anspruchsvollsten Komponente im Schaltkreis wieder und kann wie die Menge der Simulationszwecke vom Benutzer wahlweise vorgegeben werden. Falls keine Definition erfolgt, gilt die Default-Ordnung 4.1.

$$\text{Arbeits-} \rightarrow \text{Steuer-} \rightarrow \text{Versorgungs-} \rightarrow \text{Verbindungselemente.} \tag{4.1}$$

- *Einflußfaktoren:*

  Die Einflußfaktoren repräsentieren das bei der Modelltiefenbestimmung einfliessende modellbasierte Wissen. Dieses Wissen besteht zum einen aus Komponentenparametern und zum anderen aus Schaltkreisinformationen.

  - *Eigenfrequenz ν:* Die Eigenfrequenz ist das bedeutendste Merkmal, gemäß dem das dynamische Verhalten einer Komponente beurteilt werden kann. Je geringer ihr Wert, desto größer ist die Trägheit der Komponente, d.h. sie kann schnellen Eingangssignalen nur zeitverzögert oder gar nicht mehr folgen. Komponenten mit niedrigen Eigenfrequenzen sind demnach dynamisch genauer zu modellieren, um das zeitverzögernde Verhalten besser aufzulösen. Dementgegen stellen Komponenten mit hohen Eigenfrequenzen im allgemeinen niedrigere Ansprüche bezüglich der Modellierungsgenauigkeit.
  - *Verstärkung K:* Die Verstärkung kennzeichnet den Einfluß des dynamischen Verhaltens einer Komponente auf das Verhalten des Gesamtsystems. Der Verstärkungsfaktor stellt demnach ein Maß für den Grad der Einflußnahme dar. Betrachtet man nur die Verstärkung, ist eine Komponente umso genauer zu modellieren, je höher ihr Verstärkungsfaktor ist.
  - *Systemordnung N:* Die Systemordnung gibt die maximale Anzahl der Energiespeicher, also die Anzahl der energiespeichernden Zustandsgrößen, einer Komponente an und ist identisch mit der Anzahl der Differentialgleichungen. Mit steigender Systemordnung nimmt die Komplexität des dynamischen Verhaltens der Komponente zu, so daß entsprechend eine höhere Modellierungsgenauigkeit erforderlich ist.
  - *Nichtlinearität NL:* Dieser Einflußfaktor beurteilt, inwieweit das dynamische Verhalten einer Komponente als linear oder überwiegend nichtlinear einzustufen ist. Höhere Nichtlinearitätswerte stellen im allgemeinen auch höhere Ansprüche an das dynamische Modell, das in der Lage sein muß, das nichtlineare Verhalten nachzubilden.
  - *Schaltkreisstruktur:* Der Schaltkreisstruktur kommt in Bezug auf den oben beschriebenen Einflußfaktoren eine besondere Bedeutung zu. Sie ist vor Beginn des Modellbildungsprozesses z.B. mit dem in [Hoff93] vorgestellten Algorithmus zur Strukturverarbeitung zu analysieren. Ergibt die Topologieuntersuchung, daß der Schaltkreis eine einfache Baumstruktur[8] besitzt, was in praktischen Anwendungen der überwiegende Fall sein wird, entsprechen die Einflußfaktoren den jeweiligen Komponentenparametern. Im anderen Fall, wenn also Parallel- oder Netzstrukturen auftauchen, werden die Einflußfaktoren mit den aus der identifizierten Struktur berechneten Werten besetzt, wobei ein vernetzter Teilschaltkreis vereinfachend wie ein paralleler behandelt wird. Die Strukturwerte der Einflußfaktoren berechnen sich aus den jeweiligen Parametern der Komponenten innerhalb der Struktur. Die

[8] Ein Schaltkreis besitzt eine *Baumstruktur*, wenn das zugehörige Widerstandsnetz mit Flußquellen und -senken keine Zyklen enthält, eine *Parallelstruktur*, wenn Zyklen vorhanden sind und eine *Netzstruktur*, wenn zusätzlich Subzyklen auftreten (vgl. [Hoff93]).

Verstärkung einer Parallelstruktur ergibt sich beispielsweise als Summe aus den einzelnen Verstärkungsfaktoren der parallel geschalteten Komponenten. Die Abbildung 4.7 zeigt exemplarisch zwei einfache Schaltkreise der in diesem Kontext unterschiedenen Strukturen. Der linke Schaltkreis repräsentiert den einfachen Fall einer baumartigen Komposition. Der rechte Schaltkreis enthält zwei parallel geschaltete 4/3 Wege-Proportionalventile, die eine Substruktur im Schaltkreis bilden. Gemäß den Vorbemerkungen gelten für diese Ventile die Einflußfaktoren der Parallelstruktur. Der Verstärkungsfaktor der beiden Ventile ergibt sich so aus der Summe der beiden Verstärkungsfaktoren der Komponenten.

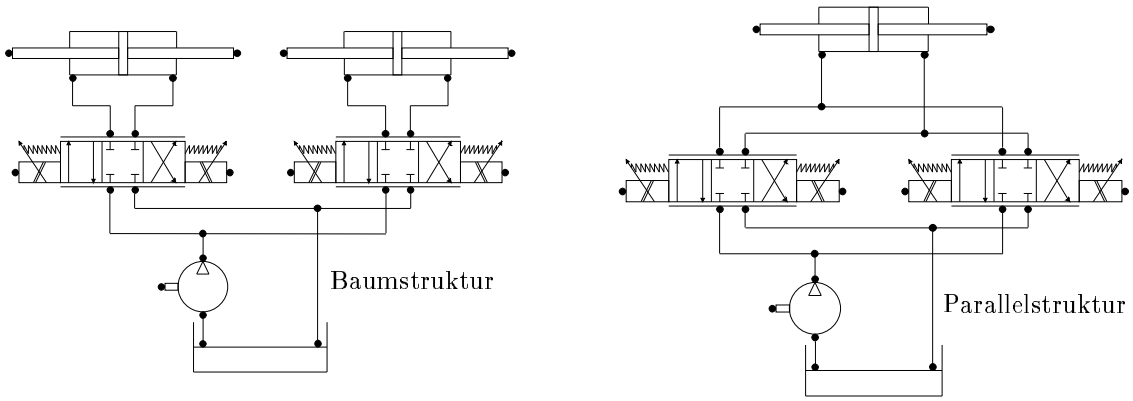


Abbildung 4.7: Einfache Schaltkreisstrukturen

- *Durchschnittlich gewählte, relative Modelltiefe* $\overline{MD}_{rel}$*:* Die Größe wird in diesem Kontext als ein Schaltkreisfaktum aufgefaßt und stellt neben der Eigenfrequenz den mit am wichtigsten bewerteten Einflußfaktor dar. Er berechnet sich durch Mittelwertbildung der im betrachteten Schaltkreis bereits verteilten relativen Modelltiefen und wird laufend aktualisiert. Stehen beispielsweise für eine Komponente 5 Modelltiefen zur Verfügung und hat die Heuristik für die Komponente die Modelltiefe 3 ermittelt, beträgt der aktuelle Wert von $MD_{rel} = \frac{3}{5}$. Dieser Wert wird dann mit dem bisherigen Mittelwert $\overline{MD}_{rel}$ verrechnet. Dadurch daß die Modelltiefenbestimmung von der dynamisch anspruchslosten hin zur dynamisch anspruchsvollsten Komponente verläuft, spiegelt der Mittelwert die durchschnittliche „Verteilung“ der gewählten Modelltiefen wieder und trägt so dazu bei, daß eine möglichst ausgewogene Modelltiefenzuordnung im Schaltkreis erfolgt.

Bestimmung der Abschneidefrequenz

Die Abschneidefrequenz ν_{cut} dient, wie bereits geschildert, zur Ermittlung der Komponenten in einem Schaltkreis, deren Verhalten bei der Simulationsberechnung durch ihr statisches Modell beschrieben wird. Sie ist abhängig vom jeweils gewählten Simulationszweck SI, der Menge der verfügbaren Simulationszwecke $\mathbb{SI}$ sowie der Eigenfrequenz ν_{min} der dynamisch anspruchsvollsten Komponente im Schaltkreis und berechnet sich nach der Beziehung 4.2.

$$\nu_{cut} = \beta_{cut}\, \nu_{min} \tag{4.2}$$

Die Abschneidekonstante β_{cut} definiert in dieser Gleichung das Eigenfrequenzspektrum der dynamisch zu berücksichtigenden Komponenten und wird gemäß der Tabelle 4.1 bestimmt. Diese Tabelle enthält das erforderliche Erfahrungswissen zur Festlegung des relevanten Frequenzspektrums und kann je nach Anwendungssituation vom Systemverwalter beliebig ergänzt bzw. modifiziert werden. Das Spektrum nimmt erwartungs-

	$\lvert\mathbb{SI}\rvert$					
SI	2	3	4	5	6	7
1	5	5	5	5	5	5
2	40	20	10	10	10	10
3	–	100	30	20	20	20
4	–	–	100	50	40	30
5	–	–	–	100	100	50
6	–	–	–	–	500	100
7	–	–	–	–	–	1000

Tabelle 4.1: Wertetabelle der Abschneidekonstanten β_{cut}

gemäß mit steigender Kardinalität von $\mathbb{SI}$ zu und kommt somit einer höheren Modellierungsgenauigkeit entgegen, wobei die Kardinalität aus den Zahlenwerten in der Kopfzeile der Tabelle hervorgeht. Die Zahlenwerte der Kopfspalte bezeichnen entsprechend den Index des gewählten Simulationszwecks.

Bestimmung der Bewertungsfaktoren

Die Bewertungsfaktoren $\nu_{val}, MD_{val}, K_{val}, N_{val}, NL_{val} \in [0,1]$ mit

$$\nu_{val} + MD_{val} + K_{val} + N_{val} + NL_{val} = 1 \tag{4.3}$$

bieten dem Anwender die Möglichkeit, die entsprechend bezeichneten Einflußfaktoren bei der Modelltiefenauswahl je nach Simulationsintention und Menge der verfügbaren

Simulationszwecke noch verschieden stark zu gewichten, so daß hier anwendungsbezogen Schwerpunkte gesetzt werden können. Die Tabelle 4.2 zeigt die Default-Werte der Bewertungsfaktoren in Abhängigkeit von SI und $\mathbb{SI}$. Dieser Wertebelegung liegt

Bedingung $(SI, \mathbb{SI})$	ν_{val}	MD_{val}	K_{val}	N_{val}	NL_{val}
$SI < \lfloor \frac{1}{2} \lvert \mathbb{SI} \rvert \rfloor$	0.5	0.5	0.0	0.0	0.0
$\lfloor \frac{1}{2} \lvert \mathbb{SI} \rvert \rfloor \leq SI < \lfloor \frac{2}{3} (\lvert \mathbb{SI} \rvert + 1) \rfloor$	0.2	0.5	0.3	0.0	0.0
$\lfloor \frac{2}{3} (\lvert \mathbb{SI} \rvert + 1) \rfloor \leq SI < \lvert \mathbb{SI} \rvert$	0.2	0.4	0.2	0.2	0.0
$SI = \lvert \mathbb{SI} \rvert$	0.1	0.4	0.1	0.1	0.3

Tabelle 4.2: Wertetabelle der Bewertungsfaktoren

die folgende Überlegung zugrunde: Unabhängig vom Simulationszweck und der Menge der Simulationszwecke wird die Eigenfrequenz einer Komponente und die durchschnittlich gewählte, relative Modelltiefe stets bei der Modelltiefenauswahl, wenn auch unterschiedlich stark, berücksichtigt. Die Verstärkung, Systemordnung und Nichtlinearität hingegen, finden erst bei höherer Untersuchungsgenauigkeit Berücksichtigung. Der durch eine höhere Untersuchungsgenauigkeit neu hinzukommende Einflußfaktor wird dann im Vergleich zu den bisherigen Einflußfaktoren mindestens gleichstark wenn nicht sogar stärker gewichtet, um so die durch diese Einflußfaktoren beschriebenen Eigenschaften der Komponente aufzulösen.

Modelltiefenzuordnung

Bei der Modelltiefenzuordnung einer Komponente unterscheidet man grundsätzlich zwei Fälle. Der erste Fall beschreibt die initiale Modelltiefenbestimmung bei der dynamisch anspruchslosten Komponente und der zweite die Modelltiefenbestimmung aller restlichen Komponenten im Schaltkreis. Im folgenden sind die Zuordnungsvorschriften als Funktionen mit $f : \mathbb{R} \rightarrow \{1, 2, \ldots, \lvert \mathbb{MD} \rvert\}$ dargestellt, wobei die Anzahl der verfügbaren Modelltiefen $\lvert \mathbb{MD} \rvert$ natürlich von Komponente zu Komponente variieren kann.

1. *Modelltiefenzuordnung bei der dynamisch anspruchslosten Komponente:*

$$MD = \begin{cases} \left\lfloor SI \frac{\lvert \mathbb{MD} \rvert}{\lvert \mathbb{SI} \rvert} \right\rfloor & \text{falls } \lvert \mathbb{MD} \rvert > \lvert \mathbb{SI} \rvert \\ \left\lceil SI \frac{\lvert \mathbb{MD} \rvert}{\lvert \mathbb{SI} \rvert} \right\rceil & \text{sonst,} \end{cases} \tag{4.4}$$

2. *Allgemeine Modelltiefenzuordnung:*

$$\begin{aligned} MD = \ & \text{round}\Big(\lvert \mathbb{MD} \rvert \Big((1 - \frac{\nu}{\nu_{cut}}) \, \nu_{val} \\ & + \overline{MD}_{rel} \, MD_{val} + \frac{K}{K_{max}} \, K_{val} + \frac{N}{N_{max}} \, N_{val} + \frac{NL}{NL_{max}} \, NL_{val} \Big) \Big) \end{aligned} \tag{4.5}$$

Mit Hilfe dieser Funktionen werden die Modelltiefen der einzelnen Komponenten im Schaltkreis in Abhängigkeit von den Einflußfaktoren und ihrer Bewertung sowie ihren Normierungswerten K_{max}, N_{max}, NL_{max} bestimmt. Die Normierungen stellen den maximal auftretenden Wert des entsprechenden Einflußfaktors im Schaltkreis dar. Zur Ermittlung der Werte muß die Komponente mit dem jeweiligen, maximalen Parameter- bzw. dem maximalen Strukturwert im Schaltkreis gesucht werden. Die Einflußfaktoren in der Gleichung 4.5 können ebenso wie die Normierungen Werte der jeweiligen Komponentenparameter oder der jeweiligen Struktur sein, je nachdem ob der Schaltkreis eine Baum- oder Parallel- bzw. Netzstruktur besitzt.

Die initiale Modelltiefenzuordnung nach 4.4 wird einzig von der Simulationsintention, der Menge der Simulationszwecke sowie der Modelltiefenanzahl abhängig gemacht, um hier richtungsweisend für alle nachfolgenden Komponenten eine Modelltiefe zu ermitteln, die stärker von der Simulationsintention des Anwenders geprägt ist als von allen anderen Einflußfaktoren. Sie ist insofern richtungsweisend, als daß die berechnete, relative Modelltiefe dieser Komponente gleichzeitig den Anfangswert von $\overline{MD}_{rel}$ darstellt und dieser Einflußfaktor im Vergleich zu den anderen Einflußfaktoren in der Funktion 4.5 ein größeres Gewicht zukommt. Die Vernachlässigung der Einflußfaktoren in der Modellzuordnungsvorschrift 4.4 ist hierbei mit den geringeren Ansprüchen der Komponente bezüglich der Modellierunggenauigkeit zu rechtfertigen.

4.2.3 Wissensrepräsentation, Inferenzmechanismus

Das heuristische Domänenwissen umfaßt die Werte der Abschneidefrequenz und der Bewertungsfaktoren. Diese Werte werden in Abhängigkeit vom Simulationszweck und der Menge der verfügbaren Simulationszwecke definiert und sind im Softwaresystem in Form von aussagenlogischen Regeln formuliert, die um die Möglichkeit von quantitativen Aussagen und Funktionsaufrufen im Konklusionsteil einer Regel erweitert worden sind. Die Syntax der Regelsprache orientiert sich an der LISP-Syntax, so daß die Regelsprache einfach in LISP zu realisieren ist und den Vorteil einer schnellen Regelverarbeitung bietet. Ein Beispiel einer Regel zur Bestimmung der Abschneidekonstante β_{cut} zeigt die Abbildung 4.8. Die vollständige Regelmenge und die formale Syntaxdefinition der Regelsprache findet der Leser im Anhang B.

```
(rule-cut-1 (IF (AND (>= simulation.intention.number 2)
                     (<= simulation.intention.number 7)
                     (= simulation.intention 1)))
            (THEN (cut.constant 5)))
```

Abbildung 4.8: Regel zur Bestimmung der Abschneidekonstante β_{cut}

Eine Bedingung einer Regel ist genau dann erfüllt, wenn das Faktum in der Wissensbasis enthalten ist, oder der Boolean-Ausdruck `TRUE` zurückgibt. Für den Fall, daß im Boolean-Ausdruck ein Faktum auftritt, das nicht in der Datenbasis vorkommt, ergibt die Auswertung des Ausdrucks entsprechend `NIL`.

Der Aufruf einer Funktion im Konklusionsteil einer Regel setzt sich aus dem Schlüsselwort `CALL-FUNCTION` gefolgt von dem Funktionsnamen zusammen. Übergabeparameter sind nicht vorgesehen und werden auch nicht benötigt, da die Funktionen nur auf den Daten der Wissensbasis operieren. Als Ergebnis fügt die Funktion dann die ermittelten Fakten, soweit noch nicht vorhanden, in die Datenbasis ein.

Die Regelmenge beschreibt eine heuristische Parameterbelegung. Die Parameterwerte fließen u.a. in die Bewertungsfunktion ein, mit der dann einer Komponente eine Modelltiefe zugeordnet wird. Da stets die Werte sämtlicher Parameter bekannt sein müssen, bietet sich vom Prinzip her das *Forward-Chaining* als Inferenzmechanismus an und wird hier auch eingesetzt.

Die Bewertungsfunktionen 4.4 und 4.5 sind nach außen, d.h. für den Anwender, nicht sichtbar. Aufgrund ihrer Konzipierung unterliegen sie keiner strukturellen Änderung, so daß sie fest kodiert sind.

4.2.4 Der Algorithmus

Hat der Anwender den Simulationszweck und eventuell die Menge der verfügbaren Simulationszwecke sowie die Suchordnung vorgegeben, werden die Modelltiefen der Komponenten im selektierten Teilschaltkreis nach dem folgenden Algorithmus bestimmt. Der Algorithmus setzt für eine Parallel- bzw. Netzstruktur des Schaltkreises eine Berechnung der entsprechenden Strukturwerte für die Einflußfaktoren voraus, um im Schritt 10b eine im Sinne der Heuristik korrekte Modelltiefe zu bestimmen.

Bei der Suche nach der dynamisch anspruchslosten Komponente im Schritt 7 und 10a dient, wie bereits beschrieben, die minimale Eigenfrequenz einer Komponente als primärer Suchschlüssel. Falls mehrere Komponenten gleicher, minimaler Eigenfrequenz auftauchen, ist diejenige Komponente zu wählen, die den geringsten Verstärkungsfaktor besitzt. Sollte für diesen Parameter ebenfalls Gleichheit vorliegen, bestimmt die kleinere Systemordnung der beiden Komponenten die dynamisch anspruchsloste Komponente. Falls immer noch keine Unterscheidung möglich ist, wird die Komponente mit der kleineren Nichtlinearität gewählt. Bei gleichen Nichtlinearitätswerten gibt letztlich die Reihenfolge der Komponenten den Ausschlag.

1. Initialisiere die Wissensbasis mit den gegebenen bzw. vordefinierten Fakten $|\mathbb{SI}|$, SI und SO.

2. Bestimme heuristisch mittels forward-chaining Werte für die Parameter β_{cut}, ν_{val}, MD_{val}, K_{val}, N_{val} und NL_{val}.

3. Suche im Teilschaltkreis die Komponente mit dem maximalen Verstärkungsfaktor, der maximalen Systemordnung und der maximalen Nichtlinearität, und initialisiere die Parameter K_{max}, N_{max} und NL_{max} mit den entsprechenden Werten.

4. Suche im Teilschaltkreis gemäß der in Schritt 1 aufgeführten Ordnung die Komponente mit der kleinsten Eigenfrequenz, und initialisiere den Parameter ν_{min} mit dem entsprechenden Wert.

5. Berechne die Abschneidefrequenz zu

$$\nu_{cut} = \beta_{cut}\, \nu_{min}.$$

6. Ordne allen Komponenten im Teilschaltkreis mit einer Eigenfrequenz $\nu \geq \nu_{cut}$ die Modelltiefe $MD = 0$ zu.

7. Suche im Teilschaltkreis die dynamisch anspruchsloste Komponente.

8. Wähle für diese Komponente eine Modelltiefe gemäß

$$MD = \begin{cases} \left\lfloor SI \frac{|\mathbb{MD}|}{|\mathbb{SI}|} \right\rfloor & \text{falls } |\mathbb{MD}| > |\mathbb{SI}| \\ \left\lceil SI \frac{|\mathbb{MD}|}{|\mathbb{SI}|} \right\rceil & \text{sonst,} \end{cases}$$

   und fasse alle Komponenten, die mit dieser Komponente verbunden sind und denen noch keine Modelltiefe zugeordnet worden ist, in der Menge $\mathbb{CC}$ zusammen.

9. Initialisiere die durchschnittlich gewählte, relative Modelltiefe mit

$$\overline{MD}_{rel} = \frac{MD}{|\mathbb{MD}|}$$

   und setze die Anzahl der Komponenten, deren Modelltiefe bereits bestimmt ist, auf $cnt = 1$.

10. Wiederhole solange $\mathbb{CC} \neq \emptyset$ gilt:

    (a) Suche aus der Menge $\mathbb{CC}$ die dynamisch anspruchsloste Komponente heraus.

    (b) Ordne dieser Komponente in Abhängigkeit ihrer Parameter ν, K, N und NL die Modelltiefe

$$\begin{aligned} MD = \text{round}\Big(|\mathbb{MD}| \Big(& (1 - \frac{\nu}{\nu_{cut}})\, \nu_{val} \\ & + \overline{MD}_{rel}\, MD_{val} + \frac{K}{K_{max}}\, K_{val} + \frac{N}{N_{max}}\, N_{val} + \frac{NL}{NL_{max}}\, NL_{val} \Big) \Big) \end{aligned}$$

    zu.

    (c) Aktualisiere die durchschnittlich gewählte, relative Modelltiefe

$$\overline{MD}_{rel} = \frac{\overline{MD}_{rel}\, cnt + \frac{MD}{|\mathbb{MD}|}}{cnt + 1}.$$

    (d) Setze $cnt = cnt + 1$

4.3 Bestimmung des Informationsflusses

Die grundlegende Problematik der Informationsflußbestimmung ist bereits im Unterabschnitt 4.1 am Beispiel einer vereinfachten Verhaltensbeschreibung eines Gleichgangzylinders vorgestellt worden. Dieser Abschnitt greift das Beispiel wieder auf und beschreibt die allgemeine Vorgehensweise, nach der eine redundante Verhaltensbeschreibung in Abhängigkeit von den Benutzervorgaben in eine problemangepaßte Beschreibung, d.h. reduzierte Form, transformiert wird.

Seien die Ein- und Ausgangsgrößen wie im Unterabschnitt 4.1 vorausgesetzt, d.h. seien $\underline{F_1}(t)$, $\underline{F_2}(t)$ als Eingangsgrößen und $\overline{x_{02}}(t)$, $\overline{x_{12}}(t)$, $\overline{x_{22}}(t)$ als Ausgangsgrößen klassifiziert, dann präsentiert sich die initiale Verhaltensbeschreibung wie in der Abbildung 4.9. Ausgehend von dieser Verhaltensbeschreibung sind die unbekannten Größen

<u>Modelltiefe 1:</u> Berücksichtigung der Massenträgheit

$$\overline{\dot{x}_{02}}(t) = \overline{x_{12}}(t) \tag{4.6}$$
$$\overline{\dot{x}_{12}}(t) = \overline{x_{22}}(t) \tag{4.7}$$
$$x_{01}(t) = -\overline{x_{02}}(t) \tag{4.8}$$
$$x_{11}(t) = -\overline{x_{12}}(t) \tag{4.9}$$
$$x_{21}(t) = -\overline{x_{22}}(t) \tag{4.10}$$
$$\overline{x_{22}}(t) = \frac{A_k/10\,(p_3(t) - p_4(t)) - \underline{F_2}(t) + \underline{F_1}(t) + d\,\overline{x_{12}}(t)}{m} \tag{4.11}$$
$$x_{11}(t) = \frac{Q_4(t)}{0.06\,A_k} \tag{4.12}$$
$$\overline{x_{12}}(t) = \frac{Q_3(t)}{0.06\,A_k} \tag{4.13}$$
$$Q_3(t) = 0.06\,A_k\,\overline{x_{12}}(t) \tag{4.14}$$
$$Q_4(t) = 0.06\,A_k\,x_{11}(t) \tag{4.15}$$
$$\underline{F_1}(t) = \underline{F_2}(t) - \frac{A_k}{10}\,(p_3(t) - p_4(t)) \tag{4.16}$$
$$\underline{F_2}(t) = \underline{F_1}(t) + \frac{A_k}{10}\,(p_3(t) - p_4(t)) \tag{4.17}$$
$$p_3(t) = p_4(t) + \frac{10}{A_k}\,(\underline{F_2}(t) - \underline{F_1}(t)) \tag{4.18}$$
$$p_4(t) = p_3(t) - \frac{10}{A_k}\,(\underline{F_2}(t) - \underline{F_1}(t)) \tag{4.19}$$

Abbildung 4.9: Initiale Verhaltensbeschreibung des Gleichgangzylinders

zu klassifizieren und dann das System zu reduzieren. Die Klassifizierung muß hierbei mit den folgenden *Markierungsregeln* konform sein:

1. Alle Größen, die auf der linken Gleichungsseite stehen, sind als Ausgangsgrößen zu markieren, falls alle Größen der rechten Seite bereits markiert sind.

2. Eingangsgrößen dürfen nur auf der rechten Seite einer Gleichung stehen.

3. Ausgangsgrößen dürfen, falls sie differentielle Größen sind, auch auf der rechten Gleichungsseite auftauchen.

Die Anwendung der Regel 1 auf die Gleichung 4.14 sowie 4.8, 4.9 und 4.10 klassifiziert den Volumenstrom $\overline{Q_3}(t)$ sowie die Position $\overline{x_{01}}(t)$, die Geschwindigkeit $\overline{x_{11}}(t)$ und Beschleunigung $\overline{x_{21}}(t)$ eindeutig als Ausgangsgrößen. Als Konsequenz dieser Markierungen kann dann auch der Volumenstrom $\overline{Q_4}(t)$ in Gleichung 4.15 als Ausgangsgröße identifiziert werden, so daß sich die initiale Verhaltensbeschreibung zu dem in der Abbildung 4.10 dargestellten, teilbestimmten Differentialgleichungssystem ergibt. In diesem System sind nur noch die Drücke $p_3(t)$, $p_4(t)$ unbekannt. Ihre Klassifizierung

Modelltiefe 1: Berücksichtigung der Massenträgheit

$$\overline{\dot{x}_{02}}(t) = \overline{x_{12}}(t) \tag{4.20}$$
$$\overline{\dot{x}_{12}}(t) = \overline{x_{22}}(t) \tag{4.21}$$
$$\overline{x_{01}}(t) = -\overline{x_{02}}(t) \tag{4.22}$$
$$\overline{x_{11}}(t) = -\overline{x_{12}}(t) \tag{4.23}$$
$$\overline{x_{21}}(t) = -\overline{x_{22}}(t) \tag{4.24}$$
$$\overline{x_{22}}(t) = \frac{A_k/10\,(p_3(t) - p_4(t)) - \underline{F_2}(t) + \underline{F_1}(t) + d\,\overline{x_{12}}(t)}{m} \tag{4.25}$$
$$\overline{x_{11}}(t) = \frac{\overline{Q_4}(t)}{0.06\,A_k} \tag{4.26}$$
$$\overline{x_{12}}(t) = \frac{\overline{Q_3}(t)}{0.06\,A_k} \tag{4.27}$$
$$\overline{Q_3}(t) = 0.06\,A_k\,\overline{x_{12}}(t) \tag{4.28}$$
$$\overline{Q_4}(t) = 0.06\,A_k\,\overline{x_{11}}(t) \tag{4.29}$$
$$\underline{F_1}(t) = \underline{F_2}(t) - \frac{A_k}{10}\,(p_3(t) - p_4(t)) \tag{4.30}$$
$$\underline{F_2}(t) = \underline{F_1}(t) + \frac{A_k}{10}\,(p_3(t) - p_4(t)) \tag{4.31}$$
$$p_3(t) = p_4(t) + \frac{10}{A_k}\,(\underline{F_2}(t) - \underline{F_1}(t)) \tag{4.32}$$
$$p_4(t) = p_3(t) - \frac{10}{A_k}\,(\underline{F_2}(t) - \underline{F_1}(t)) \tag{4.33}$$

Abbildung 4.10: Teilbestimmte Verhaltensbeschreibung des Gleichgangzylinders

könnte aus den lokalen Untersuchungen der mit dem Gleichgangzylinder verbundenen Komponenten hervorgehen, so daß an dieser Stelle die Informationsflußbestimmung zunächst mit den noch nicht betrachteten Komponenten des Schaltkreises fortgesetzt wird, für die auch Größenklassifizierungen existieren. Sind alle möglichen Markierungen im Schaltkreis durchgeführt und die Drücke immer noch unbekannt, ist der Schaltkreis

unterbestimmt. In diesem Fall wird der Anwender dazu aufgefordert, die noch fehlenden Definitionen einzugeben. Treten Widersprüche bei der Untersuchung auf, d.h. ist eine Zustandsgröße gleichzeitig als Ein- und Ausgangsgröße markiert worden, muß der Anwender seine Vorgaben verifizieren und die Informationsflußbestimmung erneut durchführen. Angenommen die Informationsflußbestimmung des restlichen Schaltkreises hätte die Drücke $p_3(t)$, $p_4(t)$ bezüglich des Gleichgangzylinders als Eingangsgrößen identifiziert, so daß sich die noch unvollständig markierten Gleichungen 4.25, 4.32 und 4.33 in der Verhaltensbeschreibung 4.10 vollständig bestimmen lassen. Dann sind aufgrund der 2-ten Markierungsregel die Gleichungen 4.30, 4.31, 4.32 sowie 4.33 und aufgrund der 3-ten Regel die Gleichungen 4.26 und 4.27 redundant und können entfernt werden, so daß die in diesem Fall erfolgreiche Informationsflußbestimmung als Ergebnis die in der Abbildung 4.4 dargestellte Verhaltensbeschreibung des Gleichgangzylinders liefert.

In einem nachgeschalteten Prozeß werden dann die reduzierten Verhaltensbeschreibungen zu einem globalen Differentialgleichungssystem zusammengefaßt, wobei die lokalen Zustandsgrößen noch entsprechend zu unifizieren sind. Die Information, welche Variablen unifiziert werden können, ist in den Konnektoren des Bausteinmodells hinterlegt, die jeweils zwei bzw. drei Komponenten des Schaltkreises miteinander verbinden (vgl. [KlBü93]).

5 Wissensbasierte Simulation

Im Abschnitt 3 sind bereits ausführlich die mathematische Modellierung der Schaltkreissimulation sowie die numerischen Lösungsmethoden vorgestellt worden. Dieser Abschnitt soll nun weiterführend die Realisierung der wissensbasierten Simulationsberechnung behandeln. Hierzu wird dem Leser zunächst die prinzipielle Vorgehensweise der Berechnung erläutert, um dann näher auf die einzelnen wissensbasierten Aspekte einzugehen. Die Wissensrepräsentation und den Inferenzmechanismus findet der Leser im nachfolgenden Unterabschnitt 5.3. Abschließend wird dann der Ablauf der wissensbasierten Simulationsberechnung algorithmisch formuliert.

5.1 Grundidee, prinzipielle Vorgehensweise

Ziel des hier eingesetzten, heuristischen Wissens ist es, den Anwender soweit es geht vor einer direkten Konfrontation mit den mathematischen Details der Simulationsberechnung zu bewahren. Darüberhinaus soll das heuristische Wissen eine effiziente Simulation sicherstellen und dafür sorgen, daß die Genauigkeit des Simulationsergebnisses der Simulationsintention des Anwenders angepaßt ist. Dieser Abschnitt beschreibt im folgenden die Konzeption der wissensbasierten Simulationsberechnung, mit der versucht wird, die angestrebten Ziele zu erreichen.

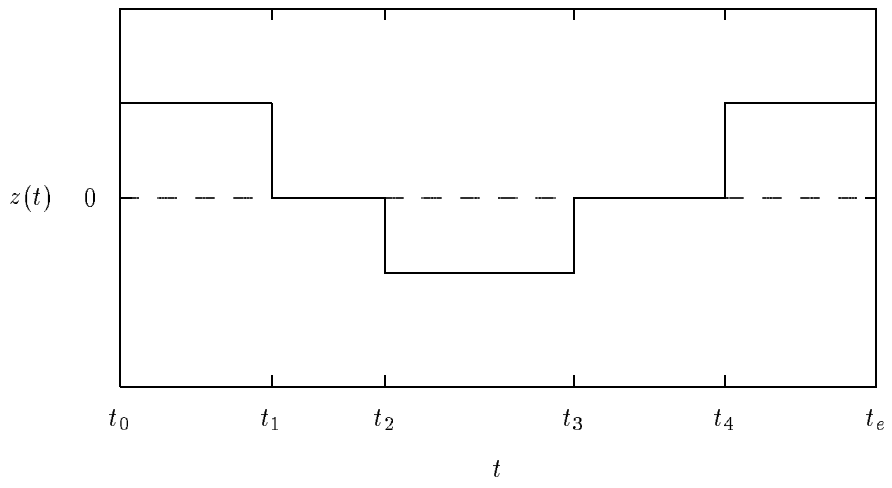


Abbildung 5.1: Funktionsdiagramm eines statischen Zustandsverlaufs

Eine wesentliche Voraussetzung für das heuristische Vorgehen bilden die Funktionsdiagramme der statischen Zustandsverläufe. Sie ergeben sich aus der statischen Funktionsprüfung, wenn man für die stationären Zustandswerte an unterschiedlichen Zeitpunkten verschiedene Eingangswerte vorgibt und die einzelnen Punkte dann durch Linien miteinander verbindet. Auf diese Weise entsteht z.B. ein Kurvenverlauf, wie in der Abbildung 5.1 zu sehen ist. In dieser Abbildung findet jeweils eine Zustandsänderung an den diskreten Zeitpunkten $t_0, t_1, \ldots, t_e$ des Simulationszeitintervalls $[t_0, t_e]$ statt.

Diese Zeitpunkte sind im dynamischen Sinne von besonderem Interesse, da sich hier die Trägheit der Komponenten am stärksten auswirkt. Sie müssen unmittelbar auf das geänderte Eingangssignal reagieren, können aber aufgrund ihrer Trägheit dem schnellen Signalwechsel nicht folgen, so daß eine gewisse Zeitspanne vergeht, bis sich das System wieder eingespielt hat, d.h. seinen stationären Zustand erreicht hat. Der stationäre Fall bereitet der dynamischen Simulationsberechnung im allgemeinen keine Probleme, problematisch sind hingegen die oben beschriebenen Zeitpunkte, an denen Schaltprozesse stattfinden. Zu diesen Zeitpunkten ist zunächst die Stabilität des Differentialgleichungssystems gemäß der Stabilitätsbedingung 3.7 zu überprüfen. Ist das System stabil, kann die Steifheit des Differentialgleichungssystems in diesem Zeitpunkt nach der Gleichung 3.12 untersucht werden. Ansonsten läßt sich das Anfangswertproblem numerisch nicht lösen.

Hat der Steifheitstest ergeben, daß ein steifes Anfangswertproblem vorliegt, steht auch die numerische Verfahrensklasse fest, die zur Lösung des Problems herangezogen wird. In diesem Fall ist die Klasse der impliziten Runge-Kutta-Verfahren zu wählen. Ansonsten kommen die expliziten Runge-Kutta-Verfahren zum Einsatz. Aus der einen oder anderen Klasse ist dann noch das optimale Verfahren zu wählen, mit dem die Lösung bis zum nächsten Schaltzeitpunkt berechnet wird. Sollte die Schrittweite über diesen Zeitpunkt hinausragen, ist sie entsprechend zu korrigieren, so daß der letzte Integrationsschritt mit dem Zeitwert abschließt. In diesem Punkt angelangt, wird die Überprüfung der Stabilität und der Steifheit des Differentialgleichungssystems sowie die Bestimmung der Verfahrensklasse und des optimalen Verfahrens für das nächste Teilintervall wiederholt und das Intervall dann mit diesem Verfahren berechnet. So fortfahrend erreicht man schließlich den Simulationsendzeitpunkt.

Da nur eine lokale Aussage über die Steifheit eines Differentialgleichungssystems möglich ist, müßte man eigentlich in jedem Integrationsschritt das System auf Steifheit untersuchen und dann das entsprechende, optimale Verfahren auswählen. Diese mathematisch korrekte Vorgehensweise würde aber einen erheblichen, zusätzlichen Rechenaufwand erfordern, denn in jedem Schritt müßte zumindest der betragsgrößte und -kleinste Eigenwert des Systems berechnet werden. Diesem erhöhten Aufwand steht in den meisten Fällen auf der anderen Seite nur ein geringer Zeitgewinn durch eine optimalere Verfahrensauswahl gegenüber, so daß man vereinfachend annimmt, daß sich das prinzipielle, dynamische Verhalten des Systems zwischen zwei Schaltzeitpunkten nur unwesentlich ändert. Genauer beschrieben: Ein zum Zeitpunkt t_s als nicht steif beurteiltes Differentialgleichungssystem wird nicht ohne Änderung eines Eingangssignals plötzlich ein steifes Verhalten zeigen und umgekehrt. Hat man also an einem Schaltzeitpunkt aufgrund der Untersuchung des Differentialgleichungssystems eine Verfahrensauswahl getroffen, kann man im allgemeinen davon ausgehen, daß diese Auswahl auch während des ganzen Teilintervalls, also bis zum nächsten Schaltzeitpunkt, ihre Berechtigung behält. In manchen Fällen könnte sich allerdings im Laufe des Teilintervalls ein anderes Verfahren aus der gleichen Verfahrensklasse als weniger aufwendig erweisen. Diesen Ausnahmefällen wird hier jedoch keine weitere Beachtung geschenkt, so daß der eventuell etwas erhöhte Rechenaufwand in Kauf genommen wird.

Das optimale, numerische Verfahren aus einer Verfahrensklasse wird, wie bereits im Unterabschnitt 3.3.4 beschrieben, über den minimalen Aufwand des Verfahrens zu

einer vorgegebenen Genauigkeitsanforderung ermittelt. Der je nach Verfahrensklasse anzuwendende Algorithmus kennzeichnet das optimale Verfahren anhand der Verfahrensstufe. Da nicht jede Stufe in der Menge der verfügbaren Verfahren vertreten ist, stellt die beschriebene Verfahrensauswahl im streng mathematischen Sinne nur eine *quasi-optimale* Auswahl dar. Sie wird sich in der Anwendungspraxis jedoch nur unwesentlich von einer optimalen Auswahl unterscheiden, wenn die Menge der verfügbaren Verfahren nach heuristischen Gesichtspunkten zusammengestellt ist, so daß hier einer reduzierten Verfahrensanzahl gegenüber einer optimalen Verfahrensauswahl den Vorzug gegeben wird.

Ebenso wie die Verfahrensauswahl wird auch die Fehlerschranke vom System heuristisch festgelegt. Eine detaillierte Erläuterung sämtlicher, heuristischer Aspekte bei der wissensbasierten Simulationsberechnung findet der Leser im nachfolgenden Unterabschnitt.

5.2 Einsatz von heuristischem Wissen

Die wissensbasierte Simulationsberechnung greift ebenfalls wie der Modul der Modellbildung auf regelbasiertes Wissen zurück. Im Unterschied hierzu unterliegt dieses Wissen im allgemeinen jedoch einer zeitlichen Änderung, so daß die abgeleiteten, zeitabhängigen Fakten innerhalb des Simulationszeitintervalls immer wieder aktualisiert werden müssen. Im folgenden sind die Voraussetzungen und die einzelnen Aspekte des heuristischen Vorgehens beschrieben, die zu Zeitpunkten des Ableitungsprozesses gelten bzw. berücksichtigt werden.

5.2.1 Voraussetzungen, Basiswissen

Als Basiswissen reicht der Heuristik der Simulationszweck SI, die Menge der verfügbaren Simulationszwecke $\mathbb{SI}$ sowie die Liste der Schaltzeitpunkte aus den statischen Funktionsdiagrammen. Der Simulationszweck und die Menge der Simulationszwecke dienen in diesem Kontext ausschließlich zur Festlegung der Fehlerschranke für die numerische Lösung. Diese Fehlerschranke stellt ein zeitunabhängiges Faktum in der Wissensbasis dar. Die Liste der Schaltzeitpunkte enthält sämtliche Zeitwerte im Simulationszeitintervall, an denen Wissen über das mathematische Modell des Schaltkreises abgeleitet und der Wissensbasis zugefügt wird. Ist ein Zeitpunkt abgearbeitet, d.h. die Integration bis zu diesem Zeitpunkt fortgeschritten, wird der Zeitwert aus der Liste entfernt. Die Liste der Schaltzeitpunkte stellt somit ein zeitabhängiges Faktum der Wissensbasis dar. Das Ende der Simulationsberechnung zeigt die leere Liste an.

5.2.2 Festlegung der Fehlerschranken

Die Fehlerschranke ε setzt sich gemäß der Gleichung 3.27 aus dem absoluten ε_{abs} und relativen Fehler ε_{rel} zusammen und definiert bei der Simulationsberechnung die einzuhaltende Genauigkeit. Sie ist abhängig vom Simulationszweck SI und der Menge

der verfügbaren Simulationszwecke $\mathbb{SI}$. Die Tabelle 5.1 zeigt die heuristischen Werte für die beiden Größen der Fehlerschranke, die hier standardmäßig jeweils mit gleichen Werten belegt sind. Mit steigender Kardinalität von $\mathbb{SI}$ und höherem *SI* nehmen auch die Genauigkeitsanforderungen zu, d.h. die Werte der Fehlergrößen werden entsprechend kleiner, denn im allgemeinen stellt der Anwender, wenn er schon ein genaues Modell des Schaltkreises zugrunde legt, auch bezüglich der numerischen Genauigkeit höhere Ansprüche. Die Wertetabelle kann als Orientierungshilfe bei der Belegung sinnvoller Fehlerschranken dienen und jederzeit vom Anwender nach seinen Vorstellungen entsprechend modifiziert werden.

	$\lvert\mathbb{SI}\rvert$					
SI	2	3	4	5	6	7
1	10^{-3} 10^{-3}	10^{-3} 10^{-3}	10^{-3} 10^{-3}	10^{-3} 10^{-3}	10^{-3} 10^{-3}	10^{-3} 10^{-3}
2	10^{-5} 10^{-5}	10^{-4} 10^{-4}	10^{-5} 10^{-5}	10^{-5} 10^{-5}	10^{-5} 10^{-5}	10^{-5} 10^{-5}
3	–	10^{-6} 10^{-6}	10^{-6} 10^{-6}	10^{-7} 10^{-7}	10^{-7} 10^{-7}	10^{-7} 10^{-7}
4	–	–	10^{-7} 10^{-7}	10^{-8} 10^{-8}	10^{-9} 10^{-9}	10^{-9} 10^{-9}
5	–	–	–	10^{-9} 10^{-9}	10^{-10} 10^{-10}	10^{-11} 10^{-11}
6	–	–	–	–	10^{-11} 10^{-11}	10^{-12} 10^{-12}
7	–	–	–	–	–	10^{-13} 10^{-13}

Tabelle 5.1: Wertetabelle der Fehlerschranken $\varepsilon_{abs}, \varepsilon_{rel}$

5.2.3 Bestimmung der Steifheit und Fehlergewichte

Zur Überprüfung der Steifheit des Differentialgleichungssystems dient die Funktion `check-stiffness`, die ebenfalls wie die Funktion `check-stability` zu einem Schaltzeitpunkt innerhalb des Ableitungsprozesses aufgerufen wird und den Steifheitsgrad κ liefert. Die Funktion `check-stability` untersucht zuvor die Stabilität des Differentialgleichungssystems und muß daher die Eigenwerte des Systems berechnen, auf die dann die Funktion `check-stiffness` zurückgreifen kann. Als numerisches Verfahren zur Eigenwertberechnung bietet sich z.B. das Verfahren von Martin, Parlett, Peters, Reinsch

und Wilkinson aus [Eng91] an, das als numerisch besonders stabil gilt. In Abhängigkeit vom Steifheitsgrad κ ist dann zu entscheiden, ob ein steifes Problem vorliegt oder nicht. Derzeit wird ein System mit einem Steifheitsgrad von $\kappa \geq 100$ als steif beurteilt, so daß in diesem Fall dann implizite Verfahren zur Lösungsberechnung herangezogen werden. Der Grenzwert sollte nach Möglichkeit so groß gewählt sein, daß erst ab diesem Wert die impliziten Verfahren ihre Vorteile gegenüber den expliziten Verfahren bezüglich der Rechenzeit ausspielen und somit ihre Anwendung auch wirklich gerechtfertigt ist. Im anderen Fall würde man durch den verfrühten Einsatz von impliziten Verfahren wertvolle Simulationszeit verschenken, denn diese Verfahren benötigen im direkten Vergleich zu den expliziten Verfahren einen höheren Rechenaufwand. In diesem Sinne stellt der hier angegebene Steifheitswert ein Default-Wert dar, der noch anhand von empirischen Tests entsprechend nach unten oder oben zu korrigieren ist.

Sollte ein steifes Problem vorliegen, müssen auch die Fehlergewichte der einzelnen Komponenten der Lösungsfunktion bestimmt werden. Diese Aufgabe übernimmt im Ableitungsprozeß die Funktion `compute-weights`, die die Gewichte σ_i nach der Beziehung 5.1 bestimmt.

$$\sigma_i = \left| \frac{\lambda_{min}}{\lambda_i} \right| \quad \text{für } i = 1, 2, \ldots, n \tag{5.1}$$

Diese Abbildung ordnet jeder Lösungskomponente entsprechend dem Verhältnis von aktuellem zu minimalem Eigenwert ein Gewicht zu. Dem kleinsten Eigenwert (niedrigste Frequenz) wird demzufolge das größte Gewicht und dem größten Eigenwert (höchste Frequenz) das niedrigste Gewicht zugewiesen.

5.2.4 Quasi-optimale Verfahrensauswahl

Die Wahl eines optimalen Verfahrens hängt im allgemeinen von der Anfangswertproblemklasse, dem Anfangswertproblem und der vorgegebenen Fehlerschranke ab. Die Problemklasse legt die Verfahrensklasse fest. Das Anfangswertproblem und die Fehlerschranke bestimmen dann das optimale Verfahren der gewählten Klasse, wobei optimal, wie bereits beschrieben, ein Verfahren mit minimalem Aufwand bezeichnet. Der Aufwand eines Verfahrens wird je nach Verfahrensklasse mit dem entsprechenden Algorithmus aus 3.3.4 abgeschätzt. Praktische Tests haben gezeigt, daß die Fehlerordnung des Verfahrens und die positive Potenz der Fehlerschranke ungefähr die gleiche Größenordnung haben sollten, um eine effiziente Berechnung der Lösungsfunktion zu ermöglichen. Unter diesem Gesichtspunkt sind die Mengen der hier verfügbaren Verfahren zusammengestellt. Sie enthalten eine Auswahl von Verfahren verschiedener Ordnungen, die den unterschiedlichen Fehlerschranken angepaßt sind. Hierbei haben bei den Einbettungsformeln solche Verfahren den Vorzug bekommen, die sich in umfangreichen Tests im Vergleich zu anderen Verfahren gleicher Ordnung durch ihre hohe Zuverlässigkeit und Genauigkeit sowie ihren niedrigen Rechenaufwand ausgezeichnet haben (vgl. [Eng91]). Bei den impliziten Runge-Kutta-Formeln sind die Verfahren nach der Gauß-Legendre Stützstellenverteilung gewählt worden, da sie in Abhängigkeit von der maximalen Verfahrensstufe die höchste Fehlerordnung erzielen, so daß sie in puncto Genauigkeit und Effizienz von keinem anderen, impliziten Verfahren übertroffen werden. Die Tabelle 5.2 zeigt eine Zusammenstellung der Verfahrensstufen und -ordnungen

der hier eingesetzten Runge-Kutta-Verfahren. Die Runge-Kutta-Formeln 2./3. Ordnung

Einbettungsformeln	$(m/\widehat{m})$	2/3	6/6	6/8	12/13
	$(e_g/\hat{e}_g)$	2/3	4/5	5/6	7/8
Implizite Formelpaare	$(m/\widehat{m})$	2/3	5/6	8/9	11/12
	$(e_g/\hat{e}_g)$	4/6	10/12	16/18	22/24

Tabelle 5.2: Verfahrensstufe und -ordnung der verwendeten Runge-Kutta-Verfahren

decken im Normalfall den unteren Genauigkeitsbereich $\varepsilon \approx 10^{-1} \ldots 10^{-3}$ ab. Die Verfahren 4./5. und 5./6. Ordnung sind für den Bereich von $\varepsilon \approx 10^{-3} \ldots 10^{-6}$ und die Formeln 5./6. und 8./9. Ordnung für das Fehlerintervall $\varepsilon \approx 10^{-6} \ldots 10^{-9}$ gedacht. Entsprechend gelten die Formeln 7./8. und 11./12. Ordnung für Genauigkeitsanforderungen von $\varepsilon \geq 10^{-9}$. Die beschriebenen Zuordnungen von Verfahrensordnungen auf Genauigkeitsbereiche sind natürlich nicht starr und können von Fall zu Fall variieren. Anhand der Bereiche sollten lediglich die Überlegungen erläutert werden, die zu der Vorauswahl der Menge der verfügbaren Verfahren geführt haben. Wie man sieht, basiert die Vorauswahl bereits auf heuristischem Wissen.

Die Aufwandsabschätzung zur optimalen Verfahrensauswahl gibt die Stufe des optimalen Verfahrens zurück. Da jedoch nicht jede Verfahrensstufe zur Verfügung steht, kann die Verfahrensauswahl im allgemeinen nicht als optimal sondern nur als quasi-optimal angesehen werden. Sie ist insofern schon nur als quasi-optimal zu beurteilen, als daß der Auswahlprozeß nicht in jedem Integrationsschritt, sondern nur an den Schaltzeitpunkten erfolgt. In der Praxis wird die Anwendung eines quasi-optimalen anstelle eines optimalen Verfahrens nicht ins Gewicht fallen, zumal sich bereits die Vorauswahl der Verfahren an empirischen Untersuchungen orientiert hat, so daß man hier einer einfacheren Verfahrenssteuerung den Vorrang gibt, um an dieser Stelle Rechenzeit einzusparen.

5.3 Wissensrepräsentation, Inferenzmechanismus

Wie für die wissensbasierte Modellbildung ist auch das heuristische Wissen der Simulationsberechnung in erweiterten[9], aussagenlogischen Regeln formuliert. Diese Regeln entsprechen ebenfalls der im Anhang B aufgeführten, formalen Syntaxdefinition. Neben der Sprachdefinition findet der Leser dort auch eine zusammenhängende Darstellung sämtlicher Regeln der wissensbasierten Simulationsberechnung. Die Abbildung 5.2 zeigt exemplarisch eine Regel aus dieser Regelmenge. Sie enthält im Konklusionsteil einen Funktionsaufruf zur Bestimmung des optimalen, impliziten Verfahrens. Nach Anwendung der Regel findet sich das Ergebnis der Funktionsauswertung als Faktum

[9] (vgl. Unterabschnitt 4.2.3)

bzw. Fakten in der Wissensbasis wieder, in diesem Fall der Name des optimalen, impliziten Verfahrens sowie die Anfangsschrittweite des Verfahrens.

```
(rule-method-1 (IF (method.class is implicit))
               (THEN (CALL-FUNCTION determine-optimal-implicit-method)))
```

Abbildung 5.2: Regel zur Bestimmung des optimalen, impliziten Verfahrens

Die in den Regeln auftauchenden Funktionen sind für den Anwender im allgemeinen nicht zugänglich, da sie den allgemeingültigen Teil des Wissens repräsentieren, der keinen Änderungen unterliegt. Diese Funktionen werden zur Berechnung der Parameter-Wert-Tupel benötigt, auf denen dann weitere Funktionen im Konklusionsteil von Regeln zurückgreifen oder mit denen dann weitere Schlußfolgerungen möglich sind.

In gleicher Weise wie bei der wissensbasierten Modellbildung sind auch für das heuristische Vorgehen bei der Simulationsberechnung sämtliche Parameter mit Werten zu belegen, wie zum Beispiel die Fehlerschranken, die Fehlergewichte etc., so daß sich auch hier als Inferenzmechanismus das *Forward-Chaining* anbietet.

5.4 Der Algorithmus

Der nachfolgende Algorithmus veranschaulicht den groben Ablauf der Simulationsberechnung. Die hierin auftretenden Funktionen first und rest entsprechen den gleichnamigen LISP-Funktionen und werden als bekannt vorausgesetzt. Den Ausgangspunkt der Berechnung bildet die mit den vorgegebenen Fakten initialisierte Wissensbasis. Alle weiteren, benötigten Fakten liefert der Ableitungsprozeß im Schritt 2b.

Die Berechnung des Näherungswertes $\boldsymbol{z}(t_{act}+h_{act})$ im Schritt 2(e)ii setzt die im Unterabschnitt 3.3.3 beschriebenen Schrittweitensteuerungen des Verfahrens voraus. Sie sorgen dafür, daß der in 2(e)ii berechnete Funktionswert die Genauigkeitanforderung erfüllt, was unter Umständen eine mehrfache Wiederholung des Integrationsschrittes erfordert. In diesem Kontext steht also nach der Ausführung der Anweisung stets ein gültiger Funktionswert zur Verfügung.

Im Schritt 2(e)iii werden die Werte von Zustandsgrößen überprüft, für die eine Begrenzungsfunktion definiert ist, wie beispielsweise die Hubbegrenzung des Gleichgangzylinders oder 4/3 Wege-Proportionalventils, die dafür sorgt, daß die Geschwindigkeit unmittelbar auf Null gesetzt wird, sobald der Kolben den maximalen Hub erreicht hat. Bei der Simulation z.B. eines Gleichgangzylinders kann nun folgende Situation auftreten. Im letzten Integrationsschritt befand sich der Kolben noch vor der Hubbegrenzung. Der nächste Integrationsschritt würde aber mit der aktuellen Schrittweite einen Positionswert außerhalb des Hubes liefern, so daß die Begrenzungsfunktion innerhalb dieses Integrationschrittes die Kolbengeschwindigkeit auf Null setzt. Dies kann zur Folge haben, daß die vom Verfahren berechneten Funktionszwischenwerte $\boldsymbol{k}_i$ untereinander stark abweichen, so daß der berechnete Funktionswert aufgrund dieser Abweichungen außerhalb der Fehlertoleranz liegt und somit verworfen wird. In diesem

Fall verringert die Schrittweitensteuerung die aktuelle Schrittweite und veranlaßt die wiederholte Berechnung des letzten Integrationsschrittes. Die neue Schrittweite kann jetzt so gewählt sein, daß der entsprechend berechnete Positionswert des Kolbens noch innerhalb des Hubes liegt. Da aber zuvor die Begrenzungsfunktion fälschlicherweise die Geschwindigkeit bereits auf Null gesetzt hat, liefert das Verfahren nun einen falschen Geschwindigkeitswert. Um eine falsche Berechnung zu verhindern, müßte die Nullzuweisung der Begrenzungsfunktion in ungültigen Integrationsschritten wieder rückgängig gemacht werden. Da dies jedoch einen direkten Eingriff in das Verfahren bedeutet, läßt man das Verfahren zunächst einen vollständigen Integrationsschritt durchführen und testet anschließend die Gültigkeit der kritischen Werte. Sollte sich jetzt ein unzulässiger Wert zeigen, kann die Schrittweite korrigiert und der letzte Integrationsschritt erneut berechnet werden. Dieser Vorgang ist gegebenenfalls solange zu wiederholen bis die Zustandsgröße innerhalb einer vorgegebenen Fehlerschranke mit der Begrenzung übereinstimmt. Erst dann kann die Begrenzungsfunktion aufgerufen und die Geschwindigkeit korrekterweise auf Null gesetzt werden.

Sind keine ungültigen Zustandswerte aufgetreten, wird der Wert der Lösungsfunktion $\boldsymbol{z}(t_{act}+h_{act})$ im Schritt 2(e)v des Algorithmus gespeichert und die Zeit in 2(e)vi für den nächsten Schleifendurchlauf aktualisiert. Nach Terminierung der inneren Schleife, die für die Integration zwischen zwei Schaltzeitpunkten zuständig ist, wird in der äußeren Schleife abschließend die Wissensbasis gelöscht, so daß im nächsten Schleifendurchlauf die Berechnung des nachfolgenden Teilintervalls beginnen kann.

1. Initialisiere die Wissensbasis mit den gegebenen Fakten SI, $|\mathbb{SI}|$, $timepoints = [t_0, t_1, \ldots, t_e]$
2. Wiederhole solange $timepoints \neq [\,]$ gilt:
   (a) Setze
   $$\begin{aligned} t_{act} &= \text{first}(timepoints) \\ timepoints &= \text{rest}(timepoints) \end{aligned}$$
   (b) Leite mittels forward-chaining alle gültigen Fakten ab, d.h. die Fehlerschranken ε_{abs}, ε_{rel}, die Anfangswertproblemklasse, das optimale Verfahren, Anfangsschrittweite h_{act} etc.
   (c) Falls das Differentialgleichungssystem nicht stabil ist, gehe zu 3
   (d) Falls $timepoints \neq [\,]$ gilt, dann setze $t_{nxt} = \text{first}(timepoints)$ sonst setze $t_{nxt} = t_{act}$
   (e) Wiederhole solange $t_{act} < t_{nxt}$ gilt:
      i. Falls $(t_{act} + h_{act}) > t_{nxt}$ gilt, dann setze $h_{act} = t_{nxt} - t_{act}$
      ii. Berechne mit dem gewählten Verfahren $\boldsymbol{z}(t_{act} + h_{act})$
      iii. Überprüfe die Gültigkeit kritischer Größen $z_i(t)$ für $i \in \{1, \ldots, n\}$.
      iv. Falls eine Zustandsgröße $z_i(t)$ einen unzulässigen Wert besitzt, setze
      $$h_{act} = \frac{1}{2}\, h_{act}$$

und gehe zu 2(e)ii.

v. Speichere Funktionswert $\boldsymbol{z}(t_{act} + h_{act})$

vi. Setze $t_{act} = t_{act} + h_{act}$

(f) Entferne alle abgeleiteten Fakten aus der Wissensbasis.

3. Breche die Simulationsberechnung mit einer entsprechenden Fehlermeldung ab.

6 Perspektiven

In dieser Diplomarbeit sind die Grundlagen und die Konzeption einer wissensbasierten Modellbildung und Simulation erarbeitet worden, auf denen dann eine dynamische Funktionsprüfung im Softwaresystem ^{art}*deco* aufbauen kann. Teile dieses Entwurfs sind bereits implementiert und soweit möglich ersten Tests unterzogen worden. Andere Bereiche wiederum konnten im Rahmen dieser Arbeit nur angeschnitten werden, ihre Realisierung steht noch aus. Die prototypische Umsetzung der Konzeption läßt natürlich noch Verbesserungsansätze bzw. Erweiterungen erkennen. Beispielsweise berücksichtigt die heuristische Modelltiefenzuordnung derzeit noch keine Parallel- und Netzstrukturen von hydraulischen Schaltkreisen, da die Topologieuntersuchung gemäß [Hoff93] noch nicht zur Verfügung stand. Die Realisierung der Modelltiefenbestimmung ist jedoch so ausgelegt, daß die Ergebnisse der Strukturanalyse ohne weiteres in das bestehende Programm integriert werden können.

Auch eine anwenderfreundlichere Syntax der Regelsprache ist in zukünftigen Programmversionen wünschenswert. Hierbei bietet sich wie bei der technischen Beschreibungssprache im Anhang A.1 eine Unterscheidung in externer und interner Syntax an. Die externe Syntax definiert die Schnittstelle zum Anwender. Sie sollte dem Ingenieur eine einfache Möglichkeit zur Pflege und Wartung des Domänenwissens bereitstellen und eine gut leserliche Form haben. Die externe Syntax der Regelsprache wird nach Abschluß eines Benutzerdialoges in die interne Form der Sprache übersetzt, die die Sicht des Systems auf die Regeln definiert. Diese Syntax sollte aufgrund ihrer systemangepaßten Form eine effiziente Verarbeitung der Regeln erlauben.

Die Wissensakquisition in ^{art}*deco* verlangt bei der Definition einer dynamischen Komponentenverhaltensbeschreibung, daß der Ingenieur die Gleichungen, aufgelöst nach allen darin enthaltenen Variablen, eingibt. Dieser aus Anwendersicht zusätzliche Aufwand ist zwar vertretbar, da die Definition neuer Komponenten bzw. die Modifikation bekannter Bauteile in der Regel eher selten vorkommt, wünschenswert wäre aber auch hier eine Entlastung des Anwenders. Hierzu müßte das System in der Lage sein, eine Gleichung nach jeder Variablen umzuformen. Im Fall nichtlinearer Gleichungen stellt die Lösung dieser Aufgabe kein leichtes Unterfangen dar.

Die Informationsflußbestimmung konnte im Rahmen dieser Arbeit nur sehr allgemein behandelt werden, so daß der vorgestellte Lösungsansatz in Zukunft noch zu konkretisieren ist. Hier stellt sich dann auch die Frage nach einer geeigneten Modellierung des Problems. Sie sollte selbstverständlich eine effiziente Berechnung des Informationsflusses ermöglichen. Auch in diesem Kontext könnte der Suchbaum eventuell durch wissensbasierte Techniken beschränkt bzw. die Lösungssuche zielgerichtet gesteuert werden. Beispielsweise könnten Regeln, die auf Komponentenebene definiert sind, offensichtliche Zusammenhänge bezüglich der Klassifizierung zwischen einzelnen Zustandsgrößen formulieren. Bezogen auf den Gleichgangzylinder würde das z.B. bedeuten, daß, falls nur die Position bzw. die Geschwindigkeit, die Beschleunigung als Ausgangsgröße an einem Gate vorgegeben ist, dann auch die anderen beiden Größen des Gates als Ausgangsgröße klassifiziert werden. Entsprechend sind dann auch die Größen am gegenüberliegenden Gate als Ausgangsgrößen zu markieren.

Im Bereich der numerischen Verfahren kann eventuell eine speziellere Schrittweitensteuerung zu noch kürzeren Simulationszeiten führen. Die im Unterabschnitt 3.3.3 vorgestellte Schrittweitensteuerung der expliziten Runge-Kutta-Verfahren vergrößert bzw. verringert die aktuelle Schrittweite maximal um den Faktor 2 bzw. $\frac{1}{2}$. Wenn Sprünge im Funktionsverlauf auftauchen, wie sie z.B. im Geschwindigkeitsdiagramm der Abbildung 1.2 zu sehen sind, wird die Schrittweite im allgemeinen bis zur ihrer unteren Schranke verkleinert, um anschließend wieder auf eine normale Größe anzuwachsen. Wird in jedem Integrationsschritt die Schrittweite nur halbiert bzw. verdoppelt, nimmt die Berechnung der Lösungsfunktion an dieser Stelle entsprechende Zeit in Anspruch. Größere Schrittweitensprungbeträge könnten hier Abhilfe schaffen. Auf der anderen Seite darf eine zu große Schrittweitenänderung nicht dazu führen, daß in einem Integrationsschritt mehrfache Korrekturen der Schrittweite erforderlich werden und so die gewonnene Zeitersparnis wieder verloren geht. Ein ausgeklügelteres Steuerungsverfahren, das die Anzahl und Art der Schrittweitenkorrekturen auswertet und anhand dieser Werte dann Schlußfolgerungen über den vermeindlichen Kurvenverlauf zieht, d.h. Funktionssprünge identifiziert, könnte hier vielleicht den gewünschten Erfolg bringen.

A Die technische Beschreibungssprache

Die funktionale Beschreibung einer beliebigen Komponente basiert im allgemeinen auf ein System von gewöhnlichen, nichtlinearen Differentialgleichungen. Die Sicht des Benutzers auf solch ein System definiert die im folgenden Unterabschnitt dargestellte, externe Form der technischen Beschreibungssprache. Das Softwaresystem hat die durch die interne Form spezifizierte Sicht auf die Gleichungen. In dieser Darstellung wird ein Differentialgleichungssystem auf eine Liste von Lisp-Formen abgebildet, wobei jede Lisp-Form eine Gleichung repräsentiert. Eine Auswahl von dynamischen Verhaltensbeschreibungen, formuliert in der technischen Beschreibungssprache, zeigt der Abschnitt A.2.

A.1 Definition der Syntax

Die formale Definition der technischen Beschreibungssprache ist in EBNF[10]-Notation gegeben. Klein- und kursivgedruckte Bezeichner kennzeichnen die Nichtterminale dieser Grammatik. Die Terminale sind in normaler Schrift gesetzt.

Externe Darstellung

$$
\begin{aligned}
\textit{equation–system} &::= \textit{equation}\ \{\textit{eq–delimiter equation}\}_0^* \\
\textit{equation} &::= \textit{expr} = \textit{expr} \\
\textit{expr} &::= \{\ +\ |\ -\ |\ /\ \}_0^1\ \textit{term}\ \{\textit{op term}\}_0^* \\
\textit{term} &::= \textit{variable}\ |\ (\textit{expr})\ |\ \textit{funcall}\ |\ \textit{if–form}\ |\ \textit{case–form} \\
\textit{variable} &::= \textit{attribute}\ |\ \textit{number} \\
\textit{funcall} &::= \textit{function–name argument–list} \\
\textit{if–form} &::= (\text{IF}\ \textit{boolean–expr}\ \text{THEN}\ \textit{expr}\ \text{ELSE}\ \textit{expr}) \\
\textit{case–form} &::= (\text{CASE}\ \{\text{WHEN}\ \textit{boolean–expr} : \textit{expr}\}_1^*\ \{\text{OTHERWISE} : \textit{expr}\}_0^1) \\
\textit{boolean–expr} &::= \textit{expr}\ \{\textit{boolean-op expr}\}_1^*\ |\ \text{NOT}\ \textit{boolean–expr}\ | \\
&\qquad \{\textit{attribute}\ |\ \textit{constant}\ \text{IS}\ \textit{attribute}\ |\ \textit{constant}\} \\
\textit{op} &::= +\ |\ -\ |\ *\ |\ / \\
\textit{boolean–op} &::= <\ |\ \leq\ |\ =\ |\ \geq\ |\ >\ |\ \text{AND}\ |\ \text{OR} \\
\textit{attribute} &::= \textit{attribute–name}\ [\textit{gate–name}\ |\ \text{SELF}\ |\ \text{ABS}\ ] \\
\textit{argument–list} &::= (\textit{expr}\ \{,\ \textit{expr}\}_0^*) \\
\textit{gate–name} &::= \text{GATE}.\textit{number} \\
\textit{attribute–name} &::= \textit{symbol} \\
\textit{function–name} &::= \textit{symbol} \\
\textit{constant} &::= \textit{symbol}
\end{aligned}
$$

[10]EBNF (Extended-Backus-Naur-Form)

$$
\begin{aligned}
\textit{number} &::= \{\textit{digit}\}_1^* \\
\textit{digit} &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\textit{symbol} &::= \{\textit{char}\}_1^* \\
\textit{char} &::= \mathrm{a} \mid \mathrm{b} \mid \ldots \mid \mathrm{z} \\
\textit{eq–delimiter} &::= \mathrm{EMPTYLINE}
\end{aligned}
$$

Interne Darstellung

$$
\begin{aligned}
\textit{equation-system} &::= (\{\textit{equation}\}_1^*) \\
\textit{equation} &::= (= \textit{expr expr}) \\
\textit{expr} &::= \textit{arithmetic–lisp–form} \mid \textit{conditional–expr} \\
\textit{conditional–expr} &::= (\mathrm{IF}\ \textit{boolean–lisp–form expr expr}) \mid \\
&\quad\ (\mathrm{COND}\ \{(\textit{boolean–lisp–form expr})\}_1^*)
\end{aligned}
$$

Eine *arithmetic–lisp–form* bezeichnet eine beliebige Lisp-Form, deren Evaluierung für den Fall, daß alle Variablen dieser Form gebunden sind, eine Zahl ergibt. Entsprechend dieser Definition steht eine *boolean–lisp–form* für eine beliebige Lisp-Form, die den Wert `TRUE` oder `NIL` zurückgibt.

A.2 Eine Auswahl dynamischer Verhaltensbeschreibungen

Hydraulische Komponenten können in Arbeits-, Steuer-, Versorgungs- und Verbindungselemente klassifiziert werden. Jede dieser Klassen umfaßt eine Vielzahl unterschiedlichster Elemente, so daß im Rahmen dieser Arbeit eine dynamische Verhaltensbeschreibung sämtlicher Komponenten nicht sinnvoll erscheint. Die getroffene Auswahl an Komponenten zeigt deshalb jeweils ein oder zwei Vertreter einer Klasse, an denen exemplarisch für die gesamte Klasse die Modellierung dargestellt wird. Neben der externen Repräsentation der Differentialgleichungssysteme in ^{art}*deco* gliedert sich eine Komponentenbeschreibung zusätzlich in die folgenden Paragraphen:

- Schaltzeichen und Parameter
- Mathematische Darstellung

Der erste Paragraph zeigt das Schaltzeichen und die Geometrie der betrachteten Komponente sowie deren Parameter. Neben den zeitlichen Konstanten sind auch die von den Parametern und Größen einer Komponente abgeleiteten Größen wie z.B. die Reibkraft $F_R(\ldots)$ in der Parametertabelle aufgeführt. Eine Ausnahme bilden die global gültigen, abgeleiteten Größen, die für mehrere Komponenten unterschiedlicher Klassenzugehörigkeit gelten können. Für die getroffene Komponentenauswahl beschränken sich diese Größen auf den Elastizitätsmodul von Öl, der wie in [Lau90] zitiert, definiert ist als

$$E_{\ddot{O}l}(p(t)) = 9000\ \log(\frac{9}{28}\,p(t) + 3) \quad [\mathrm{bar}].$$

Im zweiten Paragraphen findet der Leser die dynamische Komponentenbeschreibung in mathematischer Notation vor. Die dem Ingenieur und Naturwissenschaftler vertraute Darstellung eines Differentialgleichungssystems soll zum einen das Verständnis der beschriebenen, funktionalen Zusammenhänge erleichtern und zum anderen die große Ähnlichkeit zur externen Darstellung in ^{art}*deco* verdeutlichen. Abweichend von den allgemeinen Konventionen der Mathematik werden innerhalb eines Differentialgleichungssystems zwei äquivalente Bezeichnungen sowohl für die Geschwindigkeit als auch für die Beschleunigung verwendet. Die Mehrfachbezeichnungen resultieren aus der Forderung, jede am Gate einer Komponente anliegende Größe im Differentialgleichungssystem zu manifestieren bei gleichzeitiger transparenter Darstellung des gesamten Gleichungssystems. Die verwendeten Bezeichnungen und ihre jeweilige Bedeutung sind in der nachfolgenden Tabelle aufgeführt. Die differentielle Notation einer Größe

Bezeichnung:	Bedeutung:
$\dot{x}_{0i}(t)$, $x_{1i}(t)$	Geschwindigkeit am Gate i
$\dot{x}_{1i}(t)$, $x_{2i}(t)$	Beschleunigung am Gate i

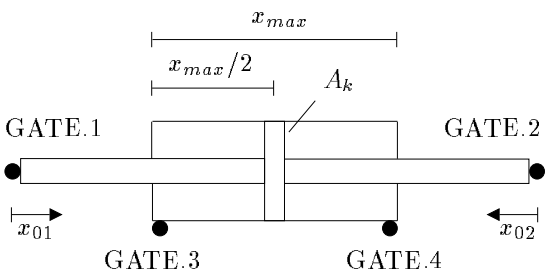
Tabelle A.1: Äquivalente Bezeichnungen und ihre Bedeutung

tritt ausschließlich auf der linken Seite einer Differentialgleichung auf und kennzeichnet somit eindeutig die zu integrierenden Differentialgleichungen innerhalb des Gleichungssystems.

Eine weitere Besonderheit der dynamischen Beschreibung stellt die Formulierung der Hubbegrenzung für den Gleichgangzylinder und das 4/3-Wege-Proportionalventil dar. Die Begrenzungsfunktion ist implizit als Fallunterscheidung in der Differentialgleichung zur Berechnung der Beschleunigung realisiert, wobei die Funktion $h(\ldots)$ als eine Hilfsfunktion anzusehen ist, die den eigentlichen Beschleunigungswert bestimmt. Für den Fall, daß der Kolben den maximalen Hub überschreitet, bekommt die Beschleunigung den Wert Null zugewiesen. Folglich erfährt auch die Geschwindigkeit keinen Zuwachs mehr. Die Begrenzungsfunktion setzt jedoch die Geschwindigkeit nicht auf Null zurück. Dies geschieht in einer der Integration nachgeschalteten Überprüfung des Zustandsvektors. Stellt sich eine Verletzung des Gültigkeitsbereichs einer Zustandsgröße heraus, wird ihr Wert entsprechend angepaßt und der letzte Integrationsschritt gegebenenfalls mit einer kleineren Schrittweite wiederholt. In Bezug auf die Kolbengeschwindigkeit sorgt die Zustandsvektorüberprüfung dafür, daß, falls der Kolben die Hubbegrenzung innerhalb einer gewissen Genauigkeitsschranke erreicht hat, die Geschwindigkeit auf Null zurückgesetzt wird, so daß der Kolben in maximaler Auslenkung verharrt und sich erst dann wieder in Bewegung setzt, wenn die Beschleunigung ihr Vorzeichen, d.h. ihre Richtung, ändert.

A.2.1 Gleichgangzylinder

Schaltzeichen und Parameter



Parameter:		Beschreibung:
A_k	[mm^2]	Kolbenfläche
d	$[\frac{\mathrm{N\,s}}{\mathrm{m}}]$	viskose Reibung
V_0	[mm^3]	Ölvolumen Zylinderkammer + Leitungsvolumen für $x = \frac{x_{max}}{2}$
F_{C0}	[N]	Coulombsche Reibkraft
F_{H0}	[N]	Haftreibkraft
$F_R(\dot{x}(t), F_L(t))$	[N]	Gesamtreibkraft, wobei $F_L(t)$ die result. Lastkraft bezeichnet
v_{ab}	$[\frac{\mathrm{m}}{\mathrm{s}}]$	Abklingkonstante
Q_L	$[\frac{\mathrm{l}}{\mathrm{min}}]$	interner Leckagefluß
m	[kg]	bewegte Masse
x_{max}	[mm]	Hub

Mathematische Darstellung

<u>Modelltiefe 0:</u> Statische Verhaltensbeschreibung

$$
\begin{aligned}
x_{01}(t) &= -x_{02}(t) \\
x_{02}(t) &= \begin{cases} \frac{x_{max}}{2} & \text{für } h(t, x_{12}(t)) \geq \frac{x_{max}}{2} \\ -\frac{x_{max}}{2} & \text{für } h(t, x_{12}(t)) \leq -\frac{x_{max}}{2} \\ h(t, x_{12}(t)) & \text{sonst} \end{cases} \\
x_{11}(t) &= -x_{12}(t) \\
x_{21}(t) &= x_{22}(t) = 0 \\
Q_3(t) &= 0.06\, A_k\, x_{12}(t) \\
Q_4(t) &= 0.06\, A_k\, x_{11}(t) \\
F_2(t) - F_1(t) &= \frac{A_k}{10}\,(p_3(t) - p_4(t)) \\
h(t, v(t)) &= x_0 + t\, v(t)
\end{aligned}
$$

<u>Modelltiefe 1:</u> Berücksichtigung der Massenträgheit

$$
\begin{aligned}
\dot{x}_{02}(t) &= x_{12}(t) \\
\dot{x}_{12}(t) &= x_{22}(t) \\
x_{01}(t) &= -x_{02}(t) \\
x_{11}(t) &= -x_{12}(t) \\
x_{21}(t) &= -x_{22}(t)
\end{aligned}
$$

$$
\begin{aligned}
x_{22}(t) &= \begin{cases} 0 \text{ für } (x_{02}(t) \leq \frac{x_{max}}{-2} \wedge h(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)) \leq 0) \vee \\ \qquad (x_{02}(t) \geq \frac{x_{max}}{2} \wedge h(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)) \geq 0) \\ -h\big(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)\big) \text{ sonst} \end{cases} \\
Q_3(t) &= 0.06\, A_k\, x_{12}(t) \\
Q_4(t) &= 0.06\, A_k\, x_{11}(t) \\
F_2(t) - F_1(t) &= \frac{A_k}{10}\,(p_3(t) - p_4(t)) \\
h\big(p_3(t),p_4(t),F_2(t),F_1(t),v(t)\big) &= \frac{A_k/10\,(p_3(t) - p_4(t)) - F_2(t) + F_1(t) + d\,v(t)}{m}
\end{aligned}
$$

<u>Modelltiefe 2:</u> Zusätzlich zur Modelltiefe 1, Berücksichtigung des Druckaufbaus

$$
\begin{aligned}
\dot{x}_{02}(t) &= x_{12}(t) \\
\dot{x}_{12}(t) &= x_{22}(t) \\
\dot{p}_3(t) &= \frac{16660\, E_{\ddot{O}l}(p_3(t))}{V_0 + 1000\, A_k\, x_{02}(t)}\,(Q_3(t) + 0.06\, A_k\, x_{12}(t)) \\
\dot{p}_4(t) &= \frac{16660\, E_{\ddot{O}l}(p_4(t))}{V_0 + 1000\, A_k\, x_{01}(t)}\,(Q_4(t) + 0.06\, A_k\, x_{11}(t)) \\
x_{01}(t) &= -x_{02}(t) \\
x_{11}(t) &= -x_{12}(t) \\
x_{21}(t) &= -x_{22}(t) \\
x_{22}(t) &= \begin{cases} 0 \text{ für } (x_{02}(t) \leq \frac{x_{max}}{-2} \wedge h(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)) \leq 0) \vee \\ \qquad (x_{02}(t) \geq \frac{x_{max}}{2} \wedge h(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)) \geq 0) \\ -h\big(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)\big) \text{ sonst} \end{cases} \\
Q_3(t) &= 0.06\, A_k\, x_{12}(t) \\
Q_4(t) &= 0.06\, A_k\, x_{11}(t) \\
F_2(t) - F_1(t) &= \frac{A_k}{10}\,(p_3(t) - p_4(t)) \\
h\big(p_3(t),p_4(t),F_2(t),F_1(t),v(t)\big) &= \frac{A_k/10\,(p_3(t) - p_4(t)) - F_2(t) + F_1(t) + d\,v(t)}{m}
\end{aligned}
$$

<u>Modelltiefe 3:</u> Zusätzlich zur Modelltiefe 2, Berücksichtigung der Reibungskräfte

$$
\begin{aligned}
\dot{x}_{02}(t) &= x_{12}(t) \\
\dot{x}_{12}(t) &= x_{22}(t) \\
\dot{p}_3(t) &= \frac{16660\, E_{\ddot{O}l}(p_3(t))}{V_0 + 1000\, A_k\, x_{02}(t)}\,(Q_3(t) + 0.06\, A_k\, x_{12}(t)) \\
\dot{p}_4(t) &= \frac{16660\, E_{\ddot{O}l}(p_4(t))}{V_0 + 1000\, A_k\, x_{01}(t)}\,(Q_4(t) + 0.06\, A_k\, x_{11}(t)) \\
x_{01}(t) &= -x_{02}(t) \\
x_{11}(t) &= -x_{12}(t) \\
x_{21}(t) &= -x_{22}(t)
\end{aligned}
$$

$$
\begin{aligned}
x_{22}(t) &= \begin{cases} 0 \text{ für } (x_{02}(t) \leq \frac{x_{max}}{-2} \wedge h(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)) \leq 0) \vee \\ \qquad (x_{02}(t) \geq \frac{x_{max}}{2} \wedge h(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)) \geq 0) \\ -h\big(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)\big) \text{ sonst} \end{cases} \\
Q_3(t) &= 0.06\, A_k\, x_{12}(t) \\
Q_4(t) &= 0.06\, A_k\, x_{11}(t) \\
F_2(t) - F_1(t) &= \frac{A_k}{10}\,(p_3(t) - p_4(t)) - F_R(t) \\
F_R\big(p_3(t),p_4(t),F_2(t),F_1(t),v(t)\big) &= (F_{C0} + F_{H0}\exp(-\frac{x_{12}(t)}{v_{ab}}))\,\mathrm{sign}(x_{12}(t)) + (F_{C0} + F_{H0}) \\
&\quad * \,\mathrm{sn}(x_{12}(t))\,\mathrm{sign}(\frac{A_k}{10}\,(p_3(t) - p_4(t)) - F_2(t) + F_1(t)) \\
h\big(p_3(t),p_4(t),F_2(t),F_1(t),v(t)\big) &= \frac{A_k/10\,(p_3(t) - p_4(t)) - F_2(t) + F_1(t) + d\,v(t)}{m}
\end{aligned}
$$

<u>Modelltiefe 4:</u> Zusätzlich zur Modelltiefe 3, Berücksichtigung der Leckagen

$$
\begin{aligned}
\dot{x}_{02}(t) &= x_{12}(t) \\
\dot{x}_{12}(t) &= x_{22}(t) \\
\dot{p}_3(t) &= \frac{16660\, E_{Öl}(p_3(t))}{V_0 + 1000\, A_k\, x_{02}(t)}\,(Q_3(t) + 0.06\, A_k\, x_{12}(t) \\
&\quad - Q_L\,\mathrm{sign}(p_3(t) - p_4(t))) \\
\dot{p}_4(t) &= \frac{16660\, E_{Öl}(p_4(t))}{V_0 + 1000\, A_k\, x_{01}(t)}\,(Q_4(t) + 0.06\, A_k\, x_{11}(t) \\
&\quad + Q_L\,\mathrm{sign}(p_3(t) - p_4(t))) \\
x_{01}(t) &= -x_{02}(t) \\
x_{11}(t) &= -x_{12}(t) \\
x_{21}(t) &= -x_{22}(t) \\
x_{22}(t) &= \begin{cases} 0 \text{ für } (x_{02}(t) \leq \frac{x_{max}}{-2} \wedge h(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)) \leq 0) \vee \\ \qquad (x_{02}(t) \geq \frac{x_{max}}{2} \wedge h(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)) \geq 0) \\ -h\big(p_3(t),p_4(t),F_2(t),F_1(t),x_{12}(t)\big) \text{ sonst} \end{cases} \\
Q_3(t) &= 0.06\, A_k\, x_{12}(t) \\
Q_4(t) &= 0.06\, A_k\, x_{11}(t) \\
F_2(t) - F_1(t) &= \frac{A_k}{10}\,(p_3(t) - p_4(t)) - F_R(t) \\
F_R\big(p_3(t),p_4(t),F_2(t),F_1(t),v(t)\big) &= (F_{C0} + F_{H0}\exp(-\frac{x_{12}(t)}{v_{ab}}))\,\mathrm{sign}(x_{12}(t)) + (F_{C0} + F_{H0}) \\
&\quad * \,\mathrm{sn}(x_{12}(t))\,\mathrm{sign}(\frac{A_k}{10}\,(p_3(t) - p_4(t)) - F_2(t) + F_1(t)) \\
h\big(p_3(t),p_4(t),F_2(t),F_1(t),v(t)\big) &= \frac{A_k/10\,(p_3(t) - p_4(t)) - F_2(t) + F_1(t) + d\,v(t)}{m}
\end{aligned}
$$

Externe Darstellung in ^{art}*deco*

Modelltiefe 0: Statische Verhaltensbeschreibung

```
position[GATE.1] =
   (CASE
      WHEN h(t,velocity[GATE.1]) >= xmax[SELF] / 2: xmax[SELF] / 2
      WHEN h(t,velocity[GATE.1]) <= -xmax[SELF] / 2: -xmax[SELF] / 2
      OTHERWISE: h(t,velocity[GATE.1]))

position[GATE.2] =
   (CASE
      WHEN h(t,velocity[GATE.2]) >= xmax[SELF] / 2: xmax[SELF] / 2
      WHEN h(t,velocity[GATE.2]) <= -xmax[SELF] / 2: -xmax[SELF] / 2
      OTHERWISE: h(t,velocity[GATE.2]))

velocity[GATE.1] = -velocity[GATE.2]
velocity[GATE.2] = -velocity[GATE.1]
velocity[GATE.1] = flow[GATE.4] / (0.06 * Ak[SELF])
velocity[GATE.2] = flow[GATE.3] / (0.06 * Ak[SELF])

acceleration[GATE.1] = 0
acceleration[GATE.2] = 0

flow[GATE.3] = 0.06 * Ak[SELF] * velocity[GATE.2]
flow[GATE.4] = 0.06 * Ak[SELF] * velocity[GATE.1]

pressure[GATE.3] =
   pressure[GATE.4] + 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1])

pressure[GATE.4] =
   pressure[GATE.3] - 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1])

force[GATE.1] =
   force[GATE.2] - Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])

force[GATE.2] =
   force[GATE.1] + Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])

h(t,v) = position[SELF] + t * v
```

Modelltiefe 1: Berücksichtigung der Massenträgheit

```
diff(position[GATE.1]) = velocity[GATE.1]
diff(position[GATE.2]) = velocity[GATE.2]

diff(velocity[GATE.1]) = acceleration[GATE.1]
diff(velocity[GATE.2]) = acceleration[GATE.2]

position[GATE.1] = -position[GATE.2]
position[GATE.2] = -position[GATE.1]

velocity[GATE.1] = -velocity[GATE.2]
```

```
velocity[GATE.2] = -velocity[GATE.1]
velocity[GATE.1] = flow[GATE.4] / (0.06 * Ak[SELF])
velocity[GATE.2] = flow[GATE.3] / (0.06 * Ak[SELF])

acceleration[GATE.1] = -acceleration[GATE.2]
acceleration[GATE.2] = -acceleration[GATE.1]

acceleration[GATE.1] =
   (IF (position[GATE.1] <= -xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.1]) <= 0) OR
       (position[GATE.1] >= xmax[SELF] / 2  AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.1]) >= 0)
    THEN 0
    ELSE -h(pressure[GATE.3],pressure[GATE.4],
            force[GATE.2],force[GATE.1],velocity[GATE.1]))

acceleration[GATE.2] =
   (IF (position[GATE.2] <= -xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.2]) <= 0) OR
       (position[GATE.2] >= xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.2]) >= 0)
    THEN 0
    ELSE -h(pressure[GATE.3],pressure[GATE.4],
            force[GATE.2],force[GATE.1],velocity[GATE.2]))

flow[GATE.3] = 0.06 * Ak[SELF] * velocity[GATE.2]
flow[GATE.4] = 0.06 * Ak[SELF] * velocity[GATE.1]

pressure[GATE.3] =
   pressure[GATE.4] + 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1])

pressure[GATE.4] =
   pressure[GATE.3] - 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1])

force[GATE.1] =
   force[GATE.2] - Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])

force[GATE.2] =
   force[GATE.1] + Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])

h(p3,p4,F2,F1,v) =
   (Ak[SELF] / 10 * (p3 - p4) - F2 + F1 + d[SELF] * v) / m[SELF]
```

Modelltiefe 2: Zusätzlich zur Modelltiefe 1, Berücksichtigung des Druckaufbaus

```
diff(position[GATE.1]) = velocity[GATE.1]
diff(position[GATE.2]) = velocity[GATE.2]

diff(velocity[GATE.1]) = acceleration[GATE.1]
diff(velocity[GATE.2]) = acceleration[GATE.2]
```

```
diff(pressure[GATE.3]) =
   (16660 * Eoel(pressure[GATE.3]))
   / (V0[SELF] + 1000 * Ak[SELF] * position[GATE.2])
   * (flow[GATE.3] + 0.06 * Ak[SELF] * velocity[GATE.2])

diff(pressure[GATE.4]) =
   (16660 * Eoel(pressure[GATE.4]))
   / (V0[SELF] + 1000 * Ak[SELF] * position[GATE.1])
   * (flow[GATE.3] + 0.06 * Ak[SELF] * velocity[GATE.1])

position[GATE.1] = -position[GATE.2]
position[GATE.2] = -position[GATE.1]

velocity[GATE.1] = -velocity[GATE.2]
velocity[GATE.2] = -velocity[GATE.1]
velocity[GATE.1] = flow[GATE.4] / (0.06 * Ak[SELF])
velocity[GATE.2] = flow[GATE.3] / (0.06 * Ak[SELF])

acceleration[GATE.1] = -acceleration[GATE.2]
acceleration[GATE.2] = -acceleration[GATE.1]

acceleration[GATE.1] =
   (IF (position[GATE.1] <= -xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.1]) <= 0) OR
       (position[GATE.1] >= xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.1]) >= 0)
    THEN 0
    ELSE -h(pressure[GATE.3],pressure[GATE.4],
            force[GATE.2],force[GATE.1],velocity[GATE.1]))

acceleration[GATE.2] =
   (IF (position[GATE.2] <= -xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.2]) <= 0) OR
       (position[GATE.2] >= xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.2]) >= 0)
    THEN 0
    ELSE -h(pressure[GATE.3],pressure[GATE.4],
            force[GATE.2],force[GATE.1],velocity[GATE.2]))

flow[GATE.3] = 0.06 * Ak[SELF] * velocity[GATE.2]
flow[GATE.4] = 0.06 * Ak[SELF] * velocity[GATE.1]

pressure[GATE.3] =
   pressure[GATE.4] + 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1])

pressure[GATE.4] =
   pressure[GATE.3] - 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1])

force[GATE.1] =
   force[GATE.2] - Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])
```

```
force[GATE.2] =
   force[GATE.1] + Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])

h(p3,p4,F2,F1,v) =
   (Ak[SELF] / 10 * (p3 - p4) - F2 + F1 + d[SELF] * v) / m[SELF]
```

Modelltiefe 3: Zusätzlich zur Modelltiefe 2, Berücksichtigung der Reibungskräfte

```
diff(position[GATE.1]) = velocity[GATE.1]
diff(position[GATE.2]) = velocity[GATE.2]

diff(velocity[GATE.1]) = acceleration[GATE.1]
diff(velocity[GATE.2]) = acceleration[GATE.2]

diff(pressure[GATE.3]) =
   (16660 * Eoel(pressure[GATE.3]))
   / (V0[SELF] + 1000 * Ak[SELF] * position[GATE.2])
   * (flow[GATE.3] + 0.06 * Ak[SELF] * velocity[GATE.2])

diff(pressure[GATE.4]) =
   (16660 * Eoel(pressure[GATE.4]))
   / (V0[SELF] + 1000 * Ak[SELF] * position[GATE.1])
   * (flow[GATE.3] + 0.06 * Ak[SELF] * velocity[GATE.1])

position[GATE.1] = -position[GATE.2]
position[GATE.2] = -position[GATE.1]

velocity[GATE.1] = -velocity[GATE.2]
velocity[GATE.2] = -velocity[GATE.1]
velocity[GATE.1] = flow[GATE.4] / (0.06 * Ak[SELF])
velocity[GATE.2] = flow[GATE.3] / (0.06 * Ak[SELF])

acceleration[GATE.1] = -acceleration[GATE.2]
acceleration[GATE.2] = -acceleration[GATE.1]

acceleration[GATE.1] =
   (IF (position[GATE.1] <= -xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.1]) <= 0) OR
       (position[GATE.1] >= xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.1]) >= 0)
    THEN 0
    ELSE -h(pressure[GATE.3],pressure[GATE.4],
            force[GATE.2],force[GATE.1],velocity[GATE.1]))

acceleration[GATE.2] =
   (IF (position[GATE.1] <= -xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.2]) <= 0) OR
       (position[GATE.1] >= xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.2]) >= 0)
    THEN 0
```

```
    ELSE -h(pressure[GATE.3],pressure[GATE.4],
            force[GATE.2],force[GATE.1],velocity[GATE.2]))

flow[GATE.3] = 0.06 * Ak[SELF] * velocity[GATE.2]
flow[GATE.4] = 0.06 * Ak[SELF] * velocity[GATE.1]

pressure[GATE.3] =
   pressure[GATE.4]
   + 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1]
                      + FR(pressure[GATE.3],pressure[GATE.4],
                           force[GATE.2],force[GATE.1],velocity[GATE.1]))

pressure[GATE.4] =
   pressure[GATE.3]
   - 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1]
                      + FR(pressure[GATE.3],pressure[GATE.4],
                           force[GATE.2],force[GATE.1],velocity[GATE.2]))

force[GATE.1] =
   force[GATE.2]
   + FR(pressure[GATE.3],pressure[GATE.4],
        force[GATE.1],force[GATE.2],velocity[GATE.1])
   - Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])

force[GATE.2] =
   force[GATE.1]
   - FR(pressure[GATE.3],pressure[GATE.4],
        force[GATE.1],force[GATE.2],velocity[GATE.2])
   + Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])

FR(p3,p4,F2,F1,v) =
   FC0[SELF] + FH0[SELF] * exp(-v / vab[SELF]) * sign(v)
   + (FC0[SELF] + FH0[SELF]) * sn(v) * sign(Ak[SELF] * (p3 - p4) - F2 + F1)

h(p3,p4,F2,F1,v) =
   (Ak[SELF] / 10 * (p3 - p4) - F2 + F1 - FR(p3,p4,F2,F1,v) + d[SELF] * v)
   / m[SELF]
```

Modelltiefe 4: Zusätzlich zur Modelltiefe 3, Berücksichtigung der Leckagen

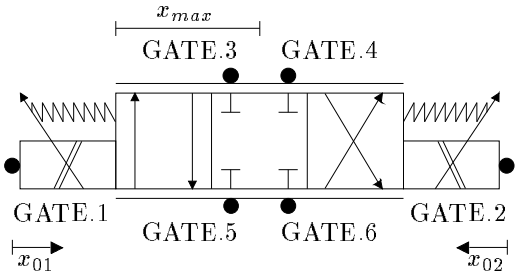
```
diff(position[GATE.1]) = velocity[GATE.1]
diff(position[GATE.2]) = velocity[GATE.2]

diff(velocity[GATE.1]) = acceleration[GATE.1]
diff(velocity[GATE.2]) = acceleration[GATE.2]

diff(pressure[GATE.3]) =
   (16660 * Eoel(pressure[GATE.3]))
   / (V0[SELF] + 1000 * Ak[SELF] * position[GATE.2])
   * (flow[GATE.3] + 0.06 * Ak[SELF] * velocity[GATE.2]
      - QL[SELF] * sign(pressure[GATE.3] - pressure[GATE.4]))
```

```
diff(pressure[GATE.4]) =
   (16660 * Eoel(pressure[GATE.4])) /
   (V0[SELF] + 1000 * Ak[SELF] * position[GATE.1])
   * (flow[GATE.3] + 0.06 * Ak[SELF] * velocity[GATE.1]
      + QL[SELF] * sign(pressure[GATE.3] - pressure[GATE.4]))

position[GATE.1] = -position[GATE.2]
position[GATE.2] = -position[GATE.1]

velocity[GATE.1] = -velocity[GATE.2]
velocity[GATE.2] = -velocity[GATE.1]
velocity[GATE.1] = flow[GATE.4] / (0.06 * Ak[SELF])
velocity[GATE.2] = flow[GATE.3] / (0.06 * Ak[SELF])

acceleration[GATE.1] = -acceleration[GATE.2]
acceleration[GATE.2] = -acceleration[GATE.1]

acceleration[GATE.1] =
   (IF (position[GATE.1] <= -xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.1]) <= 0) OR
       (position[GATE.1] >= xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.1]) >= 0)
    THEN 0
    ELSE -h(pressure[GATE.3],pressure[GATE.4],
            force[GATE.2],force[GATE.1],velocity[GATE.1]))

acceleration[GATE.2] =
   (IF (position[GATE.1] <= -xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.2]) <= 0) OR
       (position[GATE.1] >= xmax[SELF] / 2 AND
        h(pressure[GATE.3],pressure[GATE.4],
          force[GATE.2],force[GATE.1],velocity[GATE.2]) >= 0)
    THEN 0
    ELSE -h(pressure[GATE.3],pressure[GATE.4],
            force[GATE.2],force[GATE.1],velocity[GATE.2]))

flow[GATE.3] = 0.06 * Ak[SELF] * velocity[GATE.2]
flow[GATE.4] = 0.06 * Ak[SELF] * velocity[GATE.1]

pressure[GATE.3] =
   pressure[GATE.4]
   + 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1]
                      + FR(pressure[GATE.3],pressure[GATE.4],
                           force[GATE.2],force[GATE.1],velocity[GATE.1]))

pressure[GATE.4] =
   pressure[GATE.3]
   - 10 / Ak[SELF] * (force[GATE.2] - force[GATE.1])
                      + FR(pressure[GATE.3],pressure[GATE.4],
                           force[GATE.2],force[GATE.1],velocity[GATE.2]))
```

```
force[GATE.1] =
   force[GATE.2]
   + FR(pressure[GATE.3],pressure[GATE.4],
        force[GATE.1],force[GATE.2],velocity[GATE.1])
   - Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])

force[GATE.2] =
   force[GATE.1]
   - FR(pressure[GATE.3],pressure[GATE.4],
        force[GATE.1],force[GATE.2],velocity[GATE.2])
   + Ak[SELF] / 10 * (pressure[GATE.3] - pressure[GATE.4])

FR(p3,p4,F2,F1,v) =
   FC0[SELF] + FH0[SELF] * exp(-v / vab[SELF]) * sign(v)
   + (FC0[SELF] + FH0[SELF]) * sn(v) * sign(Ak[SELF] * (p3 - p4) - F2 + F1)

h(p3,p4,F2,F1,v) =
   (Ak[SELF] / 10 * (p3 - p4) - F2 + F1 - FR(p3,p4,F2,F1,v) + d[SELF] * v)
   / m[SELF]
```

A.2.2 4/3 Wege-Proportionalventil

Schaltzeichen und Parameter



Parameter:		Beschreibung:
c	$[\frac{\mathrm{N}}{\mathrm{mm}}]$	result. Federsteifigkeit
d	$[\frac{\mathrm{N\,s}}{\mathrm{m}}]$	viskose Reibung
K_F	$[\frac{\mathrm{N}}{\mathrm{A}}]$	Kraftverstärkung
K_P	$[\frac{\mathrm{m}}{\mathrm{A\,s}}]$	Druckverstärkung
$R_h(x(t))$	$[\frac{\mathrm{bar\,min}^2}{\mathrm{l}^2}]$	hydr. Widerstand
m	[kg]	bewegte Masse
i_{max}	[A]	max. Ansteuerstrom
x_{max}	[mm]	Hub

Mathematische Darstellung

<u>Modelltiefe 0:</u> Statische Verhaltensbeschreibung

$$
\begin{aligned}
x_{01}(t) &= -x_{02}(t) \\
x_{02}(t) &= \begin{cases} x_{max} & \text{für } h(t, x_{12}(t)) \geq x_{max} \\ -x_{max} & \text{für } h(t, x_{12}(t)) \leq -x_{max} \\ h(t, x_{12}(t)) & \text{sonst} \end{cases} \\
x_{11}(t) &= -x_{12}(t) \\
x_{12}(t) &= K_P\,(i_{a_2}(t) - i_{a_1}(t)) \\
x_{21}(t) &= x_{22}(t) = 0 \\
R_h(x(t)) &= R_{h_{min}}\,\frac{x_{max}}{x(t)}
\end{aligned}
$$

$$h(t, v(t)) = x_0 + t\, v(t)$$

$$i_{a_1}(t) = \begin{cases} Q_3(t)\, i_{max}\, \mathrm{sign}(p_5(t) - p_3(t))\sqrt{|p_5(t) - p_3(t)|} & \text{für } x_{02}(t)<0 \\ Q_3(t)\, i_{max}\, \mathrm{sign}(p_3(t) - p_6(t))\sqrt{|p_3(t) - p_6(t)|} & \text{für } x_{02}(t)>0 \\ 0 & \text{sonst} \end{cases}$$

$$i_{a_2}(t) = \begin{cases} Q_4(t)\, i_{max}\, \mathrm{sign}(p_5(t) - p_4(t))\sqrt{|p_5(t) - p_4(t)|} & \text{für } x_{02}(t)>0 \\ Q_4(t)\, i_{max}\, \mathrm{sign}(p_4(t) - p_6(t))\sqrt{|p_4(t) - p_6(t)|} & \text{für } x_{02}(t)<0 \\ 0 & \text{sonst} \end{cases}$$

$$x_{02}(t) < 0 \;:\; \begin{cases} p_5(t) - p_3(t) = Q_3(t)^2\, R_h(-x_{02}(t)) \\ p_4(t) - p_6(t) = Q_4(t)^2\, R_h(-x_{02}(t)) \\ \qquad Q_5(t)^2 = Q_3(t)^2 \\ \qquad Q_6(t)^2 = Q_4(t)^2 \end{cases}$$

$$x_{02}(t) > 0 \;:\; \begin{cases} p_5(t) - p_4(t) = Q_4(t)^2\, R_h(x_{02}(t)) \\ p_3(t) - p_6(t) = Q_3(t)^2\, R_h(x_{02}(t)) \\ \qquad Q_5(t)^2 = Q_4(t)^2 \\ \qquad Q_6(t)^2 = Q_3(t)^2 \end{cases}$$

$$x_{02}(t) = 0 \;:\; \begin{cases} p_3(t) = \text{const.} \\ p_4(t) = \text{const.} \\ p_5(t) = \text{const.} \\ p_6(t) = \text{const.} \\ Q_3(t) = Q_4(t) = Q_5(t) = Q_6(t) = 0 \end{cases}$$

<u>Modelltiefe 1:</u> Berücksichtigung der Massenträgheit

$$\begin{aligned} \dot{x}_{02}(t) &= x_{12}(t) \\ \dot{x}_{12}(t) &= x_{22}(t) \\ x_{01}(t) &= -x_{02}(t) \\ x_{11}(t) &= -x_{12}(t) \\ x_{21}(t) &= -x_{22}(t) \end{aligned}$$

$$x_{22}(t) = \begin{cases} 0 \text{ für } (x_{02}(t) \le -x_{max} \wedge h(i_{a_1}(t), i_{a_2}(t), x_{12}(t), x_{02}(t)) \le 0) \vee \\ \qquad (x_{02}(t) \ge x_{max} \wedge h(i_{a_1}(t), i_{a_2}(t), x_{12}(t), x_{02}(t)) \ge 0) \\ -h\big(i_{a_1}(t), i_{a_2}(t), x_{12}(t), x_{02}(t)\big) \text{ sonst} \end{cases}$$

$$R_h(x(t)) = R_{h_{min}} \frac{x_{max}}{x(t)}$$

$$h\big(i_{a_1}(t), i_{a_2}(t), v(t), x(t)\big) = \frac{K_F\,(i_{a_1}(t) - i_{a_2}(t)) + d\, v(t) + c\, x(t)}{m}$$

$$i_{a_1}(t) = \begin{cases} Q_3(t)\, i_{max}\, \mathrm{sign}(p_5(t) - p_3(t))\sqrt{|p_5(t) - p_3(t)|} & \text{für } x_{02}(t)<0 \\ Q_3(t)\, i_{max}\, \mathrm{sign}(p_3(t) - p_6(t))\sqrt{|p_3(t) - p_6(t)|} & \text{für } x_{02}(t)>0 \\ 0 & \text{sonst} \end{cases}$$

$$i_{a_2}(t) = \begin{cases} Q_4(t)\, i_{max}\, \mathrm{sign}(p_5(t) - p_4(t))\sqrt{|p_5(t) - p_4(t)|} & \text{für } x_{02}(t)>0 \\ Q_4(t)\, i_{max}\, \mathrm{sign}(p_4(t) - p_6(t))\sqrt{|p_4(t) - p_6(t)|} & \text{für } x_{02}(t)<0 \\ 0 & \text{sonst} \end{cases}$$

$$x_{02}(t) < 0 \;:\; \begin{cases} p_5(t) - p_3(t) = Q_3(t)^2\, R_h(-x_{02}(t)) \\ p_4(t) - p_6(t) = Q_4(t)^2\, R_h(-x_{02}(t)) \\ \qquad Q_5(t)^2 = Q_3(t)^2 \\ \qquad Q_6(t)^2 = Q_4(t)^2 \end{cases}$$

$$x_{02}(t) > 0 \;:\; \begin{cases} p_5(t) - p_4(t) = Q_4(t)^2\, R_h(x_{02}(t)) \\ p_3(t) - p_6(t) = Q_3(t)^2\, R_h(x_{02}(t)) \\ \qquad Q_5(t)^2 = Q_4(t)^2 \\ \qquad Q_6(t)^2 = Q_3(t)^2 \end{cases}$$

$$x_{02}(t) = 0 \;:\; \begin{cases} p_3(t) = \text{const.} \\ p_4(t) = \text{const.} \\ p_5(t) = \text{const.} \\ p_6(t) = \text{const.} \\ Q_3(t) = Q_4(t) = Q_5(t) = Q_6(t) = 0 \end{cases}$$

<u>Modelltiefe 2:</u> Zusätzlich zur Modelltiefe 1, Berücksichtigung des Druckaufbaus

$$\begin{aligned} \dot{x}_{02}(t) &= x_{12}(t) \\ \dot{x}_{12}(t) &= x_{22}(t) \\ x_{01}(t) &= -x_{02}(t) \\ x_{11}(t) &= -x_{12}(t) \\ x_{21}(t) &= -x_{22}(t) \end{aligned}$$

$$x_{22}(t) = \begin{cases} 0 \text{ für } (x_{02}(t) \le -x_{max} \wedge h(i_{a_1}(t), i_{a_2}(t), x_{12}(t), x_{02}(t)) \le 0) \vee \\ \qquad (x_{02}(t) \ge x_{max} \wedge h(i_{a_1}(t), i_{a_2}(t), x_{12}(t), x_{02}(t)) \ge 0) \\ -h\big(i_{a_1}(t), i_{a_2}(t), x_{12}(t), x_{02}(t)\big) \text{ sonst} \end{cases}$$

$$R_h(x(t)) = R_{h_{min}} \frac{x_{max}}{x(t)}$$

$$h\big(i_{a_1}(t), i_{a_2}(t), v(t), x(t)\big) = \frac{K_F\,(i_{a_1}(t) - i_{a_2}(t)) + d\,v(t) + c\,x(t)}{m}$$

$$i_{a_1}(t) = \begin{cases} Q_3(t)\, i_{max}\, \mathrm{sign}(p_5(t) - p_3(t))\sqrt{|p_5(t) - p_3(t)|} & \text{für } x_{02}(t)<0 \\ Q_3(t)\, i_{max}\, \mathrm{sign}(p_3(t) - p_6(t))\sqrt{|p_3(t) - p_6(t)|} & \text{für } x_{02}(t)>0 \\ 0 & \text{sonst} \end{cases}$$

$$i_{a_2}(t) = \begin{cases} Q_4(t)\, i_{max}\, \mathrm{sign}(p_5(t) - p_4(t))\sqrt{|p_5(t) - p_4(t)|} & \text{für } x_{02}(t)>0 \\ Q_4(t)\, i_{max}\, \mathrm{sign}(p_4(t) - p_6(t))\sqrt{|p_4(t) - p_6(t)|} & \text{für } x_{02}(t)<0 \\ 0 & \text{sonst} \end{cases}$$

$$
x_{02}(t) < 0 \;:\; \begin{cases}
\dot{p}_3(t) = \frac{E_{\ddot{O}l}(p_3(t))\,(p_5(t)-p_3(t))}{V_0\,R_h(-x_{02}(t))} - \frac{E_{\ddot{O}l}(p_3(t))\,Q_3(t)^2}{V_0} \\
\dot{p}_4(t) = \frac{E_{\ddot{O}l}(p_4(t))\,(p_6(t)-p_4(t))}{V_0\,R_h(-x_{02}(t))} + \frac{E_{\ddot{O}l}(p_4(t))\,Q_4(t)^2}{V_0} \\
\dot{p}_5(t) = \frac{E_{\ddot{O}l}(p_5(t))\,(p_3(t)-p_5(t))}{V_0\,R_h(-x_{02}(t))} + \frac{E_{\ddot{O}l}(p_5(t))\,Q_5(t)^2}{V_0} \\
\dot{p}_6(t) = \frac{E_{\ddot{O}l}(p_6(t))\,(p_4(t)-p_6(t))}{V_0\,R_h(-x_{02}(t))} - \frac{E_{\ddot{O}l}(p_6(t))\,Q_6(t)^2}{V_0} \\
Q_5(t)^2 = Q_3(t)^2 \\
Q_6(t)^2 = Q_4(t)^2
\end{cases}
$$

$$
x_{02}(t) > 0 \;:\; \begin{cases}
\dot{p}_3(t) = \frac{E_{\ddot{O}l}(p_3(t))\,(p_6(t)-p_3(t))}{V_0\,R_h(x_{02}(t))} + \frac{E_{\ddot{O}l}(p_3(t))\,Q_3(t)^2}{V_0} \\
\dot{p}_4(t) = \frac{E_{\ddot{O}l}(p_4(t))\,(p_5(t)-p_4(t))}{V_0\,R_h(x_{02}(t))} - \frac{E_{\ddot{O}l}(p_4(t))\,Q_4(t)^2}{V_0} \\
\dot{p}_5(t) = \frac{E_{\ddot{O}l}(p_5(t))\,(p_4(t)-p_5(t))}{V_0\,R_h(x_{02}(t))} + \frac{E_{\ddot{O}l}(p_5(t))\,Q_5(t)^2}{V_0} \\
\dot{p}_6(t) = \frac{E_{\ddot{O}l}(p_6(t))\,(p_3(t)-p_6(t))}{V_0\,R_h(x_{02}(t))} - \frac{E_{\ddot{O}l}(p_6(t))\,Q_6(t)^2}{V_0} \\
Q_5(t)^2 = Q_4(t)^2 \\
Q_6(t)^2 = Q_3(t)^2
\end{cases}
$$

$$
x_{02}(t) = 0 \;:\; \begin{cases}
p_3(t) = \text{const.} \\
p_4(t) = \text{const.} \\
p_5(t) = \text{const.} \\
p_6(t) = \text{const.} \\
Q_3(t) = Q_4(t) = Q_5(t) = Q_6(t) = 0
\end{cases}
$$

Externe Darstellung in ^{art}*deco*

<u>Modelltiefe 0:</u> Statische Verhaltensbeschreibung

```
position[GATE.1] =
   (CASE
      WHEN h(t,velocity[GATE.1]) >= xmax[SELF]: xmax[SELF]
      WHEN h(t,velocity[GATE.1]) <= -xmax[SELF]: -xmax[SELF]
      OTHERWISE: h(t,velocity[GATE.1]))

position[GATE.2] =
   (CASE
      WHEN h(t,velocity[GATE.2]) >= xmax[SELF]: xmax[SELF]
      WHEN h(t,velocity[GATE.2]) <= -xmax[SELF]: -xmax[SELF]
      OTHERWISE: h(t,velocity[GATE.2]))

position[GATE.1] = -position[GATE.2]
position[GATE.2] = -position[GATE.1]

velocity[GATE.1] =
   Kp[SELF] * (signal.current[GATE.2] - signal.current[GATE.1])
```

```
velocity[GATE.2] =
   Kp[SELF] * (signal.current[GATE.1] - signal.current[GATE.2])

velocity[GATE.1] = -velocity[GATE.2]
velocity[GATE.2] = -velocity[GATE.1]

acceleration[GATE.1] = 0
acceleration[GATE.2] = 0

rh(x) = rhmin[SELF] * xmax[SELF] / x

signal.current[GATE.1] =
   (CASE
      WHEN position[GATE.2] < 0:
        flow[GATE.3] * imax[SELF] * sign(pressure[GATE.5] - pressure[GATE.3])
        * sqrt(abs(pressure[GATE.5] - pressure[GATE.3]))
      WHEN position[GATE.2] > 0:
        flow[GATE.3] * imax[SELF] * sign(pressure[GATE.3] - pressure[GATE.6])
        * sqrt(abs(pressure[GATE.3] - pressure[GATE.6]))
     OTHERWISE: 0)

signal.current[GATE.2] =
   (CASE
      WHEN position[GATE.2] > 0:
        flow[GATE.4] * imax[SELF] * sign(pressure[GATE.5] - pressure[GATE.4])
        * sqrt(abs(pressure[GATE.5] - pressure[GATE.4]))
      WHEN position[GATE.2] < 0:
        flow[GATE.4] * imax[SELF] * sign(pressure[GATE.4] - pressure[GATE.6])
        * sqrt(abs(pressure[GATE.4] - pressure[GATE.6]))
      OTHERWISE: 0)

pressure[GATE.3] =
   (CASE
      WHEN position[GATE.2] < 0:
         (pressure[GATE.5] - sqr(flow[GATE.3]) * rh(-position[GATE.2])
      WHEN position[GATE.2] > 0:
         (pressure[GATE.6] + sqr(flow[GATE.3]) * rh(position[GATE.2])
      OTHERWISE: const(pressure[GATE.3]))

pressure[GATE.4] =
   (CASE
      WHEN position[GATE.2] < 0:
         (pressure[GATE.6] + sqr(flow[GATE.4]) * rh(-position[GATE.2])
      WHEN position[GATE.2] > 0:
         (pressure[GATE.5] - sqr(flow[GATE.4]) * rh(position[GATE.2])
      OTHERWISE: const(pressure[GATE.4]))

pressure[GATE.5] =
   (CASE
      WHEN position[GATE.2] < 0:
         (pressure[GATE.3] + sqr(flow[GATE.3]) * rh(-position[GATE.2])
      WHEN position[GATE.2] > 0:
         (pressure[GATE.4] + sqr(flow[GATE.4]) * rh(position[GATE.2])
      OTHERWISE: const(pressure[GATE.5]))
```

```
pressure[GATE.6] =
   (CASE
      WHEN position[GATE.2] < 0:
         (pressure[GATE.4] - sqr(flow[GATE.4]) * rh(-position[GATE.2])
      WHEN position[GATE.2] > 0:
         (pressure[GATE.3] - sqr(flow[GATE.3]) * rh(position[GATE.2])
      OTHERWISE: const(pressure[GATE.6]))

flow[GATE.3] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.5] - pressure[GATE.3]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.3] - pressure[GATE.6]) / rh(position[GATE.2]))
      OTHERWISE: 0)

flow[GATE.4] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.4] - pressure[GATE.6]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.5] - pressure[GATE.4]) / rh(position[GATE.2]))
      OTHERWISE: 0)

flow[GATE.5] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.5] - pressure[GATE.3]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.5] - pressure[GATE.4]) / rh(position[GATE.2]))
      OTHERWISE: 0)

flow[GATE.6] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.4] - pressure[GATE.6]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.3] - pressure[GATE.6]) / rh(position[GATE.2]))
      OTHERWISE: 0)
```

Modelltiefe 1: Berücksichtigung der Massenträgheit

```
diff(position[GATE.1]) = velocity[GATE.1]
diff(position[GATE.2]) = velocity[GATE.2]

diff(velocity[GATE.1]) = acceleration[GATE.1]
diff(velocity[GATE.2]) = acceleration[GATE.2]

position[GATE.1] = -position[GATE.2]
position[GATE.2] = -position[GATE.1]

velocity[GATE.1] = -velocity[GATE.2]
velocity[GATE.2] = -velocity[GATE.1]
```

```
acceleration[GATE.1] = -acceleration[GATE.2]
acceleration[GATE.2] = -acceleration[GATE.1]

acceleration[GATE.1] =
   (IF (position[GATE.1] <= -xmax[SELF] AND
        h(signal.current[GATE.1],signal.current[GATE.2],
          velocity[GATE.1],position[GATE.1]) <= 0) OR
       (position[GATE.1] >= xmax[SELF] AND
        h(signal.current[GATE.1],signal.current[GATE.2],
          velocity[GATE.1],position[GATE.1]) >= 0))
    THEN 0
    ELSE -h(signal.current[GATE.1],signal.current[GATE.2],
            velocity[GATE.1],position[GATE.1]))

acceleration[GATE.2] =
   (IF (position[GATE.2] <= -xmax[SELF] AND
        h(signal.current[GATE.1],signal.current[GATE.2],
          velocity[GATE.2],position[GATE.2]) <= 0) OR
       (position[GATE.2] >= xmax[SELF] AND
        h(signal.current[GATE.1],signal.current[GATE.2],
          velocity[GATE.2],position[GATE.2]) >= 0))
    THEN 0
    ELSE -h(signal.current[GATE.1],signal.current[GATE.2],
            velocity[GATE.2],position[GATE.2]))

rh(x) = rhmin[SELF] * xmax[SELF] / x

h(i1,i2,v,x) = Kf[SELF] * (i1 - i2) + d[SELF] * v + c[SELF] * x

signal.current[GATE.1] =
   (CASE
      WHEN position[GATE.2] < 0:
        flow[GATE.3] * imax[SELF] * sign(pressure[GATE.5] - pressure[GATE.3])
        * sqrt(abs(pressure[GATE.5] - pressure[GATE.3]))
      WHEN position[GATE.2] > 0:
        flow[GATE.3] * imax[SELF] * sign(pressure[GATE.3] - pressure[GATE.6])
        * sqrt(abs(pressure[GATE.3] - pressure[GATE.6]))
     OTHERWISE: 0)

signal.current[GATE.2] =
   (CASE
      WHEN position[GATE.2] > 0:
        flow[GATE.4] * imax[SELF] * sign(pressure[GATE.5] - pressure[GATE.4])
        * sqrt(abs(pressure[GATE.5] - pressure[GATE.4]))
      WHEN position[GATE.2] < 0:
        flow[GATE.4] * imax[SELF] * sign(pressure[GATE.4] - pressure[GATE.6])
        * sqrt(abs(pressure[GATE.4] - pressure[GATE.6]))
      OTHERWISE: 0)

pressure[GATE.3] =
   (CASE
      WHEN position[GATE.2] < 0:
         (pressure[GATE.5] - sqr(flow[GATE.3]) * rh(-position[GATE.2])
      WHEN position[GATE.2] > 0:
         (pressure[GATE.6] + sqr(flow[GATE.3]) * rh(position[GATE.2])
```

```
      OTHERWISE: const(pressure[GATE.3]))

pressure[GATE.4] =
   (CASE
      WHEN position[GATE.2] < 0:
         (pressure[GATE.6] + sqr(flow[GATE.4]) * rh(-position[GATE.2])
      WHEN position[GATE.2] > 0:
         (pressure[GATE.5] - sqr(flow[GATE.4]) * rh(position[GATE.2])
      OTHERWISE: const(pressure[GATE.4]))

pressure[GATE.5] =
   (CASE
      WHEN position[GATE.2] < 0:
         (pressure[GATE.3] + sqr(flow[GATE.3]) * rh(-position[GATE.2])
      WHEN position[GATE.2] > 0:
         (pressure[GATE.4] + sqr(flow[GATE.4]) * rh(position[GATE.2])
      OTHERWISE: const(pressure[GATE.5]))

pressure[GATE.6] =
   (CASE
      WHEN position[GATE.2] < 0:
         (pressure[GATE.4] - sqr(flow[GATE.4]) * rh(-position[GATE.2])
      WHEN position[GATE.2] > 0:
         (pressure[GATE.3] - sqr(flow[GATE.3]) * rh(position[GATE.2])
      OTHERWISE: const(pressure[GATE.6]))

flow[GATE.3] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.5] - pressure[GATE.3]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.3] - pressure[GATE.6]) / rh(position[GATE.2]))
      OTHERWISE: 0)

flow[GATE.4] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.4] - pressure[GATE.6]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.5] - pressure[GATE.4]) / rh(position[GATE.2]))
      OTHERWISE: 0)

flow[GATE.5] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.5] - pressure[GATE.3]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.5] - pressure[GATE.4]) / rh(position[GATE.2]))
      OTHERWISE: 0)

flow[GATE.6] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.4] - pressure[GATE.6]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
```

```
         sqrt((pressure[GATE.3] - pressure[GATE.6]) / rh(position[GATE.2]))
      OTHERWISE: 0)
```

Modelltiefe 2: Zusätzlich zur Modelltiefe 1, Berücksichtigung des Druckaufbaus

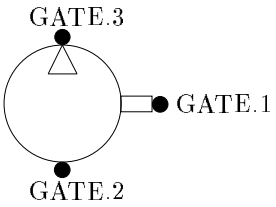
```
diff(position[GATE.1]) = velocity[GATE.1]
diff(position[GATE.2]) = velocity[GATE.2]

diff(velocity[GATE.1]) = acceleration[GATE.1]
diff(velocity[GATE.2]) = acceleration[GATE.2]

diff(pressure[GATE.3]) =
   (CASE
      WHEN position[GATE.2] < 0:
         (Eoel(pressure[GATE.3]) * (pressure[GATE.5] - pressure[GATE.3])
         / (V0[SELF] * rh(-position[GATE.2]))
         - (Eoel(pressure[GATE.3]) * sqr(flow[GATE.3])) / V0[SELF]
      WHEN position[GATE.2] > 0:
         (Eoel(pressure[GATE.3]) * (pressure[GATE.6] - pressure[GATE.3])
         / (V0[SELF] * rh(position[GATE.2]))
         + (Eoel(pressure[GATE.3]) * sqr(flow[GATE.3])) / V0[SELF]
      OTHERWISE: const(pressure[GATE.3]))

diff(pressure[GATE.4]) =
   (CASE
      WHEN position[GATE.2] < 0:
         (Eoel(pressure[GATE.4]) * (pressure[GATE.6] - pressure[GATE.4])
         / (V0[SELF] * rh(-position[GATE.2]))
         + (Eoel(pressure[GATE.4]) * sqr(flow[GATE.4])) / V0[SELF]
      WHEN position[GATE.2] > 0:
         (Eoel(pressure[GATE.4]) * (pressure[GATE.5] - pressure[GATE.4])
         / (V0[SELF] * rh(position[GATE.2]))
         - (Eoel(pressure[GATE.4]) * sqr(flow[GATE.4])) / V0[SELF]
      OTHERWISE: const(pressure[GATE.4]))

diff(pressure[GATE.5]) =
   (CASE
      WHEN position[GATE.2] < 0:
         (Eoel(pressure[GATE.5]) * (pressure[GATE.3] - pressure[GATE.5])
         / (V0[SELF] * rh(-position[GATE.2]))
         + (Eoel(pressure[GATE.5]) * sqr(flow[GATE.5])) / V0[SELF]
      WHEN position[GATE.2] > 0:
         (Eoel(pressure[GATE.5]) * (pressure[GATE.4] - pressure[GATE.5])
         / (V0[SELF] * rh(position[GATE.2]))
         + (Eoel(pressure[GATE.5]) * sqr(flow[GATE.5])) / V0[SELF]
      OTHERWISE: const(pressure[GATE.5]))

diff(pressure[GATE.6]) =
   (CASE
      WHEN position[GATE.2] < 0:
         (Eoel(pressure[GATE.6]) * (pressure[GATE.4] - pressure[GATE.6])
         / (V0[SELF] * rh(-position[GATE.2]))
         - (Eoel(pressure[GATE.6]) * sqr(flow[GATE.6])) / V0[SELF]
      WHEN position[GATE.2] > 0:
```

```
          (Eoel(pressure[GATE.6]) * (pressure[GATE.3] - pressure[GATE.6])
          / (V0[SELF] * rh(position[GATE.2]))
          - (Eoel(pressure[GATE.6]) * sqr(flow[GATE.6])) / V0[SELF]
       OTHERWISE: const(pressure[GATE.6]))

position[GATE.1] = -position[GATE.2]
position[GATE.2] = -position[GATE.1]

velocity[GATE.1] = -velocity[GATE.2]
velocity[GATE.2] = -velocity[GATE.1]

acceleration[GATE.1] = -acceleration[GATE.2]
acceleration[GATE.2] = -acceleration[GATE.1]

acceleration[GATE.1] =
   (IF (position[GATE.1] <= -xmax[SELF] AND
        h(signal.current[GATE.1],signal.current[GATE.2],
          velocity[GATE.1],position[GATE.1]) <= 0) OR
       (position[GATE.1] >= xmax[SELF] AND
        h(signal.current[GATE.1],signal.current[GATE.2],
          velocity[GATE.1],position[GATE.1]) >= 0))
    THEN 0
    ELSE -h(signal.current[GATE.1],signal.current[GATE.2],
            velocity[GATE.1],position[GATE.1]))

acceleration[GATE.2] =
   (IF (position[GATE.2] <= -xmax[SELF] AND
        h(signal.current[GATE.1],signal.current[GATE.2],
          velocity[GATE.2],position[GATE.2]) <= 0) OR
       (position[GATE.2] >= xmax[SELF] AND
        h(signal.current[GATE.1],signal.current[GATE.2],
          velocity[GATE.2],position[GATE.2]) >= 0))
    THEN 0
    ELSE -h(signal.current[GATE.1],signal.current[GATE.2],
            velocity[GATE.2],position[GATE.2]))

rh(x) = rhmin[SELF] * xmax[SELF] / x

h(i1,i2,v,x) = Kf[SELF] * (i1 - i2) + d[SELF] * v + c[SELF] * x

signal.current[GATE.1] =
   (CASE
      WHEN position[GATE.2] < 0:
        flow[GATE.3] * imax[SELF] * sign(pressure[GATE.5] - pressure[GATE.3])
        * sqrt(abs(pressure[GATE.5] - pressure[GATE.3]))
      WHEN position[GATE.2] > 0:
        flow[GATE.3] * imax[SELF]) * sign(pressure[GATE.3] - pressure[GATE.6])
        * sqrt(abs(pressure[GATE.3] - pressure[GATE.6]))
     OTHERWISE: 0)

signal.current[GATE.2] =
   (CASE
      WHEN position[GATE.2] > 0:
        flow[GATE.4] * imax[SELF] * sign(pressure[GATE.5] - pressure[GATE.4])
        * sqrt(abs(pressure[GATE.5] - pressure[GATE.4]))
```

```
      WHEN position[GATE.2] < 0:
        flow[GATE.4] * imax[SELF] * sign(pressure[GATE.4] - pressure[GATE.6])
        * sqrt(abs(pressure[GATE.4] - pressure[GATE.6]))
      OTHERWISE: 0)

flow[GATE.3] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.5] - pressure[GATE.3]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.3] - pressure[GATE.6]) / rh(position[GATE.2]))
      OTHERWISE: 0)

flow[GATE.4] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.4] - pressure[GATE.6]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.5] - pressure[GATE.4]) / rh(position[GATE.2]))
      OTHERWISE: 0)

flow[GATE.5] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.5] - pressure[GATE.3]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.5] - pressure[GATE.4]) / rh(position[GATE.2]))
      OTHERWISE: 0)

flow[GATE.6] =
   (CASE
      WHEN position[GATE.2] < 0:
         sqrt((pressure[GATE.4] - pressure[GATE.6]) / rh(-position[GATE.2]))
      WHEN position[GATE.2] > 0:
         sqrt((pressure[GATE.3] - pressure[GATE.6]) / rh(position[GATE.2]))
      OTHERWISE: 0)
```

A.2.3 Pumpe

Schaltzeichen und Parameter



Parameter:		Beschreibung:
c	$[\frac{\mathrm{N}}{\mathrm{mm}}]$	resultierende Federsteifigkeit
d	$[\frac{\mathrm{N\,s}}{\mathrm{m}}]$	viskose Reibung
V_F	$[\frac{\mathrm{cm}^3}{\mathrm{U}}]$	Fördervolumen
Θ	$[\mathrm{kg\,m}^2]$	Trägheitsmoment
M_{C0}	$[\mathrm{N\,m}]$	Coulombsches Reibmoment
M_{H0}	$[\mathrm{N\,m}]$	Haftmoment
$M_R(\omega(t), M_L(t))$	$[\mathrm{N\,m}]$	Gesamtreibmoment, wobei $M_L(t)$ das result. Lastmoment bezeichnet
ω_{ab}	$[\frac{\mathrm{rad}}{\mathrm{s}}]$	Abklingkonstante

Mathematische Darstellung

Modelltiefe 0: Statische Verhaltensbeschreibung

$$
\begin{aligned}
\omega_{01}(t) &= \omega_0 + t\,\omega_{11}(t) \\
\omega_{11}(t) &= \frac{Q_2(t)}{V_F} \\
\omega_{21}(t) &= 0 \\
Q_2(t) &= V_F\,\omega_{11}(t) \\
Q_3(t) &= -V_F\,\omega_{11}(t) \\
p_3(t) - p_2(t) &= \frac{M_1(t)}{V_F}
\end{aligned}
$$

Modelltiefe 1: Berücksichtigung der Massenträgheit

$$
\begin{aligned}
\dot{\omega}_{01}(t) &= \omega_{11}(t) \\
\dot{\omega}_{11}(t) &= \omega_{21}(t) \\
\omega_{21}(t) &= \frac{M_1(t) - d\,\omega_{11}(t) - c\,\omega_{01}(t)}{\Theta} \\
Q_2(t) &= V_F\,\omega_{11}(t) \\
Q_3(t) &= -V_F\,\omega_{11}(t) \\
p_3(t) - p_2(t) &= \frac{M_1(t)}{V_F}
\end{aligned}
$$

Modelltiefe 2: Zusätzlich zur Modelltiefe 1, Berücksichtigung der Reibungskräfte

$$\dot{\omega}_{01}(t) = \omega_{11}(t)$$

$$
\begin{aligned}
\dot{\omega}_{11}(t) &= \omega_{21}(t) \\
\omega_{21}(t) &= \frac{M_1(t) - d\,\omega_{12}(t) - M_R(p_3(t),p_2(t),M_1(t),\omega_{11}(t)) - c\,\omega_{01}(t)}{\Theta} \\
Q_2(t) &= V_F\,\omega_{11}(t) \\
Q_3(t) &= -V_F\,\omega_{11}(t) \\
p_3(t) - p_2(t) &= \frac{M_1(t)}{V_F} \\
M_R(p_3(t),p_2(t),M_1(t),\omega(t)) &= (M_{C0} + M_{H0}\exp(-\frac{\omega(t)}{\omega_{ab}}))\,\mathrm{sign}(\omega(t)) + (M_{C0} + M_{H0}) \\
&\quad * \mathrm{sn}(\omega(t))\,\mathrm{sign}(V_F\,(p_3(t) - p_4(t)) - M_1(t))
\end{aligned}
$$

Externe Darstellung in ^{art}*deco*

<u>Modelltiefe 0:</u> Statische Verhaltensbeschreibung

```
angle.position[GATE.1] = position[SELF] + t * angle.velocity[GATE.1]

angle.velocity[GATE.1] = flow[GATE.2] / Vf[SELF]

angle.acceleration[GATE.1] = 0

flow[GATE.2] = Vf[SELF] * angle.velocity[GATE.1]
flow[GATE.3] = -Vf[SELF] * angle.velocity[GATE.1]

flow[GATE.2] = -flow[GATE.3]
flow[GATE.3] = -flow[GATE.2]

pressure[GATE.2] = pressure[GATE.3] - torque[GATE.1] / Vf[SELF]
pressure[GATE.3] = pressure[GATE.2] + torque[GATE.1] / Vf[SELF]

torque[GATE.1] = Vf[SELF] * (pressure[GATE.3] - pressure[GATE.2])
```

<u>Modelltiefe 1:</u> Berücksichtigung der Massenträgheit

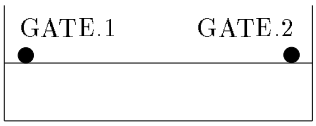
```
diff(angle.position[GATE.1]) = angle.velocity[GATE.1]
diff(angle.velocity[GATE.1]) = angle.acceleration[GATE.1]

angle.acceleration[GATE.1] =
   (torque[GATE.1] - c[SELF] * angle.position[GATE.1]
    - d[SELF] * angle.velocity[GATE.1]) / Theta[SELF]

flow[GATE.2] = Vf[SELF] * angle.velocity[GATE.1]
flow[GATE.3] = -Vf[SELF] * angle.velocity[GATE.1]

flow[GATE.2] = -flow[GATE.3]
flow[GATE.3] = -flow[GATE.2]

pressure[GATE.2] = pressure[GATE.3] - torque[GATE.1] / Vf[SELF]
pressure[GATE.3] = pressure[GATE.2] + torque[GATE.1] / Vf[SELF]

torque[GATE.1] = Vf[SELF] * (pressure[GATE.3] - pressure[GATE.2])
```

Modelltiefe 2: Zusätzlich zur Modelltiefe 1, Berücksichtigung der Reibungskräfte

```
diff(angle.position[GATE.1]) = angle.velocity[GATE.1]
diff(angle.velocity[GATE.1]) = angle.acceleration[GATE.1]

angle.acceleration[GATE.1] =
   (torque[GATE.1] - c[SELF] * angle.position[GATE.1]
    - MR(pressure[GATE.3],pressure[GATE.2],torque[GATE.1],
         angle.velocity[GATE.1]) - d[SELF] * angle.velocity[GATE.1])
   / Theta[SELF]

flow[GATE.2] = Vf[SELF] * angle.velocity[GATE.1]
flow[GATE.3] = -Vf[SELF] * angle.velocity[GATE.1]

flow[GATE.2] = -flow[GATE.3]
flow[GATE.3] = -flow[GATE.2]

pressure[GATE.2] = pressure[GATE.3] - torque[GATE.1] / Vf[SELF]
pressure[GATE.3] = pressure[GATE.2] + torque[GATE.1] / Vf[SELF]

torque[GATE.1] = Vf[SELF] * (pressure[GATE.3] - pressure[GATE.2])

MR(p3,p2,M1,w) =
   MC0[SELF] + MH0[SELF] * exp(-w / wab[SELF]) * sign(w)
   + (MC0[SELF] + MH0[SELF]) * sn(w) * sign(Vf[SELF] * (p3 - p2) - M1)
```

A.2.4 Tank

Schaltzeichen und Parameter



Parameter:	Beschreibung:
P_T [bar]	Tankdruck

Mathematische Darstellung

Modelltiefe 0: Statische Verhaltensbeschreibung

$$p_1(t) = p_2(t) = P_T$$
$$Q_1(t) = \text{const.}$$
$$Q_2(t) = \text{const.}$$

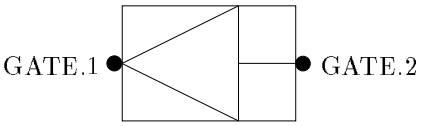
Externe Darstellung in ^{art}*deco*

Modelltiefe 0: Statische Verhaltensbeschreibung

```
pressure[GATE.1] = PT[SELF]
pressure[GATE.2] = PT[SELF]

flow[GATE.1] = const(flow[GATE.1])
flow[GATE.2] = const(flow[GATE.2])
```

A.2.5 Ventilverstärker

Schaltzeichen und Parameter



Parameter:		Beschreibung:
T	[s]	Zeitverzögerung
k	$[\frac{1}{\Omega}]$	Proportionalitätsfaktor
u_{max}	[V]	maximale Spannung
i_{max}	[A]	maximaler Strom

Mathematische Darstellung

Modelltiefe 0: Statische Verhaltensbeschreibung

$$\begin{aligned} i_1(t) &= k\,u_2(t) \\ i_2(t) &= \text{const.} \\ u_1(t) &= \text{const.} \end{aligned}$$

Modelltiefe 1: Berücksichtigung der Spannungsbegrenzung

$$\begin{aligned} i_1(t) &= \begin{cases} k\,u_2(t) & \text{für } |\,u_2(t)\,| \leq u_{max} \\ k\,\text{sign}(u_2(t))\,u_{max} & \text{sonst} \end{cases} \\ i_2(t) &= \text{const.} \\ u_1(t) &= \text{const.} \end{aligned}$$

Modelltiefe 2: Zusätzlich zur Modelltiefe 1, Berücksichtigung zeitlicher Verzögerung

$$\begin{aligned} \dot{i}_1(t) &= \begin{cases} \frac{k\,u_2(t) - i_1(t)}{T} & \text{für } |\,u_2(t)\,| \leq u_{max} \\ \frac{k\,\text{sign}(u_2(t))\,u_{max} - i_1(t)}{T} & \text{sonst} \end{cases} \\ i_2(t) &= \text{const.} \\ u_1(t) &= \text{const.} \end{aligned}$$

Externe Darstellung in ^{art}*deco*

Modelltiefe 0: Statische Verhaltensbeschreibung

```
signal.current[GATE.1] = k[SELF] * signal.voltage[GATE.2]
signal.current[GATE.2] = const(signal.current[GATE.2])

signal.voltage[GATE.1] = const(signal.voltage[GATE.1])
signal.voltage[GATE.2] = signal.current[GATE.1] / k[SELF]
```

Modelltiefe 1: Berücksichtigung der Spannungsbegrenzung

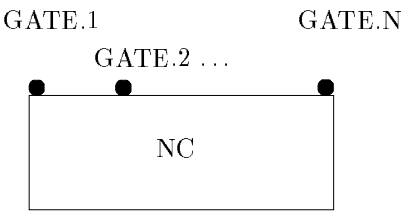
```
signal.current[GATE.1] =
   (IF abs(signal.voltage[GATE.2]) <= umax[SELF] THEN
       k[SELF] * signal.voltage[GATE.2]
    ELSE
       k[SELF] * sign(signal.voltage[GATE.2]) * umax[SELF])

signal.current[GATE.2] = const(signal.current[GATE.2])

signal.voltage[GATE.1] = const(signal.voltage[GATE.1])

signal.voltage[GATE.2] =
   (IF abs(signal.current[GATE.1]) <= imax[SELF] THEN
       signal.current[GATE.1] / k[SELF]
    ELSE
       sign(signal.current[GATE.1]) * umax[SELF])
```

Modelltiefe 2: Zusätzlich zur Modelltiefe 1, Berücksichtigung zeitlicher Verzögerung

```
diff(signal.current[GATE.1]) =
   (IF abs(signal.voltage[GATE.2]) <= umax[SELF] THEN
       (k[SELF] * signal.voltage[GATE.2] - signal.current[GATE.1]) / T[SELF]
    ELSE
       (k[SELF] * sign(signal.voltage[GATE.2]) * umax[SELF]
       - signal.current[GATE.1]) / T[SELF])

diff(signal.voltage[GATE.2]) =
   (IF abs(signal.current[GATE.1]) <= imax[SELF] THEN
       (signal.current[GATE.1] / k[SELF] - signal.voltage[GATE.2]) / T[SELF]
    ELSE
       sign(signal.current[GATE.1]) * umax[SELF] / T[SELF])

signal.current[GATE.2] = const(signal.current[GATE.2])

signal.voltage[GATE.1] = const(signal.voltage[GATE.1])
```

A.2.6 Numeric Control Einheit

Schaltzeichen und Parameter



Parameter:	Beschreibung:
input-function$_1(t)$ [V]	Eingangsfunktion
input-function$_2(t)$ [V]	Eingangsfunktion
$\vdots$ $\vdots$	$\vdots$
input-function$_n(t)$ [V]	Eingangsfunktion

Mathematische Darstellung

<u>Modelltiefe 0:</u> Statische Verhaltensbeschreibung

$$
\begin{aligned}
u_1(t) &= \text{input-function}_1(t) \\
i_1(t) &= \text{const.} \\
u_2(t) &= \text{input-function}_2(t) \\
i_2(t) &= \text{const.} \\
&\vdots \; \vdots \; \vdots \\
u_n(t) &= \text{input-function}_n(t) \\
i_n(t) &= \text{const.}
\end{aligned}
$$

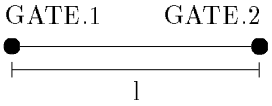
Externe Darstellung in ^{art}*deco*

<u>Modelltiefe 0:</u> Statische Verhaltensbeschreibung

```
signal.voltage[GATE.1] = f1(t)
signal.current[GATE.1] = const(signal.current[GATE.1])

signal.voltage[GATE.2] = f2(t)
signal.current[GATE.2] = const(signal.current[GATE.2])
                        ...
signal.voltage[GATE.N] = fn(t)
signal.current[GATE.N] = const(signal.current[GATE.N])
```

A.2.7 Hydraulische Leitung

Schaltzeichen und Parameter



Parameter:	Beschreibung:
A [mm^2]	Leitungsquerschnitt
l [m]	Leitungslänge
R_h [$\frac{10^5\,\mathrm{N\,min}}{\mathrm{l\,m^2}}$]	hydraulischer Widerstand

Mathematische Darstellung

Modelltiefe 0: Statische Verhaltensbeschreibung

$$p_1(t) = p_2(t)$$
$$Q_1(t) = -Q_2(t)$$

Modelltiefe 1: Berücksichtigung des hydraulischen Widerstands

$$p_2(t) - p_1(t) = \begin{cases} R_h\, Q_2(t)^2 \text{ für } p_2(t) \geq p_1(t) \\ R_h\, Q_1(t)^2 \text{ sonst} \end{cases}$$
$$Q_1(t) = -Q_2(t)$$

Modelltiefe 2: Zusätzlich zur Modelltiefe 1, Berücksichtigung des Druckaufbaus

$$\dot{p}_1(t) = \begin{cases} \frac{E_{\ddot{O}l}(p_1(t))\,(p_2(t)-p_1(t))}{A\,l\,R_h} - \frac{E_{\ddot{O}l}(p_1(t))\,Q_2(t)^2}{A\,l} \text{ für } p_2(t) \geq p_1(t) \\ \frac{E_{\ddot{O}l}(p_1(t))\,(p_2(t)-p_1(t))}{A\,l\,R_h} + \frac{E_{\ddot{O}l}(p_1(t))\,Q_1(t)^2}{A\,l} \text{ sonst} \end{cases}$$
$$\dot{p}_2(t) = \begin{cases} \frac{E_{\ddot{O}l}(p_2(t))\,(p_1(t)-p_2(t))}{A\,l\,R_h} + \frac{E_{\ddot{O}l}(p_2(t))\,Q_2(t)^2}{A\,l} \text{ für } p_2(t) \geq p_1(t) \\ \frac{E_{\ddot{O}l}(p_2(t))\,(p_1(t)-p_2(t))}{A\,l\,R_h} - \frac{E_{\ddot{O}l}(p_2(t))\,Q_1(t)^2}{A\,l} \text{ sonst} \end{cases}$$
$$Q_1(t) = -Q_2(t)$$

Externe Darstellung in ^{art}*deco*

Modelltiefe 0: Statische Verhaltensbeschreibung

```
pressure[GATE.1] = pressure[GATE.2]
pressure[GATE.2] = pressure[GATE.1]

flow[GATE.1] = -flow[GATE.2]
flow[GATE.2] = -flow[GATE.1]
```

Modelltiefe 1: Berücksichtigung des hydraulischen Widerstands

```
pressure[GATE.1] =
   (IF pressure[GATE.2] >= pressure[GATE.1]
    THEN pressure[GATE.2] - Rh[SELF] * sqr(flow[GATE.2])
    ELSE pressure[GATE.2] - Rh[SELF] * sqr(flow[GATE.1]))

pressure[GATE.2] =
   (IF pressure[GATE.2] >= pressure[GATE.1]
    THEN pressure[GATE.1] + Rh[SELF] * sqr(flow[GATE.2])
    ELSE pressure[GATE.1] + Rh[SELF] * sqr(flow[GATE.1]))

flow[GATE.1] = -flow[GATE.2]
flow[GATE.2] = -flow[GATE.1]
```

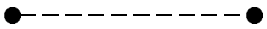
Modelltiefe 2: Zusätzlich zur Modelltiefe 1, Berücksichtigung des Druckaufbaus

```
diff(pressure[GATE.1]) =
   (IF pressure[GATE.2] >= pressure[GATE.1]
    THEN
       (Eoel(pressure[GATE.1]) * (pressure[GATE.2] - pressure[GATE.1]))
       / (A[SELF] * l[SELF] * Rh[SELF])
       - (Eoel(pressure[GATE.1] * sqr(flow[GATE.2])) / (A[SELF] * l[SELF])
    ELSE pressure[GATE.2] - Rh[SELF] * sqr(flow[GATE.1]))

diff(pressure[GATE.2]) =
   (IF pressure[GATE.2] >= pressure[GATE.1]
    THEN pressure[GATE.1] + Rh[SELF] * sqr(flow[GATE.2])
    ELSE pressure[GATE.1] + Rh[SELF] * sqr(flow[GATE.1]))

flow[GATE.1] = -flow[GATE.2]
flow[GATE.2] = -flow[GATE.1]
```

A.2.8 Elektrische Leitung

Schaltzeichen und Parameter

GATE.1 GATE.2



Mathematische Darstellung

Modelltiefe 0: Statische Verhaltensbeschreibung

$$i_1(t) = -i_2(t)$$
$$u_1(t) = u_2(t)$$

Externe Darstellung in ^{art}*deco*

Modelltiefe 0: Statische Verhaltensbeschreibung

```
signal.current[GATE.1] = -signal.current[GATE.2]
signal.current[GATE.2] = -signal.current[GATE.1]

signal.voltage[GATE.1] = signal.voltage[GATE.2]
signal.voltage[GATE.2] = signal.voltage[GATE.1]
```

B Die Regelmenge

Das Domänenwissen ist, wie bereits in Abschnitt 4.2.3 beschrieben, in aussagenlogischen Regeln gefaßt, die zusätzlich um Funktionsaufrufe im Konklusionsteil sowie um Formulierungen quantitativer Aussagen im Konditionsteil einer Regel erweitert worden sind. Eine genaue Definition der Regelsprache beschreibt der nachfolgende Unterabschnitt. Dem schließt sich dann eine Darstellung der gesamten Regelmenge, getrennt nach ihrer jeweiligen Zuständigkeit, an.

B.1 Definition der Syntax

Die formale Beschreibung der Regelsprache ist durch die folgende EBNF[11]-Notation gegeben. In dieser Notation stellen klein- und kursivgedruckte Bezeichner die Nichtterminale der Grammatik dar. Die Produktionen, die die als Lisp-Form erkennbaren Nichtterminale in Terminale überführen, finden in der Grammatik keine Berücksichtigung, ihre Definition wird als bekannt vorausgesetzt.

$$\begin{aligned}
\textit{rule} &::= (\textit{rule–name}\ (\text{IF}\ \textit{condition})\ (\text{THEN}\ \textit{conclusion})) \\
\textit{rule-name} &::= \textit{lisp–atom} \\
\textit{condition} &::= \textit{boolean–lisp–form} \\
\textit{conclusion} &::= \{\textit{fact}\}^{+} \mid (\text{CALL–FUNCTION}\ \textit{lisp–atom}) \\
\textit{fact} &::= (\textit{lisp–atom}\ \{\{\textit{lisp–atom}\}^{+} \mid (\{\textit{lisp–atom}\}^{+})\})
\end{aligned}$$

Unter einer *boolean–lisp–form* ist in diesem Kontext eine beliebige Lisp-Form zu verstehen, deren Auswertung für den Fall, daß alle Variablen der Form gebunden sind, den Wert `TRUE` oder `NIL` zurückgibt. Hat eine Variable in der Lisp-Form keine gültige Bindung, ist das Faktum (Variable Wert) nicht in der Wissensbasis enthalten. Die Bedingung wird entsprechend mit `NIL` bewertet.

B.2 Regeln zur Modelltiefenbestimmung

Die Regeln dieser Menge lassen sich in zwei Teilmengen gruppieren. Die erste Menge umfaßt die Regeln `rule-cut-1` bis `rule-cut-15`. Sie dienen zur Bestimmung der Abschneidekonstanten, mit der dann die Abschneidefrequenz berechnet wird. Die zweite Menge enthält die Regeln `rule-val-1` bis `rule-val-4`, in denen die verschiedenen Bewertungsfaktoren mit den heuristischen Werten initialisiert werden.

```
(SETQ *dynamic-rulebase-1*
      '((rule-cut-1 (IF (AND (>= simulation.intention.number 2)
                             (<= simulation.intention.number 7)
                             (= simulation.intention 1)))
                    (THEN (cut.constant 5)))
```

[11] EBNF (Extended-Backus-Naur-Form)

```
(rule-cut-2 (IF (= simulation.intention.number simulation.intention 2))
            (THEN (cut.constant 40)))

(rule-cut-3 (IF (AND (= simulation.intention.number 3)
                     (= simulation.intention 2)))
            (THEN (cut.constant 20)))

(rule-cut-4 (IF (AND (>= simulation.intention.number 4)
                     (<= simulation.intention.number 7)
                     (= simulation.intention 2)))
            (THEN (cut.constant 10)))

(rule-cut-5 (IF (AND (= simulation.intention.number simulation.intention)
                     (>= simulation.intention.number 3)
                     (<= simulation.intention.number 5)))
            (THEN (cut.constant 100)))

(rule-cut-6 (IF (AND (= simulation.intention.number 4)
                     (= simulation.intention 3)))
            (THEN (cut.constant 30)))

(rule-cut-7 (IF (AND (>= simulation.intention.number 5)
                     (<= simulation.intention.number 7)
                     (= simulation.intention 3)))
            (THEN (cut.constant 20)))

(rule-cut-8 (IF (AND (= simulation.intention.number 5)
                     (= simulation.intention 4)))
            (THEN (cut.constant 50)))

(rule-cut-9 (IF (AND (= simulation.intention.number 6)
                     (= simulation.intention 4)))
            (THEN (cut.constant 40)))

(rule-cut-10 (IF (AND (= simulation.intention.number 7)
                      (= simulation.intention 4)))
             (THEN (cut.constant 30)))

(rule-cut-11 (IF (AND (= simulation.intention.number 6)
                      (= simulation.intention 5)))
             (THEN (cut.constant 100)))

(rule-cut-12 (IF (AND (= simulation.intention.number 7)
                      (= simulation.intention 5)))
             (THEN (cut.constant 50)))

(rule-cut-13 (IF (= simulation.intention.number simulation.intention 6))
             (THEN (cut.constant 500)))

(rule-cut-14 (IF (AND (= simulation.intention.number 7)
                      (= simulation.intention 6)))
             (THEN (cut.constant 100)))

(rule-cut-15 (IF (= simulation.intention.number simulation.intention 7))
             (THEN (cut.constant 1000)))
```

```
        (rule-val-1 (IF (< simulation.intention
                           (FLOOR (* 0.5 simulation.intention.number))))
                    (THEN (frequency.valuation 0.5)
                          (modeldepth.valuation 0.5)
                          (amplification.valuation 0.0)
                          (system.order.valuation 0.0)
                          (non.linearity.valuation 0.0)))

        (rule-val-2 (IF (AND (>= simulation.intention
                                 (FLOOR (* 0.5 simulation.intention.number)))
                             (< simulation.intention
                                (FLOOR (* 2/3 (+ simulation.intention.number 1))))))
                    (THEN (frequency.valuation 0.2)
                          (modeldepth.valuation 0.5)
                          (amplification.valuation 0.3)
                          (system.order.valuation 0.0)
                          (non.linearity.valuation 0.0)))

        (rule-val-3 (IF (AND (>= simulation.intention
                                 (FLOOR (* 2/3 (+ simulation.intention.number 1))))
                             (< simulation.intention simulation.intention.number)))
                    (THEN (frequency.valuation 0.2)
                          (modeldepth.valuation 0.4)
                          (amplification.valuation 0.2)
                          (system.order.valuation 0.2)
                          (non.linearity.valuation 0.0)))

        (rule-val-4 (IF (= simulation.intention simulation.intention.number))
                    (THEN (frequency.valuation 0.1)
                          (modeldepth.valuation 0.4)
                          (amplification.valuation 0.1)
                          (system.order.valuation 0.1)
                          (non.linearity.valuation 0.3)))
    ))
```

B.3 Regeln zur Simulationssteuerung

Diese Regelmenge gliedert sich in vier Teilmengen. Die erste Menge dient zur Festlegung der Fehlerschranken und umfaßt die Regeln `rule-eps-1` bis `rule-eps-31`. Die zweite Menge besteht aus den Regeln `rule-stable-1` und `rule-stable-2`. Sie überprüfen die Stabilität des Differentialgleichungssystems und ermitteln dessen Steifheitsgrad. Ob ein steifes Differentialgleichungssystem vorliegt, bestimmen die Regeln der dritten Teilmenge, `rule-stiff-1` bis `rule-stiff-5`, die zusätzlich auch noch die Fehlergewichtung der einzelnen Lösungskurven festlegen. Die Regeln `rule-method-1` und `rule-method-2` bilden die letzte Teilmenge der Regelmenge und stellen eine optimale Verfahrensauswahl sicher.

```
(SETQ *dynamic-rulebase-2*
      '((rule-eps-1 (IF (method.class is implicit))
                    (THEN (absolute.error.bound 0.0)))
```

```
(rule-eps-2 (IF (AND (>= simulation.intention.number 2)
                     (<= simulation.intention.number 7)
                     (= simulation.intention 1)
                     (method.class is explicit)))
            (THEN (absolute.error.bound 1.0E-3)
                  (relative.error.bound 1.0E-3)))

(rule-eps-3 (IF (AND (>= simulation.intention.number 2)
                     (<= simulation.intention.number 7)
                     (= simulation.intention 1)
                     (method.class is implicit)))
            (THEN (relative.error.bound 1.0E-3)))

(rule-eps-4 (IF (AND (= simulation.intention.number simulation.intention 2)
                     (method.class is explicit)))
            (THEN (absolute.error.bound 1.0E-5)
                  (relative.error.bound 1.0E-5)))

(rule-eps-5 (IF (AND (= simulation.intention.number simulation.intention 2)
                     (method.class is implicit)))
            (THEN (relative.error.bound 1.0E-5)))

(rule-eps-6 (IF (AND (= simulation.intention.number 3)
                     (= simulation.intention 2)
                     (method.class is explicit)))
            (THEN (absolute.error.bound 1.0E-4)
                  (relative.error.bound 1.0E-4)))

(rule-eps-7 (IF (AND (= simulation.intention.number 3)
                     (= simulation.intention 2)
                     (method.class is implicit)))
            (THEN (relative.error.bound 1.0E-4)))

(rule-eps-8 (IF (AND (> simulation.intention.number 3)
                     (<= simulation.intention.number 7)
                     (= simulation.intention 2)
                     (method.class is explicit)))
            (THEN (absolute.error.bound 1.0E-5)
                  (relative.error.bound 1.0E-5)))

(rule-eps-9 (IF (AND (> simulation.intention.number 3)
                     (<= simulation.intention.number 7)
                     (= simulation.intention 2)
                     (method.class is implicit)))
            (THEN (relative.error.bound 1.0E-5)))

(rule-eps-10 (IF (AND (>= simulation.intention.number 3)
                      (<= simulation.intention.number 4)
                      (= simulation.intention 3)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-6)
                   (relative.error.bound 1.0E-6)))
```

```
(rule-eps-11 (IF (AND (>= simulation.intention.number 3)
                      (<= simulation.intention.number 4)
                      (= simulation.intention 3)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-6)))

(rule-eps-12 (IF (AND (> simulation.intention.number 4)
                      (<= simulation.intention.number 7)
                      (= simulation.intention 3)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-7)
                   (relative.error.bound 1.0E-7)))

(rule-eps-13 (IF (AND (> simulation.intention.number 4)
                      (<= simulation.intention.number 7)
                      (= simulation.intention 3)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-7)))

(rule-eps-14 (IF (AND (= simulation.intention.number simulation.intention 4)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-7)
                   (relative.error.bound 1.0E-7)))

(rule-eps-15 (IF (AND (= simulation.intention.number simulation.intention 4)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-7)))

(rule-eps-16 (IF (AND (= simulation.intention.number 5)
                      (= simulation.intention 4)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-8)
                   (relative.error.bound 1.0E-8)))

(rule-eps-17 (IF (AND (= simulation.intention.number 5)
                      (= simulation.intention 4)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-8)))

(rule-eps-18 (IF (AND (> simulation.intention.number 5)
                      (<= simulation.intention.number 7)
                      (= simulation.intention 4)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-9)
                   (relative.error.bound 1.0E-9)))

(rule-eps-19 (IF (AND (> simulation.intention.number 5)
                      (<= simulation.intention.number 7)
                      (= simulation.intention 4)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-9)))
```

```
(rule-eps-20 (IF (AND (= simulation.intention.number simulation.intention 5)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-9)
                   (relative.error.bound 1.0E-9)))

(rule-eps-21 (IF (AND (= simulation.intention.number simulation.intention 5)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-9)))

(rule-eps-22 (IF (AND (= simulation.intention.number 6)
                      (= simulation.intention 5)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-10)
                   (relative.error.bound 1.0E-10)))

(rule-eps-23 (IF (AND (= simulation.intention.number 6)
                      (= simulation.intention 5)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-10)))

(rule-eps-24 (IF (AND (= simulation.intention.number 7)
                      (= simulation.intention 5)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-11)
                   (relative.error.bound 1.0E-11)))

(rule-eps-25 (IF (AND (= simulation.intention.number 7)
                      (= simulation.intention 5)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-11)))

(rule-eps-26 (IF (AND (= simulation.intention.number simulation.intention 6)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-11)
                   (relative.error.bound 1.0E-11)))

(rule-eps-27 (IF (AND (= simulation.intention.number simulation.intention 6)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-11)))

(rule-eps-28 (IF (AND (= simulation.intention.number 7)
                      (= simulation.intention 6)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-12)
                   (relative.error.bound 1.0E-12)))

(rule-eps-29 (IF (AND (= simulation.intention.number 7)
                      (= simulation.intention 6)
                      (method.class is implicit)))
             (THEN (relative.error.bound 1.0E-12)))

(rule-eps-30 (IF (AND (= simulation.intention.number simulation.intention 7)
                      (method.class is explicit)))
             (THEN (absolute.error.bound 1.0E-13)
                   (relative.error.bound 1.0E-13)))
```

```
        (rule-eps-31 (IF (AND (= simulation.intention.number simulation.intention 7)
                              (method.class is implicit)))
                     (THEN (absolute.error.bound 1.0E-13)
                           (relative.error.bound 1.0E-13)))

        (rule-stable-1 (IF (T)) (THEN (CALL-FUNCTION check-stability)))

        (rule-stable-2 (IF (dgl.system is stable))
                       (THEN (CALL-FUNCTION check-stiffness)))

        (rule-stiff-1 (IF (dgl.system is stiff))
                      (THEN (CALL-FUNCTION compute-weights)))

        (rule-stiff-2 (IF (>= stiffness.factor 100))
                      (THEN (dgl.system is stiff)))

        (rule-stiff-3 (IF (< stiffness.factor 100))
                      (THEN (dgl.system is not stiff)))

        (rule-stiff-4 (IF (dgl.system is stiff))
                      (THEN (method.class is implicit)))

        (rule-stiff-5 (IF (dgl.system is not stiff))
                      (THEN (method.class is explicit)))

        (rule-method-1 (IF (method.class is implicit))
                       (THEN (CALL-FUNCTION determine-optimal-implicit-method)))

        (rule-method-2 (IF (method.class is implicit))
                       (THEN (CALL-FUNCTION determine-optimal-explicit-method)))
))
```

C Runge-Kutta-Verfahren

Die nachfolgenden Tabellen stellen die Koeffizientenschemata der Runge-Kutta-Formeln dar, die zur Lösung der Differentialgleichungssysteme Verwendung finden. Die Schemata der expliziten Runge-Kutta-Formeln, die aus [Eng91] entnommen sind, zeigen den exakten Zahlenwert eines jeden Koeffizienten. Die Koeffizienten der impliziten Verfahren lassen sich dagegen im allgemeinen nicht als gebrochen-rationale Zahl darstellen. Aus Gründen einer einheitlichen Präsentation sind sie deshalb sämtlich als Gleitkommazahlen auf 10 Stellen normiert aufgeführt. Berechnet wurden diese Zahlenwerte mit dem Programm `machstuetz.c` aus oben erwähnter Literatur. Die Beschränkung auf 10 signifikante Stellen stellt hier einen Kompromiß der unterschiedlichsten Genauigkeitsanforderungen dar. Je nach Anwendungsbereich und Verfahren kann eine mehr oder weniger genaue Zahlendarstellung Sinn machen.

C.1 Explizite Verfahren

Jedes Koeffizientenschema der expliziten Verfahren stellt eine Runge-Kutta-Einbettungsformel gemäß der Definition 3.8 dar, bestehend aus einer m- und $\widehat{m}$-stufigen, expliziten Runge-Kutta-Formel der globalen Fehlerordnung e_g und $\hat{e}_g$ mit $\widehat{m} \geq m$ sowie $\hat{e}_g > e_g$. Die maximale Stufe des jeweiligen Verfahrens ist am oberen Index der Gewichte γ_i bzw. $\hat{\gamma}_i$ zu erkennen. Die globale Fehlerordnung geht entsprechend ihrer Kennzeichnung aus der Tabellenbeschriftung hervor. Im oberen Teil eines Schemas sind die $\widehat{m} \times \widehat{m}$ und $m \times m$ Koeffizienten β_{ij} in einer einzigen Matrix dargestellt. Die Indizes i, j bezeichnen den Zeilen- und Spaltenindex dieser Matrix. Unterhalb dieses Tabellenteils befinden sich die gemeinsamen Stützstellen α_i und die Gewichte γ_i, $\hat{\gamma}_i$ waagerecht angeordnet. Der laufende Index i entspricht hierbei dem Zeilenindex der Matrix β_{ij}.

Stellvertretend für alle anderen expliziten Verfahren soll anhand des Koeffizientenschemas C.1 der Runge-Kutta-Einbettungsformel $2./\hat{3}$. Ordnung die Konstruktion dieses Formelpaares gezeigt werden. Ausgehend von der allgemeinen Formel

$$\begin{aligned}
\boldsymbol{z}_{s+1} &= \boldsymbol{z}_s + h_s \sum_{i=1}^{m} \gamma_i^{[m]} \boldsymbol{k}_i \\
\hat{\boldsymbol{z}}_{s+1} &= \boldsymbol{z}_s + h_s \sum_{i=1}^{\widehat{m}} \hat{\gamma}_i^{[\widehat{m}]} \boldsymbol{k}_i \\
\text{mit} \quad \boldsymbol{k}_i &= \boldsymbol{f}(t_s + \alpha_i\, h_s, \boldsymbol{z}_s + h_s \sum_{j=1}^{i-1} \beta_{ij}\, \boldsymbol{k}_j)
\end{aligned}$$

ergibt sich das Runge-Kutta-Formelpaar aus dem Koeffizientenschema C.1 durch Einsetzen der abgelesenen Werte zu

$$\begin{aligned}
\boldsymbol{z}_{s+1} &= \boldsymbol{z}_s + h_s\,(0\,\boldsymbol{k}_1 + 1\,\boldsymbol{k}_2) \\
\hat{\boldsymbol{z}}_{s+1} &= \boldsymbol{z}_s + h_s\,(\frac{1}{6}\,\boldsymbol{k}_1 + \frac{2}{3}\,\boldsymbol{k}_2 + \frac{1}{6}\,\boldsymbol{k}_3)
\end{aligned}$$

$$\text{mit} \quad \begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(t_s + 0\,h_s, \boldsymbol{z}_s) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(t_s + \tfrac{1}{2}\,h_s, \boldsymbol{z}_s + \tfrac{1}{2}\,h_s\,\boldsymbol{k}_1) \\ \boldsymbol{k}_3 &= \boldsymbol{f}(t_s + 1\,h_s, \boldsymbol{z}_s - 1\,h_s\,\boldsymbol{k}_1 + 2\,h_s\boldsymbol{k}_2). \end{aligned}$$

Entsprechend diesem Vorgehen lassen sich alle anderen Einbettungsformeln aus den nachfolgenden Koeffizientenschemata aufstellen.

$\begin{smallmatrix} j \\ i \end{smallmatrix} \rightarrow$	1	2	3
$\downarrow i$	β_{ij}		
1	–	–	–
2	$\frac{1}{2}$	–	–
3	-1	2	–
α_i	0	$\frac{1}{2}$	1
$\gamma_i^{[2]}$	0	1	–
$\hat{\gamma}_i^{[3]}$	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

Tabelle C.1: Koeffizientenschema Runge-Kutta-Einbettungsformel 2./$\hat{3}$. Ordnung

$\begin{smallmatrix} j \\ i \end{smallmatrix} \rightarrow$	1	2	3	4	5	6
$\downarrow i$	β_{ij}					
1	–	–	–	–	–	–
2	$\frac{1}{5}$	–	–	–	–	–
3	$\frac{3}{40}$	$\frac{9}{40}$	–	–	–	–
4	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$	–	–	–
5	$\frac{226}{729}$	$-\frac{25}{27}$	$\frac{880}{729}$	$\frac{55}{729}$	–	–
6	$-\frac{181}{270}$	$\frac{5}{2}$	$-\frac{266}{297}$	$-\frac{91}{27}$	$\frac{189}{55}$	–
α_i	0	$\frac{1}{5}$	$\frac{3}{10}$	$\frac{3}{5}$	$\frac{2}{3}$	1
$\gamma_i^{[6]}$	$\frac{31}{540}$	0	$\frac{190}{297}$	$-\frac{145}{108}$	$\frac{351}{220}$	$\frac{1}{20}$
$\hat{\gamma}_i^{[6]}$	$\frac{19}{216}$	0	$\frac{1000}{2075}$	$-\frac{125}{216}$	$\frac{81}{88}$	$\frac{5}{56}$

Tabelle C.2: Koeffizientenschema Prince-Dormand-Einbettungsformel 4./$\hat{5}$. Ordnung

$\begin{smallmatrix} j \\ i \end{smallmatrix} \rightarrow$	1	2	3	4	5	6	7	8	9	10	11	12	13
$\downarrow i$							β_{ij}						
1	—	—	—	—	—	—	—	—	—	—	—	—	—
2	$\frac{1}{18}$	—	—	—	—	—	—	—	—	—	—	—	—
3	$\frac{1}{48}$	$\frac{1}{16}$	—	—	—	—	—	—	—	—	—	—	—
4	$\frac{1}{32}$	0	$\frac{3}{32}$	—	—	—	—	—	—	—	—	—	—
5	$\frac{5}{16}$	0	$-\frac{75}{64}$	$\frac{75}{64}$	—	—	—	—	—	—	—	—	—
6	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$	—	—	—	—	—	—	—	—
7	$\frac{29443841}{614563906}$	0	0	$\frac{77736538}{692538347}$	$-\frac{28693883}{1125000000}$	$\frac{23124283}{1800000000}$	—	—	—	—	—	—	—
8	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{2278713}{633445777}$	$\frac{545815736}{2771057229}$	$-\frac{180193667}{1043307555}$	—	—	—	—	—	—
9	$\frac{39632708}{573591083}$	0	0	$-\frac{433636366}{683701615}$	$-\frac{421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$	—	—	—	—	—
10	$\frac{246121993}{1340847787}$	0	0	$-\frac{37695042795}{15268766246}$	$-\frac{309121744}{1061227803}$	$-\frac{12992083}{490766935}$	$\frac{6005943493}{2108947869}$	$\frac{393006217}{1396673457}$	$\frac{123872331}{1001029789}$	—	—	—	—
11	$-\frac{1028468189}{846180014}$	0	0	$\frac{8478235783}{508512852}$	$\frac{1311729495}{1432422823}$	$-\frac{10304129995}{1701304382}$	$-\frac{48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$-\frac{45442868181}{3398467696}$	$\frac{3065993473}{597172653}$	—	—	—
12	$\frac{185892177}{718116043}$	0	0	$-\frac{3185094517}{667107341}$	$-\frac{477755414}{1098053517}$	$-\frac{703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$-\frac{4093664535}{808688257}$	$\frac{3962137247}{1805957418}$	$\frac{65686358}{487910083}$	—	—
13	$\frac{403863854}{491063109}$	0	0	$-\frac{5068492393}{434740067}$	$-\frac{411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$-\frac{13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$-\frac{160528059}{685178525}$	$\frac{248638103}{1413531060}$	0	—
α_i	0	$\frac{1}{18}$	$\frac{1}{12}$	$\frac{1}{8}$	$\frac{5}{16}$	$\frac{3}{8}$	$\frac{59}{400}$	$\frac{93}{200}$	$\frac{5490023248}{9719169821}$	$\frac{13}{20}$	$\frac{1201146811}{1299019798}$	1	1
$\gamma_i^{[12]}$	$\frac{13451932}{455176623}$	0	0	0	0	$-\frac{808719846}{976000145}$	$\frac{1757004468}{5645159321}$	$\frac{656045339}{265891186}$	$-\frac{3867574721}{1518517206}$	$\frac{465885868}{322736535}$	$\frac{53011238}{667516719}$	$\frac{2}{45}$	—
$\hat{\gamma}_i^{[13]}$	$\frac{14005451}{335480064}$	0	0	0	0	$-\frac{59238493}{1068277825}$	$\frac{181606767}{758867731}$	$\frac{561292985}{797845732}$	$-\frac{1041891430}{1371343529}$	$\frac{760417239}{1151165299}$	$\frac{118820643}{751138087}$	$-\frac{528747749}{2220607170}$	$\frac{1}{4}$

Tabelle C.3: Koeffizientenschema Prince-Dormand-Einbettungsformel 7./$\hat{8}$. Ordnung

$\begin{smallmatrix} j \\ i \end{smallmatrix} \rightarrow$	1	2	3	4	5	6	7	8
$\downarrow i$				β_{ij}				
1	–	–	–	–	–	–	–	–
2	$\frac{1}{18}$	–	–	–	–	–	–	–
3	$-\frac{1}{12}$	$\frac{1}{4}$	–	–	–	–	–	–
4	$-\frac{2}{81}$	$\frac{4}{27}$	$\frac{8}{81}$	–	–	–	–	–
5	$\frac{40}{33}$	$-\frac{4}{11}$	$-\frac{56}{11}$	$\frac{54}{11}$	–	–	–	–
6	$-\frac{369}{73}$	$\frac{72}{73}$	$\frac{5380}{219}$	$-\frac{12285}{584}$	$\frac{2695}{1752}$	–	–	–
7	$-\frac{8716}{891}$	$\frac{656}{297}$	$\frac{39520}{891}$	$-\frac{416}{11}$	$\frac{52}{27}$	0	–	–
8	$\frac{3015}{256}$	$-\frac{9}{4}$	$-\frac{4219}{78}$	$\frac{5985}{128}$	$-\frac{539}{384}$	0	$\frac{693}{3328}$	–
α_i	0	$\frac{1}{18}$	$\frac{1}{6}$	$\frac{2}{9}$	$\frac{2}{3}$	1	$\frac{8}{9}$	1
$\gamma_i^{[6]}$	$\frac{3}{80}$	0	$\frac{4}{25}$	$\frac{243}{1120}$	$\frac{77}{160}$	$\frac{73}{700}$	–	–
$\hat{\gamma}_i^{[8]}$	$\frac{57}{640}$	0	$-\frac{16}{65}$	$\frac{1377}{2240}$	$\frac{212}{320}$	0	$\frac{891}{8320}$	$\frac{2}{35}$

Tabelle C.4: Koeffizientenschema Verner-Einbettungsformel 5./$\hat{6}$. Ordnung

C.2 Implizite Verfahren

Im Unterschied zu den Koeffizientenschemata der expliziten Runge-Kutta-Verfahren, die jeweils eine Einbettungsformel darstellen, enthalten die Schemata der impliziten Verfahren nach der Gauß-Legendre Stützstellenverteilung nur eine m-stufige Runge-Kutta-Formel der globalen Fehlerordnung e_g, die sich in keine weitere Formel einbetten läßt. Ein implizites Formelpaar bildet sich demnach aus den m- und $\widehat{m}$-stufigen Formeln zweier Koeffizientenschemata mit $\widehat{m} > m$. Diese Formeln besitzen die globale Fehlerordnung e_g und $\hat{e}_g$, wobei $\hat{e}_g > e_g$ gilt. Die folgenden Tabellen zeigen die impliziten Formelpaare 4./6., 10./12., 16./18. und 22./24. Ordnung. Der Aufbau dieser Schemata ist im wesentlichen identisch mit denen der expliziten Verfahren. Für die Tabellen ab der 6. Ordnung muß lediglich jeder Zahlenwert der Tabelle mit dem entsprechenden Maßstabsfaktor multipliziert werden.

Wie bei den expliziten Verfahren soll auch an dieser Stelle, beispielhaft für alle impliziten Verfahren, das Aufstellen eines Formelpaares anhand der Formeln 4./6. Ordnung demonstriert werden. Aus dem allgemeinen Runge-Kutta-Ansatz 3.19, hier dargestellt in der Form

$$\boldsymbol{z}_{s+1} = \boldsymbol{z}_s + h_s \sum_{i=1}^{m} \gamma_i^{[m]} \boldsymbol{k}_i$$

$$\text{mit} \quad \boldsymbol{k}_i = \boldsymbol{f}(t_s + \alpha_i\, h_s, \boldsymbol{z}_s + h_s \sum_{j=1}^{m} \beta_{ij}\, \boldsymbol{k}_j),$$

erhält man durch Einsetzen der Koeffizienten des Schemas C.5 die Formel 4. Ordnung

$$\boldsymbol{z}_{s+1} = \boldsymbol{z}_s + h_s\,(\frac{1}{2}\,\boldsymbol{k}_1 + \frac{1}{2}\,\boldsymbol{k}_2)$$

$$\begin{aligned} \text{mit} \quad \boldsymbol{k}_1 &= \boldsymbol{f}(t_s + (\tfrac{1}{2} - \tfrac{\sqrt{3}}{6})\, h_s, \boldsymbol{z}_s + h_s\,(\tfrac{1}{4}\,\boldsymbol{k}_1 + (\tfrac{1}{4} - \tfrac{\sqrt{3}}{6})\,\boldsymbol{k}_2)) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(t_s + (\tfrac{1}{2} + \tfrac{\sqrt{3}}{6})\, h_s, \boldsymbol{z}_s + h_s\,((\tfrac{1}{4} + \tfrac{\sqrt{3}}{6})\,\boldsymbol{k}_1 + \tfrac{1}{4}\,\boldsymbol{k}_2)) \end{aligned}$$

und entsprechend für die Werte der Tabelle C.6 die Formel 6. Ordnung

$$\boldsymbol{z}_{s+1} = \boldsymbol{z}_s + h_s\,(2.777777778\,\boldsymbol{k}_1 + 4.444444444\,\boldsymbol{k}_2 + 2.777777778\,\boldsymbol{k}_3)$$

$$\begin{aligned} \text{mit} \quad \boldsymbol{k}_1 &= \boldsymbol{f}(t_s + 0.1127016654\, h_s, \boldsymbol{z}_s \\ &\qquad + h_s\,(0.1388888889\,\boldsymbol{k}_1 - 0.03597666752\,\boldsymbol{k}_2 + 0.009789444015\,\boldsymbol{k}_3)) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(t_s + 0.500000000\, h_s, \boldsymbol{z}_s \\ &\qquad + h_s\,(0.3002631950\,\boldsymbol{k}_1 + 0.2222222222\,\boldsymbol{k}_2 - 0.2248541720\,\boldsymbol{k}_3)) \\ \boldsymbol{k}_3 &= \boldsymbol{f}(t_s + 0.8872983346\, h_s, \boldsymbol{z}_s \\ &\qquad + h_s\,(0.2679883338\,\boldsymbol{k}_1 + 0.4804211120\,\boldsymbol{k}_2 + 0.1388888889\,\boldsymbol{k}_3)). \end{aligned}$$

$\begin{smallmatrix} j \\ i \end{smallmatrix} \rightarrow$	1	2
$\downarrow i$	β_{ij}	
1	$\frac{1}{4}$	$\frac{1}{4} - \frac{\sqrt{3}}{6}$
2	$\frac{1}{4} + \frac{\sqrt{3}}{6}$	$\frac{1}{4}$
α_i	$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{2} + \frac{\sqrt{3}}{6}$
$\gamma_i^{[5]}$	$\frac{1}{2}$	$\frac{1}{2}$

Tabelle C.5: Koeffizientenschema Gauß-Legendre-Formel 4. Ordnung

$\begin{smallmatrix} j \\ i \end{smallmatrix} \rightarrow$	1	2	3
$\downarrow i$	$\beta_{ij} * 10^{-3}$		
1	138.8888889	−35.97666752	9.789444015
2	300.2631950	222.2222222	−22.48541720
3	267.9883338	480.4211120	138.8888889
α_i $*10^{-1}$	1.127016654	5.000000000	8.872983346
$\gamma_i^{[7]}$ $*10^{-1}$	2.777777778	4.444444444	2.777777778

Tabelle C.6: Koeffizientenschema Gauß-Legendre-Formel 6. Ordnung

$\frac{j}{i} \rightarrow$	1	2	3	4	5
$\downarrow i$	$\beta_{ij} * 10^{-3}$				
1	59.23172126	−19.57036436	11.25440082	−5.593793661	1.588112968
2	128.1510057	119.6571676	−24.59211462	10.31828067	−2.768994399
3	113.7762880	260.0046517	142.2222222	−20.69031643	4.687154524
4	121.2324369	228.9960546	309.0365591	119.6571676	−9.687563142
5	116.8753296	244.9081289	273.1900436	258.8846996	59.23172126
α_i $*10^{-2}$	4.691007703	23.07653449	50.00000000	76.92346551	95.30899230
$\gamma_i^{[11]}$ $*10^{-1}$	1.184634425	2.393143352	2.844444444	2.393143352	1.184634425

Tabelle C.7: Koeffizientenschema Gauß-Legendre-Formel 10. Ordnung

$\frac{j}{i} \rightarrow$	1	2	3	4	5	6
$\downarrow i$	$\beta_{ij} * 10^{-4}$					
1	428.3112309	−147.6372600	93.25050706	−56.68858049	28.54433315	−8.127801713
2	926.7349143	901.9039326	−203.0010229	103.6315624	−48.87192928	13.55561055
3	822.4792261	1960.321623	1169.784836	−204.8252775	79.89991900	−20.75625785
4	877.3787197	1723.907946	2544.394950	1169.784836	−156.5137581	34.14323577
5	843.0668513	1852.679795	2235.938110	2542.570696	901.9039326	−70.11245241
6	864.7502636	1775.263532	2396.258253	2246.319166	1951.445125	428.3112309
α_i $*10^{-2}$	3.376524290	16.93953068	38.06904070	61.93095930	83.06046932	96.62347571
$\gamma_i^{[13]}$ $*10^{-2}$	8.566224619	18.03807865	23.39569673	23.39569673	18.03807865	8.566224619

Tabelle C.8: Koeffizientenschema Gauß-Legendre-Formel 12. Ordnung

$\overset{j}{i} \rightarrow$	1	2	3	4	5	6	7	8
$\downarrow i$	$\beta_{ij} * 10^{-4}$							
1	253.0713407	−91.05943306	62.80831147	−44.83015613	30.78491368	−19.17675255	9.727576641	−2.775083271
2	547.5932177	555.9525861	−136.3979624	81.49708858	−52.15352089	31.39752985	−15.64934911	4.428023043
3	485.8753600	1208.595250	784.2666147	−159.7510336	83.71732720	−46.43465862	22.25714775	−6.188056953
4	518.6552097	1061.934901	1706.711343	906.7094584	−160.2104132	72.41206561	−31.97814310	8.592365843
5	497.5503156	1143.883315	1496.121164	1973.629330	906.7094584	−138.1781134	49.97027078	−12.51252825
6	512.3307384	1089.648025	1614.967888	1729.701590	1973.169951	784.2666147	−96.69007770	20.26732146
7	501.7146584	1127.554521	1537.135700	1865.572438	1731.921828	1704.931192	555.9525861	−41.45053622
8	508.9177647	1102.177596	1587.709982	1782.634003	1858.249073	1505.724918	1202.964605	253.0713407
α_i $*10^{-2}$	1.985507175	10.16667613	23.72337950	40.82826788	59.17173212	76.27662050	89.83332387	98.01449282
γ_i $*10^{-2}$	5.061426815	11.11905172	15.68533229	18.13418917	18.13418917	15.68533229	11.11905172	5.061426815

Tabelle C.9: Koeffizientenschema Gauß-Legendre-Formel 16. Ordnung

$\frac{j}{i} \rightarrow$	1	2	3	4	5	6	7	8	9
$\downarrow i$	$\beta_{ij} * 10^{-4}$								
1	203.1859709	−73.97868566	52.22003592	−38.73451292	28.31369450	−19.62194188	12.26609789	−6.229640916	1.777784627
2	439.6552723	451.6204017	−113.3546401	70.34766802	−47.88276131	32.03360152	−19.64405740	9.871721037	−2.802742376
3	390.0865340	981.8151196	651.5267410	−137.7083399	76.64175625	−47.13586468	27.70828893	−13.61671983	3.825321129
4	416.4508703	862.5547277	1417.952160	780.8676926	−146.1895788	73.00428262	−39.32839915	18.52686201	−5.105734749
5	399.4037287	929.4337227	1242.571104	1700.004453	825.5983875	−138.2690674	60.48237791	−26.19291925	6.968213110
6	411.4776766	884.7139415	1342.381881	1488.731103	1797.386354	780.8676926	−114.8986784	40.68607575	−10.07892847
7	402.5466207	916.8575233	1275.345193	1608.871250	1574.555019	1699.443725	651.5267410	−78.57431612	16.28540784
8	409.1746842	893.3690824	1322.697539	1529.701784	1699.079536	1491.387717	1416.408122	451.6204017	−33.28333046
9	404.5941572	909.4704444	1290.787384	1581.357327	1622.883081	1600.469898	1250.833446	977.2194891	203.1859709
α_i $*10^{-2}$	1.591988025	8.198444634	19.33142836	33.78732883	50.00000000	66.21267117	80.66857164	91.80155537	98.40801198
γ_i $*10^{-2}$	4.063719418	9.032408035	13.03053482	15.61735385	16.51196775	15.61735385	13.03053482	9.032408035	4.063719418

Tabelle C.10: Koeffizientenschema Gauß-Legendre-Formel 18. Ordnung

$\overset{j}{i} \rightarrow$	1	2	3	4	5	6	7	8	9	10	11
$\downarrow i$	$\beta_{ij} * 10^{-5}$										
1	1391.714178	−514.3255630	373.5603749	−290.0379051	227.2742153	−175.2401163	130.2489117	−90.92796577	57.01360623	−28.98753250	8.274889308
2	3011.423919	3139.509237	−810.5626979	526.2821274	−383.7436682	285.3546764	−207.7832917	143.2677813	−89.14667120	45.11849309	−12.84989370
3	2671.798385	6825.428092	4657.255273	−1028.952482	612.6673248	−418.0537537	291.0667954	−195.5739875	119.8216565	−60.09479430	17.03721172
4	2852.666891	5995.603074	10136.50412	5829.844115	−1165.369259	643.8381298	−409.1648653	262.0618906	−156.2371080	77.15309600	−21.70653908
5	2735.286000	6462.389243	8880.218493	12693.75271	6570.113613	−1212.406158	621.9262574	−363.0633384	206.3240376	−99.28862756	27.58997299
6	2819.172106	6147.718906	9599.861090	11109.57743	14307.67523	6823.127169	−1167.448005	550.1107979	−285.3505439	131.2995669	−35.74375038
7	2755.838383	6378.307101	9108.186509	12022.75157	12518.30097	14858.66050	6570.113613	−1034.064480	434.2920530	−183.3707698	48.14235586
8	2805.134895	6201.865377	9470.747655	11397.62634	13549.39209	13002.41621	14305.59648	5829.844115	−821.9935689	283.4153996	−69.23853485
9	2766.391144	6339.113267	9194.688890	11855.26222	12849.16043	14064.30809	12527.55990	12688.64071	4657.255273	−546.4096187	111.6299707
10	2796.278250	6233.899980	9403.657219	11516.42045	13348.01052	13360.89966	13523.97089	11133.40610	10125.07324	3139.509236	−227.9955638
11	2775.153467	6308.006004	9257.496940	11750.61620	13009.97831	13821.49446	12912.95301	11949.72613	8940.950172	6793.344036	1391.714178
α_i $*10^{-2}$	1.088567093	5.646870012	13.49239972	24.04519354	36.52284220	50.00000000	63.47715780	75.95480646	86.50760028	94.35312999	98.91143291
γ_i $*10^{-2}$	2.783428356	6.279018472	9.314510548	11.65968823	13.14022723	13.64625434	13.14022722	11.65968823	9.314510547	6.279018473	2.783428356

Tabelle C.11: Koeffizientenschema Gauß-Legendre-Formel 22. Ordnung

$\frac{j}{i} \rightarrow$	1	2	3	4	5	6	7	8	9	10	11	12
$\downarrow i$	$\beta_{ij} * 10^{-5}$											
1	1179.383410	−437.9902158	321.0569641	−252.8680763	202.3245199	−160.7359021	124.7744752	−93.04998366	65.06247888	−40.82378826	20.76151085	−5.927104871
2	2551.982323	2673.483150	−696.5703663	458.7377223	−341.4878313	261.5801172	−198.8736041	146.4259365	−101.5529619	63.38807297	−32.13413373	9.158756604
3	2264.150825	5812.305775	4001.958214	−896.6328046	544.8804898	−382.8398268	278.1569166	−199.4357102	136.0626251	−84.05538400	42.34600561	−12.03083491
4	2417.482612	5105.504843	8710.385387	5079.185668	−1035.774702	588.8679696	−390.2043706	266.3886991	−176.5933661	107.2055592	−53.45347663	15.10746273
5	2317.888736	5503.367352	7630.356058	11059.62362	5837.313413	−1107.541998	591.6908830	−367.5974985	231.7911626	−136.7295744	67.06291855	−18.80001967
6	2389.201739	5234.690429	8249.893093	9678.177844	12712.57929	6228.676145	−1108.191773	554.5135189	−318.1928873	178.7173279	−85.36087240	23.62571798
7	2335.141101	5432.327172	7825.199099	10476.56422	11120.11331	13565.54406	6228.676145	−1037.952466	480.1934924	−245.9766655	112.2758704	−30.43491937
8	2377.566839	5279.903381	8140.646001	9926.580174	12042.22433	11865.66141	13564.89429	5837.313413	−901.2522802	373.5603689	−156.4010527	40.87808298
9	2343.659357	5400.419776	7896.710868	10334.96470	11408.23813	12847.55666	11868.48432	12710.40153	5079.185667	−706.4689593	241.4614569	−58.71579299
10	2370.797654	5304.620294	8087.971812	10022.30872	11874.06253	12179.19537	12840.19212	11129.74634	11055.00414	4001.958216	−465.3394754	94.61599449
11	2349.608063	5379.100434	7940.528353	10259.92431	11528.20088	12656.22589	12195.77217	12016.11466	9699.633607	8700.486797	2673.483149	−193.2155043
12	2364.693924	5326.204790	8044.740216	10093.30886	11767.67680	12332.57781	12618.08819	11472.30231	10411.23941	7682.859468	5784.956516	1179.383409
α_i $*10^{-3}$	9.219682877	47.94137181	115.0486629	206.3410229	316.0842505	437.3832957	562.6167043	683.9157495	793.6589771	884.9513371	952.0586282	990.7803171
γ_i $*10^{-2}$	2.358766818	5.346966300	8.003916432	10.15837134	11.67462682	12.45735229	12.45735229	11.67462683	10.15837133	8.003916434	5.346966301	2.358766819

Tabelle C.12: Koeffizientenschema Gauß-Legendre-Formel 24. Ordnung

Literatur

[Berg93] M. Berger. Aufbereitung und Strukturierung der Wissensbasis für die Prüfung und Diagnose der Arbeitselemente einer hydraulischen Anlage. Studienarbeit, Universität-Gesamthochschule Duisburg 1993.

[Clem92] D. Clemens. Parallele Simulation mit Transputer-Systemen. Forschungsbericht Nr. 5/92, Universität-Gesamthochschule Duisburg 1992.

[Eng85] G. Engeln-Müllges, F. Reuter. Numerische Mathematik für Ingenieure. BI Mannheim, Wien, Zürich 1985.

[Eng91] G. Engeln-Müllges, F. Reuter. Formelsammlung zur numerischen Mathematik mit Turbo PASCAL-Programmen. BI Mannheim, Wien, Zürich 1991.

[For84] O. Forster. Analysis 2, Differentialrechnung im $\mathbb{R}^n$, Gewöhnliche Differentialgleichungen. Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig 1984.

[Glas67] W. Glasmacher. Beiträge zur numerischen Integration von Anfangswertaufgaben bei gewöhnlichen Differentialgleichungen erster Ordnung. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen 1967.

[Hall76] G. Hall, J. M. Watt. Modern numerical methods for ordinary differential equations. Claredon Press, Oxford 1976.

[Hoff93] M. Hoffmann. Algorithmen zur Verarbeitung von topologischen Informationen in Netzwerken. Diplomarbeit, Universität-Gesamthochschule Duisburg 1993.

[Janz94] F. Janz. Aufbereitung und Strukturierung der Wissensbasis für die Prüfung und Diagnose der Versorgungselemente einer hydraulischen Anlage. Studienarbeit, Universität-Gesamthochschule Duisburg 1994.

[KlBü93] H. Kleine Büning, B. Stein. Ein wissensbasiertes System zur Inbetriebnahmeunterstützung. Arbeitsbericht, Universität-Gesamthochschule Paderborn 1993.

[Koe90] N. Köckler. Numerische Algorithmen in Softwaresystemen. B. G. Teubner Stuttgart 1990.

[Lau90] H. Lausch. Digitale Regelung hydraulischer Antriebe mittels pulsbreitenmoduliert angesteuerter Proportionalventile. Forschungsberichte, VDI Reihe 8 Nr. 213, VDI-Verlag Düsseldorf 1990.

[Lem91] R. Lemmen. Akquisition und Analyse von Wissen zur Inbetriebnahme von hydraulischen, translatorischen Anlagen. Diplomarbeit, Universität-Gesamthochschule Duisburg 1991.

[Luth87] W. Luther, K. Niederdrenk, F. Reutter, H. Yserentant. Gewöhnliche Differentialgleichungen, Reihe: Rechnerorientierte Ingenieurmathematik. Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig-Wiesbaden 1987.

[Min94] D. Minninger. Aufbereitung und Strukturierung der Wissensbasis für die Prüfung und Diagnose der Steuerelemente einer hydraulischen Anlage. Studienarbeit, Universität-Gesamthochschule Duisburg 1994.

[Schm80] G. Schmidt. Simulationstechnik, (Methoden der Regelungstechnik). R. Oldenbourg Verlag München, Wien 1980.

[Schw93] H. R. Schwarz. Numerische Mathematik. B. G. Teubner Stuttgart 1993.

[Som67] D. Sommer. Neue implizite RUNGE-KUTTA-Formeln und deren Anwendungsmöglichkeiten. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen 1967.

[Stee84] G. L. Steele Jr.. Common Lisp, The Language. Digital Press 1984.

[Stum71] F. Stummel, K. Hainer. Praktische Mathematik. B. G. Teubner Stuttgart 1971.

[Win87] P. H. Winston, B. K. Horn. LISP. Addison-Wesley 1987.