TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

DIPLOMARBEIT

Quantized State Systems in der Systemsimulation - Algorithmenanalyse und Implementierung in MATLAB/Simulink

Ausgeführt am Institut für
Analysis und Scientific Computing
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker

durch
Patrick Grabher, BSc.
Wien

Wien, am 7. Mai 2014

Danksagung

An dieser Stelle möchte ich mich bei Herrn Prof. Felix Breitenecker für die Betreuung dieser Diplomarbeit bedanken.
Vielen Dank Dipl.-Ing. Bernhard Heinzl und Dipl.-Ing. Matthias Rössler, die mich umfangreich bei dieser Arbeit unterstützt haben.

Kurzfassung

In der heutigen Zeit ist MATLAB/Simulink wohl eine der meist benutzten Simulationsumgebungen. Sowohl im Umfang der Bibliothek als auch in der Vielseitigkeit der numerischen Verfahren bietet Simulink sehr viele Möglichkeiten, Modelle einfach und effizient zu simulieren.
Ein wichtiger Bestandteil von Simulink sind numerische Differentialgleichungslöser. Sie haben die Aufgabe die Lösung von kontinuierlichen Systemen approximativ zu berechnen, da zumeist keine Möglichkeit besteht diese auf analytischem Weg zu ermittelt. Diese Verfahren müssen aufgrund begrenzter Ressourcen (Computersystem) entweder die Zeit oder den Zustand des kontinuierlichen Systems diskretisieren. Tatsache ist, dass jeder in Simulink integrierter Differentialgleichungslöser die Zeit diskretisiert. In dieser Arbeit werden wir sehen, dass es eine Vielzahl von Systemen gibt, bei denen diese Löser Schwächen aufweisen. Ein Beispiel dafür sind Modelle, in denen die jeweiligen Parameter einen Zeitpunkt definieren, in welchem es zu einer unstetigen Änderung des Zustandes kommt. Dieser Zeitpunkt wird künftig auch als Diskontinuität bezeichnet. Modelle mit dieser Eigenschaft werden in dieser Arbeit öfter herangezogen.
Damit Simulink auch mit dieser Problemstellung in Zukunft besser umgehen kann, wurde im Laufe dieser Arbeit ein Differentialgleichungslöser nach dem *Quantized State System* (kurz: QSS) Formalismus implementiert. Dieser unterscheidet sich ganz wesentlich von den in Simulink zur Verfügung stehenden Lösern, da beim QSS-Verfahren nicht die Zeit, sondern der Zustand diskretisiert wird. Anders als bei zeitorientierten Verfahren werden hier Diskontinuitäten immer auf direktem Weg erkannt.
Diese Arbeit beschäftigt sich im Wesentlichen mit der Umsetzung und Anwendung des QSS-Verfahrens. Die ausgewählten Beispiele in dieser Arbeit sollen einen Eindruck schaffen, wie sehr sich das Verhalten des QSS-Verfahrens von dem Verhalten zeitdiskreter Verfahren unterscheidet. Außerdem beinhaltet diese Arbeit einen theoretischen Abschnitt, der den Fehler und die Stabilität des QSS-Verfahrens untersucht.

Abstract

Nowadays MATLAB/Simulink is probably one of the most used simulation software. Simulink provides a large library of blocks and many capabilities of numerical tools for a simple and efficient simulation.
An important part of Simulink are the numerical differential solvers. Their task is to solve continuous systems in an approximatelly way, because in most cases there exists no analytical methode to get the solution of this systems. Numerical solvers are always faced with finite ressources (like a computersystem). This is the reason, that they have to discretise the time or the state. The fact is, that all in Simulink integrated solvers discretise the time. But not always this method leads to good results. Later we will see, that there exists many continuous systems, where these solvers get in trouble. One example are models, which have discontinuities.
To deal with this problem this master thesis presents a solver based on the *Quantized State System* (QSS) which solves such systems by discretising the state rather then the time. The big advantage of this approach is that the solver handles discontinuities in a direct way.
This master thesis contains the implementation of this solver and its usage with some selected models. This models show the difference in the behaviour between the QSS method and the conventional time-discrete methods. Beneath that this master thesis encloses a theoretical part where the error and the stability of the QSS method are discussed.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Ausgangslage

Für das Lösen von gewöhnlichen Differentialgleichungssystemen werden meist konventionelle Verfahren eingesetzt. Gemeint sind numerische Verfahren, welche die Zeitachse diskretisieren, während die Zustandsvariable beliebige Werte annehmen kann. Da die Lösung dieser Systeme von der Abtastung der Zeitachse abhängt, können sie auch als zeitorientierte Verfahren bezeichnet werden. Beispiele sind die Runge-Kutta-Verfahren, welche wohl die bekanntesten Einschrittverfahren sind.
Die Bandbreite von zeitorientierten Verfahren ist heutzutage groß und sehr vielseitig, sodass es für zahlreiche Probleme ein geeignetes gibt. Trotz allem gibt es noch immer viele Modelle einer speziellen Klasse, die mit den bewährten Verfahren nur mit hohem Aufwand simuliert werden können. Die Ursache dafür finden wir in der grundsätzlichen Vorgehensweise, nämlich in der Orientierung nach der Zeit.
Denken wir an ein Modell, das aufgrund bestimmter Werte der Parametern eine unstetige Änderung im Zustand hat, so sind zumeist Probleme an diesem Zeitpunkt (Diskontinuität) vorprogrammiert. Die Orientierung nach der Zeit bringt mit sich, dass Diskontinuitäten übersehen werden und deshalb große Fehler auftreten. Dies führt dazu, dass Integrationsschritte über Diskontinuitäten speziell behandelt werden müssen. Der zusätzlich betriebene Aufwand, diese Zeitpunkte zu lokalisieren kann sich sehr rasch negativ auf die Performance auswirken.
Ein weiterer Kritikpunkt vieler Modelle stellt die synchrone Auswertung der Zustände nach jedem Zeitschritt dar. Das im Kapitel 2.1 vorgestellte Verfahren wird uns zeigen, dass Auswertungen auch asynchron erfolgen können. Man könnte sich dies auch so vorstellen, dass jeder Integrator mit seiner eigenen Schrittlänge arbeitet. Gerade in größeren Modellen kann ein selbständig arbeitender Integrator wesentliche Vorteile in der Performance schaffen.

Im nächsten Abschnitt werden wir die Idee eines nicht zeitorientierten Verfahrens kennenlernen.

1.2 Motivation

Betrachten wir zunächst eine gewöhnlichen DGL. Dafür sei $G \subset \mathbb{R}^{n+1}$ ein Gebiet, $(t_0, \boldsymbol{x_0}) \in G$, die Funktion $f \in C(G, \mathbb{R}^n)$ und $\boldsymbol{x_0} \in \mathbb{R}^n$. Sei nun $\boldsymbol{x}(t)$ die Lösung des Anfangswertproblems

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f(t, \boldsymbol{x}(t)), \\ \boldsymbol{x}(t_0) &= \boldsymbol{x_0}. \end{aligned}$$

Dieses kontinuierliche System wollen wir jetzt numerisch lösen und zwar nicht mit Hilfe eines zeitorientierten Verfahrens. Wir gehen deshalb von einem System aus, in dem die Zeitachse kontinuierlich belassen wird, dafür aber die Zustände diskretisiert werden. Mit diesem Tausch bekommt das System eine neue Orientierung. Formuliert man dieses Problem in eine Frage, so wollen wir zukünftig nicht mehr wissen:

- „Welchen Zustand nimmt das System nach einem bestimmten Zeitschritt an?“

sondern:

- „Wieviel Zeit muss vergehen, damit das System einen bestimmten Zustand annimmt?“

Ein Zustandswechsel ist somit ein Ereignis für unser zukünftiges System, weshalb wir es auch als ein ereignisorientiertes System bezeichnen werden. Auswertungen werden somit nur bei Ereignissen durchgeführt.

Zur Veranschaulichung dient uns folgendes Beispiel.

Beispiel 1.2.1. Gegeben sei eine gewöhnliche Differentialgleichung:

$$\begin{aligned} \dot{x}(t) &= -x(t) + 3 \\ x(0) &= 0 \end{aligned}$$

Wir wollen dieses System in seinem Zustand diskretisieren und machen dies mit $q(t) = \lfloor x(t) \rfloor$ ($x(t)$ wird ganzzahlig abgerundet). Dadurch erhalten wir ein neues, ein quantisiertes System:

$$\begin{aligned} \dot{x}(t) &= -q(t) + 3 \\ x(0) &= 0 \end{aligned} \tag{1.1}$$

Die Größe $x(t)$ stellt nun die Lösung des Systems (1.1) dar. Dieses quantisierte System kann mit dem in Abschnitt 2.2 vorgestelltem DEVS-Formalismus beschrieben werden und anschließend mit diskreten Simulationsverfahren exakt gelöst werden. In diesem Fall ist das System (1.1) aber nicht allzu schwer zu lösen, weshalb wir es auch manuell nachrechnen können:

$$\begin{aligned}
&\Rightarrow q(0) = 0 \\
\dot{x}(0) = 3 &\Rightarrow \text{der nächste Zustand wird bei } t = 1/3 \text{ erreicht: } q(1/3) = 1 \\
\dot{x}(1/3) = -1 + 3 = 2 &\Rightarrow q(1/2 + 1/3) = q(5/6) = 2 \\
\dot{x}(5/6) = -2 + 3 = 1 &\Rightarrow q(1 + 5/6) = 3 \\
\dot{x}(11/6) = -3 + 3 = 0 &\Rightarrow q(\Delta t + 11/6) = 3, \qquad \text{für jedes } \Delta t \geq 0
\end{aligned}$$

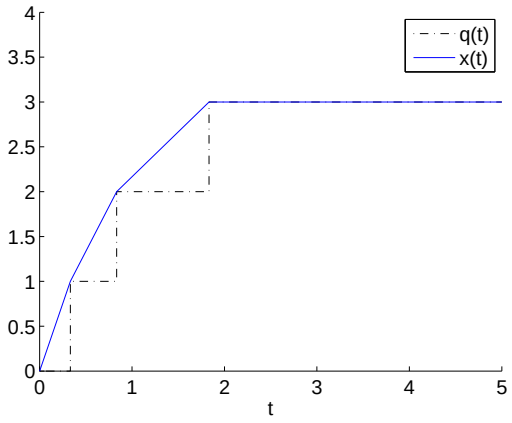

Abbildung 1.1: Lösung x(t) des quantisierten Systems (1.1)

Diese einfache Methode aus dem obigen Beispiel, nur den Zustand $x(t)$ abzurunden, funktioniert nicht in jedem System. In vielen Fällen kann dies zu unendlich vielen Auswertungen in einem beschränkten Zeitintervall führen, sodass diese Systeme für diskrete Simulationsverfahren nicht legitim sind. Folgendes Beispiel demonstriert dieses Problem.

Beispiel 1.2.2. Das System (1.1) des vorangegangenen Beispiels wird hier geringfügig geändert:

$$\begin{aligned}
\dot{x}(t) &= -q(t) + 5/2 \\
x(0) &= 0
\end{aligned} \tag{1.2}$$

Die Lösung dieses Systems ist

$$\begin{aligned}
&\Rightarrow q(0) = 0 \\
\dot{x}(0) = 5/2 &\Rightarrow q(2/5) = 1 \\
\dot{x}(2/5) = -1 + 5/2 = 3/2 &\Rightarrow q(2/3 + 2/5) = q(1 + 1/15) = 2 \\
\dot{x}(1 + 1/15) = -2 + 5/2 = 1/2 &\Rightarrow q(3 + 1/15) = 3
\end{aligned}$$

Das heißt, des System (1.1) nimmt im Zeitpunkt $\hat{t} = 3 + 1/15$ den Zustand 3 an. Es gilt $q(\hat{t}) = x(\hat{t}) = 3$. Werten wir nun die Funktion $\dot{x}(t)$ an $\hat{t} = 3 + 1/15$ aus, so kommt es zu einem Vorzeichenwechsel bei $\dot{x}(t)$. Da $x(\hat{t}) = 3$, bedeutet das für die Abrundungsfunktion $q(t)$, dass sie im gleichem Augenblick den Wert von 3 auf 2 revidieren muss, hingegen der Wert von $x(\hat{t})$ unverändert bleibt. Dies ist nun der Auslöser für eine endlose Oszillation zwischen den Werten $q(t) = 2$ und $q(t) = 3$ im Zeitpunkt $\hat{t}$. Dieses System hat sich leider festgelaufen.

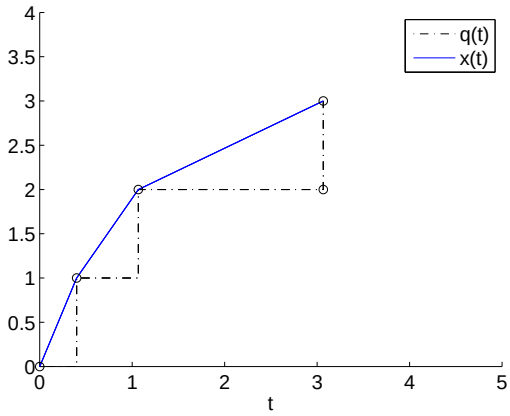

Abbildung 1.2: Lösung des quantisierten Systems (1.2)

1.3 Aufgabenstellung

Die Lösung des Problems aus dem vorangegangenen Beispiel liegt im *Quantized State System* (kurz: QSS), welches wir im Kapitel 2.1 kennenlernen. Mithilfe dieser Übersetzung können wir ein kontinuierliches System in ein quantisiertes umwandeln und anschließend mit einem diskreten Simulationsverfahren lösen.
Ein Teilbereich dieser Arbeit befasst sich mit der Implementierung des quantisierten Integrators, basierend auf dem QSS-Formalismus, in Simulink. Diese Implementierung, die im Wesentlichen aus den Blöcken der SimEvents Bibliothek besteht, ist für kontinuierliche Systeme einsetzbar. Im Kapitel 4 wird die Funktionsweise des Integrators beschrieben.
Dieser QSS-Integrator wird anhand von Modellen getestet (siehe Kapitel 5). Anschließend werden die daraus resultierenden Lösungen und die dafür benötigten Ressourcen des QSS-Verfahrens mit denen der zeitdiskreten Lösern gegenübergestellt.
Weiters wird im Kapitel 3 der Fehleranalyse und Stabilität des QSS-Formalismus ein theoretischer Abschnitt gewidmet. Die wichtigsten Aspekte, die wir dort diskutieren, sind Genauigkeit und Effizienz dieses Verfahrens.

Kapitel 2

Zustandsdiskrete Simulation

2.1 Quantized State System (QSS)

Die Definitionen und Sätze in diesem Textabschnitt sind aus der Arbeit von Ernesto Kofman und Sergio Junco [1] entnommen. Wir lernen hier ein Methode kennen, die ein kontinuierliches in ein ereignisorientiertes System zu wandeln ermöglicht und gleichzeitig garantiert keine unendlich schnellen Oszillationen im Zustand zu haben. Der Schlüssel dafür besteht in einer neue Quantisierungsfunktion, welche gewisse Zustandswechsel verzögert.

Definition 2.1.1 (hysteretische Quantisierungsfunktion)**.** Sei $Q = \{Q_0, Q_1, Q_2, ..., Q_r\}$, wobei $Q_0 \leq Q_1 \leq \cdots \leq Q_r$ und $Q_i \in \mathbb{R}$ für $0 \leq i \leq r$. Sei Ω eine Menge von stückweise linearen Trajektorien, $x \in \Omega$ und $b : \Omega \to \Omega$ eine Abbildung: $x \mapsto b(x) = q$. Erfüllt die Trajetorie q nun die Bedingung

$$q(t) = \begin{cases} Q_m & \text{falls } t = t_0 \\ Q_{i+1} & \text{falls } x(t) = Q_{i+1} \ \wedge \ q(t^-) = Q_i \ \wedge \ i < r \\ Q_{i-1} & \text{falls } x(t) = Q_i - \epsilon \ \wedge \ q(t^-) = Q_i \ \wedge \ i > 0 \\ q(t^-) & \text{sonst} \end{cases} \tag{2.1a}$$

und

$$m = \begin{cases} 0 & \text{falls } x(t_0) < Q_0 \\ r & \text{falls } x(t_0) \geq Q_r \\ j & \text{falls } Q_j \leq x(t_0) < Q_{j+1} \end{cases}, \tag{2.1b}$$

so ist b die *hysteretische Quantisierungsfunktion.*

Q_0 und Q_r geben dabei den minimal bzw. maximal möglichen Zustand und ϵ die Hysteresebreite an. Würden wir von einer einheitlichen Größe des Quantums ausgehen, das heißt $\Delta Q = Q_i - Q_{i-1}$ für alle $1 \leq i \leq r$, so hätte die hysteretische Quantisierungsfunktion die Gestalt, wie sie in Abb. 2.1 dargestellt ist.

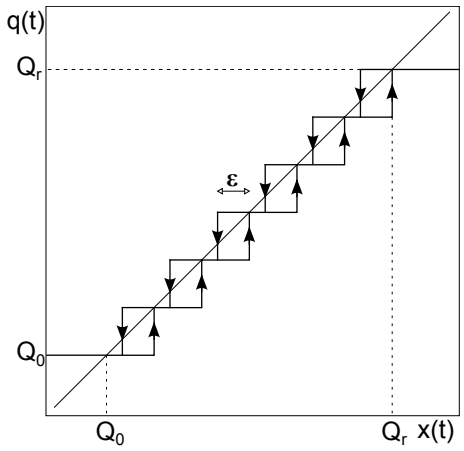

Abbildung 2.1: hysteretische Quantisierungsfunktion

Definition 2.1.2 (Quantized State System)**.** Betrachten wir ein kontinuierliches System in der Zustandsraumdarstellung

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$$

und transformieren dieses mit Hilfe der oben definierten hysteretischen Quantisierungsfunktion in

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{q}(t), \boldsymbol{u}(t)), \tag{2.2}$$

so ist dieses das *Quantized State System*. In diesem System kann jede Komponente ihre eigene Zustandsmenge verwenden.

Abb. 2.2 zeigt das Blockdiagramm des Systems (2.2).

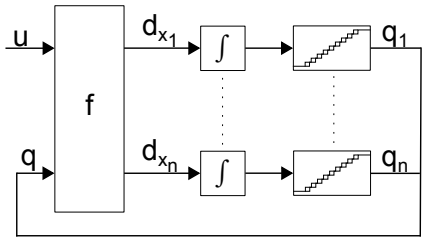

Abbildung 2.2: Blockdiagramm des Quantized State Systems

In diesem Diagramm sehen wir die *stationäre Funktion* $f(\boldsymbol{q}(t), \boldsymbol{u}(t))$ und die quantisierten Integratoren. Die Ableitung von $\boldsymbol{x}(t)$ wird hier als $\boldsymbol{d_x}$ bezeichnet. Später werden wir sehen, dass der Zustand des quantisierten Integrators nur bei Ein- und Ausgangsereignissen aktualisiert wird.
In allen zukünftigen Beispielen werden wir mit der Variablen $q(t)$ (oder $q_x(t)$) immer den quantisierten Zustand von $x(t)$ beschreiben.

Grundsätzliche Eigenschaften des QSS-Verfahrens

Wir möchten hier genauer diskutieren ob das System (2.2) alle Voraussetzungen mitbringt, immer ein legitimes Modell für diskrete Simulationsverfahren zu beschreiben. Dabei spielt der nächste Satz eine entscheidende Rolle.

Satz 2.1.3. *Sei ein QSS-System gegeben wie (2.2), f stetig und beschränkt in einem beschränkten Gebiet und die Eingangsvariable $\boldsymbol{u}(t)$ beschränkt und stückweise konstant, dann sind auch die Trajektorien von $\boldsymbol{q}(t)$ stückweise konstant.*

Beweis. Sei $\boldsymbol{q}(t) \in \mathbb{R}^n$ und $q_j(t)$ der j-te Eintrag von $\boldsymbol{q}(t)$. Aus der Defintion von $b(x)$ (2.1) folgt einerseits

$$Q_{0_j} \leq q_j(t) \leq Q_{r_j}, \qquad \forall j$$

und andererseits $\|\boldsymbol{q}(t)\| < \infty$ (Q_{i_j} steht für den j-ten Eintrag des i-ten Zustandes).
Aus der Beschränktheit von f folgt, dass ein $M \in \mathbb{R}^+$ existiert mit

$$-M \leq \dot{x}_j(t) \leq M. \tag{2.3a}$$

Nach der Integration beider Seiten

$$\begin{aligned} \int_{t_0}^{t} -M dt \leq & \int_{t_0}^{t} \dot{x}_j(t) dt \leq \int_{t_0}^{t} M dt \\ -M(t-t_0) \leq & x_j(t) - x_j(t_0) \leq M(t-t_0) \\ x_j(t_0) - M(t-t_0) \leq & \quad x_j(t) \quad \leq x_j(t_0) + M(t-t_0) \end{aligned} \tag{2.3b}$$

sehen wir, dass die Trajektorien von $x_j(t)$ stetig und beschränkt bleiben. Gehen wir davon aus, dass $x_j(t) = q_j(t) = Q_{i_j}$, dann folgt aus (2.1) und (2.3a), dass

$$q_j(t) = q_j(t + \Delta t)$$

für jedes

$$\Delta t \leq \Delta t_{min} = \frac{\min\limits_{1 \leq i \leq r} (Q_{i-1_j} - Q_{i_j}, \epsilon)}{M}.$$

Mit Δt_{min} haben wir also ein Zeitintervall gefunden, welches mindestens durchlaufen werden muss, damit $q_j(t)$ einen zweiten Wert annehmen kann. □

Jetzt wissen wir:

- $\boldsymbol{q}(t)$ hat stückweise konstante Ausgangstrajektorien.

Daraus folgen direkt die Eigenschaften der anderen Größen:

- $\boldsymbol{d_x}(t)$ hat stückweise konstante Trajektorien.
- $\boldsymbol{x}(t)$ hat stetige und stückweise lineare Trajektorien.

$\boldsymbol{q}(t)$ und $\boldsymbol{d_x}(t)$ zeigen uns, dass das QSS-Verfahren stückweise konstante Eingangs- und Ausgangsgrößen hat. Es lässt sich also mit einem diskreten Simulationsverfahren simulieren.

2.2 Discrete Event System Specification (DEVS)

Der DEVS-Formalismus wurde von Bernard Zeigler [2] im Jahr 1976 vorgestellt. In dieser Arbeit werden wir nur die einfachste Variante, das Atomare DEVS-Modell kennenlernen.

Definition 2.2.1 (atomares DEVS-Modell)**.** Ein atomares DEVS-Modell ist ein Tupel folgender Form:

$$M = \langle X, Y, S, ta, \delta_{int}, \delta_{ext}, \lambda \rangle$$

Dabei gilt:

- X ist die Menge der Eingangsereignisse.
- Y ist die Menge der Ausgangsereignisse.
- S ist die Menge der (aufeinanderfolgenden) Zustände.
- $ta : S \to \mathbb{R}_+ \cup \{+\infty\}$ ist jene Funktion, welche die Laufzeit des Zustandes vorgibt.
- $\delta_{int} : S \to S$ ist die interne Transitionsfunktion (oder interne Zustandsübergangsfunktion), welche bei einem Ausgangsereignis ausgeführt wird. Sie gibt jenen Zustand vor, den das Modell nach Ablauf von $ta(s)$ einnimmt.
- $\delta_{ext} : S \times \mathbb{R}_0^+ \times X \to S$ ist die externe Transitionsfunktion (oder externe Zustandsübergangsfunktion). Sie wird für jedes Eingangsereignis benötigt. $\delta_{ext}(s, t_e, x)$ ist neben dem neuen Eingangssignal x, auch vom derzeitigen Zustand s und von der Zeit t_e, die seit dem letzten Ereignis vergangen ist, abhängig.
- $\lambda : S \to Y$ ist die Ausgangsfunktion. Ist die Laufzeit des Zustandes erreicht, so setzt diese Funktion das entsprechende Ausgangsereignis.

Um sich ein besseres Bild der Funktionsweise dieses Formalismus machen zu können, sehen wir uns nun ein Beispiel an.

Beispiel 2.2.2. Gegeben sei folgendes DEVS-Modell:

$$\begin{aligned} X = Y = S &= \mathbb{N}_0 \\ ta(s) &= \begin{cases} \infty & \text{falls } s = 0 \\ \frac{2}{s} & \text{sonst} \end{cases} \\ \delta_{int}(s) &= s - 1 \\ \delta_{ext}(s, t_e, x) &= x + 2 \\ \lambda(s) &= 2 \cdot s \end{aligned}$$

Wir nehmen an, dass es am Eingang X zu 2 Ereignissen

$$x(0) = 0 \quad \text{und} \quad x(2) = 1$$

kommt. Da bereits zu Beginn $t = 0$ ein Eingangsereignis vorliegt und δ_{ext} unabhängig vom Zustand ist, ist die Angabe eines Anfangswertes $s_0 = s(0)$ überflüssig. Abb. 2.3 zeigt das Verhalten dieses Modells.

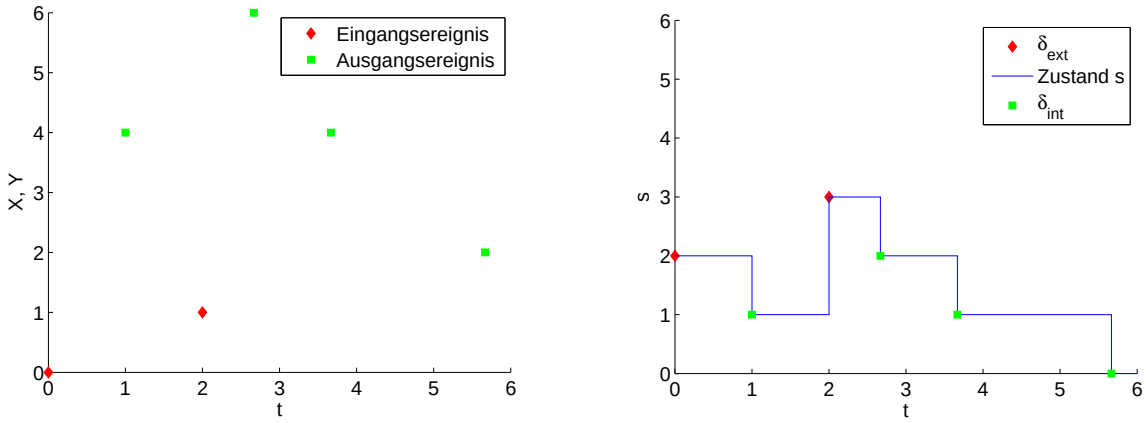

Abbildung 2.3: Verhalten des DEVS-Modells

Dieses Modell ist so konstruiert, dass es beim letzten Eingangsereignis $x = n$ nach $n + 2$ internen Zustandswechseln die Ruhelage $s = 0$ einnimmt, da $ta(0) = \infty$. Für den Ausgang bedeutet dies, dass der Wert $y = 2$ das kleinst mögliche Ereignis ist.

Jetzt kennen wir die Funktionsweise des DEVS-Formalismus und können ihn nun auf unser ursprüngliches Problem anwenden.

2.3 Umsetzung des QSS-Verfahrens in den DEVS-Formalismus

Prinzipiell gibt es einige Möglichkeiten, ein QSS-System als ein DEVS-Modell zu beschreiben. Aufgrund der asynchronen Arbeitsweise ist es oft effizienter, für jedes Subsystem einen eigenen quantisierten Integrator zu verwenden und diesen mithilfe der Ein- und Ausgangsgrößen zu einem gekoppelten DEVS-Modell zu verknüpfen. Wir können diesen Integrator des QSS-Verfahrens wie in Abb. 2.2 als Blockdiagramm dargestellt, als ein atomares DEVS-Modell beschreiben.

Wir betrachten also nur eine Komponente des QSS-Systems. Jede Änderung von $q(t)$ definiert somit ein Ausgangsereignis im DEVS-Modell und jede Änderung von $d_x(t)$ ein Eingangsereignis. Der stückweise lineare Zustand $x(t)$ stellt im DEVS-Modell den Zustand s_j dar, welcher nur im Zeitpunkt eines Ereignisses aktualisiert werden muss. Ernesto Kofman und Sergio Junco [1] definieren das atomare DEVS-Modell des QSS-Systems wie folgt:

$$M_j = \langle X, Y, S, ta, \delta_{int}, \delta_{ext}, \lambda \rangle \tag{2.4a}$$

wobei:

$$\begin{aligned} X = Y &= \mathbb{R} \\ S &= \mathbb{R}^2 \times \mathbb{Z} \times \mathbb{R}_+ \cup \{+\infty\}, \quad \text{mit } s \in S \text{ und } s = (x, d_x, i, \sigma) \\ ta(s) &= \sigma \\ \delta_{int}(s) &= (x + \sigma \cdot d_x, d_x, i + sgn(d_x), \sigma_1) \\ \delta_{ext}(s, t_e, \hat{d}_x) &= (x + t_e \cdot d_x, \hat{d}_x, i, \sigma_2) \\ \lambda(s) &= Q_{i+sgn(d_x)} \end{aligned} \tag{2.4b}$$

Die interne Transitionsfunktion δ_{int} wird also bei jedem neuen Ausgangsereignis (= Zustandsänderung) ausgeführt. Die Variable $\hat{d}_x$, so wie d_x, beschreiben die Ableitung des Zustandes am Integratoreingang. Kommt es zu einem Eingangsereignis, so wird die externe Transitionsfunktion δ_{ext} ausgeführt, für welche $\hat{d}_x$ den aktuellen und d_x den vorigen Wert der Ableitung angeben. Die benötigten Zeiten σ_1 und σ_2 für einen Zustandswechsel von $q(t)$ werden dabei wie folgt berechnet:

$$\sigma_1 = \begin{cases} \frac{Q_{k+2} - (x + \sigma \cdot d_x)}{d_x} & \text{falls } d_x > 0 \\ \frac{(x + \sigma \cdot d_x) - (Q_{k-1} - \epsilon)}{|d_x|} & \text{falls } d_x < 0 \\ \infty & \text{falls } d_x = 0 \end{cases} \tag{2.4c}$$

$$\sigma_2 = \begin{cases} \frac{Q_{k+1} - (x + t_e \cdot d_x)}{\hat{d}_x} & \text{falls } \hat{d}_x > 0 \\ \frac{(x + t_e \cdot d_x) - (Q_k - \epsilon)}{|\hat{d}_x|} & \text{falls } \hat{d}_x < 0 \\ \infty & \text{falls } \hat{d}_x = 0 \end{cases} \tag{2.4d}$$

Kapitel 3

Eigenschaften des QSS-Verfahrens

Wie man ein kontinuierliches in ein ereignisorientiertes System umformen kann, zeigt Kapitel 2. Am Beginn einer Simulation besteht die Möglichkeit zu wählen, ob die Zeit oder der Zustand diskretisiert wird. Das QSS-System geht zwar sehr effizient mit Ereignissen um, kann jedoch bei glatten Lösungen zumeist nicht mit einem zeitorientierten Verfahren mithalten. Man denke z.B. an die Verfahren höherer Ordnung mit adaptiver Schrittweitensteuerung. Möchte man auf die Vorteile der zeitorientierten Verfahren nicht verzichten und trotzdem an den Ereignissen interessiert sein, so könnte folgende Methode von Interesse sein.

Henon-Methode

Die Methode von Michel Hénon [3] ermöglicht mit einem beliebigen zeitorientierten Verfahren die Lösung einer DGL zu finden. Wurde bei einem Integrationsschritt ein Ereignis (z.B. eine Nullstelle) verpasst, so wie es üblicherweise bei einem zeitorientierten Verfahren vorkommt, so wird hier das Ereignis nicht durch iteratives Annähern lokalisiert. Die Henon-Methode wandelt stattdessen das zeitorientierte System in ein ereignisorientiertes um. Die Differenz des Zustandes zum verpassten Ereignis stellt somit einen Integrationsschritt dar, in dem der Konsistenzfehler gleicher Ordnung wie der im zeitorientierten Fall ist. Anschließend wechselt man zurück in das ursprüngliche System und orientiert sich wieder nach der Zeit. Der grundsätzliche Vorgang dieser Methode wird anhand folgenden Beispiels gezeigt:

Gegeben sei ein autonomes DGL-System:

$$\begin{aligned} \frac{dx_1}{dt} &= f_1(x_1, \ldots, x_N) \\ &\vdots \\ \frac{dx_N}{dt} &= f_N(x_1, \ldots, x_N) \end{aligned} \tag{3.1a}$$

Für dieses Bsp. betrachten wir zunächst die Ereignisfunktion $S(\boldsymbol{x}) = x_N - a$, wobei a eine Konstante ist. Die Frage lautet nun, in welchem Zeitpunkt t ist $S(\boldsymbol{x}) = 0$. Dazu wird die DGL (3.1a) so lange integriert, bis es zu einem Vorzeichenwechsel bei $S(\boldsymbol{x})$ kommt, diesen Wert bezeichnen wir im Folgenden mit S. Da $\boldsymbol{x}$ eine von t abhängige Größe im System (3.1a) beschreibt, muss dieses umgewandelt werden. Dazu dividieren wir die $(N-1)$-ten Gleichungen durch die N-te Gleichung und invertieren die N-te Gleichung dieses Systems:

$$\begin{aligned} \frac{dx_1}{dx_N} &= \frac{f_1}{f_N} \\ &\vdots \\ \frac{dx_{N-1}}{dx_N} &= \frac{f_{N-1}}{f_N} \\ \frac{dt}{dx_N} &= \frac{1}{f_N} \end{aligned} \tag{3.1b}$$

Dieses System wird nun entweder an dem Zeitpunkt vor bzw. nach dem Erkennen des Ereignisses gelöst. Je nach Festlegung müssen die entsprechenden Anfangswerte (für diesen Zeitpunkt berechneten Zustände) gewählt werden. Der Wert der Ereignisfunktion an dem entsprechenden Zeitpunkt dient als Schrittweite für das neue System, so ist z.B. $\Delta x_N = -S$ jene Schrittweite, falls wir vom zuletzt berechneten Wert ausgehen. Nach diesem Integrationsschritt wechselt man wieder in das ursprüngliche System (3.1a).

Mit der Henon-Methode ist es also möglich, die Orientierung bei auftretenden Ereignissen kurzzeitig zu ändern. Man sollte damit rechnen, dass diese Methode nicht in jeder Situation funktioniert, z.B. wenn der Zeitpunkt der Transformation im Verhältnis zur Krümmung der Lösungskurve noch zu weit vom Ereigniszeitpunkt weg ist. Ein weiterer Unterschied zum QSS-Verfahren besteht darin, dass die direkte Erkennung der Ereignisse nicht gegeben ist, da die Henon-Methode immer erst nach dem Verpassen in die Ereignisorientierung wechselt. Zusätzlich muss noch erwähnt werden, dass es sich bei der Henon-Methode, im Gegensatz zum QSS-Verfahren, um ein synchrones Verfahren handelt. Welches der beiden Verfahren schlussendlich effizienter arbeitet, ist von der Häufigkeit der auftretenden Ereignisse abhängig.

Nun werden wir auf die wesentlichen Eigenschaften des QSS-Verfahrens eingehen.

3.1 Stabilität und Konvergenz

Das QSS-Verfahren behält einige Eigenschaften des kontinuierlichen Systems bei, um eine gute Qualität der numerischen Lösung zu garantieren [1]. In diesem Abschnitt werden wir die Stabilität und Konvergenz des QSS-System genauer diskutieren. Um beginnen zu können, müssen wir noch folgenden Begriff einführen.

Definition 3.1.1. Betrachten wir ein System, welches gegeben ist durch die Differentialgleichung

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t)),$$

so spricht man von einer *Ruhelage* $\hat{\boldsymbol{x}}$ (auch Gleichgewicht oder Fixpunkt), falls für alle t gilt: $f(\hat{\boldsymbol{x}}) = \mathbf{0}$.

Eine wichtige Eigenschaft des Quantized State Systems ist, dass Ruhelagen des originalen Systems erhalten bleiben.

Satz 3.1.2 (Erhaltung der Ruhelage [1])**.** *Sei ein kontinuierliches System ohne Eingangssignal (3.2a) und das dementsprechende QSS-System (3.2b) gegeben*

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)), \tag{3.2a}$$

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{q}(t)), \tag{3.2b}$$

dann ist $\boldsymbol{x} = \hat{\boldsymbol{x}}$ genau dann eine Ruhelage von (3.2a), falls $\boldsymbol{q} = \hat{\boldsymbol{x}}$ eine Ruhelage von (3.2b) ist.

Dieser Satz gilt ebenso für Systeme mit konstanter Eingangsvariable $\boldsymbol{u}$. Dieser Satz sagt nichts darüber aus, dass das quantisierte System dadurch automatisch eine Ruhelage hat (womöglich ist $\hat{\boldsymbol{x}} \notin \boldsymbol{Q}$, wobei $\boldsymbol{Q}$ die Menge aller Zustände ist, welche $\boldsymbol{q}(t)$ annehmen kann).

Nun können wir folgenden Stabilitätsbegriff definieren.

Definition 3.1.3. Sei $\hat{\boldsymbol{x}}$ eine Ruhelage einer gewöhnlichen Differentialgleichung $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t))$, so ist $\hat{\boldsymbol{x}}$

- *stabil* (im Sinne von Ljapunow), falls für jedes $\epsilon > 0$ ein $\delta > 0$ existiert, sodass $\forall t > 0$ und für alle Trajektorien $\boldsymbol{x}(t)$ mit $\|\boldsymbol{x}(0) - \hat{\boldsymbol{x}}\| < \delta$, die Ungleichung $\|\boldsymbol{x}(t) - \hat{\boldsymbol{x}}\| < \epsilon$ erfüllt ist,
- *asymptotisch stabil*, falls $\hat{\boldsymbol{x}}$ stabil ist und die Trajektorien gleichmäßig gegen die Ruhelage konvergieren: $\lim_{t\to\infty} \|\boldsymbol{x}(t) - \hat{\boldsymbol{x}}\| = \mathbf{0}$.

Um zu zeigen, dass eine Ruhelage eines dynamisches Systems asymptotisch stabil ist, wird die Existenz einer zugehörigen Ljapunow-Funktion bewiesen, welche hier nicht weiter beschrieben wird. Die Definition kann bei Rolf Unbehauen [4] nachgelesen werden. Bevor wir uns mit der Stabilität des QSS-Systems beschäftigen, sei noch erwähnt, dass dieses System nicht asymptotisch stabil sein kann, solange die Ruhelage $\hat{\boldsymbol{x}} \notin \boldsymbol{Q}$.

Falls das kontinuierliche System (3.2a) eine asymptotisch stabile Ruhelage $\hat{\boldsymbol{x}}$ hat (u.a. existiert dann eine Ljapunow-Funktion), dann stellt der nächste Satz sicher, dass für eine beliebige Umgebung D, mit $\hat{\boldsymbol{x}} \in D$, eine Quantisierung gefunden werden kann, sodass die Lösung von (3.2b) in dieser Umgebung bleibt. Anwendungen werden zeigen (Abb. 3.2 im Beispiel 3.3.1), dass die Lösungskurve des QSS-Systems um den Ruhepunkt oszilliert, aber beschränkt bleibt. Die Abweichung zur exakten Lösung hängt dann von der gewählten Quantisierung ab.

Satz 3.1.4 (Stabilität des QSS-Verfahrens [1])**.** *Es sei ein kontinuierliches System wie in (3.2a) gegeben und zusätzlich sei $f(\boldsymbol{x}(t))$ stetig differenzierbar. Weiters sei das System in der Ruhelage $\hat{\boldsymbol{x}} = 0$ asymptotisch stabil und D eine offene Umgebung von $\hat{\boldsymbol{x}}$ in der die zeitliche Ableitung der entsprechenden Ljapunov-Funktion $V(\boldsymbol{x})$ negativ definit ist. Zusätzlich betrachtet man eine weitere Umgebung $D_1 \subset D$, welche durch die Niveaulinie zum Niveau V_1 beschränkt ist, d.h. $D_1 = \{\boldsymbol{x} \in D | V(\boldsymbol{x}) \leq V_1\}$.*
Dann findet man zu einer beliebigen offenen Umgebung $D_2 \subset D_1$, die durch ein Niveaulinie zum Niveau $V_2 < V_1$ beschränkt ist, eine geeignete Quantisierung, dessen Trajektorie in D_1 startet und in das Innere von D_2 konvergiert.

Beweis. Definieren wir uns eine neue Variable $\boldsymbol{\Delta x}(t) = \boldsymbol{q}(t) - \boldsymbol{x}(t)$ und setzten dies in (3.2b) ein, so erhalten wir

$$\dot{\boldsymbol{x}}(t) = f((\boldsymbol{x}(t) + \boldsymbol{\Delta x}(t)). \tag{3.3a}$$

Betrachten wir die Umgebung $D_3 = D_1 \setminus D_2$, so folgt aus der negativen Definitheit von $\dot{V}(\boldsymbol{x})$, dass ein $s > 0$ existiert mit:

$$\dot{V}(\boldsymbol{x}) < -s, \quad \forall \boldsymbol{x}(t) \in D_3$$

Nun definieren wir die neuen Funktionen

$$\alpha(\boldsymbol{x}, \boldsymbol{\Delta x}) = \langle \nabla V(\boldsymbol{x}),\, f(\boldsymbol{x} + \boldsymbol{\Delta x}) \rangle\,, \tag{3.3b}$$

$$\alpha_M(\boldsymbol{\Delta x}) = \sup_{\boldsymbol{x} \in D_3} \alpha(\boldsymbol{x},\, \boldsymbol{\Delta x}), \tag{3.3c}$$

welche in $\boldsymbol{\Delta x}(t)$ stetig sind und folgende Beziehungen erfüllen:

$$\alpha_M(0) = \sup_{\boldsymbol{x} \in D_3} \alpha(\boldsymbol{x},\, 0) = \sup_{\boldsymbol{x} \in D_3} \langle \nabla V(\boldsymbol{x}),\, f(\boldsymbol{x}) \rangle = \sup_{\boldsymbol{x} \in D_3} \dot{V}(\boldsymbol{x}) < -s \tag{3.3d}$$

Für eine Zahl $s_1 : 0 < s_1 < s$ kann jetzt eine positive Zahl r gefunden werden mit

$$\|\boldsymbol{\Delta x}\| < r, \tag{3.3e}$$

sodass

$$\alpha_M(\boldsymbol{\Delta x}) < -s_1. \tag{3.3f}$$

Sei $\boldsymbol{x}(t)$ eine Lösung des Systems (3.3a) mit dem Anfangswert $\boldsymbol{x}(0) = \boldsymbol{x_0} \in D_3$ und wurde eine Quantisierung gewählt, wie es (3.3e) vorschreibt, dann folgt

$$\langle \nabla V(\boldsymbol{x}),\, \dot{\boldsymbol{x}} \rangle \overset{(3.3a),(3.3b)}{=} \alpha(\boldsymbol{x}, \boldsymbol{\Delta x}) \overset{(3.3f)}{<} -s_1. \tag{3.3g}$$

Solange $\boldsymbol{x}(t) \in D_3$, ist die obige Bedingung erfüllt. Bei der Integration der linken und rechten Seite bekommen wir folgende Ungleichung:

$$\begin{aligned}
\langle \nabla V(\boldsymbol{x}),\, \dot{\boldsymbol{x}} \rangle &< -s_1 \\
\dot{V}(\boldsymbol{x}) &< -s_1 \\
\int_0^t \dot{V}(\boldsymbol{x}) dt &< -\int_0^t s_1 dt \\
V(\boldsymbol{x}(t)) - V(\boldsymbol{x_0}) &< -s_1 \cdot t \\
V(\boldsymbol{x}(t)) &< V(\boldsymbol{x_0}) - s_1 \cdot t
\end{aligned}$$

Wir sehen, dass $V(\boldsymbol{x}(t))$ (ausgewertet entlang der Lösung $\boldsymbol{x}(t)$) durch eine streng monoton fallende Funktion beschränkt ist. Da außerdem $V(\boldsymbol{x_0}) < V_1$ (V_1 ist das Niveau, welches D_1 einschränkt), wird die Trajektorie von $\boldsymbol{x}(t)$ die Umgebung D_1 nicht mehr verlassen.
Mithilfe von V_2 (Niveau, welches D_2 einschränkt) können wir die benötigte Zeit t_2 feststellen, wann die Trajektorie D_2 erreicht:

$$\begin{aligned}
V_2 - V(\boldsymbol{x_0}) &< -s_1 \cdot t_2 \\
s_1 \cdot t_2 &< V(\boldsymbol{x_0}) - V_2 \\
t_2 &< \frac{V(\boldsymbol{x_0}) - V_2}{s_1}
\end{aligned}$$

Der Einfachheit halber wollen wir nun ein Quantum ΔQ betrachten, das einheitlich und in allen Komponenten von $\boldsymbol{x}(t) \in \mathbb{R}^n$ gleich groß ist. Selbiges wollen wir auch für die Hysteresebreite ϵ fordern. Da jede Komponente von $\boldsymbol{\Delta x}(t)$ kleiner sein muss als das Maximum von ΔQ und ϵ, wird die Bedingung (3.3e) wie folgt erreicht:

$$\begin{aligned}
\|\boldsymbol{\Delta x}(t)\| &< \sqrt{n} \cdot \max(\Delta Q,\, \epsilon) \\
\max(\Delta Q,\, \epsilon) &< \frac{r}{\sqrt{n}}
\end{aligned}$$

□

Dieser Satz gilt ebenso für Systeme, die ihre Ruhelage $\hat{\boldsymbol{x}}$ nicht im Ursprung oder einen konstanten Eingang $\boldsymbol{u}(t)$ haben.

Der nächste Satz beschäftigt sich mit der Konvergenz des QSS-Systems.

Satz 3.1.5 (Konvergenz des QSS-Verfahrens [1])**.** *Betrachten wir ein kontinuierliches System wie* (3.2*a*) *und das zugehörige QSS-System* (3.2*b*)*. Sei D eine (beschränkte) Umgebung*

$$D = \{\boldsymbol{x} = (x_1, ..., x_n) \,|\, Q_{0_j} \leq x_j \leq Q_{r_j}\}$$

auf welcher die Funktion $f(\boldsymbol{x})$ lipschitzstetig ist. Haben wir zu einem Anfangswert $\boldsymbol{x}(0) = \boldsymbol{x_0}$ eine Lösung $\boldsymbol{\phi}(t)$ des kontinuierlichen Systems vorliegen, mit $\boldsymbol{\phi}(t) \in D_1 \subset D$. Sei $\boldsymbol{\phi_q}(t)$ die Lösung des QSS-Verfahrens mit dem gleichen Anfangswert $\boldsymbol{q}(0) = \boldsymbol{x_0}$, so konvergiert $\boldsymbol{\phi_q}(t) \longrightarrow \boldsymbol{\phi}(t)$, falls $\max(Q_{i-1_j} - Q_{i_j}, \epsilon) \longrightarrow 0$.

Der Beweis dieses Satzes haben Ernesto Kofman und Sergio Junco [1] erbracht. Genauso wie in den vorigen Sätzen lässt sich dieser für Systeme $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ mit einer Eingangsvariable $\boldsymbol{u}(t)$ erweitern. Die Funktion $f(\boldsymbol{x}, \boldsymbol{u})$ muss dann in beiden Variablen $\boldsymbol{x}$ und $\boldsymbol{u}$ lokal lipschitzstetig sein. Wir wissen jetzt, dass die Lösung des quantisierten Systems (3.2*b*) gegen die Lösung des kontinuierlichen Systems (3.2*a*) konvergiert, falls das maximale Quantum ΔQ und die Hysteresebreite ϵ hinreichend klein gewählt werden.

3.2 Fehleranalyse

Aus dem vorangegangenen Abschnitt ist ersichtlich, dass der Fehler des QSS-Verfahrens von der Wahl der Quantisierung abhängt. Auf diesen Zusammenhang zwischen dem Simulationsfehler und der Quantisierung gehen wir näher ein, betrachten allerdings nur lineare zeitinvariante Systeme (kurz: LZI-Systeme), die eine wichtige Rolle in der Systemtheorie einnehmen. Für diese Systeme fordern wir außerdem, dass sie asymptotisch stabil [5] sind. Folgender Satz inklusive Beweis stammt von François E. Cellier und Ernesto Kofman [6].

Satz 3.2.1. *Sei $\boldsymbol{x_a}(t)$ die (analytische) Lösung des LZI-Systems*

$$\dot{\boldsymbol{x}}_{\boldsymbol{a}}(t) = \boldsymbol{A} \cdot \boldsymbol{x_a}(t) + \boldsymbol{B} \cdot \boldsymbol{u}(t), \tag{3.4a}$$

in welchem die Matrix $\boldsymbol{A}$ diagonalisierbar ist, $\boldsymbol{A} = \boldsymbol{V} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{V}^{-1}$, und das Hurwitzkriterium[1] erfüllt. Sei $\boldsymbol{x}(t)$ die Lösung des entsprechenden QSS-Systems

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A} \cdot \boldsymbol{q}(t) + \boldsymbol{B} \cdot \boldsymbol{u}(t), \tag{3.4b}$$

[1] Erfüllt $\boldsymbol{A}$ dieses Kriterium, so besitzt sie nur Eigenwerte mit negativen Realteil und gewährleistet dadurch die asymptotische Stabilität des Systems (nachzulesen in Rolf Unbehauen [4]).

welche vom selben Anfangswert startet wie $\boldsymbol{x_a}(t)$. Dann ist der maximale Fehler beschränkt durch:

$$|\boldsymbol{x}(t) - \boldsymbol{x_a}(t)| \preceq |\boldsymbol{V}| \cdot \left|\mathrm{Re}\{\boldsymbol{\Lambda}\}^{-1} \cdot \boldsymbol{\Lambda}\right| \cdot \left|\boldsymbol{V}^{-1}\right| \cdot \boldsymbol{\Delta Q} \tag{3.4c}$$

Der Betrag $|.|$ wird dabei komponentenweise angewendet. Für eine Matrix M heißt das, dass in $|M|$ für jeden Eintrag $m_{i,j} \in \mathbb{C}$ der Betrag gebildet wird. Für die Vergleichsoperation $\preceq$ für zwei Vektoren $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}$ gilt: $\boldsymbol{a} \preceq \boldsymbol{b} \Longleftrightarrow a_i \leq b_i$ für $1 \leq i \leq n$.

Beweis. Betrachten wir die Lösung $\boldsymbol{x}(t)$ des quantisierten Systems (3.4*b*), dann lässt sich dieses auch anschreiben als:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A} \cdot [\boldsymbol{x}(t) + \boldsymbol{\Delta x}(t)] + \boldsymbol{B} \cdot \boldsymbol{u}(t) \tag{3.5a}$$

Wir definieren den Fehler nun mit $\boldsymbol{e}(t) = \boldsymbol{x}(t) - \boldsymbol{x_a}(t)$. Wenn wir die Gleichung (3.4*a*) von (3.5*a*) subtrahieren erhalten wir

$$\dot{\boldsymbol{e}}(t) = \boldsymbol{A} \cdot [\boldsymbol{e}(t) + \boldsymbol{\Delta x}(t)]\,. \tag{3.5b}$$

Zunächst betrachten wir wieder den eindimensionalen Fall:

$$\dot{e}(t) = a \cdot [e(t) + \Delta x(t)] \tag{3.5c}$$

Seien alle vorkommenden Größen $e(t)$, a und $\Delta x(t)$ aus $\mathbb{C}$. Aufgrund des asymptotisch stabilen Systems gilt $\mathrm{Re}\{a\} < 0$. Wir können außerdem eine positive Zahl w finden, sodass $|\Delta x| \leq w$ (w hängt von der gewählten Quantisierung ab). Drücken wir den Fehler in Polarkoordinaten aus

$$e(t) = \rho(t) \cdot e^{i\theta(t)}$$

und setzen diesen Fehler in die Gleichung (3.5c) ein, so erhalten wir:

$$\dot{\rho}(t) \cdot e^{i\theta(t)} + \rho(t) \cdot e^{i\theta(t)} \cdot i \cdot \dot{\theta}(t) = a \cdot \left[\rho(t) \cdot e^{i\theta(t)} + \Delta x(t)\right]$$

Eine Division beider Seiten durch $e^{i\theta(t)}$ führt zu

$$\dot{\rho}(t) + \rho(t) \cdot i \cdot \dot{\theta}(t) = a \cdot \left[\rho(t) + \Delta x(t) \cdot e^{-i\theta(t)}\right].$$

Betrachten wir davon nur den Realteil, dann gilt:

$$\begin{aligned}
\dot{\rho}(t) &= \mathrm{Re}\{a\} \cdot \rho(t) + \mathrm{Re}\left\{a \cdot \Delta x(t) \cdot e^{-i\theta(t)}\right\} \\
&\leq \mathrm{Re}\{a\} \cdot \rho(t) + |a| \cdot |\Delta x(t)| \\
&\leq \mathrm{Re}\{a\} \cdot \left[\rho(t) + \frac{|a|}{\mathrm{Re}\{a\}} \cdot w\right] \\
&= \mathrm{Re}\{a\} \cdot \left[\rho(t) - \left|\frac{a}{\mathrm{Re}\{a\}}\right| \cdot w\right]
\end{aligned}$$

Da die Lösungen den selben Anfangswert haben, ist der Fehler zu Beginn gleich 0. Da $\rho(t) = |e(t)|$, gilt insbesonders $\rho(0) = 0$ und da außerdem $\mathrm{Re}\{a\} < 0$, muss in jedem Zeitpunkt gelten, dass

$$|e(t)| = \rho(t) \leq \left| \frac{a}{\mathrm{Re}\{a\}} \right| \cdot w.$$

Falls $\rho(t)$ diese Schranke erreicht, würde $\dot{\rho}(t)$ negativ werden und demnach $\rho(t)$ wieder fallen.
Gehen wir nun vom allgemeinen, mehrdimensionalen Fall aus und beschränken uns zunächst darauf, dass die Matrix $\boldsymbol{A}$ aus der Gleichung (3.5*b*) eine Diagonalmatrix ist, deren Einträge $a_{i,i}$, $\mathrm{Re}\{a_{i,i}\} < 0$ erfüllen. Dadurch können wir den vorigen, eindimensionalen Fall für jede Komponente anwenden. Machen wir die Einschränkung

$$\boldsymbol{\Delta x}(t) \preceq \boldsymbol{w},$$

dann gilt auch hier für den Fehler:

$$\boldsymbol{e}(t) \preceq \left| \mathrm{Re}\{\boldsymbol{A}\}^{-1} \cdot \boldsymbol{A} \right| \cdot \boldsymbol{w} \tag{3.5d}$$

Sei die Matrix $\boldsymbol{A}$ aus der Gleichung (3.5*b*) zwar keine Diagonalmatrix, aber zumindest diagonalisierbar und erfülle das Hurwitz-Kriterium, dann können wir sie mit ihrer diagonalen Eigenwertmatrix $\boldsymbol{\Lambda}$ (und der Eigenvektormatrix $\boldsymbol{V}$) darstellen:

$$\boldsymbol{A} = \boldsymbol{V} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{V}^{-1} \tag{3.5e}$$

Betrachten wir unser QSS-System (3.5*a*) und definieren $\boldsymbol{\Delta Q}$ als jenen Vektor, welcher in jeder Komponente das maximale Quantum bzw. die maximale Hysteresebreite enthält, so gilt

$$|\boldsymbol{\Delta x}(t)| \preceq \boldsymbol{\Delta Q}.$$

Wir definieren nun eine neue Variable

$$\boldsymbol{z}(t) = \boldsymbol{V}^{-1} \cdot \boldsymbol{e}(t).$$

Schreiben wir die Gleichung (3.5*b*) nun mit dieser neuen Variable an, so erhalten wir

$$\dot{\boldsymbol{e}}(t) = \boldsymbol{V} \cdot \dot{\boldsymbol{z}}(t) = \boldsymbol{A} \cdot \left[\boldsymbol{V} \cdot \boldsymbol{z}(t) + \boldsymbol{\Delta x}(t) \right].$$

Formen wir diese nach $\dot{\boldsymbol{z}}(t)$ um und setzen (3.5*e*) ein, so ergibt dies

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= \boldsymbol{V}^{-1} \cdot \boldsymbol{A} \left[\boldsymbol{V} \cdot \boldsymbol{z}(t) + \boldsymbol{\Delta x}(t) \right] \\ &= \boldsymbol{\Lambda} \cdot \left[\boldsymbol{z}(t) + \boldsymbol{V}^{-1} \cdot \boldsymbol{\Delta x}(t) \right]. \end{aligned} \tag{3.5f}$$

Nun gilt zum einen

$$\left| \boldsymbol{V}^{-1} \cdot \boldsymbol{\Delta x}(t) \right| \preceq \left| \boldsymbol{V}^{-1} \right| \cdot \boldsymbol{\Delta Q}$$

und zum anderen ist aufgrund der Vorausetzung die Matrix $\boldsymbol{\Lambda}$ eine Diagonalmatrix. Damit hat die Gleichung (3.5*f*) die Gestalt von (3.5b) und es gilt die Ungleichung (3.5*d*) auch hier:

$$|\boldsymbol{z}(t)| \preceq \left| \mathrm{Re}\{\boldsymbol{\Lambda}\}^{-1} \cdot \boldsymbol{\Lambda} \right| \cdot \left| \boldsymbol{V}^{-1} \right| \cdot \boldsymbol{\Delta Q}$$

Die Multiplikation beider Seiten mit $|\boldsymbol{V}|$ ergibt

$$|\boldsymbol{e}(t)| \preceq |\boldsymbol{V}| \cdot |\boldsymbol{z}(t)| \preceq |\boldsymbol{V}| \cdot \left|\mathrm{Re}\{\boldsymbol{\Lambda}\}^{-1} \cdot \boldsymbol{\Lambda}\right| \cdot \left|\boldsymbol{V}^{-1}\right| \cdot \boldsymbol{\Delta Q},$$

sodass wir schlussendlich die Aussage des Satzes bewiesen haben:

$$|\boldsymbol{x}(t) - \boldsymbol{x_a}(t)| \preceq |\boldsymbol{V}| \cdot \left|\mathrm{Re}\{\boldsymbol{\Lambda}\}^{-1} \cdot \boldsymbol{\Lambda}\right| \cdot \left|\boldsymbol{V}^{-1}\right| \cdot \boldsymbol{\Delta Q}$$

□

Auffallend ist, dass der maximale Fehler nur von der Matrix $\boldsymbol{A}$ und vom gewählten Quantum abhängt und über die gesamte Simulationsdauer konstant bleibt. Es muss noch erwähnt werden, dass im QSS-Verfahren erster Ordnung die Anzahl der Schritte linear mit der gewünschten Genauigkeit wächst.

3.3 Verhalten des QSS-Verfahrens bei steifen Differentialgleichungen

In diesem Abschnitt gehen wir auf das Verhalten des QSS-Verfahrens bei steifen Problemen ein. Erwähnenswert ist, dass es sich bei diesem QSS-System nach wie vor um ein explizites Verfahren handelt. Aus diesem Grund werden wir in diesem Abschnitt auch keine impliziten Verfahren als Vergleich wählen. Wir beginnen mit einem einfachen Beispiel.

Beispiel 3.3.1. Gegeben sei folgende Differentialgleichung:

$$\dot{x}(t) = \lambda(x(t) - p), \qquad x(0) = 1 \tag{3.6}$$

Wählen wir $\lambda = -100$, so haben wir eine steife Differentialgleichung vorliegen. Zunächst betrachten wir $p = 0$. In Abb. 3.1 sehen wir die Lösung des QSS-Verfahrens mit einem gewähltem Quantum von 1, $\Delta Q = 1$. Die Lösung wird mit nur einem Zustandswechsel erreicht.

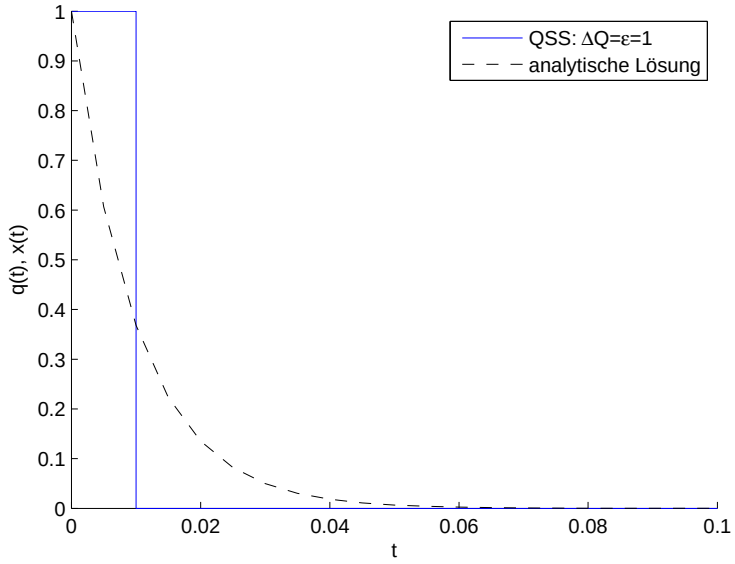

Abbildung 3.1: Lösung des QSS-Verfahrens für $p = 0$

Setzen wir nun im nächsten Schritt $p = 0.1$ und $\Delta Q = 0.2$. Das hat zur Auswirkung, dass das System keine Ruhelage $\hat{x}$ mehr besitzt.

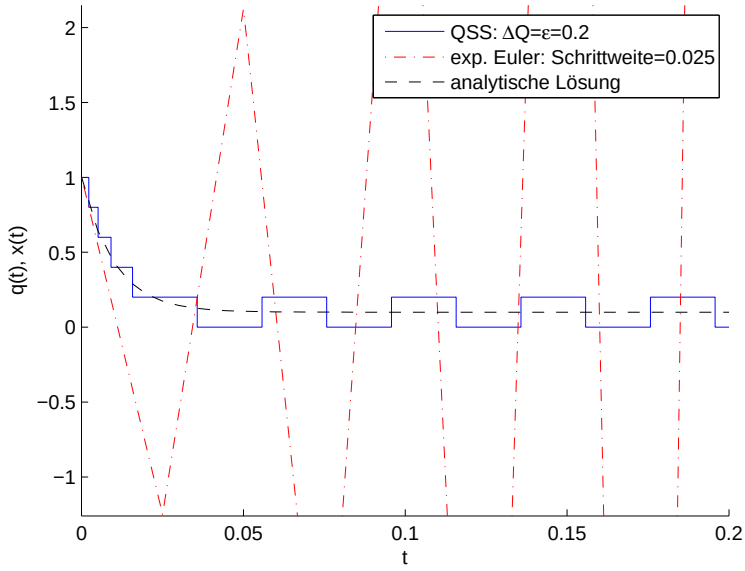

Abbildung 3.2: Vergleich der Oszillationen des QSS- und Eulerverfahrens

In Abb. 3.2 sehen wir einen Vergleich zwischen dem QSS-Verfahren und dem expliziten Eulerverfahren. Beide Verfahren oszillieren um die Ruhelage $\hat{x} = 0.1$. Dennoch ist das Verhalten der Oszillationen unterschiedlich. Während sich die numerische Lösung des

zeitorientierten Verfahrens bei äquidistanter Schrittweite immer mehr von der Lösung entfernt, bleibt der Fehler des QSS-Verfahrens beschränkt. In diesem Vergleich ist also das QSS-Verfahren ganz klar dem zeitdiskreten vorzuziehen, da der Fehler durch $|\Delta Q|$ beschränkt ist.

Im nächsten Beispiel werden wir sehen, dass das QSS-Verfahren nur begrenzt für steife Systeme einsetzbar ist. Eine höhere Steifigkeitsrate hat auch beim QSS-System negative Auswirkungen in der Oszillation.

Beispiel 3.3.2. Hier betrachten wir eine Differentialgleichung

$$\begin{aligned} \dot{x}_1(t) &= x_2(t), \\ \dot{x}_2(t) &= -10^3 \cdot (x_1(t) + x_2(t)) + p, \end{aligned} \tag{3.7}$$

welche eine Steifigkeitsrate von $\kappa \approx 1000$ hat. Im Allgemeinen spricht man von einem steifen Problem, falls $\kappa \gg 1$. Für das System $\dot{\boldsymbol{x}} = \boldsymbol{A} \cdot \boldsymbol{x}$ ist

$$\kappa = \frac{\max\{|Re\lambda_i| : Re\lambda_i < 0\}}{\min\{|Re\lambda_i| : Re\lambda_i < 0\}},$$

wobei λ_i die Eigenwerte der Matrix $\boldsymbol{A}$ sind.
Für das obige System (3.7) betrachten wir das Anfangswertproblem $x_1(0) = 0$, $x_2(0) = 10$ mit der Konstanten $p = 10^4$. Um diese DGL mit dem QSS-Verfahren zu lösen wählen wir ein Quantum von $\Delta Q = 1$. Als Vergleich verwenden wir wieder ein zeitdiskreten Verfahren, diesmal aber das Runge–Kutta-Verfahren von Dormand und Prince mit einer relativen Toleranz von 10^{-3}.

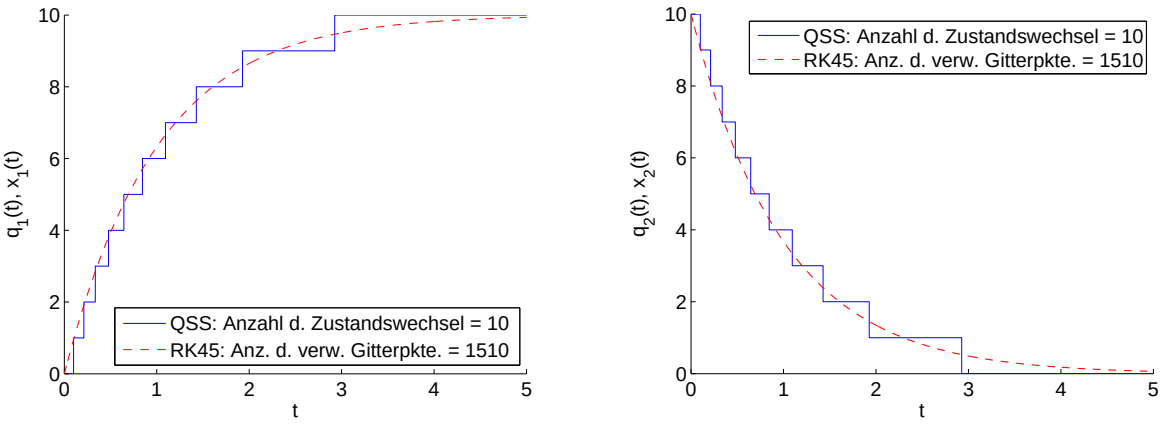

Abbildung 3.3: Ergebnisse des QSS- und RK45-Verfahrens

In Abb. 3.3 sehen wir, dass das QSS-Verfahren trotz dieser Problemstellung ein erstaunlich gutes Ergebnis liefert. Für die numerische Lösung benötigt es deutlich weniger Auswertungen als das RK45-Verfahren.
Allerdings stellt diese gute Lösung des QSS-Verfahrens eine Ausnahme dar, weshalb wir die DGL (3.7) geringfügig ändern, indem wir die Konstante p auf 9500 setzen.

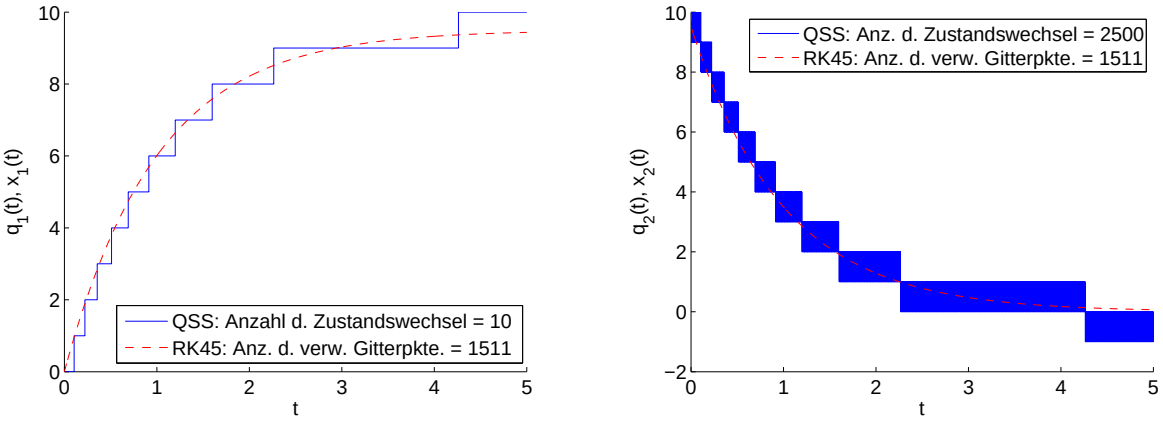

Abbildung 3.4: im rechten Bild hat die Lösung des QSS-Verfahrens sehr schnelle Oszillationen

Die Abb. 3.4 zeigt, dass die neue DGL für das QSS-Verfahren schwerer zu lösen ist. Es sind nun Oszillationen entstanden, welche aufgrund der hohen Steifigkeitsrate sehr schnell sind.

Zur Lösung dieses Problems dient eine implizite Methode des QSS-Verfahrens. Gezeigt wird die Funktionsweise des Verfahrens nach François E. Cellier und Ernesto Kofman [6].

Backward Quantized State System (BQSS)

Die Idee des impliziten QSS-Verfahrens ist dem des Backward Eulerverfahrens (BE) ähnlich. Falls wir den Zustand im nächstem Zeitpunkt t_{k+1} kennen, so können wir f_i „auswerten“ und erhalten die notwendige Ableitung zum Zeitpunkt t_{k+1}:

$$\dot{x}_{i_{k+1}} = f_i(\boldsymbol{q_{k+1}}, \boldsymbol{u_{k+1}})$$

Im Gegensatz zum BE, in welchem f_i nicht ausgewertet werden kann, sondern aufgelöst werden muss, ist dieser Vorgang beim BQSS nicht nötig. Wir wissen, dass es für den quantisierten Zustand $q_{i_{k+1}}$ nur zwei Möglichkeiten gibt, $q_{i_{k+1}} = q_{i_k} \pm \Delta Q$. Dafür muss der Integrator beim BQSS in der Lage sein, die stationäre Funktion f auszuwerten, damit das Verhalten der Ableitung bekannt ist.
Ein weiterer Vorteil ist die lokale Behandlung des Problems. Das heißt, führt ein Integrator gerade einen Zustandswechsel durch, gilt es zu berechnen, welchen Zustand er in Zukunft haben wird, während die anderen Integratoren ihren Zustand unverändert belassen.

Wir lernen nun eine Variante eines impliziten QSS-Verfahren kennen:

1. Es kommt in diesem Augenblick in der i-ten Komponente zu einem Zustandswechsel, sodass $q_i = Q$.

2. Wir überprüfen, ob der nächste Zustand von $q_i = Q + \Delta Q$ ist, indem wir ihn mit $\dot{x}_i = f_i(\boldsymbol{q}, \boldsymbol{u})$ auswerten und davon ausgehen, dass $\dot{x}_i > 0$.

3. War die Annahme richtig, so können wir die Zeit für den nächsten Zustandswechsel σ_i berechnen.

4. Wir überprüfen, ob der nächste Zustand von $q_i = Q - \Delta Q$ ist, indem wir ihn mit $\dot{x}_i = f_i(\boldsymbol{q}, \boldsymbol{u})$ auswerten und davon ausgehen, dass $\dot{x}_i < 0$.

5. War die Annahme richtig, so können wir σ_i berechnen.

6. Waren beide Annahmen richtig oder beide falsch, so ist dies ein Indiz für eine Oszillation dieses Zustandes, weshalb wir $\sigma_i = \infty$ setzen und auf das nächste Eingangsereignis warten.
   War hingegen genau eine Annahme richtig, so belassen wir das bereits berechnete σ_i.

Kapitel 4

Umsetzung des QSS-Verfahrens in Simulink

4.1 SimEvents und Simulink

Der quantisierte Integrator des QSS-Verfahrens lässt sich in der Simulationsumgebung Simulink mithilfe der SimEvents-Bibliothek implementieren. Die Blöcke aus dieser Bibliothek arbeiten entweder mit ereignisbasierten Signalen oder mit sogenanten „Entitäten“, die wir zukünftig mit S bezeichnen werden.
In der Implementierung des Integrators gibt es zu jedem Zeitpunkt genau eine Entität, welche für den Integratorzustand verantwortlich ist. Bei jedem Eingangs- oder Ausgangsereignis wird somit eine neue Entität erzeugt, die den Zustand des Integrators aktualisiert. Es sei erwähnt, dass eine Entität S selbst keinen Wert annehmen kann. Es ist aber möglich, ihr verschiedene Attribute zuzuweisen. Diese Attribute erleichtern die Umsetzung von Berechnungen, wie z.B. den Zustand $x(t)$ oder die benötigte Zeit bis zum nächsten Zustandswechsel von $q(t)$. Wir können also sagen, dass sich die Entität und ihre Attribute mit dem Zustand s des QSS-Integrators im atomaren DEVS-Modell vergleichen lassen.
Um bei einem Ereignis eine Entität erzeugen zu können, werden Entitätsgeneratoren verwendet. Für jede Aufgabe (Eingangsereignis, Ausgangsereignis, Intitialisierung und Zurücksetzen) wird ein eigener Generator benutzt.
Ein weiterer wichtiger Block in Simulink stellt der Single-Server dar, der so konfiguriert werden kann, dass die Weiterleitung der Entität für eine bestimmt Zeit verzögert wird. Simulink nennt diese Zeit „Service-Time“, welche den Zeitpunkt des nächsten Zustandswechsels beschreibt.

4.2 QSS-Modell in Simulink

Der quantisierte Integrator ist in Simulink mit einem einheitlichen Quantum ΔQ und einer fixen Hysteresebreite ϵ implementiert.
In diesem Abschnitt betrachten wir aus Gründen der einfacheren Darstellung nur den quantisierten Zustand $q(t)$, obwohl intern mit einem Index $i \in \mathbb{Z}$ gerechnet und erst für das Ausgangssignal $q = i \cdot \Delta Q$ berechnet wird. Wir umgehen damit größere Rundungsfehler von $q(t)$.

In Abb. 4.1 sehen wir einen Überblick der Implementierung des QSS-Integrators in Simulink. Bei den mit Farbe umrandeten Blöcken handelt es sich um:

- Entitätsgeneratoren und Anfangswerte (blau)
- Berechnung von $x(t)$ und σ (grün)
- Entitätsserver (rot)
- Berechnung von $q(t)$ (gelb)

Diese Blöcke nehmen eine wichtige Rolle in der Funktionsweise des Integrators ein, weswegen wir sie noch genauer beschreiben werden. Wie die Subsysteme für die Berechungen von $x(t)$, σ und $q(t)$ aussehen ist im Anhang A.1 zu finden.

Entitätsgeneratoren und Anfangswerte

Der Integrator besitzt insgesamt vier Entitätsgeneratoren.
Einer der Generatoren erzeugt am Simulationsstart eine Entität S_0, die nur den Zweck hat, das Modell zu initialisieren. Dementsprechend existiert diese Entität nur am Start der Simulation. Zusätzlich sind im Integrator die Anfangswerte

$$\begin{aligned} x(0) &= x_0, \\ q(0) &= \left\lceil \frac{x(0)}{\Delta Q} \right\rceil \cdot \Delta Q \end{aligned}$$

gesetzt, um einerseits am Beginn das gesamte System zu initialisieren und andererseits mit den Berechnungen im Integrator beginnen zu können.
Kommt es zu einem Ausgangsereignis, d.h. am Ausgang wurde $q(t)$ neu gesetzt, so erzeugt ein anderer Generator die Entität S^{int}.
Ändert sich die Ableitung $d_x(t)$ am Eingang, übernimmt dies wiederum ein anderer Generator. Die so entstandene Entität bezeichnen wir mit S^{ext}.
Falls wir den Integrator zurücksetzen möchten, wie im Beispiel 5.2.4, dann übernimmt der vierte Generator diese Aufgabe. Die dadurch entstandene Entität S^{res} hat eine ähnliche Aufgaben wie S_0.

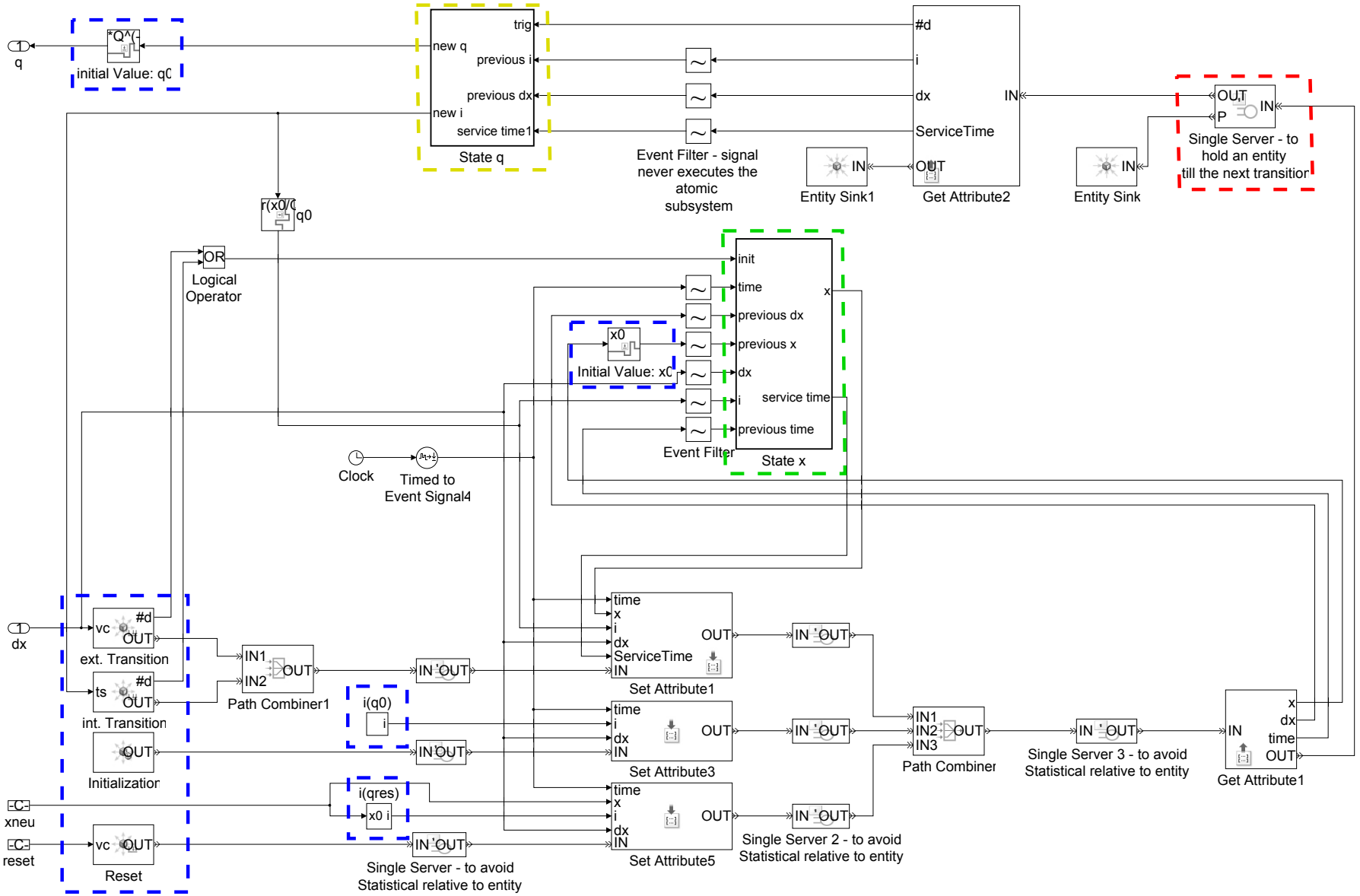

Abbildung 4.1: QSS-Implementierung in Simulink

Die Entitäten werden mithilfe des „Path-Combiner“ zusammengefasst. In vielen Modellen kann es passieren, dass zwei Generatoren eine Entität im selben Moment erzeugen. Es kommt zu einer Blockade einer dieser Generatoren, da eine Entität jenen Weg nehmen muss, der bereits von der anderen belegt ist. Um dieses Problem zu umgehen, kann man den Generatoren Prioritäten zuweisen. Eine weitere Konfiguration lässt zu, dass im Falle einer Blockade Entitäten verworfen werden. Dies ermöglicht in der Simulink-Implementierung eine kontrollierte Abwicklung der Entitäten.

Berechnung von x_n und σ_n

Wir wissen nun, dass Entitäten immer nur durch Ereignisse erzeugt werden. Auch die Berechnungen werden nur zu Zeitpunkten von Ereignissen durchgeführt. Im folgenden bezeichnen wir die Entität als S_n, mit fortlaufendem $n \in \mathbb{N}_0$ (S_{n+1} ist die auf S_n folgende Entität). Das hochgestellte *int* im Fall S_n^{int} dient nur als zusätzlicher Hinweis. Jedem S_n sind die Attribute *Zeitpunkt der Entstehung* t_n, *exakter Zustand* x_n, *quantisierter Zustand* q_n, *Ableitung* $d_{x,n}$ und *Dauer bis zum nächsten Zustandswechsel* σ_n zugewiesen. Diese Informationen dienen immer der darauffolgenden Entität. Noch zu erwähnen ist, dass in der Implementierung selbst nicht zwischen den Entitäten S_n^{ext} und S_n^{int} unterschieden wird (dies wird noch im Abschnitt 4.3.3 dargestellt).
Das Attribut x_n wird durch

$$x_n = \begin{cases} x_0 & \text{falls } n = 0 \\ x_{n-1} + d_{x,n-1} \cdot (t_n - t_{n-1}) & \text{sonst} \end{cases} \tag{4.1a}$$

und σ_n durch

$$\sigma_n = \begin{cases} 0 & \text{falls } n = 0 \\ \frac{(q_{n-1}+\Delta Q)-x_n}{d_{x,n}} & \text{falls } n > 0 \ \wedge \ d_{x,n} > 0 \\ \frac{x_n-(q_{n-1}-\epsilon)}{|d_{x,n}|} & \text{falls } n > 0 \ \wedge \ d_{x,n} < 0 \\ \infty & \text{falls } n > 0 \ \wedge \ d_{x,n} = 0 \end{cases} \tag{4.1b}$$

berechnet.
Wurden der Entität S_n alle Attribute zugewiesen, so kommt sie in den Entitätsserver.

Entitätsserver

In diesem Server verbleibt jede Entität S_n genau σ_n lang. Da wir einen „Single-Server“ verwenden (d.h. es gibt Platz für nur eine Entität), können wir diesen zusätzlich so konfigurieren, dass S_n im Falle einer neu eintreffenden Entität S_{n+1} verworfen wird. In dieser Situation handelt es sich ausschließlich um S_{n+1}^{ext}, eine durch ein Eingangsereignis erzeugte Entität.

Berechnung von q_n

Verbleibt die Entität S_n bis zum nächsten Zustandswechsel im Server (S_n verlässt den Server erst nach Ablauf von σ_n), so wird der Ausgangszustand q_{n+1} neu gesetzt. Ist diese

Entität nicht jene, welche zur Initialisierung verwendet wird, so kommt es zu einem Zustandswechsel am Ausgang, d.h. $q_{n+1} = q_n \pm \Delta Q$. Diese Änderung lässt sich formulieren als

$$q_{n+1} = \begin{cases} q_0 & \text{falls } n = 0 \\ q_n + \Delta Q & \text{falls } n > 0 \quad \wedge \quad d_{x,n} > 0 \\ q_n - \Delta Q & \text{falls } n > 0 \quad \wedge \quad d_{x,n} < 0 \end{cases} . \tag{4.2}$$

4.3 Weitere Anmerkungen zur Implementierung

Es ist von Interesse, eine möglichst hohe Kompatibilität zwischen dem Integrator und den Blöcken in Simulink zu erreichen. Es gibt verschiedene Arten von Blöcken, die einen arbeiten mit zeit-, die anderen mit ereignisbasierten und manche mit beiden Signalen. Da der implementierte QSS-Integrator Elemente aus der SimEvents-Bibliothek enthält, arbeitet dieser ausschließlich mit ereignisbasierten Signalen. Verwenden wir in einem Modell nun auch Blöcke, welche nur mit zeitbasierten Signalen arbeiten können, so müssen wir uns für diese eine Adaption überlegen.

4.3.1 Kommunikation zwischen ereignis- und zeitbasierten Blöcken

Oft ist zielführend aus Blöcken, die nur mit zeitbasierten Signalen arbeiten, ein „Atomic Subsystem“ zu erzeugen (siehe Abb. 4.2). Existiert zu einem zeitbasierten Block auch ein äquivalenter ereignisbasierter Block, ist es in dieser Situation effizienter diesen zu verwenden.

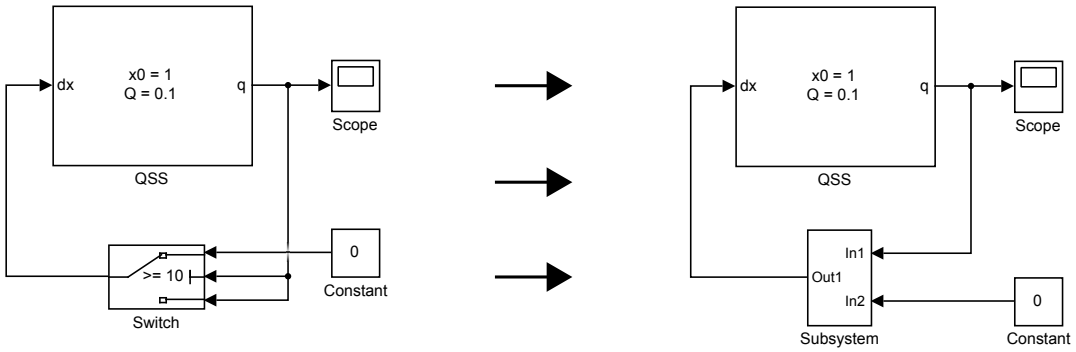

Abbildung 4.2: Der „Switch“-Block kann mit ereignisbasierten Signalen nicht umgehen, deshalb muss er in ein „Atomic Subsystem“ gelegt werden

Eine andere Kommunikationsmöglichkeit bieten auch die „Gateway“-Blöcke aus der SimEvents-Bibliothek. Dabei besteht das Problem, dass Simulink beim Lösen eine algebraische Schleife hat. Dieses Problem kann man mit dem Einbau einer Verzögerung (z.B. mit dem „Unitdelay“-Block am Integratorausgang) umgehen. Es sollte berücksichtigt werden, dass in jedem Integrationsschritt die stationäre Funktion mit dem vorherigen

Zustand ausgewertet wird und es deshalb zu einem großen Fehler kommen kann.
Um genau diesen Fehler gering zu halten, sei noch jene Methode erwähnt, die die stationäre Funktion bei jedem Ausgangsereignis doppelt ausführt. Das heißt, wenn es bei einem Integrator zu einem Zustandswechsel kommt, wird der Ausgangswert zwei Mal innerhalb kurzer Zeit für das System gesetzt. Diese Zeit beschreibt, wie lange das System mit dem vorigen und falschen Wert rechnet, der durch den „Unitdelay"-Block verursacht wird. Die Methode erfordert viel Rechenaufwand, hat aber zur vorigen Methode eine wesentlich bessere numerische Lösung.
Die soeben genannten Methoden sehen wir uns am folgenden Beispiel an:

$$\dot{x}(t) = x(t), \qquad x(0) = 0$$

Zum Lösen der DGL verwenden wir ein Quantum von $\Delta Q = 0.1$.
In Abb. 4.3 sehen wir, dass die numerischen Lösungen der Variante des „Atomic Subsystems" und der „Doppelten Auswertung" beinahe dieselben sind. Im Gegensatz zu diesen Varianten kommt es durch das Zwischenschalten des „Unitdelay"-Blocks zu einer großer Abweichung der analytischen Lösung. Deshalb kommen nur die zwei erst genannten Methoden in Frage. Vergleichen wir diese noch in der Performance, so ist die Variante des „Atomic Subsystems" die beste.

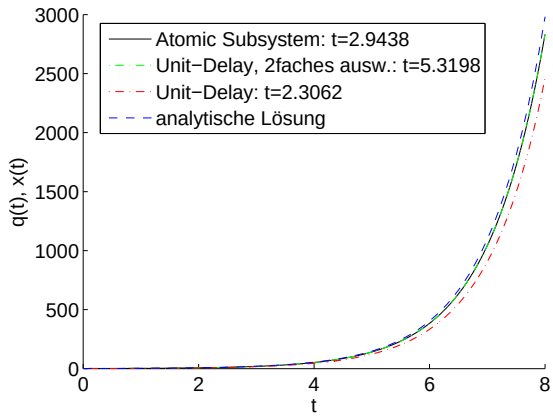

Abbildung 4.3: Gegenüberstellung der Varianten

4.3.2 Umsetzung mit und ohne eingebetteter Matlabfunktion

Im Hinblick auf die Performance in Simulink erscheint es besser zu sein, auf eingebettete Matlabfunktionen zu verzichten.
In Abb. 4.4 sehen wir die Simulation des Stabpendels (siehe auch Kapitel 5.1.1), für welche der QSS-Integrator ohne einer eingebetteten Matlabfunktion um 18% schneller arbeitet als jener, der auf diese zurückgreift. Für beide Integratoren wird das gleiche Quantum $\Delta Q = 0.01$ und die gleiche Hysteresebreite $\epsilon = 10^{-4}$ verwendet.

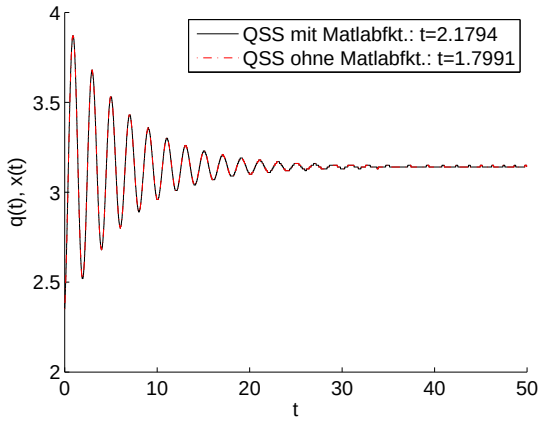

Abbildung 4.4: Stabpendel: Performancegewinn durch Verzicht auf Matlabfunktionen

4.3.3 Unterscheidung zwischen S^{ext} und S^{int}

Wie in Abschnitt 4.2 dargestellt kennt das in Simulink umgesetzte QSS-Verfahren keine Unterscheidung in den Berechnungen zwischen Eingangs- und Ausgangsereignissen. Für jede Entität, sowohl für S^{ext} als auch für S^{int}, wird x_n und σ_n auf gleichem Weg berechnet. Wollte man in der Theorie Auswertungen einsparen, müsste man im Falle eines Ausgangsereignisses die Attribute für S_n^{int} einfacher berechnen (ähnlich wie es auch im DEVS-Modell mit den Funktionen δ_{ext} und δ_{int} ist). Wurde eine Entität S_n^{int} erzeugt, so wissen wir:

$$x_n = \begin{cases} x_0 & \text{falls } n = 0 \\ q_n & \text{falls } n > 0 \quad \wedge \quad d_{x,n-1} > 0 \\ q_n + \Delta Q - \epsilon & \text{falls } n > 0 \quad \wedge \quad d_{x,n-1} < 0 \end{cases} \tag{4.3a}$$

$$\sigma_n = \begin{cases} 0 & \text{falls } n = 0 \\ \frac{\Delta Q}{|d_{x,n}|} & \text{falls } n > 0 \quad \wedge \quad sgn(d_{x,n-1}) = sgn(d_{x,n}) \\ \frac{\epsilon}{|d_{x,n}|} & \text{falls } n > 0 \quad \wedge \quad sgn(d_{x,n-1}) \neq sgn(d_{x,n}) \neq 0 \\ \infty & \text{falls } n > 0 \quad \wedge \quad d_{x,n} = 0 \end{cases} \tag{4.3b}$$

Zusätzlich lässt sich in der Berechnung von x_n und σ_n jeweils noch ein Fall einsparen, falls $\epsilon = \Delta Q$ gewählt wird.
Im Rahmen der Diplomarbeit wurde auch diese Methode, in der zwischen S^{ext} und S^{int} unterschieden wird, implementiert. Obwohl die Umsetzung einige Auswertungen einspart, ist die Rechenzeit durch die größere Anzahl von benötigten Blöcken in den meisten Simulationen gegenüber der Implementierung aus Abschnitt 4.2, in dem nicht zwischen S^{ext} und S^{int} unterschieden wird, sogar gestiegen. Aus diesem Grund wurde nicht zwischen S^{ext} und S^{int} unterschieden.

Kapitel 5

Beispiele

In diesem Kapitel beschäftigen wir uns mit verschiedenen Modellen, welche wir mit dem QSS-Verfahren simulieren. Anhand der ausgewählten Beispiele wollen wir eine bessere Vorstellung über das Verhalten des Verfahrens, einschließlich dessen Stärken und Schwächen bekommen. Als Vergleich dienen auch zeitorientierte Löser. Die dafür gewählten Modelle werden in zwei Kategorien eingeteilt, Modelle mit glatter Lösung und Modelle mit Diskontinuitäten.

5.1 Modelle mit glatter Lösung

5.1.1 Stabpendel

Hier betrachten wir ein Modell des Stabpendels [7]. Der eingeschlossene Winkel zwischen Stab und Aufhängepunkt wird als φ bezeichnet. Die Beschleunigung dieses Pendels ist durch folgende Differentialgleichung definiert:

$$\ddot{\varphi}(t) = \frac{g}{l} \cdot sin(\varphi(t)) - \frac{k}{m} \cdot \dot{\varphi}(t)$$

Der zweite Summand beschreibt die Dämpfung, in dem k für den Dämpfungsfaktor steht und m für Masse. Weiters steht g für die Erdgravitation und l für die Länge des Stabes. Für dieses Modell werden wir zukünftig l und m immer den Wert 1 zuweisen und den Dämpfungsfaktor $k = 0.3$ festlegen.
Für die Simulation betrachten wir das Modell mit den Anfangswerten $\varphi(0) = \frac{3}{4} \cdot \pi$ und $\dot{\varphi}(0) = 0$.
Die Lösungsparameter des QSS-Verfahrens wählen wir unterschiedlich. Für den ersten

Integrator, welcher die Beschleunigung zur Geschwindigkeit integriert ($\ddot{\varphi} \to \dot{\varphi}$), verwenden wir ein Quantum von $\Delta Q_1 = 0.1$ und für den zweiten ($\dot{\varphi} \to \varphi$) $\Delta Q_2 = 0.01$. Als Hysteresebreite legen wir für beide Intergratoren $\epsilon = 10^{-6}$ fest.
Als Gegenüberstellung dient die numerische Lösung von zwei zeitorientierten Verfahren, das explizite Eulerverfahren mit einer Schrittweite $h = 0.01$ und das RK-Verfahren von Dormand und Prince (kurz: RK45) mit einer relativen Toleranz von 10^{-6}.

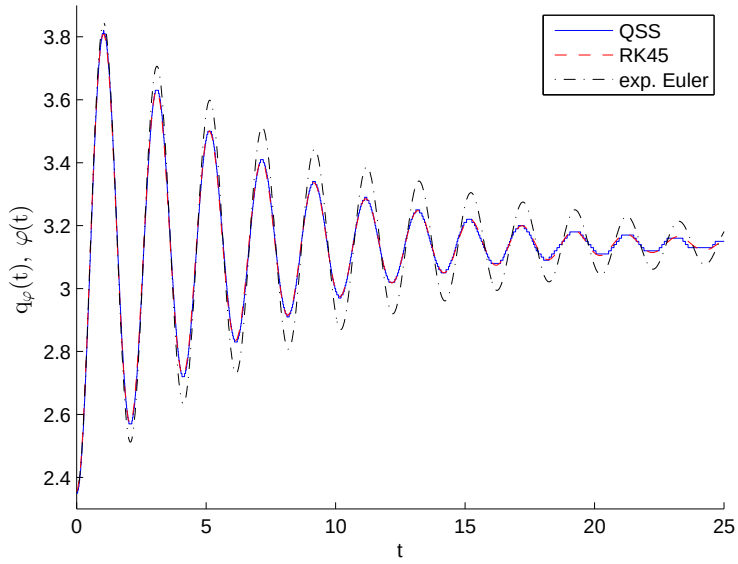

Abbildung 5.1: numerische Lösungen des Stabpendels

Abb. 5.1 stellt die verschiedenen numerischen Lösungen der Differentialgleichung dar. Vergleicht man die Anzahl der Zustandsänderungen des QSS-Systems bzw. die Anzahl der (verwendeten) Gitterpunkte der zeitorientierten Verfahren in den ersten 10 Sekunden (Tabelle 5.1), so sieht man, dass das RK45-Verfahren deutlich effizienter arbeitet. Es sei erwähnt, dass in diesem Modell ein Zustandswechsel des QSS-Systems einer Auswertung der stationären Funktion entspricht.

	Anzahl der Zustandswechsel	
	Integrator ($\ddot{\varphi} \to \dot{\varphi}$)	Integrator ($\dot{\varphi} \to \varphi$)
QSS	247	801
	Anzahl der (verwendeten) Gitterpunkte	
RK45	116	
e. Euler	1000	

Tabelle 5.1: Modell Anzahl der Gitterpunkte bzw. Zustandswechsel im Intervall $[0, 10]$

In Abb. 5.2 sehen wir den weiteren Verlauf der Lösung. Es ist zu erkennen, dass der

Fehler des QSS-Verfahrens durch das Quantum begrenzt wird und der Zustand bereits sehr bald zu oszillieren beginnt.

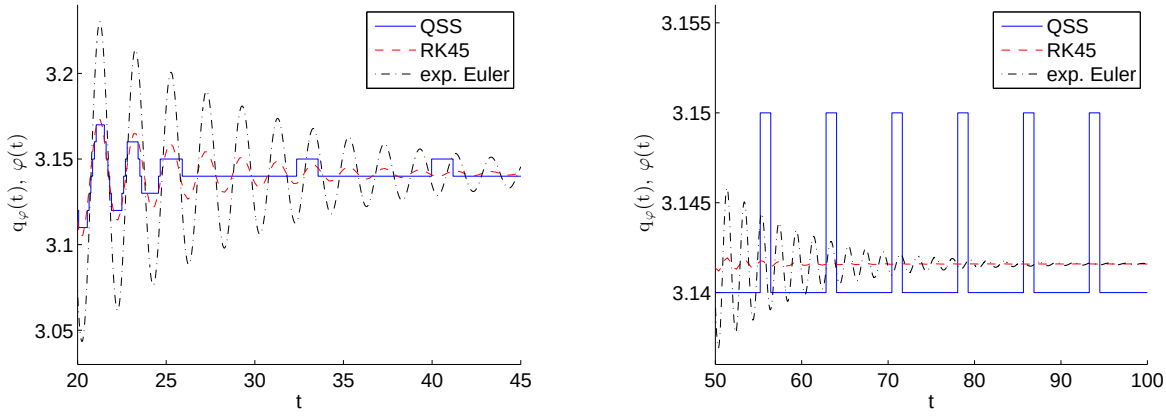

Abbildung 5.2: numerische Lösungen des Stabpendels, links ist der Ausschnitt $[20, 45]$ und rechts ist der Ausschnitt $[50, 100]$ zu sehen

Auch wenn im Gegensatz zu den anderen numerischen Lösungen die Konvergenz des QSS-Verfahrens gegen die Ruhelage nicht gegeben ist (aber höchstens um ΔQ abweicht), nimmt die Anzahl der Auswertungen bei der Oszillation deutlich ab. Einen Eindruck davon gibt die Tabelle 5.2.

	Anzahl der (verwendeten) Auswertungen	
	Integrator $(\ddot{\varphi} \to \dot{\varphi})$	Integrator $(\dot{\varphi} \to \varphi)$
QSS	4	2
RK45	14	
e. Euler	1000	

Tabelle 5.2: Anzahl der (verwendeten) Auswertungen im Intervall $t \in [90, 100]$

5.1.2 Sinusfunktion

In diesem Abschnitt wollen wir nun eine Sinusschwingung mit dem QSS-Verfahren simulieren. Dazu betrachten wir folgendes System:

$$\ddot{x}(t) = -a^2 \cdot x(t) \tag{5.1}$$

Die Lösung zu den Anfangswerten $x(0) = 0$ und $\dot{x}(0) = a$ ist die Sinusfunktion $x(t) = sin(a \cdot t)$. Die Konstante a beschreibt somit die Kreisfrequenz, welche hier immer den Wert $2 \cdot \pi$ annimmt. In Abb. 5.3 sehen wir das Simulinkmodell.

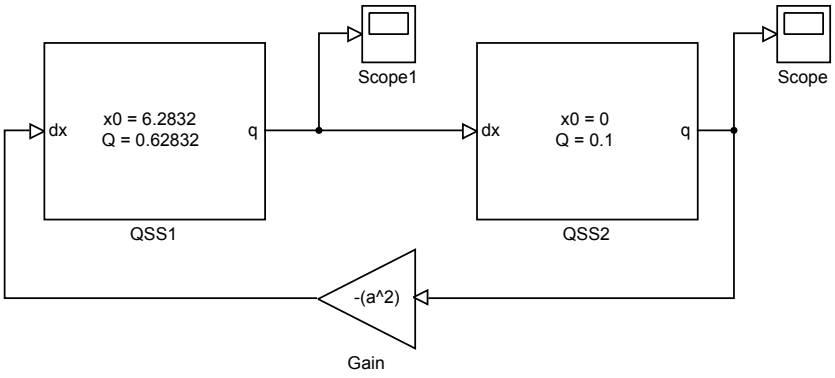

Abbildung 5.3: Simulinkmodel der Sinusfunktion

Um die Anzahl der Zustandswechsel auf beide Integratoren gleichmäßig zu verteilen, wählen wir für den ersten Integrator ($\ddot{x} \to \dot{x}$, in Abb. 5.3 als „QSS1“ bezeichnet) $\Delta Q_1 = 0.1$ und für den zweiten ($\dot{x} \to x$, „QSS2“) $\Delta Q_2 = 0.1 \cdot a$. Die Hysteresebreite ϵ wird in beiden Fällen gleich wie das Quantum gewählt.
In Abb. 5.4 zeigt die Lösung des QSS-Systems bei einer Simulationszeit von $2\,s$.

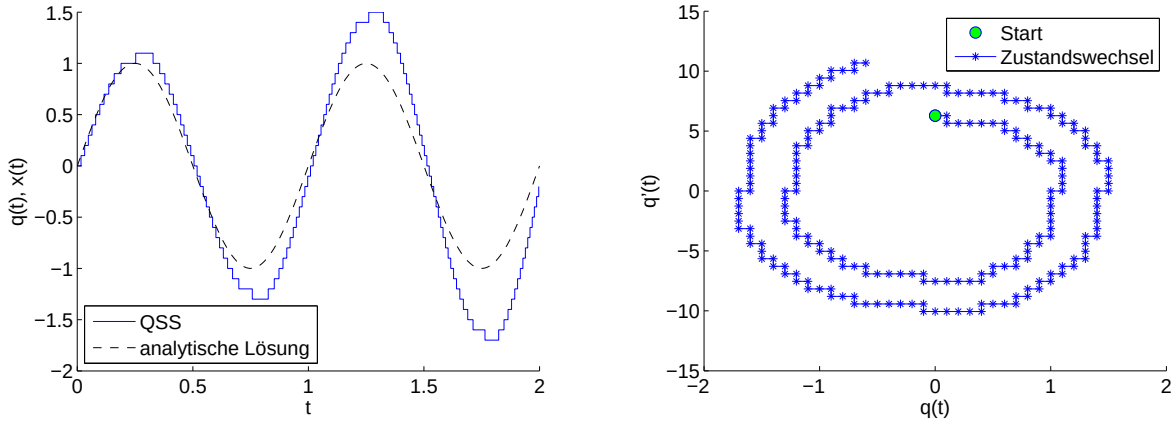

Abbildung 5.4: Sinusschwingung mit wachsender Amplitude beim QSS-Verfahren, das rechte Bild zeigt das Phasenportait mit allen Zustandswechseln

Die Amplitude der Lösung des QSS-Systems wächst sehr schnell. Dies ist die Konsequenz daraus, dass dieses Verfahren kein symmetrischer Algorithmus ist. Cellier, Kofman, Migoni, Bortolotto [8] weisen auf dieses Problem hin. Sie entwickelten zusätzlich eine zentralisierte Variante CQSS (englisch: centered QSS), welche eine Zusammensetzung zweier dezentralisierter Verfahren, QSS und BQSS, ist und mit solchen Problemen gut umgehen kann.

Man kann trotzdem mit dem klassischen QSS-Verfahren eine gute Lösung erhalten, wenn man die Hysteresebreite kleiner wählt. Abb. 5.5 zeigt die deutlich bessere Lösung durch eine 10^6-fache Minimierung von ϵ.

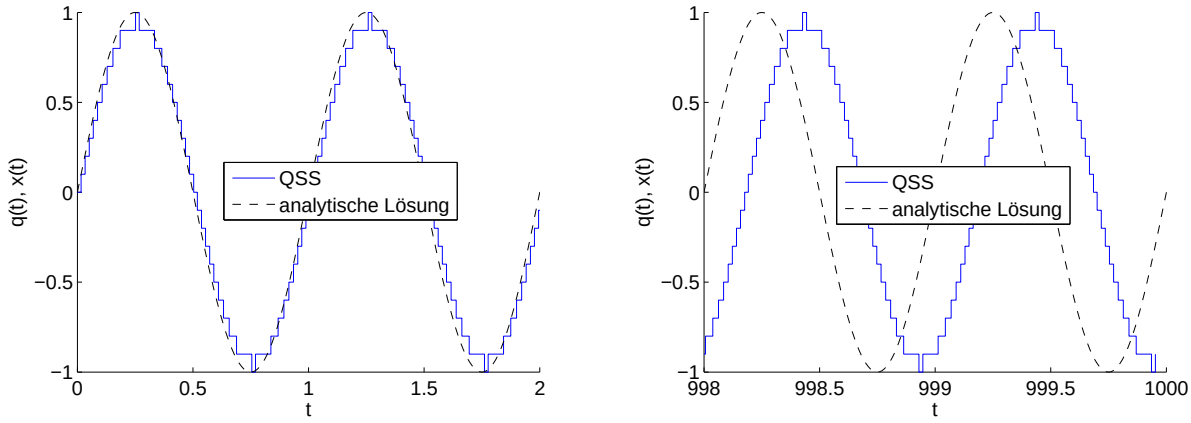

Abbildung 5.5: Sinusschwingung, links der Ausschnitt $[0, 2]$, rechts der Ausschnitt $[998, 1000]$

Sogar bei einer Simulationsdauer von 1000 s kommt es höchstens zu einer Verzögerung der Schwingung, aber zu keinem Zuwachs der Amplitude. Dies sehen wir auch im Phasenportait der Lösung, Abb. 5.6 (Anmerkung: den rot markierten Zustand nimmt das System nur einmal an).

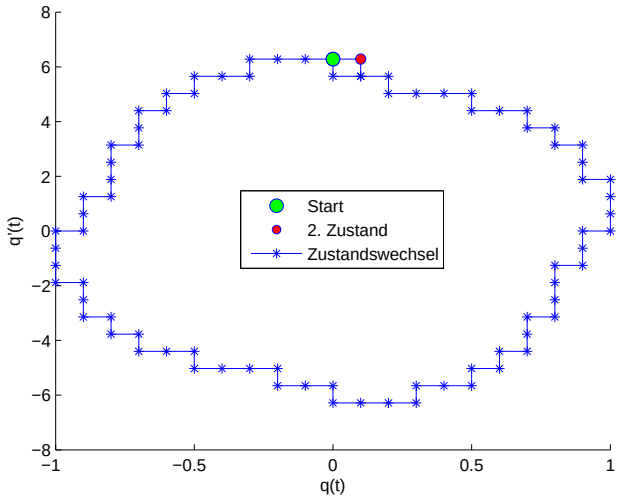

Abbildung 5.6: Phasenportait der Sinusschwinung über eine Simulationszeit von 1000 s

5.2 Modelle mit Diskontinuitäten

Wir wissen bereits, dass das QSS-Verfahren Diskontinuitäten auf direktem Weg erkennt. Für konventionelle Verfahren, die zeitorientiert arbeiten, trifft dies nicht zu. Um größere Fehler einzuschränken, führen diese zusätzliche Berechnungen durch, um den Zeitpunkt der Diskontinuität zu lokalisieren (siehe François E. Cellier und Ernesto Kofman [6]). Bei einer großen Anzahl von Diskontinuitäten sind zeitorientierte Verfahren nicht mehr effizient genug, abgesehen vom Ressourcenverbrauch gibt es weitere Aspekte, welche das QSS-Verfahren interessant machen.

Im nächste Abschnitt betrachten wir elektronische Schaltungen und wollen uns näher mit Diskontinuitäten beschäftigen, welche ein Schalter erzeugt.

5.2.1 Abwärtswandler

Ein Abwärtswandler (englisch: buck converter) ist ein Gleichspannungswandler, der die Eingangsspannung auf eine gewünschte Ausgangsspannung reduziert. Wir betrachten hier eine einfache Form des Abwärtswandlers. Die Aufgabe dieser Schaltung ist, die am Eingang anliegende Gleichspannung U_0 mit Hilfe von Schaltvorgängen und eines festgelegten Energiespeichers in Form einer Induktivität an eine gewünschten Ausgangsspannung u_R anzupassen, siehe Abb. 5.7.

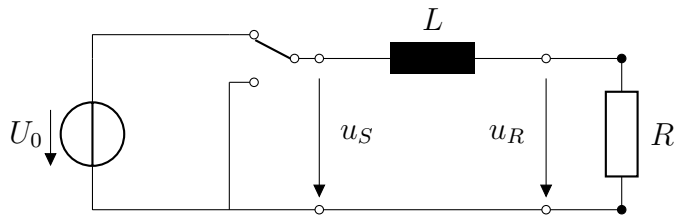

Abbildung 5.7: Schaltung eines einfachen Abwärtswandlers (Modell A)

Dieses Modell wird durch eine Differentialgleichung erster Ordnung beschrieben. Sei nun u_L die Spannung an der Induktivität, so wissen wir u.a. von der Maschenregel

$$L \cdot \frac{di}{dt} = u_L = u_S - u_R = u_S - i \cdot R.$$

Die explizite Darstellung der Differentialgleichung lautet nun

$$\frac{di}{dt} = \frac{u_S - i \cdot R}{L}.$$

Zumeist werden die Schaltzeiten im Vorhinein bestimmt. Hier wollen wir aber einen geregelten Abwärtswandler betrachten. Dazu vergleichen wir die tatsächliche Ausgangsspannung u_R mit einer Sollspannung u_{REF} und schalten U_0 entweder weg oder dazu. Die

Spannung u_S in Abb. 5.7 beschreibt dabei die Schalterstellungen:

$$u_S = \begin{cases} U_0 & \text{falls } u_R < u_{REF} \\ 0 & \text{sonst} \end{cases}$$

Beispiel 5.2.1 (geregelter Abwärtswandler (Modell A))**.** Eine Spannungsquelle $U_0 = 10\,V$, eine Induktivität $L = 0.1\,H$ und ein Widerstand $R = 100\,\Omega$ sind gegeben. Der Anfangswert des Stroms beträgt $i(0) = 0\,A$.
Wir wollen dieses Modell nun mit dem QSS-Verfahren simulieren. Haben wir das Ziel, bei einer Spannungsabweichung von z.B. $\Delta u_R = 0.1\,V$ zu schalten, so werden wir feststellen, dass dies mit dem QSS-Verfahren sehr gut funktioniert. Dafür müssen wir lediglich den richtigen Lösungsparameter für den Strom wählen:

$$\Delta Q = \epsilon = \frac{\Delta u_R}{R} = \frac{0.1}{100} = 10^{-3}$$

Für den tatsächlichen Zustand ist es wichtig, dass ϵ höchstens als 10^{-3} gewählt wird, da ϵ für die Verzögerung verantwortlich ist. Zu klein sollte sie allerdings nicht gewählt werden, da sonst eine zu hohe Schaltfrequenz erhalten wird. Die Abb. 5.8 zeigt die variierende Sollspannung u_{REF} und die zugehörige Lösung des QSS-Systems. In der Simulationszeit von $0.02\,s$ kommt es dabei zu 890 Zustandsänderungen, davon wird 770-mal die Schalterposition geändert. Der Vorteil dieses Verfahrens liegt darin, dass immer der richtige Schaltzeitpunkt gefunden wird und man dadurch ein gleichmäßiges Schaltmuster bekommt. Dass dies bei einem zeitorientierten Verfahren, wie z.B. dem expl. Eulerverfahren oder dem RK45-Verfahren nicht möglich ist, zeigt Abb. 5.8.

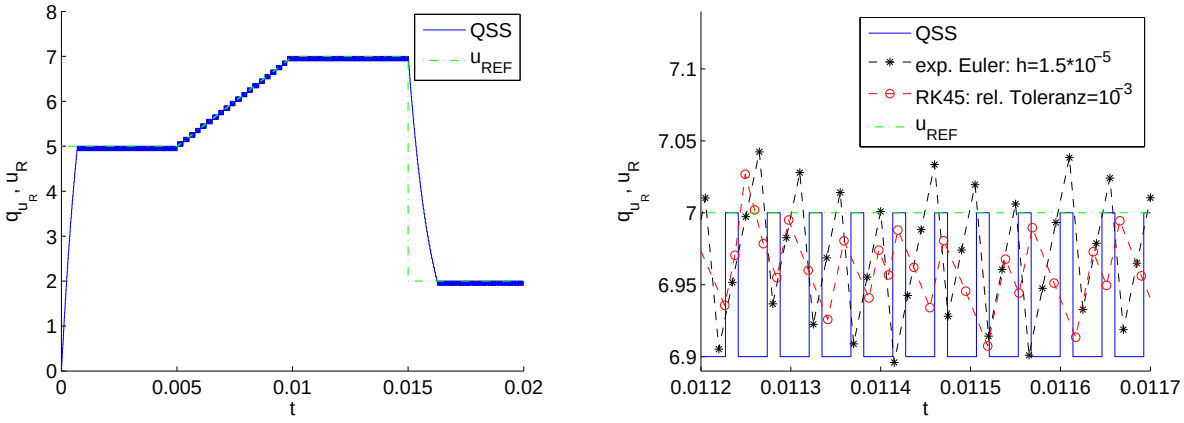

Abbildung 5.8: Ausgangsspannung des Abwärtswandlers, rechts ein Vergleich mit zwei zeitorientierten Verfahren

In Abb. 5.9 sehen wir einen Ausschnitt der Lösung für $t \in [0, 10^{-3}]$ und die Schaltzeitpunkte. Es gilt: Bei konstanter Spannung u_R muss geschaltet werden.

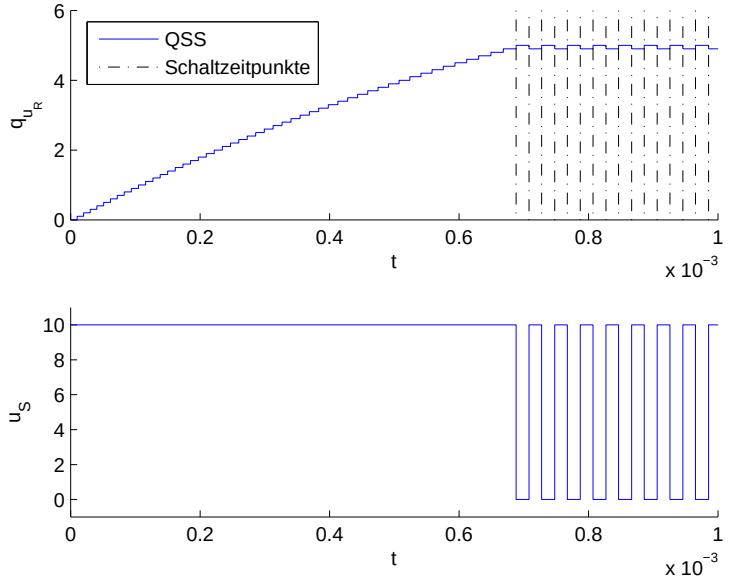

Abbildung 5.9: Spannungen des Abwärtswandlers für $t \in [0, 10^{-3}]$

Wir wollen das Modell wie in Abb. 5.7 ein wenig verändern, indem wir zusätzlich einen Kondensator C parallel zum Widerstand R schalten. Dieser hat lediglich die Aufgabe, die Spannung am Ausgang zusätzlich zu glätten. Abb. 5.10 zeigt das neue Modell.

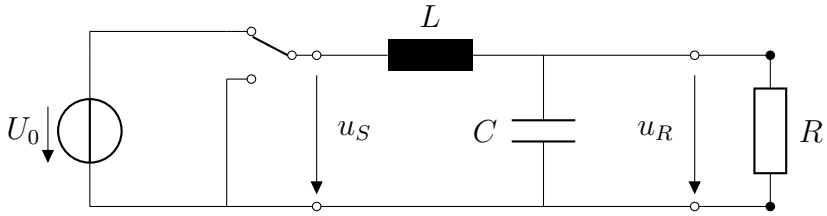

Abbildung 5.10: Schaltung eines Abwärtswandlers (Modell B)

Nun gilt folgender Zusammenhang:

$$L \cdot \frac{di}{dt} = u_S - u_R$$
$$C \cdot \frac{du_R}{dt} = i - i_R = i - \frac{u_R}{R}$$

Daraus erhalten wir ein Differentialgleichungssystem:

$$\frac{d}{dt}\begin{pmatrix} i \\ u_R \end{pmatrix} = \begin{pmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{R \cdot C} \end{pmatrix} \cdot \begin{pmatrix} i \\ u_R \end{pmatrix} + \begin{pmatrix} \frac{u_S}{L} \\ 0 \end{pmatrix}$$

Eine zusätzliche Änderung im Modell liegt im Verzicht auf die Regelung. Das heißt, wir geben die Schaltzeit vor, die im Vorhinein berechnet wurde. Das zugehörige Simulinkmodell hat folgende Gestalt:

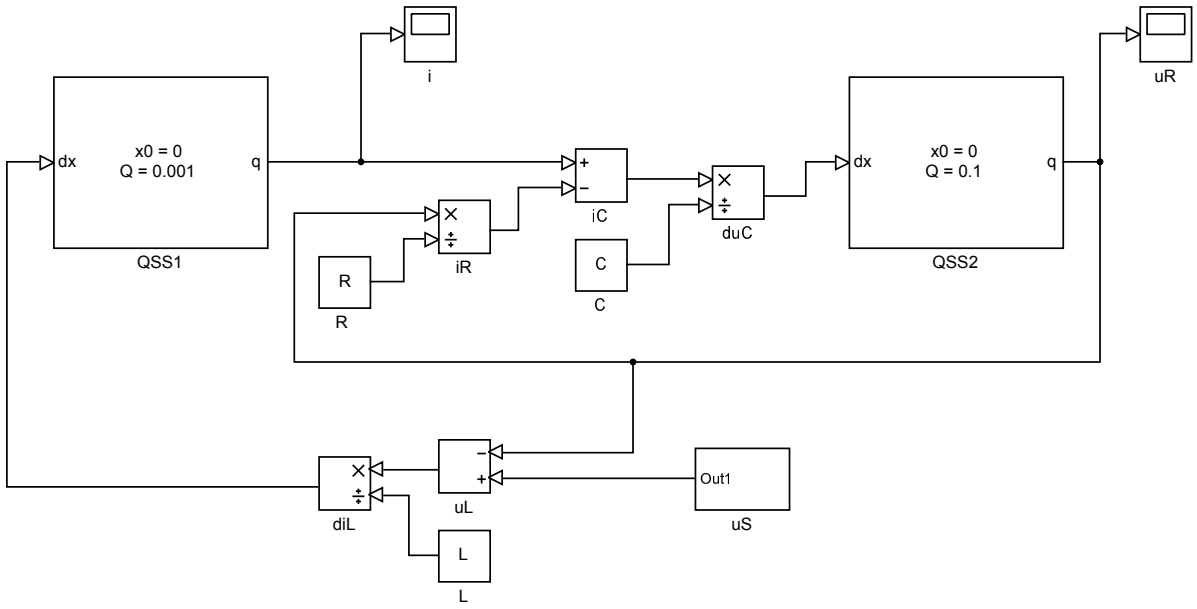

Abbildung 5.11: Simulinkmodell des Abwärtswandlers (Modell B)

Beispiel 5.2.2 (gesteuerter Abwärtswandler (Modell B))**.** Für dieses Beispiel verwenden wir die gleichen Komponenten wie im Beispiel 5.2.1. Der zusätzliche Kondensator hat den Wert $C = 10^{-5}\,F$. Der Schalter, welcher diesmal nicht geregelt wird, schaltet abwechselnd für die Dauer $t_{EIN} \approx 3,226 \cdot 10^{-3}s$ die Spannung U_0 dazu und für $t_{AUS} \approx 1,429 \cdot 10^{-3}s$ weg. Die Zeiten t_{EIN} und t_{AUS} wurden aus der Lösung des vorigen Beispiels, bei einer konstanten Ausgangsspannung $u_R = 7\,V$ entnommen. Als Lösungsparameter wählen wir für den ersten Integrator ($\frac{d}{dt}i \to i$, in Abb. 5.11 wird dieser „QSS1“ genannt) $\Delta Q_1 = 0.001$ und für den zweiten ($\dot{u}_R \to u_R$) $\Delta Q_2 = 0.1$. Die Hysteresebreite ϵ wird in beiden Fällen wieder gleich dem Quantum gewählt. In Abb. 5.12 sehen wir den Verlauf der Ausgangsspannung und des Gesamtstroms.

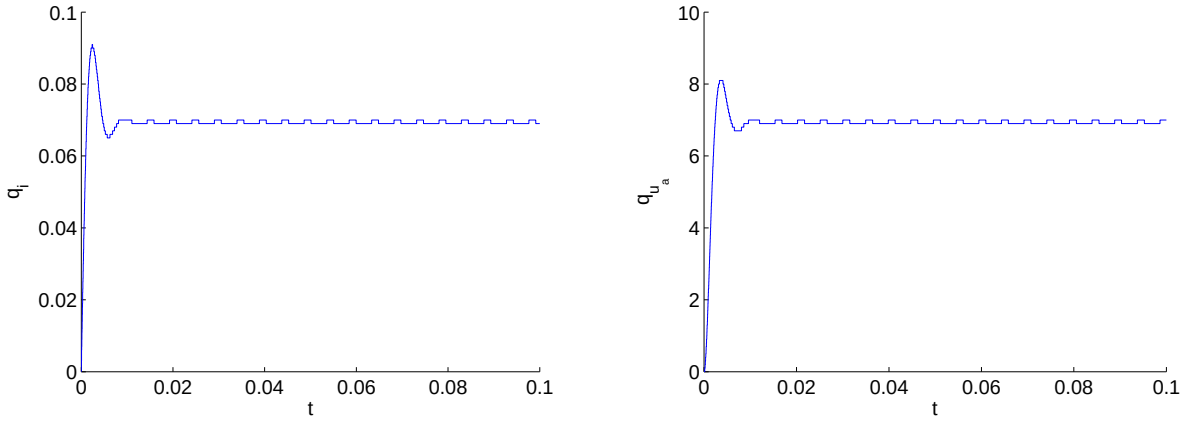

Abbildung 5.12: rechts ist der Gesamtstrom abgebildet, links die Ausgangsspannung

Abb. 5.13 zeigt das Verhalten einer zeitorientierten Methode, konkret jenes des RK45-Verfahrens. Es zeigt sich, dass Simulink diese Verfahren insofern optimiert, indem Schaltzeitpunkte für die Schrittweitensteuerung berücksichtigt werden. Das RK45-Verfahren hat mit einer relativen Toleranz von 10^{-3} deshalb gleich viele Auswertungen wie Schaltzeitpunkte.
Betrachten wir die Lösung des QSS-Verfahrens, so zeigen die Abb. 5.12 und 5.13 eine sehr langsam oszillierende Lösung des Stromes und der Spannung. Es ist bemerkenswert, dass es trotz vieler Schaltvorgänge zu wenigen Zustandsänderungen kommt. Im Zeitintervall $[0.09, 0.1]$ kommt der Schalter auf 430 Schaltvorgänge. Für den ersten Integrator „QSS1“ bedeutet dies, dass mindestens 430 Eingangsereignisse anliegen. In Abb. 5.12 sehen wir, dass er trotzdem gerade einmal 4 Zustandsänderungen hat, dass zur Folge hat, dass am zweiten Integrator „QSS2“ nur 4 Eingangsereignisse gezählt werden. Viele Auswertungen werden damit eingespart. Durch die asynchrone Arbeitsweise des QSS-Verfahrens werden somit rechenintensive Simulationen (wie bei der Spannung u_S) vereinfacht.
Diese Fähigkeit haben zeitorientierte Verfahren nicht.

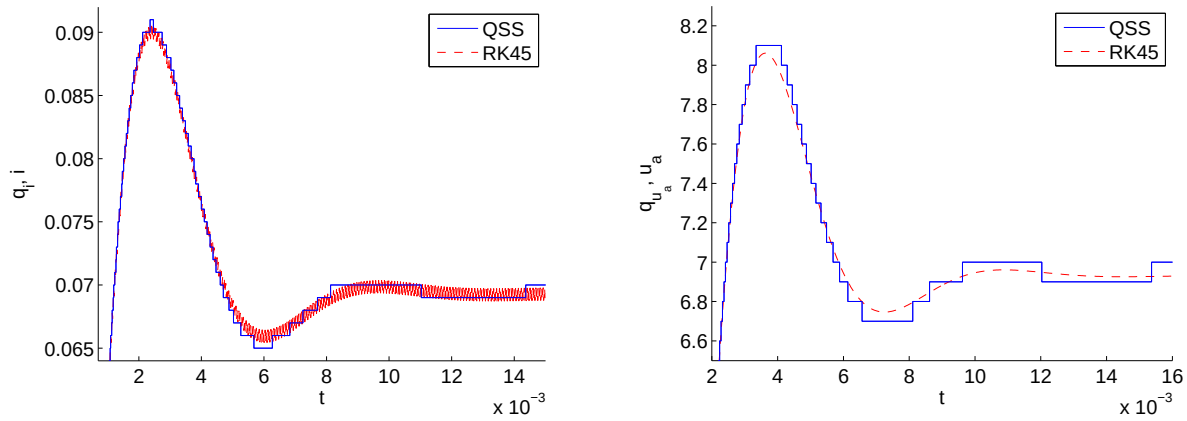

Abbildung 5.13: gesteuerter Abwärtswandler, Ausschnitt aus den Lösungen der unterschiedlichen Verfahren

Die Methode einen Rechteckpuls (wie im obigen Bsp. u_S) bei konstanter Schaltfrequenz und einem variierenden Tastverhältnis $\left(\frac{t_{EIN}}{t_{EIN}+t_{AUS}}\right)$ zu modulieren wird auch *Pulsweitenmodulation* (PWM) genannt. Die folgende Abb. zeigt, wie man 80 % einer ursprünglichen Gleichspannung mithilfe eines Sägezahnsignals von $1\,kHz$, welche gleichzeitig der Schaltfrequenz bestimmt, erreicht:

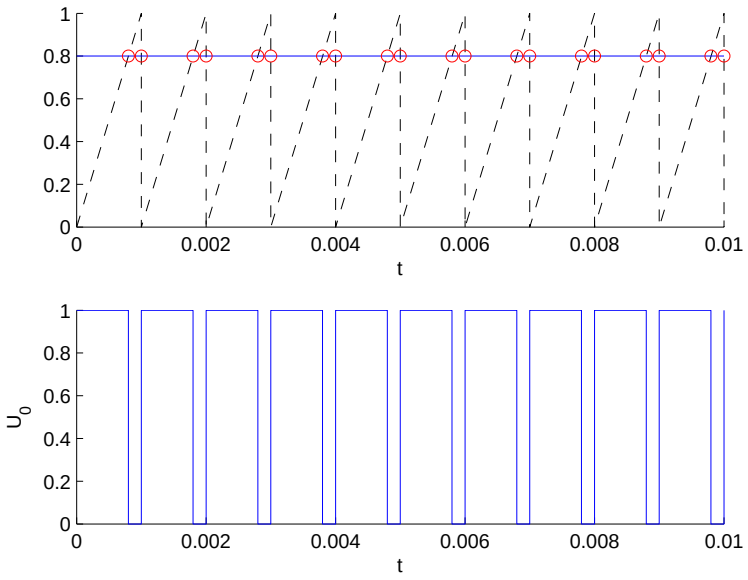

Abbildung 5.14: Pulsweitenmodulation, Abtastung eines Sägezahnsignals von $1\,kHz$

5.2.2 Vierquadrantensteller

Ein Modell aus der Antriebstechnik weist einige Ähnlichkeiten mit den zwei vorigen Beispielen auf. Der Vierquadrantensteller besteht aus vier Schaltern oder Transistoren und wird in der Praxis oft für die Ansteuerung eines Gleichstrommotors verwendet, siehe dazu Abb. 5.15.

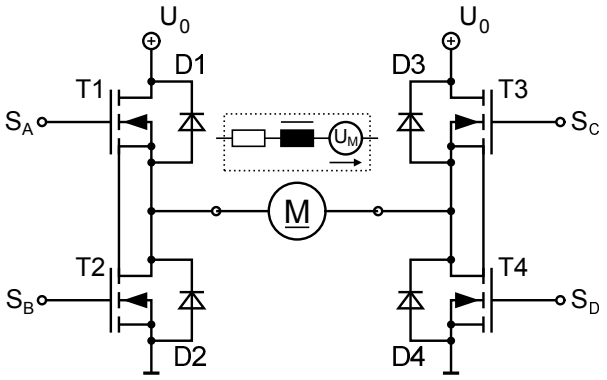

Abbildung 5.15: Vierquadrantensteller [9]

Diese vier Schalter werden mit einem PWM-Signal betrieben, sodass verschiedene Betriebsmodi wie Beschleunigung, Bremsvorgang (inklusive Rückspeisung, Notbremsung) oder Leerlauf eines Gleichstrommotors möglich sind, und das alles in Vor- und Rückwärtslauf (nachzulesen in Rolf Fischer [10]). Für unser Modell werden wir nur den Beschleunigungs- und Bremsvorgang in Vorwärtsrichtung betrachten, d.h. der Schalter „S_C“ in Abb. 5.15 wird hier nicht angesteuert.
Unser Modell [11] eines Gleichstrommotors besteht aus einem elektrotechnischen

$$u_S = u_{EMK} + u_R + u_L \tag{5.2a}$$

und einem mechanischen Teil

$$I \cdot \ddot{\omega} = M - k \cdot \dot{\omega}. \tag{5.2b}$$

In der ersten Gleichung (5.2a) beschreibt u_S die Spannung der Versorgung, welche abhängig von den Schalterstellungen entweder $100\,V$ oder $0\,V$ beträgt. u_{EMK} ist jene Spannung, die der Motor erzeugt. Sie ist somit abhängig von der Winkelgeschwindigkeit $\dot{\omega}$, mit der der Motor in Betrieb ist. Für unser Modell verwenden wir $u_{EMK} = 0.8 \cdot \dot{\omega}$.
u_R ist der Spannungsabfall beim ohm'schen Widerstand $R = 1\,\Omega$ und u_L die Spannung an der Induktivität $L = 0.5\,H$.
In der zweiten Gleichung (5.2b) beschreibt I das Masseträgheitsmoment, das wir $I = 1\,kg \cdot m^2$ setzen. Für das Drehmoment des Motors wählen wir $M = 0.8 \cdot i$ und für die Dämpfungskonstante des Motors $k = 0.01\,Nm \cdot s$.

Wir wollen dieses Modell nun simulieren. Der Motor soll aus dem Stillstand bis zu einer gewünschten Geschwindigkeit beschleunigen, danach im Leerlauf betrieben werden und zum Schluss abbremsen. Abb. 5.16 zeigt ein Signal, welches diesen Vorgang darstellt. Der

positive Anteil des Signals wird dabei von dem Sägezahnsignal abgetastet, welches die Schaltzeiten von S_A vorgibt. Der negative Anteil wird eigenständig für S_B abgetastet. Die Abtastungen haben dabei eine Frequenz von $1\,kHz$.

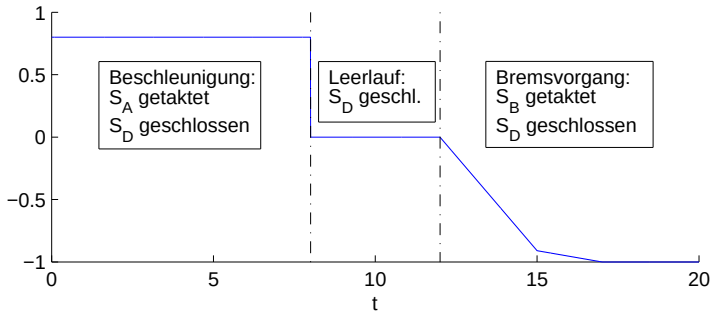

Abbildung 5.16: Referenzspannung

Für das QSS-Verfahren verwenden wir als Lösungsparameter $\Delta Q = \epsilon = 0.5$. Als Gegenüberstellung wählen wir wieder das RK45-Verfahren mit einer relativen Toleranz von 10^{-3}. Abb. 5.17 zeigt die numerischen Lösungen beider Verfahren. Wie auch im letzten Beispiel haben die QSS-Integratoren sehr wenige Zustandsänderungen. Der erste Integrator ($\ddot{\omega} \to \dot{\omega}$) weist insgesamt 397 und der zweite ($\frac{d}{dt}i \to i$) 394 Zustandsänderungen auf. Das RK45-Verfahren verwendet für diese Simulation hingegen 29082 Gitterpunkte.

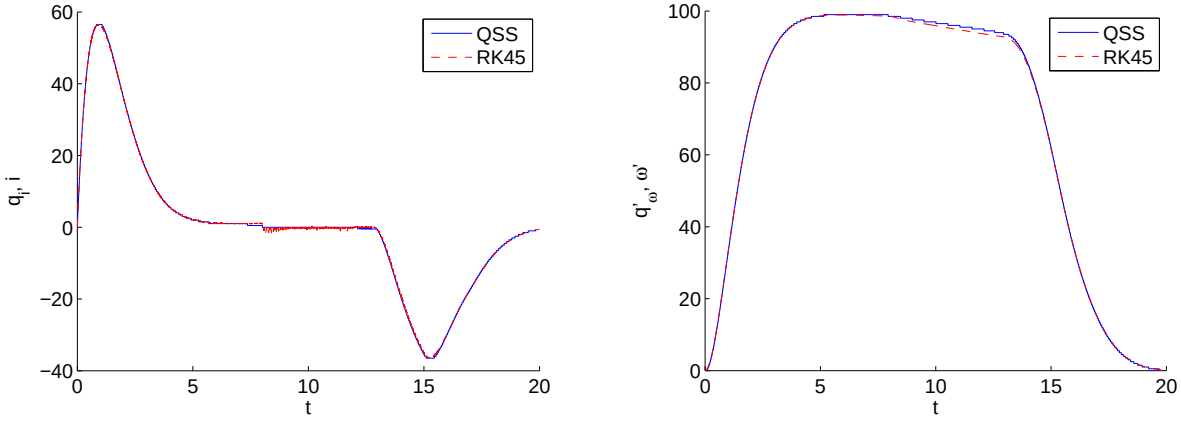

Abbildung 5.17: links ist der Gesamtstrom abgebildet, rechts die Drehzahl

Interessant ist auch das Verhalten der Verfahren im Intervall $t \in [8, 12]$, in dem der Motor im Leerlauf betrieben wird, siehe Abb. 5.18. In diesem Modell wird dafür nur der Schalter S_D geschlossen. Der Strom baut sich anschließend innerhalb kürzester Zeit ab, bis er $0\,A$ beträgt und auf den Motor nur mehr die Dämpfung wirkt. Das QSS-Verfahren setzt dies mustergültig um. In der Zeit $t \in [8, 12]$ hat der erste Intergrator ($\frac{d}{dt}i \to i$) 0 und der zweite ($\ddot{\omega} \to \dot{\omega}$) 6 Zustandsänderungen. Da in dieser Zeit auch

kein PWM-Signal anliegt, gibt es insgesamt nur 6 Zeitpunkte, zu denen Auswertungen gemacht werden. Das RK45-Verfahren, welches das Ereignis $i = 0\,A$ verpasst (der Strom wird fälschlicherweise über die Diode $D1$ zurückgespeist), muss hier mehr Auswertungen tätigen. Die Lösung des RK45-Verfahrens muss dadurch sogar im Leerlaufbetrieb auf 480 Gitterpunkte zurückgreifen.

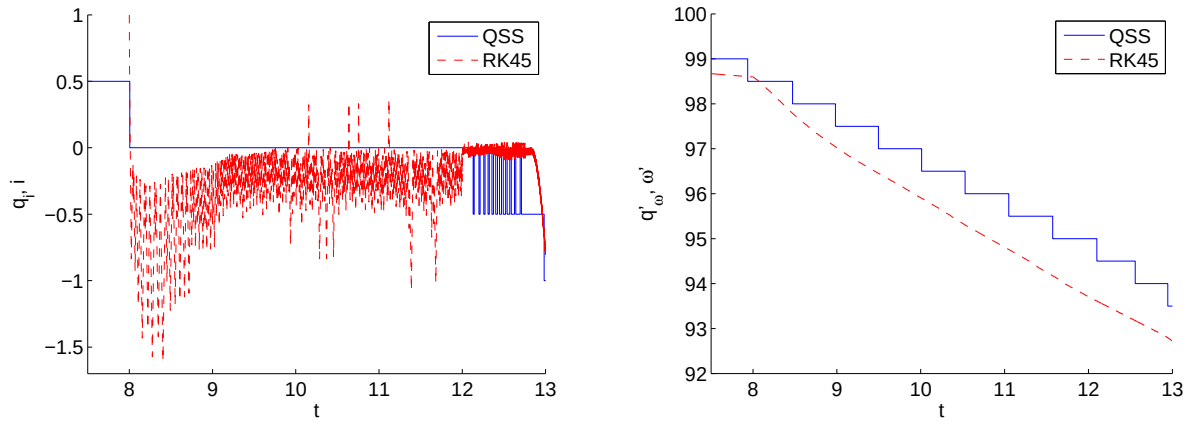

Abbildung 5.18: Ausschnitt der Lösung im Intervall $[8.5, 13]$, in welchem der Motor u.a. im Leerlauf betrieben wird

5.2.3 Fadenpendel mit Überschlag (mit kartesischen Koordinaten)

Wir wollen ein Fadenpendel mit Überschlag mit dem QSS-Verfahren simulieren. Dafür wird das Modell mit kartesischen Koordinaten betrachtet, in dem das Pendel zwei Zustände annehmen kann, die *Pendelphase* und die *Fallphase*. Als Übergang von dem einen in den anderen Zustand werden zwei Ereignisse benötigt, der *Übergang in die Fallphase* und der *Übergang in die Pendelphase*.

Zunächst wird die Pendelphase in der Kreisbahn betrachtet, welche einerseits durch die Zwangsbedingung (5.3b) und andererseits durch folgende Differentialgleichung beschrieben wird:

$$\begin{aligned} \ddot{x}(t) &= -k \cdot \dot{x}(t) \; + \; \lambda(t) \cdot x(t) \\ \ddot{y}(t) &= -k \cdot \dot{y}(t) \; + \; \lambda(t) \cdot y(t) \; - \; g \end{aligned} \tag{5.3a}$$

In der Vektorschreibweise ist diese

$$\ddot{\boldsymbol{x}}(t) = -k \cdot \dot{\boldsymbol{x}}(t) \; + \; \lambda(t) \cdot \boldsymbol{x}(t) \; - \; \begin{pmatrix} 0 \\ g \end{pmatrix},$$

wobei $\boldsymbol{x}(t)$ die Position, $\dot{\boldsymbol{x}}(t)$ die Geschwindigkeit und $\ddot{\boldsymbol{x}}(t)$ die Beschleunigung des Pendels beschreiben. Bei den zwei Konstanten steht g für die Erdgravitation und k für eine

Dämpfung ($k > 0$) bzw. Verstärkung ($k < 0$) der Geschwindigkeit.
Die Variable λ hat die Aufgabe, beide Komponenten der Differentialgleichung richtig zu koppeln, damit die Lösungskurve auf der Kreisbahn bleibt. Sei l die Fadenlänge des Pendels, welche in diesem Abschnitt immer den Wert 1 hat, dann können wir folgende Bedingung aufstellen:

$$x^2 + y^2 = l^2 \tag{5.3b}$$

Leiten wir zweimal nach t ab, so bekommen wir

$$\begin{aligned} \dot{x}x + \dot{y}y &= 0, \\ \ddot{x}x + \dot{x}^2 + \ddot{y}y + \dot{y}^2 &= 0. \end{aligned} \tag{5.3c}$$

Wir können jetzt in die Gleichung (5.3c) die Komponenten der Differentialgleichung (5.3a) für $\ddot{y}(t)$ bzw. $\ddot{x}(t)$ einsetzen. Die Gleichung, welche wir dadurch erhalten, können wir durch λ explizit darstellen und erhalten dadurch die fehlende Verknüpfungsbedingung:

$$\lambda = \frac{k(\dot{x}x + \dot{y}y) - (\dot{x}^2 + \dot{y}^2) + gy}{x^2 + y^2}$$

Es wird nun der Übergang in die Fallphase beschrieben. Dieses Ereignis tritt auf, falls das Pendel aufgrund zu geringer Geschwindigkeit keinen vollen Überschlag schafft und deshalb von der Kreisbahn abkommt. Um diesen Zeitpunkt zu erhalten werden die auf das Pendel wirkenden Kräfte beobachtet. Die Voraussetzung für den Verbleib in der Kreisbahn ist, dass die Zentrifugalkraft F_Z größer als die Radialkraft F_R ist

$$F_Z(t) \; > \; F_R(t). \tag{5.4a}$$

Diese Kräfte sind definiert durch

$$\begin{aligned} F_Z(t) &= m \cdot l \cdot \dot{\varphi}(t)^2, \\ F_R(t) &= m \cdot g \cdot cos(\varphi(t)), \end{aligned}$$

wobei $cos(\varphi) = \frac{y}{l}$ und $\dot{\varphi}$ die Winkelgeschwindigkeit ist. Da $\boldsymbol{x}$ normiert ist, lässt sich $\dot{\varphi}$ als Projektion der Geschwindigkeit auf die Tangente des Berührungspunkts der Kreisbahn ganz einfach mit dem Inneren Produkt darstellen:

$$\dot{\varphi} = \langle \dot{\boldsymbol{x}}, \boldsymbol{x}^{\perp} \rangle, \qquad \text{wobei } \boldsymbol{x}^{\perp} = \begin{pmatrix} y \\ -x \end{pmatrix}$$

Ist Bedingung (5.4a) nicht erfüllt, so nimmt das Pendel den zweiten möglichen Zustand, die Fallphase, an. Für diese wird folgende DGL verwendet:

$$\ddot{\boldsymbol{x}}(t) = -\begin{pmatrix} 0 \\ g \end{pmatrix}$$

Ist das Pendel in der Fallphase, so beschreibt das zweite Ereignis den Übergang in die

Pendelphase. Das ist jener Zeitpunkt, in dem das Pendel die ursprüngliche Kreisbahn erneut kreuzt. Dies ist erfüllt, falls der Abstand des Pendels zum Ursprung größer gleich der Fadenlänge l ist:

$$x(t)^2 + y(t)^2 \geq l^2$$

In diesem Ereignis wird der Integrator, der die Beschleunigung zur Geschwindigkeit integriert ($\ddot{\boldsymbol{x}} \rightarrow \dot{\boldsymbol{x}}$) zurückgesetzt, und das Pendel geht in die Pendelphase (5.3a) über. Den neuen Anfangswert gibt dabei der letzte Wert der Winkelgeschwindigkeit $\dot{\varphi}(t)$ vor. Das heißt, wir müssen $\dot{\varphi}(t)$ in die kartesischen Koordinaten zurücktransformieren. Es gilt nun folgender Zusammenhang zwischen $\dot{\varphi}(t)$ und $\dot{x}(t)$:

$$\begin{aligned} \frac{d}{dt}\left(\frac{x}{l}\right) &= \frac{d}{dt} sin(\varphi) \\ \frac{\dot{x}}{l} &= cos(\varphi) \cdot \dot{\varphi} = \frac{y}{l} \cdot \dot{\varphi} \\ \dot{x} &= y \cdot \dot{\varphi} \end{aligned}$$

Analog lässt sich auch $\dot{y}(t)$ berechnen, und wir erhalten dadurch den neuen Anfangswert:

$$\dot{\boldsymbol{x}}_{neu} = \begin{pmatrix} y \\ -x \end{pmatrix} \cdot \dot{\varphi} = \boldsymbol{x}^{\perp} \cdot \dot{\varphi}$$

Beispiel 5.2.3. In diesem Beispiel sehen wir das Verhalten des Modells. Als Anfangswerte und Lösungsparameter des QSS-Verfahrens wählen wir:

$$\boldsymbol{x}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \dot{\boldsymbol{x}}(0) = \begin{pmatrix} 0 \\ -30 \end{pmatrix}, \qquad \Delta Q = 10^{-3}, \qquad \epsilon = 5 \cdot 10^{-4}$$

Zusätzlich legen wir eine starke Dämpfung $k = 1.5$ fest, um eine nicht allzu lange Simulationsdauer zu haben.
Abb. 5.19 zeigt die Lösung dieses Systems. Im linken Bild sehen wir die Lösungskurve des Fadenpendels über die gesamte Simulationsdauer, rechts eine Gegenüberstellung der Position (x-Koordinate), der auf das Pendel wirkenden Kräfte und der Winkelgeschwindigkeit.

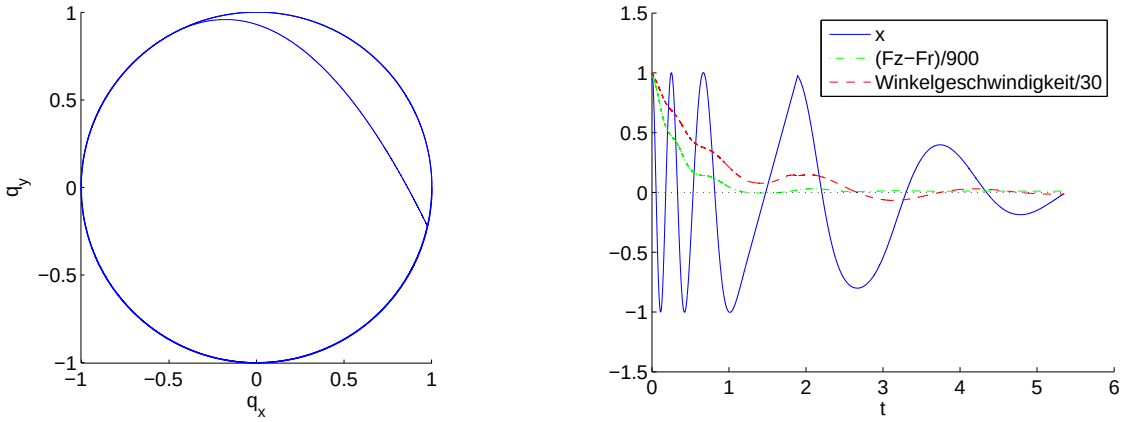

Abbildung 5.19: Lösung des Fadenpendels mit einem Fallzustand

Im nächsten Beispiel wollen wir genauer beobachten, wie sich das quantisierte System dieses Modells über einen längeren Zeitraum verhält.

Beispiel 5.2.4. Dazu betrachten wir das Modell ohne Dämpfung ($k = 0$) und mit den Anfangswerten

$$\boldsymbol{x}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{und} \qquad \dot{\boldsymbol{x}}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Als Quantum und Hysteresebreite wählen wir diesmal:

$$\Delta Q = 10^{-3}, \qquad \epsilon = 10^{-10}$$

Mit diesen Einstellungen schafft das Pendel weder einen Überschlag noch kommt es in den Fallzustand. Sie demonstrieren, dass das QSS-Verfahren beim Simulieren dieses Pendels keine Probleme hat. In Abb. 5.20 sehen wir die Lösung der Simulation über eine Dauer von 100 s. Obwohl das Pendel in dieser Zeit 42 Perioden hat, bleiben die Größen Position und Winkelgeschwindigkeit erhalten.

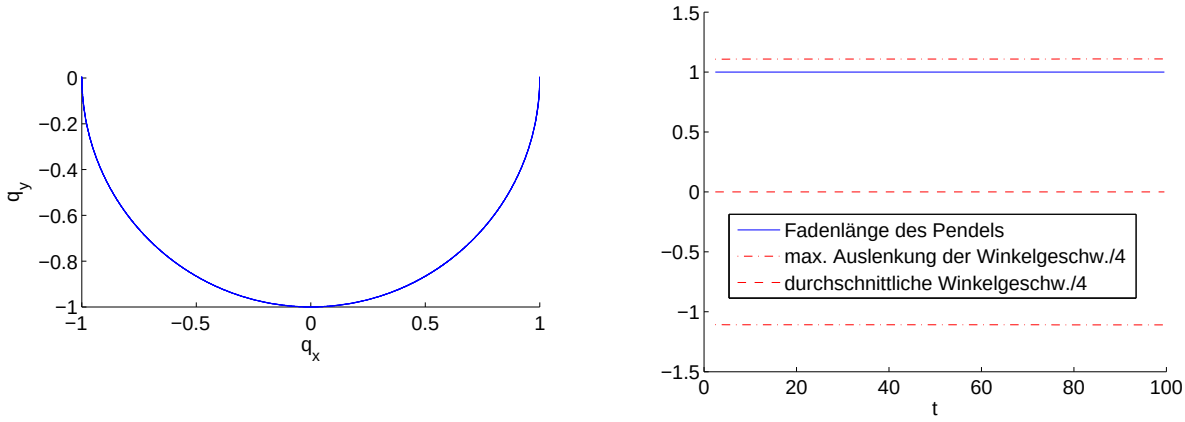

Abbildung 5.20: Lösung des Fadenpendels ohne Dämpfung und ohne Überschlag

Diese gute Lösung des QSS-Verfahrens erhält man nur in diesem Fall. Nehmen wir bei der gleichen Konfiguration eine Dämpfung von $k = 0.025$ an, so sehen wir in Abb. 5.21, dass die Fadenlänge nicht mehr erhalten bleibt.

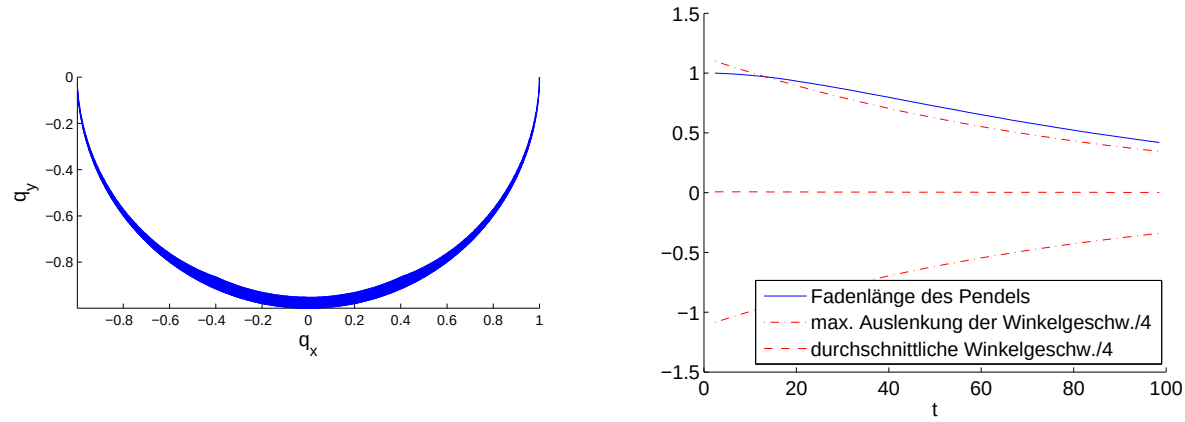

Abbildung 5.21: Lösung des Fadenpendels mit Dämpfung $k = 0.025$

Die Lösungen des QSS-Verfahrens neigen meistens dazu, in den Ursprung zu konvergieren. Wählt man die Hysteresebreite ϵ für das Verfahren nahe bei 0, so erhält man eine Lösungskurve, die sich zu Beginn umgekehrt verhält und sich ins Äußere orientiert. Das folgende Beispiel wird uns dies demonstrieren.

Beispiel 5.2.5. Wir sehen uns ein System mit folgenden Parametern an:

$$\boldsymbol{x}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \dot{\boldsymbol{x}}(0) = \begin{pmatrix} 0 \\ -6 \end{pmatrix}, \qquad \Delta Q = 10^{-3}, \qquad \epsilon = 25 \cdot 10^{-5}$$

Wählen wir für die Dämpfung wieder $k = 0$, so liegt ein Pendel mit endlosen Überschlägen vor. Abb. 5.22 zeigt die Lösung bei einer Simulationsdauer von $100\,s$.

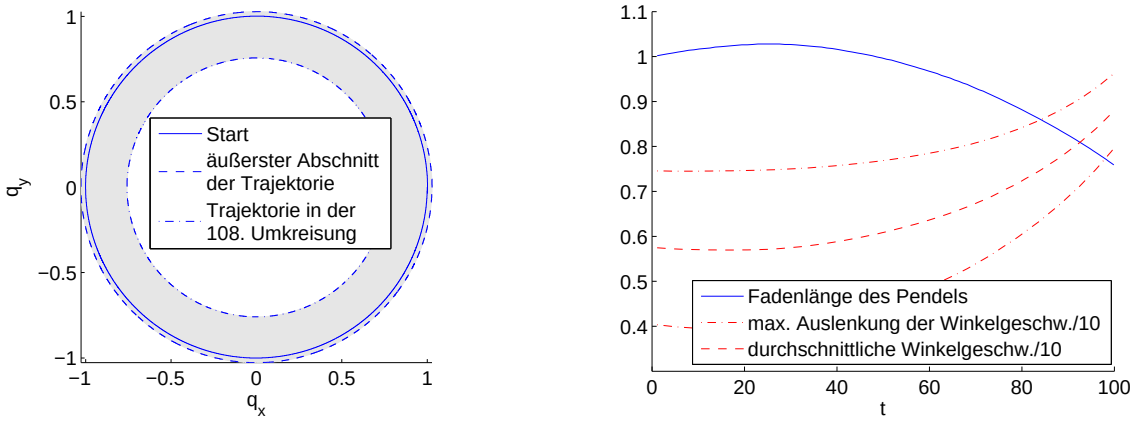

Abbildung 5.22: Lösung des QSS-Verfahrens mit Überschlag und ohne Dämpfung

Im linken Bild sind nur die wesentlichen Trajektorienabschnitte zu sehen. Die graue Fläche entspricht der gesamten Lösung des QSS-Verfahrens in der vorgegebenen Simulationszeit. Im rechten Bild sehen wir die typische Krümmung der Fadenlänge bzw. Winkelgeschwindigkeit, welche die Neigung des System zum Ursprung zeigt. Zu beachten ist, dass in dieser Abb. die durchschnittliche Winkelgeschwindigkeit nur den Mittelwert vom Maximum und Minimum pro Überschlag beschreibt.

Zum Vergleich betrachten wir, wie sich das explizite Eulerverfahren in diesem Modell verhält. Als äquidistante Schrittweite wählen wir dafür $h = 10^{-4}$. In der Simulationsdauer von $100\,s$ entspricht dies 10^6 Auswertungen, die das Verfahren benötigt. Beim QSS-Verfahren kommt der Integrator, der die Geschwindigkeit zum Ort integriert ($\dot{\boldsymbol{x}} \to \boldsymbol{x}$) hingegen mit etwa $811 \cdot 10^3$ Auswertungen (oder Zustandswechsel) aus. Mit $5.9 \cdot 10^6$ Auswertungen benötigt der zweite Integrator ($\ddot{\boldsymbol{x}} \to \dot{\boldsymbol{x}}$) deutlich mehr (der relative Fehler ist hier dafür geringer). Es sei noch erwähnen, dass die asynchrone Arbeitsweise des QSS-Verfahrens hier nicht unbedingt von Vorteil ist, da bei allen Änderungen, egal ob es sich um die Geschwindkeit oder die Position handelt, immer die gleiche stationäre Funktion ausgewertet werden muss.
In Abb. 5.23 sehen wir nun die Lösung des Eulerverfahrens. Anders als beim QSS-Verfahren tendiert hier das Pendel mit wachsender Geschwindigkeit in das Äußere zu schwingen.

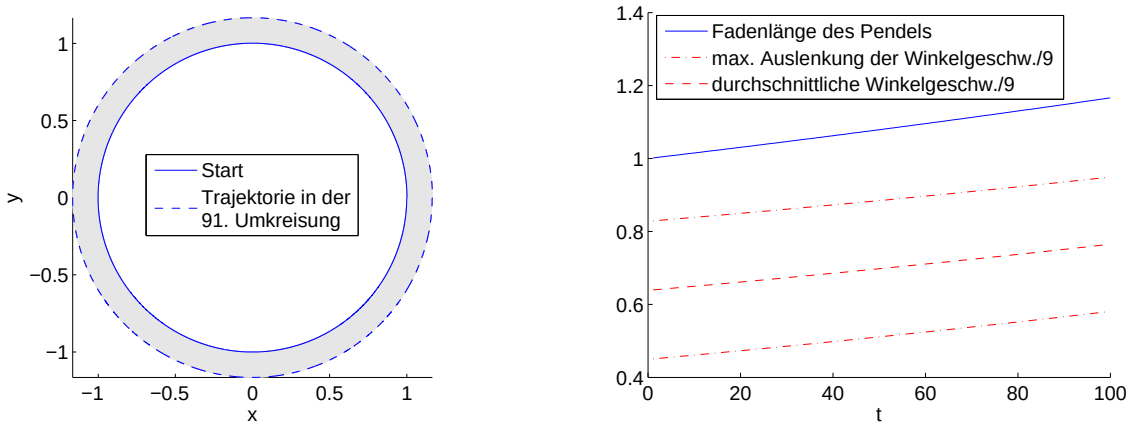

Abbildung 5.23: Lösung des expliziten Eulerverfahrens mit Überschlag und ohne Dämpfung

Anhand dieser Beispiele haben wir gesehen wie schwierig dieses Modell für das QSS-Verfahren ist. Am Rande muss allerdings erwähnt werden, das dies nicht außergewöhnlich ist und auch viele andere numerische Verfahren hierfür nicht geeignet sind. Die Ursache finden wir in der Zwangsbedingung (5.3b), die erst nach zweimaligen Ableiten für (5.3a) verwendet wird und deshalb Informationen verloren gehen. Dies wird in der Literatur auch als „Drift-Off“-Effekt bezeichnet. In [12] wird auf diese Schwierigkeit eingegangen und verschiedene Methoden, die dieses Problem speziell behandeln studiert.

5.2.4 Hüpfender Ball

In diesem Abschnitt betrachten wir die Simulation des einfachsten Modells des hüpfenden Balles (englisch: Bouncing Ball). Das Modell wird einerseits durch die Fallphase des Balles mit

$$\begin{aligned} \dot{v}(t) &= -g, \\ \dot{x}(t) &= v(t) \end{aligned} \tag{5.5}$$

und andererseits durch das Ereignis des Aufpralls beschrieben. In diesem Modell wird vorrausgesetzt, dass sich der Ball immer über dem ebenen Boden, der eine konstante Höhe von $0\,m$ hat, befindet.
In der obigen DGL (5.5) beschreibt $v(t)$ die vertikale Geschwindigkeit, g die Erdgravitation und $x(t)$ den Abstand des Balles zum Boden. Diese DGL kann auch so interpretiert werden, dass sich der Ball mit konstanter horizontaler Geschwindigkeit von $1\,m/s$ bewegt. Der Anfangswert v_0 kann beliebig gewählt werden, hingegen muss x_0 zumindest die Höhe des Bodens haben. Die einzige Reibung, die wir hier betrachten, findet im Ereignis

des Aufpralls $\hat{t}$ statt. In diesem Moment muss der Integrator, welcher die Beschleunigung zur Geschwindigkeit integriert ($\dot{v}(t) \to v(t)$, in Abb. 5.24 wird dieser „QSS1“ genannt) auf den neuen Anfangswert $v_0 = k \cdot v(\hat{t})$ zurückgesetzt werden. Die Konstante $k \in (-1, 0)$ beschreibt in dieser Gleichung die Stoßzahl.

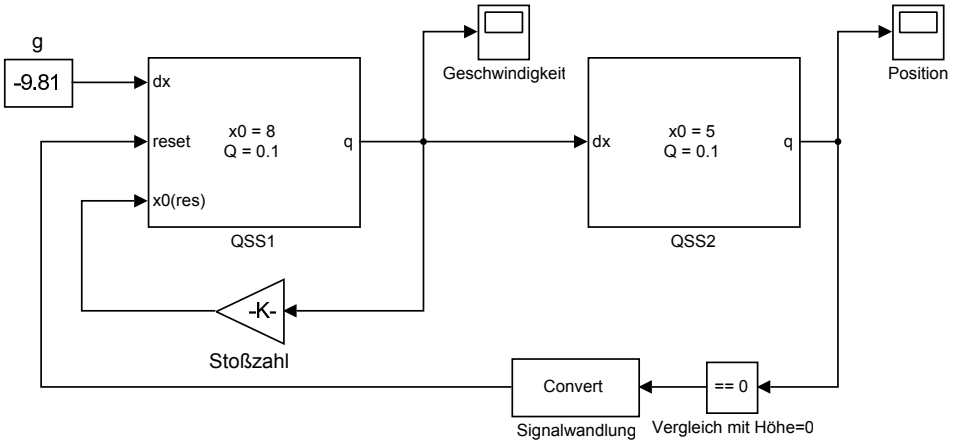

Abbildung 5.24: Simulinkmodell des Hüpfenden Balles in der Ebene (Modell A)

Beispiel 5.2.6 (Hüpfender Ball in der Ebene (Modell A))**.** In diesem Beispiel wollen wir das Verhalten verschiedener Systeme zum Zeitpunkt des Aufpralls gegenüberstellen. Als Anfangswerte wählen wir hierfür $x_0 = 5$ und $v_0 = 8$ und als Stoßkonstante $k = -0.6$. Als Vergleich zum quantiserten System wählen wir das Runge–Kutta-Verfahren von Dormand und Prince. Um gleiche Bedingungen zu schaffen, wurde hier die „Zero-Crossing“-Methode deaktiviert. In den Abb. 5.25 und 5.26 sehen wir, dass es nur ein Frage der Zeit ist, bis der Ball ausschließlich im Fallzustand ist.
Ähnliches kann auch im quantisierten System passieren, und zwar genau dann, wenn der Ball nach dem Aufprall aufgrund zu geringer Geschwindigkeit q_v keinen Zustand $q_x > 0$ mehr erhält. In diesem Fall ändert sich q_x von 0 auf $-\Delta Q$ und verpasst dadurch das Ereignis des Zurücksetzens (Aufprall). In Abb. 5.25 sehen wir rechts die Geschwindigkeit des Balles, welcher im Zeitpunkt $t = 5.71$ das lokale Maximum der Höhe von $x = 0.0642$ aufweist. Diese Höhe ist aber geringer als das gewählte Quantum $\Delta Q = 0.1$, sodass der Ball nicht mehr vom Boden „hoch kommt“. Ab diesem Zeitpunkt befindet sich der Ball nur noch im freien Fall.
Wählt man andere Lösungsparameter für das QSS-Verfahren, so sehen wir eine bessere numerische Lösung q_x, welche am Ende zwischen 0 und ΔQ oszilliert (siehe Abb. 5.27). Die Hysteresebreite ϵ wird in beiden Fällen gleich dem Quantum gewählt.

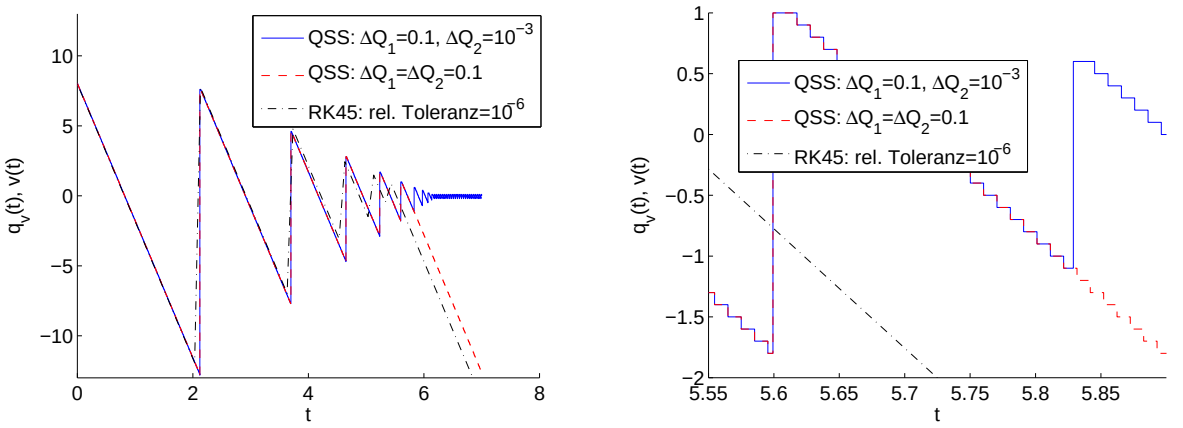

Abbildung 5.25: Geschwindigkeit des Balles, rechts der Ausschnitt $[5.55, 5.90]$

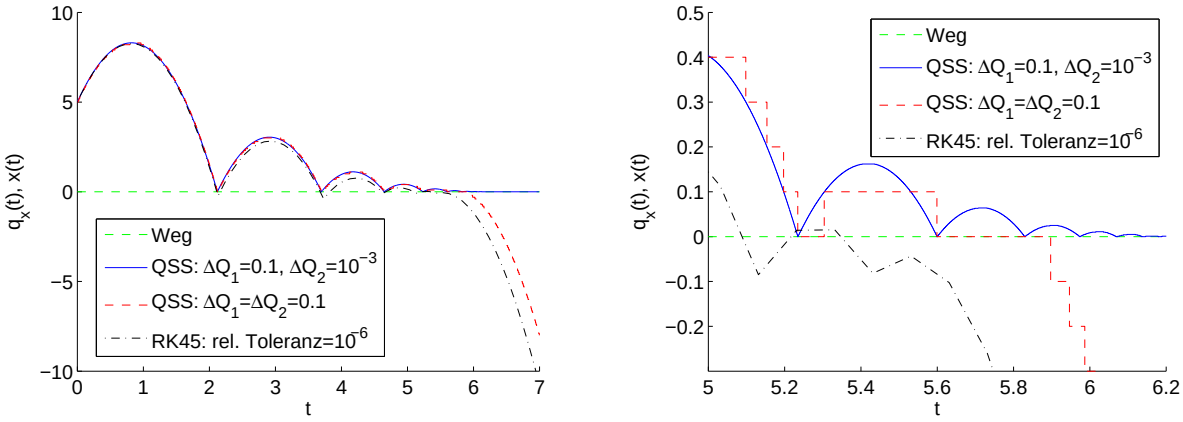

Abbildung 5.26: Position des Balles, rechts der Ausschnitt $[5.0, 6.2]$

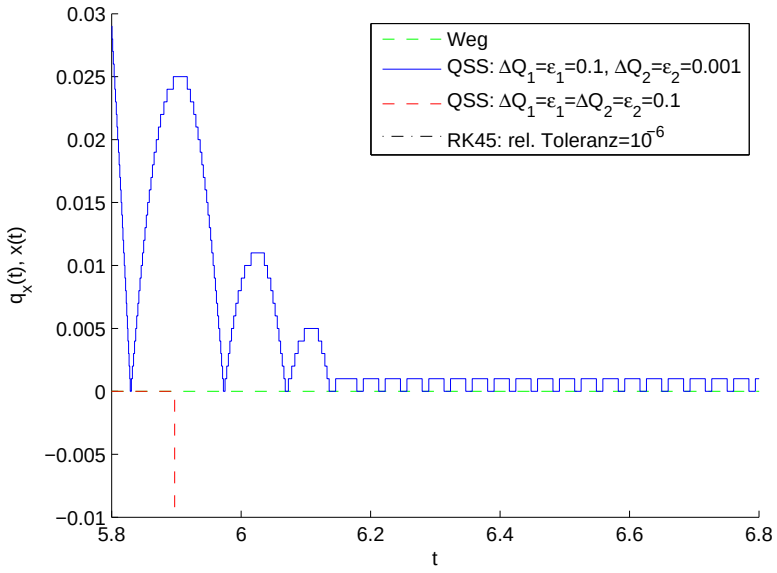

Abbildung 5.27: Position des Balles, Ausschnitt $[5.8, 6.8]$

Wir stellen somit fest, dass in diesem Modell eine richtig gewählte Quantumgröße ΔQ von großer Bedeutung ist. Gehen wir von $\Delta Q = \epsilon$ aus, so können wir folgendes garantieren:
Betrachten wir den Zeitpunkt, in dem der Integrator auf den Anfangswert $x_{v,n} = k \cdot q_{v,n-1}$ zurückgesetzt wird. Ist gleichzeitig die Bedingung

$$\Delta Q_1 \cdot \left(m \cdot \sigma_{v,n} + \frac{(m-1) \cdot m}{2} \cdot \sigma_{v,i} \right) > \Delta Q_2,$$

$$\text{mit } m = \frac{q_{v,n}}{\Delta Q_1} \text{ und } \sigma_{v,i} = \frac{\Delta Q}{g} \quad (i = n+1, \ldots, n+m-1)$$

erfüllt, so erkennt das System noch einen weiteren Aufprall des Balles.

In den nächsten Beispielen wird das Modell ein wenig verändert, um das Problem des „Durchfallens“ zu umgehen, siehe Abb. 5.28. Die Abfrage wird erweitert, sodass bei $q_x = -\Delta Q$ nun beide Integratoren zurückgesetzt werden. Das heißt, liegt der Ball um ein ΔQ „unter dem Boden“, so wird er wieder auf den Anfangswert $x_0 = 0$ zurückgesetzt. Bei den zeitbasierten Systemen wird ebenso die „Zero-Crossing“-Methode aktiviert.

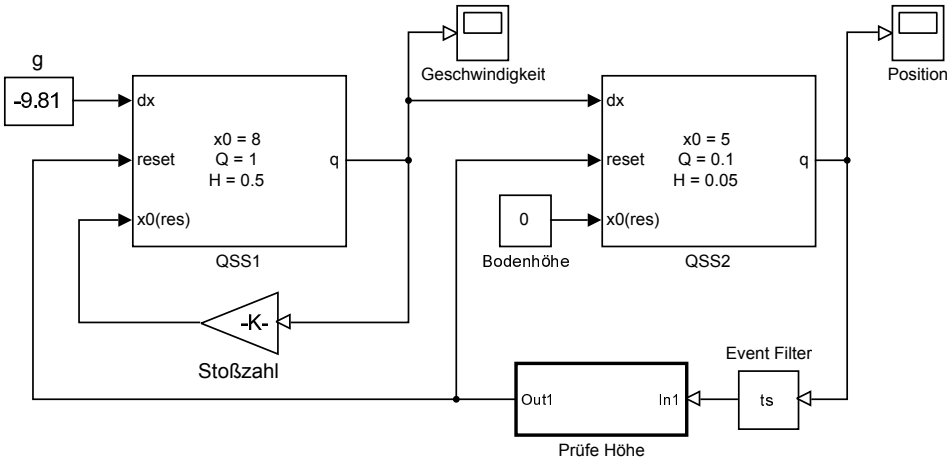

Abbildung 5.28: Simulinkmodell des Hüpfenden Balles in der Ebene (Modell B)

Beispiel 5.2.7 (Hüpfender Ball in der Ebene (Modell B))**.** Hier werden die gleichen Anfangswerte und die gleiche Stoßkonstante wie aus dem Beispiel zuvor gewählt. Als zeitbasiertes System wählen wir wieder das RK45-Verfahren mit einer relativen Toleranz von 10^{-6}. Die Lösungsparameter des QSS-Verfahrens werden wie folgt gewählt: $\Delta Q_1 = 1$, $\epsilon_1 = 0.5$, $\Delta Q_2 = 0.1$ und $\epsilon_2 = 0.05$. Die Abb. 5.29 und 5.30 zeigen die verschiedenen numerischen Lösungen.

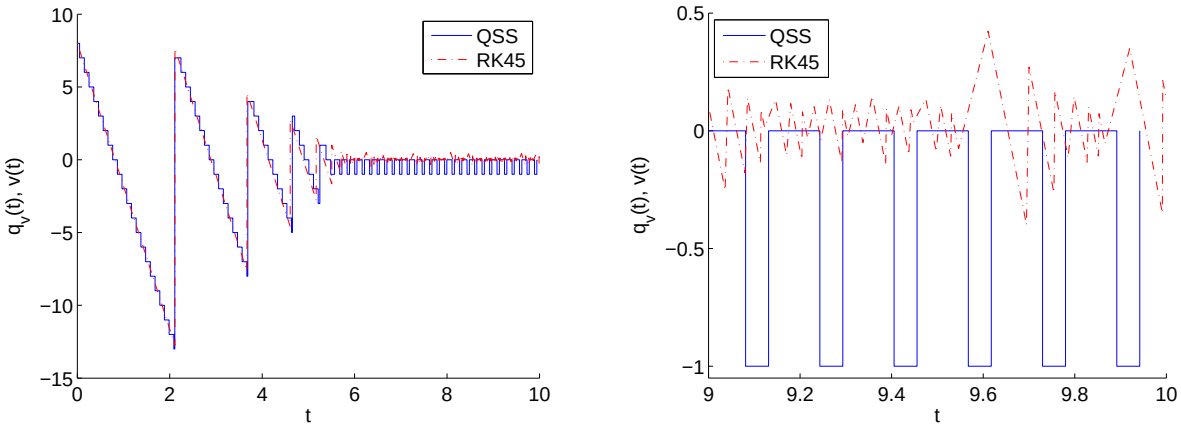

Abbildung 5.29: Geschwindigkeit des Balles, rechts der Ausschnitt $[9, 10]$

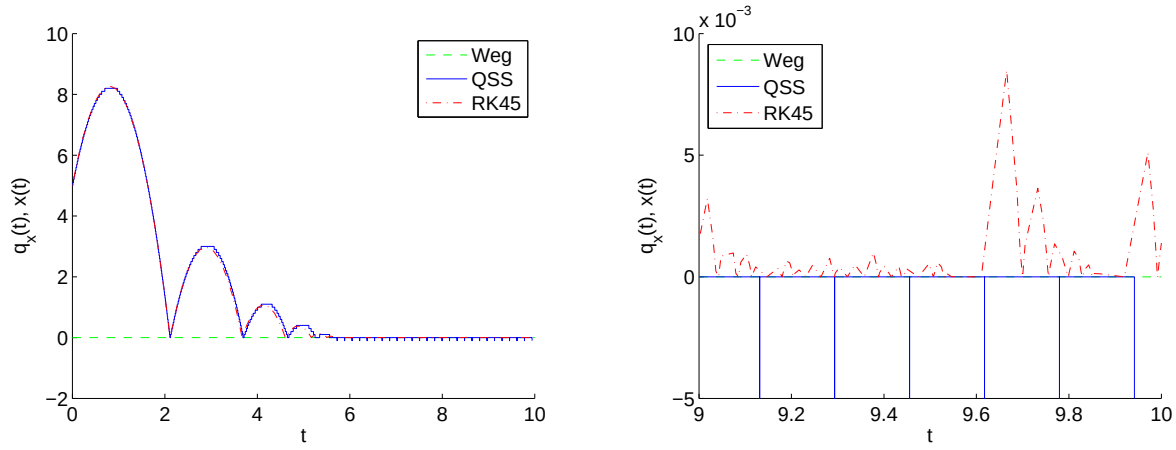

Abbildung 5.30: Position des Balles, rechts der Ausschnitt $[9, 10]$

Die Flanken in Abb. 5.30 unter 0 zeigen die Zeitpunkte $\bar{t}$, zu denen $q_x(t)$ den negativen Wert $-\Delta Q$ erhalten würde. Dies sind auch die Zeitpunkte, zu denen beide Integratoren zurückgesetzt werden und das System zu oszillieren beginnt. In den Abb. 5.29 und 5.30 sehen wir auch, dass sich die Anzahl der verwendeten Gitterpunkten (RK45) von der Anzahl der Zustandswechsel (QSS) im Intervall $[9, 10]$ deutlich unterscheidet. Die Tabelle 5.3 zeigt eine Gegenüberstellung:

	Anzahl der Zustandswechsel	
	Integrator $(\dot{v} \to v)$	Integrator $(v \to x)$
QSS	12	6
	verwendete Gitterpunkte	
RK45	999	

Tabelle 5.3: Anzahl der Gitterpunkte bzw. Zustandswechsel für $t \in [9, 10]$

Beispiel 5.2.8 (Hüpfender Ball über Stufen (Modell B))**.** Wir ändern das Modell in der Form, dass der Ball Stufen hinunter hüpft. Als Vergleich zum QSS-System nehmen wir diesmal das Verfahren von Bogacki–Shampine. Dieses ist ein vierstufiges explizites Runge-Kutta-Verfahren von dritter Ordnung, falls wie in diesem Beispiel eine konstante Schrittweite von $h = 0.01$ gewählt wird. Als Lösungsparameter wählen wir $\Delta Q = \epsilon = 0.1$.

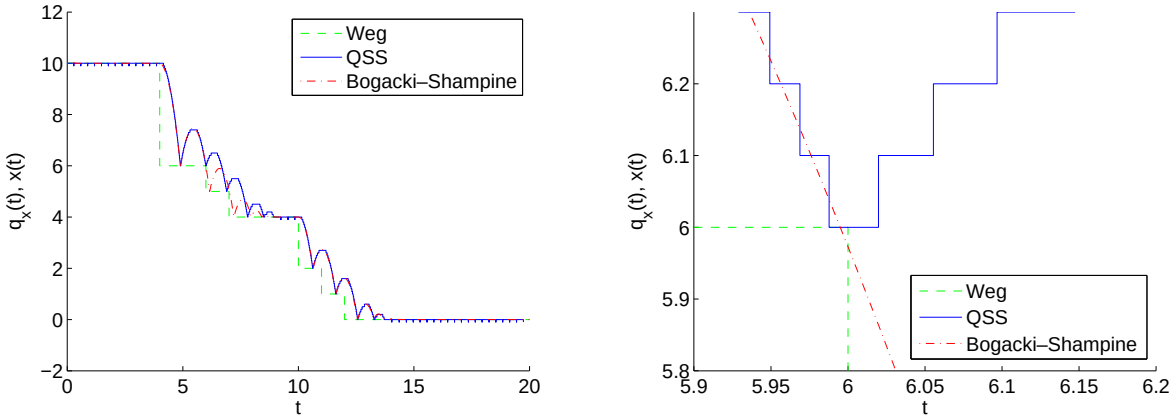

Abbildung 5.31: Hinunterfallen des Balles über Stufen, rechts der Ausschnitt $[5.9, 6.2]$

Die Abb. 5.31 zeigt, dass in der numerischen Lösung des zeitbasierten Verfahrens der Ball „durch eine Stufe fällt“, falls es in diesem Zeitintervall keine Funktionsauswertung gegeben hat (dieses Phänomen namens „Event Skipping“ wurde bereits von François E. Cellier und Ernesto Kofman [6] ausführlich behandelt). Dies ist ein weiterer Vorteil des QSS-Verfahrens, bei dem dies nicht vorkommen kann.

Kapitel 6

Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Funktionalität in MATLAB/Simulink implementiert, die kontinuierliche Systeme mithilfe des *Quantized State System* (QSS) Formalismus in zustandsdiskrete Systeme wandelt und löst. Diese Implementierung wurde an ausgewählten Beispielen getestet. Die Ergebnisse zeigen, wann das zustandsdiskrete Lösungsverfahren dem konventionellen vorzuziehen ist. Das Kapitel 5.2, dass das Lösen von Differentialgleichungen mit Diskontinuitäten zum Inhalt hat, unterstreicht die Attraktivität des QSS-Verfahrens. Hier wurden unter anderem Modelle mit einem PWM-Signal betrachtet. Sie verdeutlichen die Wichtigkeit des QSS-Verfahrens, das in der Lage ist rechenintensive Modelle durch die asynchrone Arbeitsweise der Integratoren zu vereinfachen. Als großen Vorteil stellte sich die Eigenschaft des QSS-Verfahrens bei manchen Modellen heraus, Ereignisse auf direktem Weg zu erkennen.

In dieser Arbeit haben wir uns mit dem Quantized State System erster Ordnung (QSS1) beschäftigt. Die Beispiele zeigen, dass bereits das QSS1-Verfahren oft erhebliche Vorteile mit sich bringt. Die Nachtteile wie das lineare Wachsen der benötigten Schritte für eine gewünschte Toleranz oder das Oszillieren um eine konstante Lösung, bei gleich bleibenden Fehler, bleiben aber weiterhin bei diesem Verfahren bestehen.
Deshalb wird am Ende dieser Arbeit die QSS-Variante zweiter Ordnung vorgestellt, die die oben genanten Nachteile nicht hat. Außerdem wird ein Ausblick auf eine Umsetzung dieses Verfahrens in Simulink gegeben.

Quantized State System zweiter Ordnung (QSS2) [13]

Beim QSS2-Verfahren muss wie bei QSS1 eine Beziehung zwischen dem quantisierten und dem exakten Zustand geschaffen werden (Quantisierungsfunktion). Ein Quantum beschreibt, wie sich die Steigung für einen Zustandswechsel des Integrators ändern muss.

Die Quantisierungsfunktion des QSS2-Verfahrens ist definiert durch

$$q_i(t) = \begin{cases} x_i(t) & \text{falls } t = t_0 \quad \vee \quad |q_i(t^-) - x_i(t^-)| = \Delta Q \\ q_i(t) + m_j \cdot (t - t_j) & \text{sonst} \end{cases}$$

mit den zugehörigen Zeitpunkten $t_0, ..., t_j, ...$ wobei

$$t_{j+1} = \min \{t | t > t_j \quad \wedge \quad |x_i(t_j) + m_j \cdot (t - t_j) - x_i(t)| = \Delta Q\}$$

und die Steigung definiert ist als

$$m_0 = 0, \quad m_j = \dot{x}_i(t_j^-), \quad j = 1, ..., k, ...$$

Ein großer Unterschied zum QSS1-Verfahren besteht darin, dass Ereignisse im QSS2-Verfahren außerdem die Information über die Steigung beinhalten. Für eine Implementierung in Simulink hat man dies bereits in der Signalführung zu berücksichtigen. Abb. 6.1 zeigt das Verhalten der Quantisierungsfunktion des QSS2-Verfahrens.

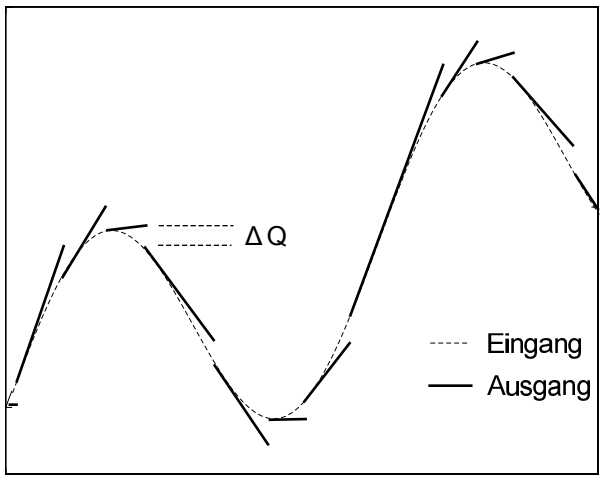

Abbildung 6.1: typische Ein- und Ausgangstrajektorien der QSS2-Quantisierungsfunktion [6]

Anders als beim QSS1-Verfahren sind beim QSS2-Verfahren die Ausgangstrajektorien nicht mehr stückweise konstant sondern stückweise linear. Eine nichtlineare stationäre Funktion würde somit keine linearen Eingangstrajektorien für den Integrator erzeugen. In dieser Eigenschaft unterscheidet sich das QSS2 vom QSS1 ganz wesentlich. Ob die Einganstrajektorien linear sind oder nicht, hängt davon ab, ob es sich um ein LZI-System handelt. Liegt das vor, so können wir das zugehörige quantisierte System (2.2) wieder exakt lösen.

Ausblick zur Umsetzung des QSS2-Verfahrens in Simulink

Die Implementierung des QSS2 Integrators in Simulink würde sich nicht wesentlich von der des Integrators erster Ordnung unterscheiden, außer dass er mit Ereignissen doppelter Dimension arbeiten müsste.

Die Schwierigkeiten werden in der Umsetzung der stationären Funktion liegen. Beim QSS1-Verfahren muss das nicht berücksichtigt werden, da das System auf stückweise konstanten Werten basiert. Das neue System stellt an die stationäre Funktion noch zusätzlich die Anforderung, sich den Zeitpunkt des letzten Ausgangsereignisses (in jeder Komponente) zu merken. Nur durch diese Eigenschaft lässt sich der Wert einer Ausgangstrajektorie bestimmen

$$x_i(t) = x_i(t_k) + m_{x_i}(t_k) \cdot (t - t_k),$$

wobei t_k der letzte Zeitpunkt eines Ausgangsereignisses ist. Die obige Anforderung an die stationäre Funktion ist auch notwendig, um die Steigung von $f(\boldsymbol{q}(t), \boldsymbol{u}(t))$ zu bestimmen, welche der Integrator des QSS2-Verfahrens benötigt.

Anhang

A.1 Subsysteme der QSS-Implementierung in Simulink

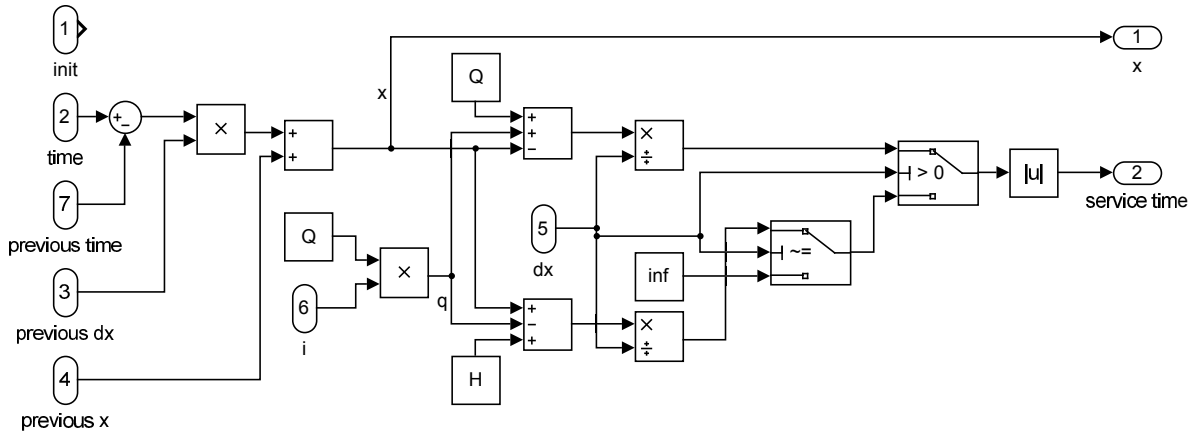

Abbildung A.1: Subsystem „State x“ der QSS-Implementierung

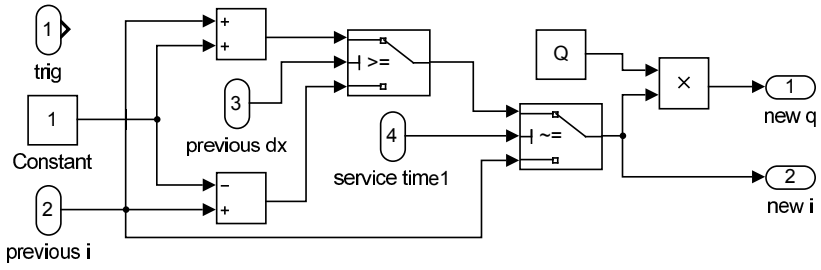

Abbildung A.2: Subsystem „State q“ der QSS-Implementierung

A.2 Anleitung zur Bedienung des QSS-Integrators in Simulink

Die Abb. A.3 zeigt das Äußere des QSS-Integrators. Der Eingang „dx" des QSS-Integrator wird für die Ableitung des Zustandes verwendet, während der Ausgang „q" den quantisierten Zustand beschreibt.

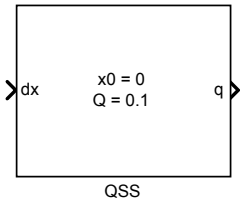

Abbildung A.3: Äußere des QSS-Integrators in Simulink

Der Anfangswert x_0 und das Quantum können in der Maske (siehe Abb. A.4) festgelegt werden. Einen optionalen Parameter stellt die Hysteresebreite dar. Ist die zugehörige Checkbox nicht markiert, so wird der Hysteresebreite der Wert des Quantum zugewiesen.

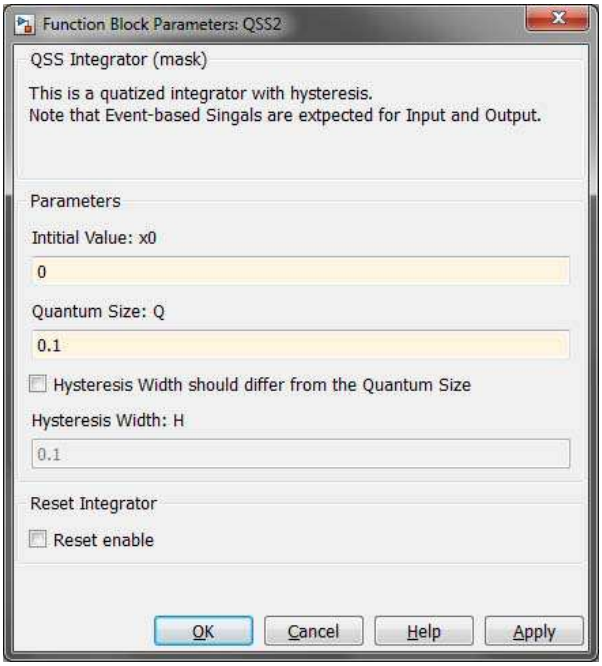

Abbildung A.4: Maske des QSS-Integrators in Simulink

Die zweite Checkbox ermöglicht ein Zurücksetzen des Integrators. Ist diese markiert, so erhält der Integrator zwei weitere Eingänge „reset" und „x0(res)" (siehe Abb. A.5). Hat das Signal an „reset" eine Zustandsänderung, so setzt der Integrator in diesem Moment den Wert an „x0(res)" als neuen Zustand.

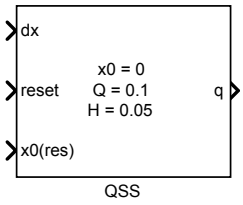

Abbildung A.5: Äußere des QSS-Integrators in Simulink, mit gewählter Option „Zurücksetzen“

Hinweis

Die Aus- und Eingangssignale des Integrators sind ereignisbasierte Signale. Blöcke aus der Simulinkbibliothek, die nur mit zeitbasierten Signalen arbeiten, können mit diesem Integrator zusammenarbeiten, falls sie in ein „Atomic Subsystem“ gelegt werden, siehe Kapitel 4.3.1 und Abb. 4.2.

Literaturverzeichnis

[1] Ernesto Kofman, Sergio Junco: *Quantized-State Systems: A DEVs Approach for Countinuous System Simulation*, (2001)

[2] Bernard Zeigler: *Theory of Modeling and Simulation*, (1976)

[3] Michel Hénon: *On the numerical computation of Poincaré maps*, (1982)

[4] Rolf Unbehauen: *Systemtheorie 1*, (8. Auflage 2002)

[5] Jan Lunze: *Regelungstechnik 1*, (9. Auflage 2013)

[6] François E. Cellier, Ernesto Kofman: *Continuous System Simulation*, (2006)

[7] Felix Breitenecker: *Modellbildung und Simulation: Unterlage zur Vorlesung, TU Wien*, (2012)

[8] François E. Cellier, Ernesto Kofman, Gustavo Migoni, Mario Bortolotto: *Quantized State System Simulation*, (2008)

[9] Wikipedia: *http://de.wikipedia.org/wiki/Vierquadrantensteller*, (06-05-2014)

[10] Rolf Fischer: *Elektrische Maschinen*, (12. Auflage 2004)

[11] Bill Messner, Dawn Tilbury: *Control Tutorials for MATLAB and Simulink, University of Michigan: http://ctms.engin.umich.edu*, (06-05-2014)

[12] Carina Pöll, Irene Hafner, Bernhard Heinzl, Felix Breitenecker: *Über die Simulation differential-algebraischer Modelle mit nichttrivialem Index*, (2014)

[13] Ernesto Kofman: *A Second Order Approximation for DEVS Simulation of Continuous Systems*, (2002)