Inhalt

1 Einleitung

In ingenieurswissenschaftlichen Anwendungen stösst man häufig auf Sachverhalte, bei denen der Zuwachs einer Größe mit dem Wert der Größe selbst zusammenhängt. So ist es beispielsweise entscheidend für den Ausfluss aus einem Gefäss, wie hoch der Füllstand des selbigen ist, welcher sich jedoch auch wieder mit fortschreitendem Entleeren ändert.

Die mathematischen Grundlagen zur Lösung eines solchen Problems liefern uns die Differentialgleichungen. Je nach Ordnung und Komplexität finden sich in vielen Fällen Lösungswege, durch die man eine „gewöhnliche“ Gleichung erhält. In diese kann man nun Werte einsetzen und sie als Ergebnis graphisch veranschaulichen.

Einen anderen Weg beschreitet das numerische Lösen von Differentialgleichungen. Hierbei wird mit Hilfe von Computern einfach ein gewisser Wertebereich direkt in die ursprüngliche Differentialgleichung eingesetzt und jeder Wert für sich berechnet. Als Ergebniss liefert einem das Programm ein Schaubild des Funktionsverlaufes mit zugehöriger Wertetabelle.

Ein Beispiel für ein solches Programm stellt Simulink dar. Basierend auf Matlab, einer leistungsfähigen mathematischen Simulations- und Rechenumgebung, stellt Simulink eine graphische Erweiterung dar. Damit kann man Beziehungen durch Modelle darstellen und die Ergebniss schließlich visualisieren.

Dieses Dokument soll - ohne Anspruch auf Vollständigkeit erheben zu wollen – mittels einiger Beispiele dem Anwender Matlab Simulink als Werkzeug zur Lösung von Differentialgleichungen näherbringen.

2 Programmeinführung

2.1 Programmstart

Gestartet wird Matlab® Simulink über den Aufruf von Matlab im Startmenü. Nun findet man sich im Eingabebereich von Matlab wieder, einer kommandozeilenorientierten Bedieneroberfläche. Wir werden uns diese später noch weiter zunutze machen wenn es um die Definition von Variablen geht. Durch Eingabe von `simulink` wird das entsprechende Modul gestartet.

In einem neuen Fenster präsentiert sich nun der *Simulink Library Browser* (**Abb. 1**). Im oberen der drei Kästen steht eine kurze Beschreibung des gewählten Blocks. Auf der linken Seite befinden sich die verschiedenen Gruppen von Blöcken. Für diese Anleitung ist nur die erste Hauptgruppe (Simulink) von Interesse. Die am häufigsten benutzten Blöcke sind dabei unter *Commonly Used Blocks* zu finden. Auf der rechten Seite befinden sich die Blöcke mit ihren Symbolen und Namen.

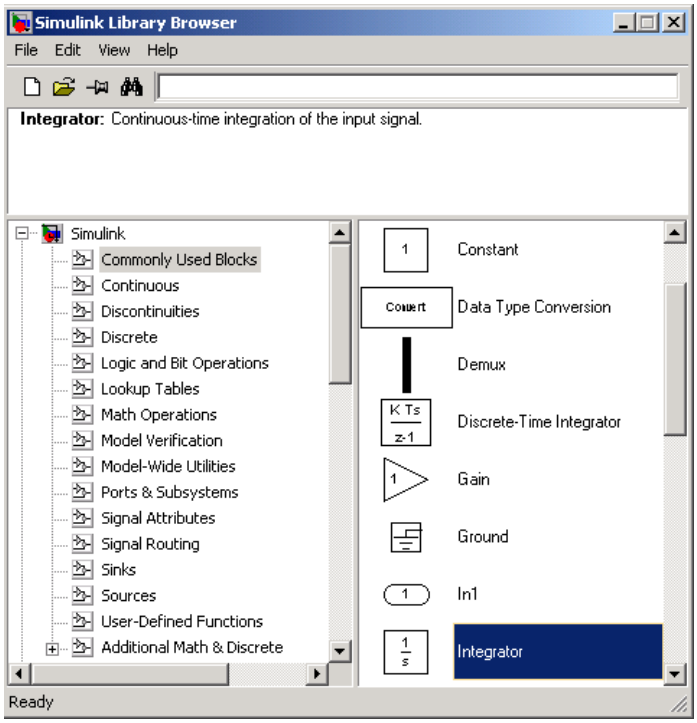
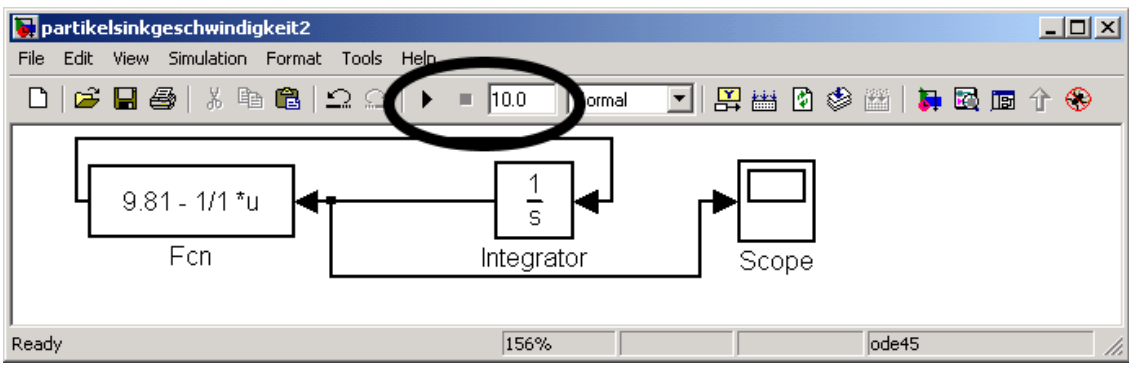


Abb. 1: Simulink Library Browser

Über *File – New – Model* öffnet man eine neue, leere Arbeitsfläche. Dorthin werden die Blöcke aus dem *Library Browser* per drag'n'drop kopiert. In **Abb. 2** sieht man die Arbeitsfläche; in dem Oval umrandeten Bereich kann man die Simulation starten, anhalten und die `Stop-Time` einstellen.



2.2 Definition der Ablaufparameter

Das Numerische Lösen von Funktionen besteht im wesentlichen darin, dass alle Werte aus einem bestimmten Bereich in die Funktion eingesetzt werden. Dieser Bereich beginnt dabei bei der *Start-Time* und endet bei der *Stop-Time*. Die *Start-Time* beginnt standartmässig bei 0.

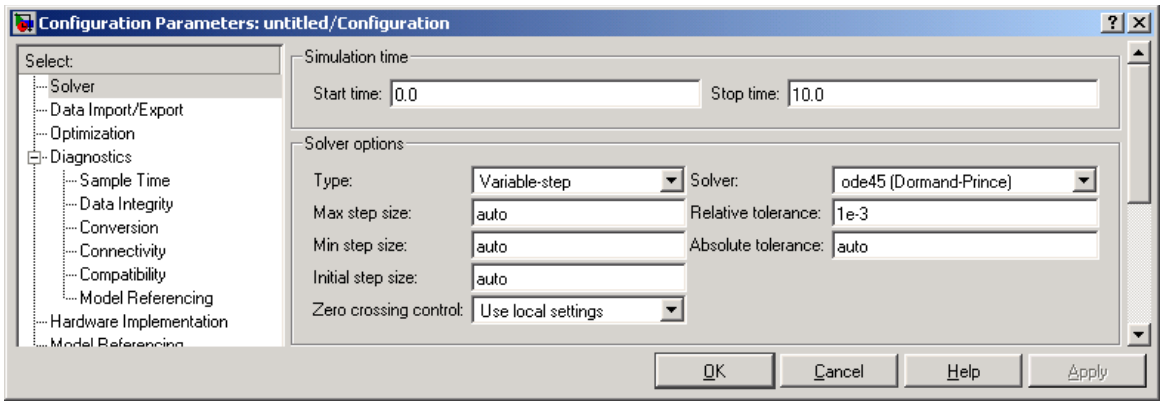


Abb. 3: Dialog Configuration Parameters

Möchte man die Standartwerte ändern, so öffnet man unter *Simulation – Configuration Parameters* den in **Abb. 3** gezeigten Dialog. Dabei ist neben Start und Stop vor allem das Feld *Relative Tolerance* von Bedeutung. Mit ihm kann eine Kurve geglättet werden, d.h die Auflösung verbessert werden.

2.3 Funktionsblöcke

Um in Matlab Simulink eine mathmatische Gleichung aufzustellen bedarf es verschiedener Funktionsblöcke. Ein jeder Block verfügt über einen Eingang und/oder Ausgang. Sie werden miteinander durch Klicken und Ziehen vom Aus- zum Eingang verbunden. Alternativ kann man den Ursprungsblock markieren, `[Strg]` gedrückt halten und den Zielblock anklicken. Im folgenden werden die wichtigsten Blöcke kurz beschrieben.

2.3.1 Signale zusammenführen (Mux)

Zur Zusammenführung mehrerer Signale bedient man sich eines `Mux` (Multiplexer, **Abb. 4**). Von Bedeutung ist dabei die Reihenfolge, in der man andere Blöcke verbindet, es wird von oben nach unten durchnummeriert. Durch Doppelklick auf den Mux lässt sich die Anzahl der möglichen Eingänge festlegen.

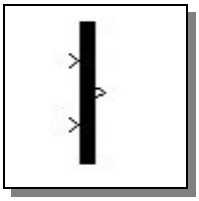


Abb. 4: Mux

2.3.2 Signale Trennen (Demux)

Um mehrere Signal zu trennen, wird ein `Demux` verwendet. Im Wesentlichen handelt

es sich dabei um die Umkehrung eines Mux.

2.3.3 Benutzerdefinierte Funktionen (fcn)

Selbst erstellte Funktionen werden im `(fcn)`-Block mit einer einfachen Formelschreibweise untergebracht. Die Eingangsvariablen mit werden dabei mit `u[n]` bezeichnet, wobei `n` die Nummer des Eingangs von oben nach unten durchnummeriert ist.

2.3.4 Integrator

Der *`Integrator`*-Block (**Abb. 5**) integriert die aus dem *`fcn`*-Block übergebene Variable nach der Zeit bzw. der Verlaufsachse.

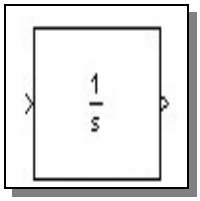


Abb. 5: Integrator

2.3.5 Diagrammanzeige (Scope)

Der `Scope` (**Abb. 6**) dient zur Veranschaulichung der numerisch bestimmten Wertetabelle in Form eines Diagramms. Er verfügt lediglich über einen Eingang, keinen Ausgang. Nach Ablauf einer Simulation kann durch einen Doppelklick auf das Symbol ein Fenster geöffnet werden. Im `Scope`-Diagramm werden die einzelnen Datenreihen in unterschiedlich farbigen Kurven dargestellt (siehe **Abb. 14** auf Seite 15).

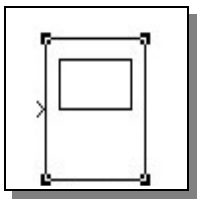


Abb. 6: Scope

2.3.6 Impulsgenerator (Pulse Generator)

Um während des Verlaufs einer Funktion eine exakt definierte Größe über eine gewisse Dauer einzubringen, gibt es den *`Pulse Generator`* (**Abb. 7**). Er besitzt nur einen Ausgang, d.h. er ist von anderen Größen im laufenden System unabhängig. Durch Doppeklicken kann man seine Parameter bestimmen.

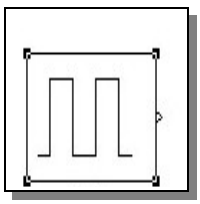


Abb. 7: Pulse Generator

- `Pulse Type`: Auswahl zwischen *`Time based`* (unabhängig von der Auflösegenauigkeit) oder *`Sample Based`* (Jede numerische Berechnung zählt als Takt).
- `Amplitude`: Ausschlag, z.B. Menge des zugegebenen Stoffes in einem System
- `Periode`: Dauer bis zum erneuten Ausschlag/Stoffzugabe
- `Puls width`: Dauer des Ausschlags/der Stoffzugabe
- `Phase delay`: Dauer bis zum ersten Ausschlag

3 Export von Daten

Simulink bietet mit dem eigenen *Scope* nur sehr beschränkte Möglichkeiten zur Ergebnisvisualisierung. Um dieses Manko auszugleichen oder die Daten generell in externen Programmen weiterzuverwerten, gibt es die Möglichkeit des Datenexports. Dabei werden die Daten in Feldern der Matlab-Oberfläche übergeben und von dort aus im Textformat exportiert.

Dazu wird die gleiche Schiene abgefasst, welche auch zu dem Scope führt. Sie wird mit dem Eingang eines *To Workspace*-Blocks (zu finden in der Gruppe *Sinks*) verbunden. Standardmässig ist dieser „simout“ (SIMulink OUTput) genannt und kann nach Bedarf verändert werden. In diesem Block muss als Parameter zwingend der Typ *Array* eingetragen werden, damit Simulink die Daten im richtigen Format übergibt. Nach einem ersten Programmdurchlauf werden dann die Daten an Matlab übergeben.

In der Kommandozeile von Matlab werden die Zeit (tout) und dazugehörige Daten (simout) zusammengefasst. Dies geschieht mit dem Befehl

```
datenverlauf = [tout,simout]
```

Daraufhin werden die Daten auf dem Bildschirm ausgegeben und sind nun zur weiteren Verarbeitung verfügbar. Als ASCII-Datei kann man sie mit

```
save c:\temp\daten.txt datenverlauf -ascii
```

ausgeben.

3.1 Diagramme in OpenOffice erstellen

Zu beachten ist bei der Textdatei, dass anstatt des Kommas ein Punkt als Dezimaltrenner fungiert, die Ursache ist der angelsächsische Ursprung des Programms. Am einfachsten ist es, die Datei in einem Texteditor mit „*replace all*“-Funktion zu öffnen und den Punkt durch ein Komma zu ersetzen. Nun kann auf vertraute Weise z.B. in OpenOffice Calc damit ein Diagramm erstellt werden.

Dazu öffnet man eine neue Tabelle und geht auf *Einfügen – Tabelle aus Datei.* Als Seperator für die Spalten kann man *Feste Breite* wählen. Für die meisten anschaulicheren Anwendungen rät es sich, das Zellenformat auf „Standart Zahlen“ umzustellen, da die Werte von Matlab in der wissenschaftlichen Exponentialschreibweise abgespeichert werden.

Als Beschriftung fügt man nun noch eine erste Zeile ein, im Feld A1 dürfte sich in den meisten Fällen die Zeit (bzw. der Verlauf) als Beschriftung eignen, die anderen Spalten werden entsprechend ihrer Messaufgabe beschriftet.

Nun werden alle Zellen mit Inhalt markiert und *Einfügen – Diagramm* gewählt. Hier eignet sich für die meisten Aufgaben das XY-Diagramm am besten. Bei allen folgenden Programmbeispielen wurde hiervon aufgrund der besseren Lesbarkeit auf Ausdrucken Gebrauch gemacht.

3.2 Plotten mit Matlab

Eine weitere Möglichkeit ist, die Daten mit der *Plot*-Funktion in Matlab direkt auszugeben. Man benutzt dazu den Befehl

```
plot (tout,simout)
```

Es öffnet sich ein Fenster mit einem schlichten Liniendiagramm, ein Beispiel für eine solche Ausgabe befindet sich auf Seite 21.

4 Programmbeispiele

Nachfolgend wird in einigen Beispielen der Weg von der Problemstellung zum Diagramm aufgezeigt. Dabei wurde bewusst auf Masseinheiten bei der Achsenbeschriftung der Diagramme verzichtet, da sie meist auf zufällig gewählten Konstanten basieren und lediglich eine qualitative Aussage liefern sollten.

4.1 Sedimentation

Gegeben sei ein Partikel, das sich in einem Fluid befindet. Es unterliegt dabei einem Kräftegleichgewicht aus Beschleunigungskraft F_B, Schwerkraft F_M und Auftriebskraft F_R. Dabei gilt die Gleichung **(1)**, für den Stillstand des Partikels dargestellt in **Abb.8**.

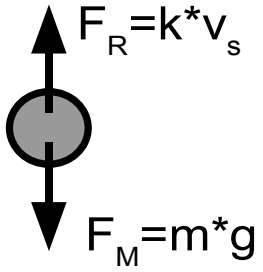


Abb. 8: Partikelsedimentation

$$F_B = F_M - F_R \qquad (1)$$

Gesucht ist dabei die Entwicklung der Sinkgeschwindigkeit über der Zeit, also das Integral dv_s/dt. Durch Einsetzen und Auflösen der Kräftebestandteile erhält man die in **(2)** angegebene Differentialgleichung.

$$\frac{dv_s}{dt} = g - \frac{k}{m} \cdot v_s \qquad (2)$$

Durch systematisches Aufintegrieren der Ableitungsfunktion der gesuchten Lösung mit dem Simulink Integrator erhält man die Lösungsfunktion. Die Anfangsbedingung setzen wir auf v_s=0 durch ändern der `Initial Condition` im `Integrator`. Die Sinkgeschwindigkeit v_s ist dabei der Ausgang des Integrators, dieser wird mit dem selbst definierten Funktionsblock verbunden. In diesem wird die Variable durch die Simulink-eigene Variablenbezeichnung `u` ersetzt. *k* und *m* sind hierbei Partikelkonstanten, zur Vereinfachung wird ihnen in diesem Beispiel der Wert 1 zugewiesen (**Abb. 9**).

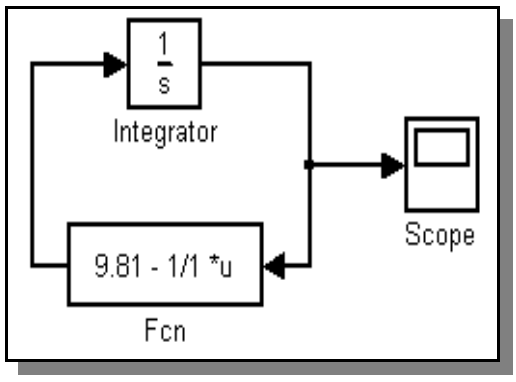


Abb. 9: Model Sedimentation

Möchte man nun die gleiche Simulation für mehrere Startgeschwindigkeiten v_0 ablaufen lassen, so kopiert man den Kreislauf aus `Integrator` und `fcn` mehrmals und ändert die `Initial Condition` auf den gewünschten Startwert. Um die verschiedenen Datenausgaben der Integrationen zusammenzufassen, benutzt man

einen *Mux*. Das Schaubild für drei Anfangsgeschwindigkeiten ist in **Abb. 10** zu sehen. Dabei wurde im ersten Integrator v_0=0, im zweiten v_0=5 und im dritten v_0=-5 gesetzt. Zusätzlich zum *Scope* wurde dabei ein *To Workspace*-Block auf die zusammengefasste Schiene nach dem Mux gehängt, um die Daten zu exportieren.

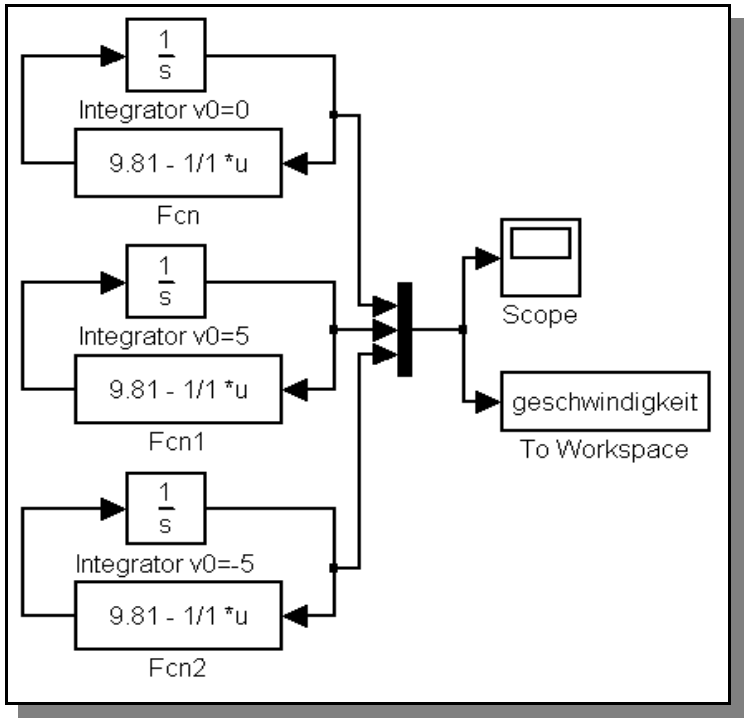


Abb. 10: Modell Sinkgeschwindigkeit mit drei Anfangsgeschwindigkeiten

In **Diagramm 1** zeigt sich nun, dass es keine Rolle spielt, welche Anfangsgeschwindigkeit die Partikel besitzen. Sie nähern sich alle der gleichen Endgeschwindigkeit im Laufe der Zeit an, selbst wenn sie in die endgegengesetzte Richtung gestartet sind.

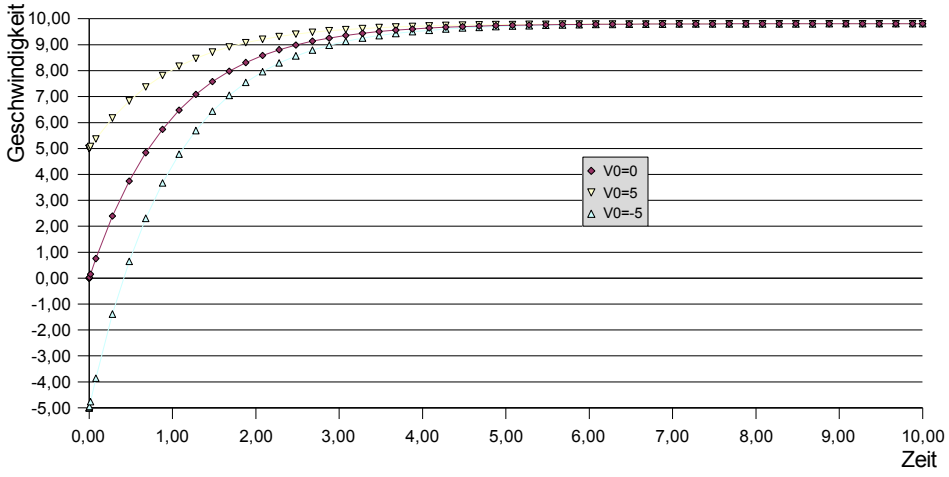


Diagramm 1: Sinkgeschwindigkeiten mit drei Startbedingungen

4.2 Gedämpfte Schwingung

Eine gedämpfte harmonische Schwingung lässt sich mit der Differentialgleichung **(3)** darstellen, wobei der Term 2μ die Dämpfung und ω_0^2 die Frequenz der ursprünglichen Schwingung beschreibt.

$$\frac{d^2x}{dt^2} + 2\mu \cdot \frac{dx}{dt} + w_0^2 \cdot x = 0 \qquad (3)$$

Der Term ω_0^2 muss damit doppelt integriert werden. Dazu wird der Ausgang der Funktion erst einmal Integriert, danach sowohl abgegriffen als auch erneut integriert. Die beiden Stränge werden dann durch einen `Mux` wieder zusammengefasst und in die Funktion zurückgeführt. Bei dem Modell in **Abb. 11** wurde zwischen den `Mux` nach den Integratoren ein weiterer `Mux` eingefügt um die Konstanten Dämpfung μ und ursprüngliche Schwingung ω besser verändern zu können.

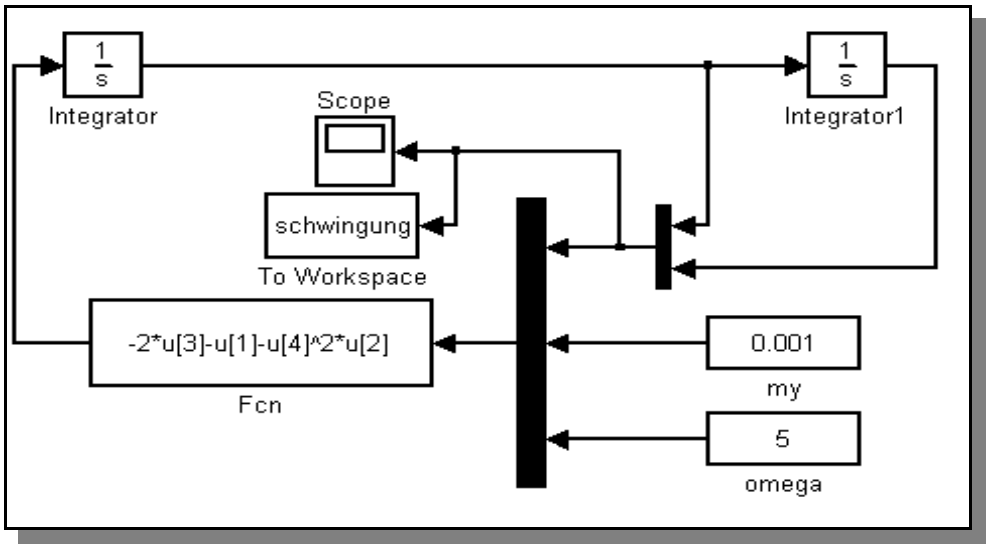


Abb. 11: Modell gedämpfte Schwingung

Da nun in die Funktion mehr als nur eine Variable führen muss man sie durchnummerieren. Dazu werden die Variablen mit `u[n]` bezeichnet, wobei `n` die Nummer des `Mux`-Eingangs von oben nach unten (bzw. Links nach rechts) durchnummeriert ist. Im Beispiel ist somit die Dämpfung μ in der Funktion mit u[4] bezeichnet. Den Verlauf einer harmonischen gedämpften Schwingung sieht man in **Diagramm 2**.

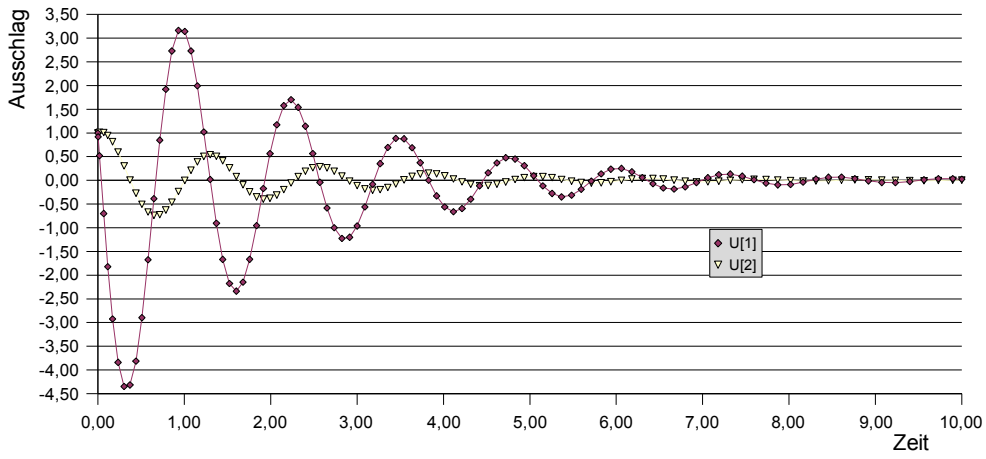


Diagramm 2: Pendelausschlag bei gedämpfter Schwingung

4.3 Chemische Reaktion

Wenn man eine chemische Reaktion betrachtet, welche als Edukt den Stoff A, als Zwischenprodukt B und als daraus entstehendes Produkt C ergibt, so schreibt man dafür in der Chemie:

$$\mathbf{A \rightarrow B \rightarrow C}$$

Möchte man sich nun ein Bild über den Konzentrationsverlauf der einzelnen Stoffe machen, so benötigt man dazu die Reaktionskonstanten $k_{x.}$ Diese beschreiben die Geschwindigkeit der Umsetzung eines Stoffes. Ausgedrückt in Differentialgleichungen erhält man **(4-6)**

Änderung der Konzentration A:

$$\frac{dC_A}{dt} = -k_1 \cdot C_A \qquad (4)$$

Änderung der Konzentration B:

$$\frac{dC_B}{dt} = -k_2 \cdot C_B + k_1 \cdot C_A \qquad (5)$$

Änderung der Konzentration C:

$$\frac{dC_C}{dt} = k_2 \cdot C_B \qquad (6)$$

An dieser Stelle ist es von entscheidender Bedeutung, dass man sich die Abhängigkeiten der einzelnen Konzentrationsverläufe deutlich macht: dC_A/dt ist nur von der Konzentration des Stoffes A abhängig, dC_B/dt sowohl von A als auch von B und dC_C/dt widerum lediglich von der Konzentration des Stoffes B. In **Abb. 12** ist ein mögliches Model in Simulink dargestellt. Die Reaktionskonstanten wurden dabei mit k_A=1,5 und k_B=0,7 willkürlich gewählt und direkt in die `fcn`-Blöcke eingetragen.

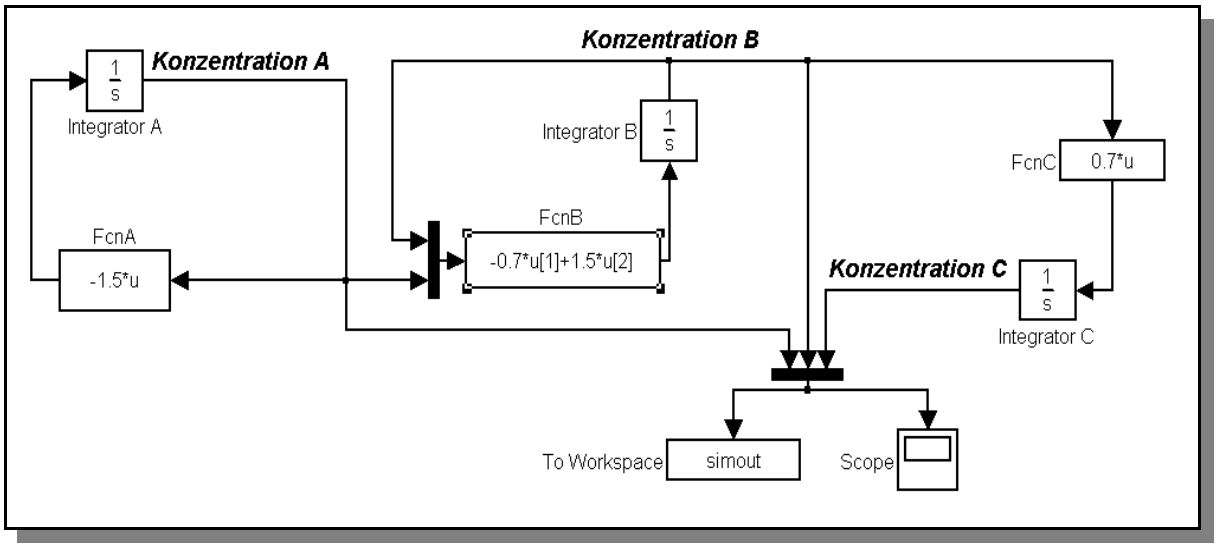


Abb. 12: Modell Chemische Reaktion

Das **Diagramm 3** liefert die passende Ausgabe: Zu Beginn der Reaktion ist am meisten von Stoff A vorhanden *(`Initial Condition=10`),* halb so viel von Stoff B und kein Stoff C. Im Verlauf wird A in B umgewandelt, weswegen die B erst ansteigt. Gegen Ende hin ist nur noch das Edukt C vorhanden, A und B sind vollständig aufgebraucht.

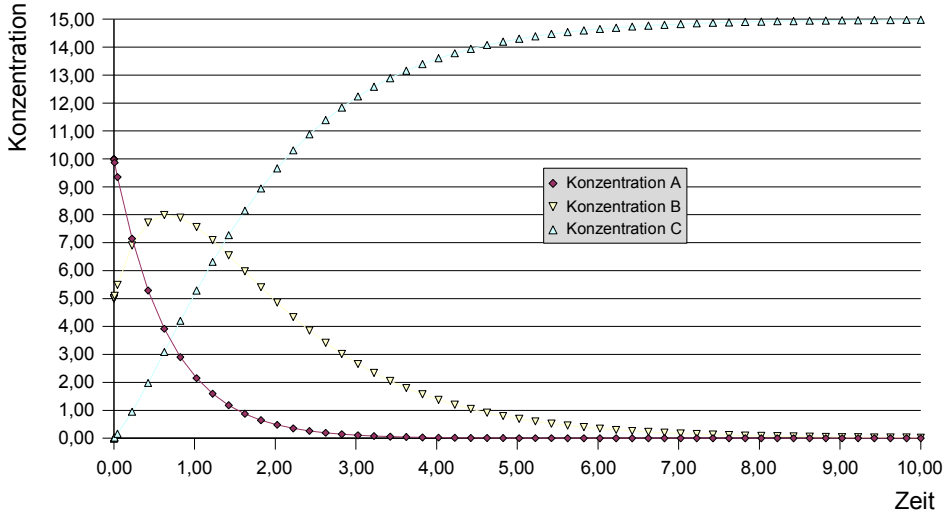


Diagramm 3: Konzentrationsverlauf einer chemischen Reaktion

4.4 Autokatalytische Reaktion

Eine weitere Form einer chemischen Reaktion ist die sog. Autokatalytische Reation.

A + B → C
C → D + B
D → B

Dabei beschleunigen die Reaktionsprodukte die Geschwindigkeit der Reaktion. Zu Beginn kommt die Reaktion in Gang, weil das Produkt B auch auf anderem Wege gebildet wird, bis soviel davon vorhanden ist, dass es seine katalyische Wirkung entfaltet. Bei vielen industriellen Prozessen spielt die Autokatalyse eine Rolle; man kann eine Reaktion optimieren, indem man dafür Sorge trägt, dass Edukte und Produkte immer in den optimalen Konzentrationen vorhanden sind.

Wie bereits im vorigen Kapitel, betrachten wir die gegenseitigen Abhängigkeiten der Reaktionen und stellen die Differentialgleichungen **(7-10)** für die Konzentrationsverläufe der vier an der Reaktion beteiligten Stoffe auf.

Änderung der Konzentration A:

$$\frac{dC_A}{dt} = -k_1 \cdot C_A \cdot C_B \qquad (7)$$

Änderung der Konzentration B:

$$\frac{dC_B}{dt} = -k_1 \cdot C_A \cdot C_B + k_2 \cdot C_C + k_3 \cdot C_D \qquad (8)$$

Änderung der Konzentration C:

$$\frac{dC_C}{dt} = k_1 \cdot C_A \cdot C_B - k_2 \cdot C_D \qquad (9)$$

Änderung der Konzentration D:

$$\frac{dC_D}{dt} = -k_2 \cdot C_C - k_3 \cdot C_D \qquad (10)$$

Vor einer Umsetzung in einem Simulink-Modell sollte man sich an dieser Stelle Gedanken über den Aufbau machen. Jeder Konzentrationsverlauf ist hier von einer gewissen Anzahl anderer Konzentrationen abhängig. Es werden dazu vier verschiedene Integrationskreisläufe gebraucht. Diese werden wir in der Mitte untereinander anordnen, rechts daneben vier vertikale „Schienen“, auf denen sich die Konzentrationen von A nach D gestaffelt befinden. Auf der linken Seite ordnen wir, ebenfalls in vertikalen Schienen, die Geschwindigkeitskonstanten an. Von diesen Schienen kann nun das jeweils benötigte Signal abgegriffen werden. Das gesamte Simulink-Modell ist in **Abb. 13** zu sehen.

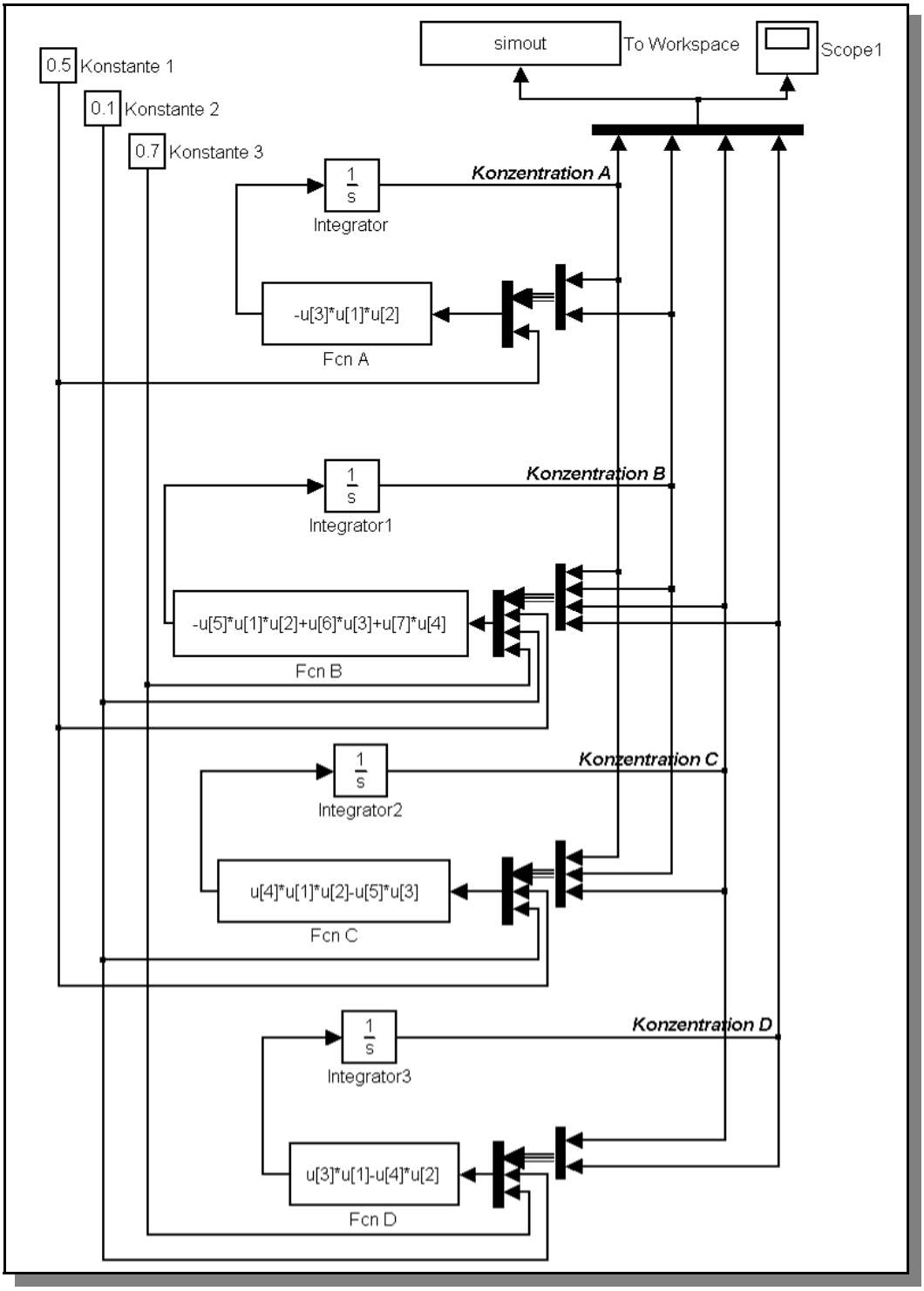


Abb. 13: Modell Autokatalytische Reaktion

Die Ausgabe in einem Simulink-Scope ist in **Abb. 14** zu sehen. Die Bezeichnungen der einzelnen Kurven wurden nachträglich eingefügt, sie werden normalerweise nur verschieden eingefärbt dargestellt.

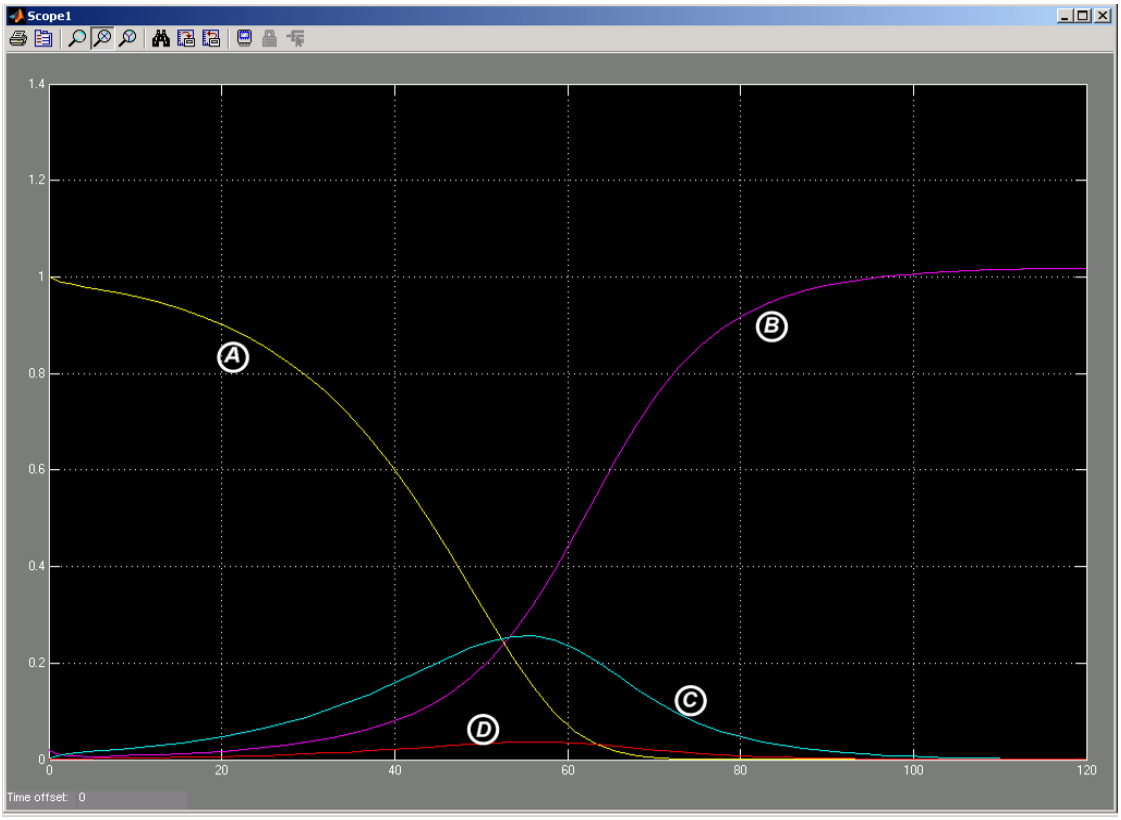


Abb. 14: Simulink-Scope zur Autokatalytischen Reaktion

4.5 Verdünnung im Durchflussbehälter

Aus einer Übungsaufgabe aus der Vorlesung ergab sich folgende Aufgabenstellung:

Die Behälterkaskade (**Abb. 15**) besteht aus drei Flüssigkeitsbehältern mit je 100 Liter Fassungsvermögen. Zum Zeitpunkt t=0 befinden sich im Behälter B1=100 L Wasser in dem 10 kg Salz gelöst sind. In den Behältern B2 und B3 befinden sich je 100 Liter reines Wasser. In den Behälter B1 fließt pro Minute 3 Liter Wasser. Ebenso viel Wasser fließt je Minute vom Behälter B1 in den Behälter B2 und von dort in den Behälter B3. Der Abfluss vom Behälter B3 wird in ein Sammelbecken geleitet. Durch den Überlauf vom Behälter B1 gelangt salzhaltiges Wasser in die nachfolgenden

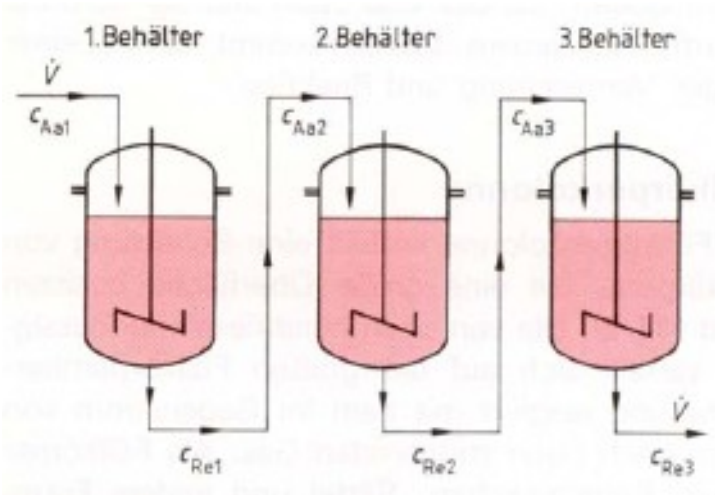


Abb. 15: Behälterkaskade

Behälter. Ein Rührwerk sorgt in jedem Behälter gleichmäßige Konzentrationsverteilung.

Nach 100 Minuten wird für 20 Minuten in den Behälter B1 statt reines Wasser eine Salzlösung mit einer Konzentration von 100 g/L gepumpt – der Volumenstrom bleibt unverändert bei 3 Liter pro Minute.

Stellen Sie das Differentialgleichungssystem auf und simulieren Sie den Konzentrationsverlauf (g/L) in den drei Behältern für einen Zeitraum von 300 Minuten. Ein Impuls kann mit Hilfe des SOURCE-Blocks DISCRETE PULSE GENERATOR simuliert werden. Stellen Sie für den Block folgende Parameter ein:

- Amplitude: 100
- Period: 1000
- Pulse width: 20
- Phase delay: 100
- Sample Time: 1

Zur Lösung der Aufgabe betrachtet man zuerst die Bilanzgrenzen jedes einzelnen Behälters. Die Speicherung der Stoffmenge im Behälter pro Zeit ist dabei die Differenz abgeführter und zugeführter Stoffmenge. Da wir von einer idealen Durchmischung ausgehen, ist die Konzentration im Behälter gleichzeitig auch die Konzentration des abgeführten Stoffes c_{ab}. Darüber hinaus ist der zugeführte Volumenstrom gleich dem abgeführten, womit sich der Term ausklammern und mit **(11)** beschreiben lässt.

$$\frac{dc_{ab}}{dt} = \frac{\dot{V}}{V}(c_{zu} - c_{ab}) \qquad (11)$$

Es ist in diesem Fall zweckmässig, mit der kleinsten Einheit der Zeitachse Minuten zu wählen. Für die Konzentrationsachse wählt man am besten g/L, da man damit nur noch ganzzahlige Werte hat.

In **Abb. 16** ist eine mögliche Lösung der Aufgabenstellung zu sehen. Das Modell ist zwar voll funktionsfähig, jedoch wäre es erheblich anschaulicher, wenn jeder Bilanzraum für sich zusammengefasst wäre. Zu diesem Zweck gibt es *Subsystems.* Dazu markiert man alle Elemente, welche zu einer logischen Einheit gehören. Dann wählt man *Edit – Create Subsystem* oder drückt alternativ [Strg]+[g]. Das optisch wesentlich vereinfachte Modell ist in **Abb. 17** dargestellt. Zur nachträglichen Änderung bzw. zum Betrachten des Subsystems braucht man nur doppelt auf den Block zu klicken.

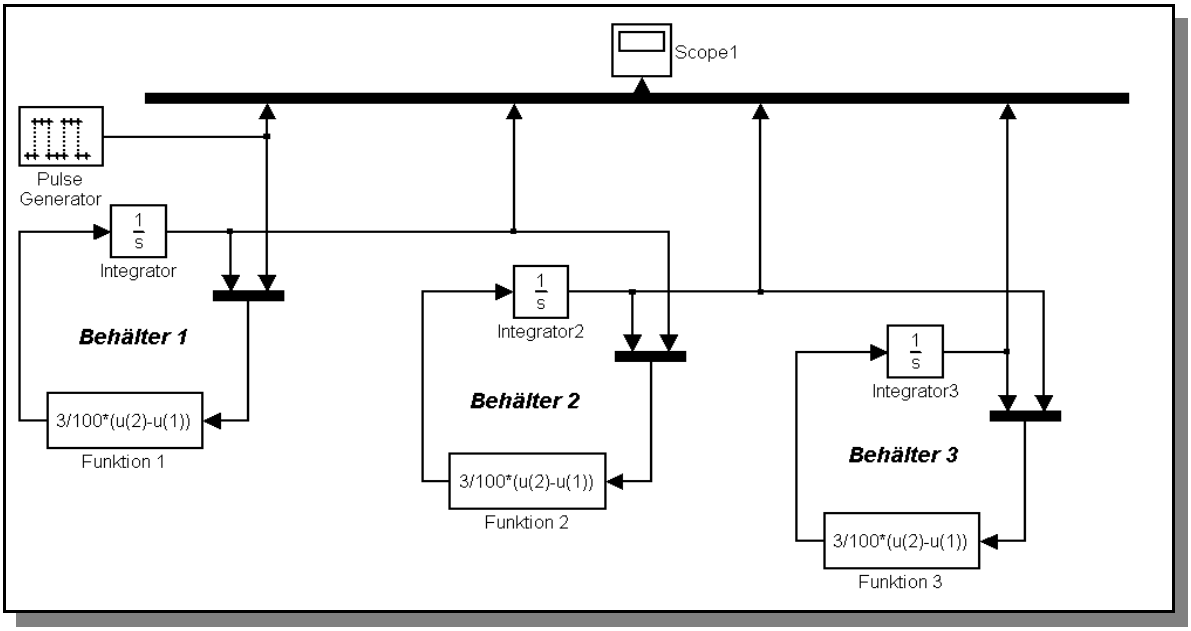


Abb. 16: Modell Behälterkaskade

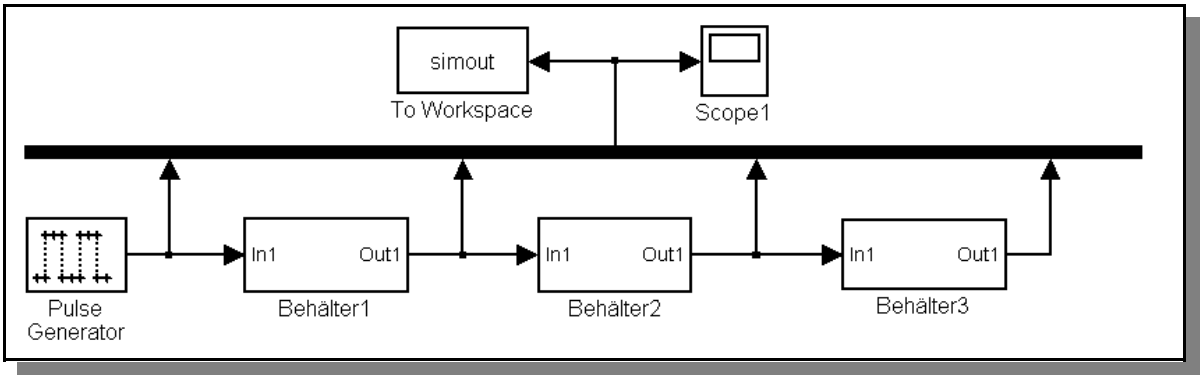


Abb. 17: Behälterkaskade mit Subsystemen

Das Ergebnis bleibt dabei in beiden Fällen natürlich dasselbe. Es ist in **Diagramm 4** dargestellt.

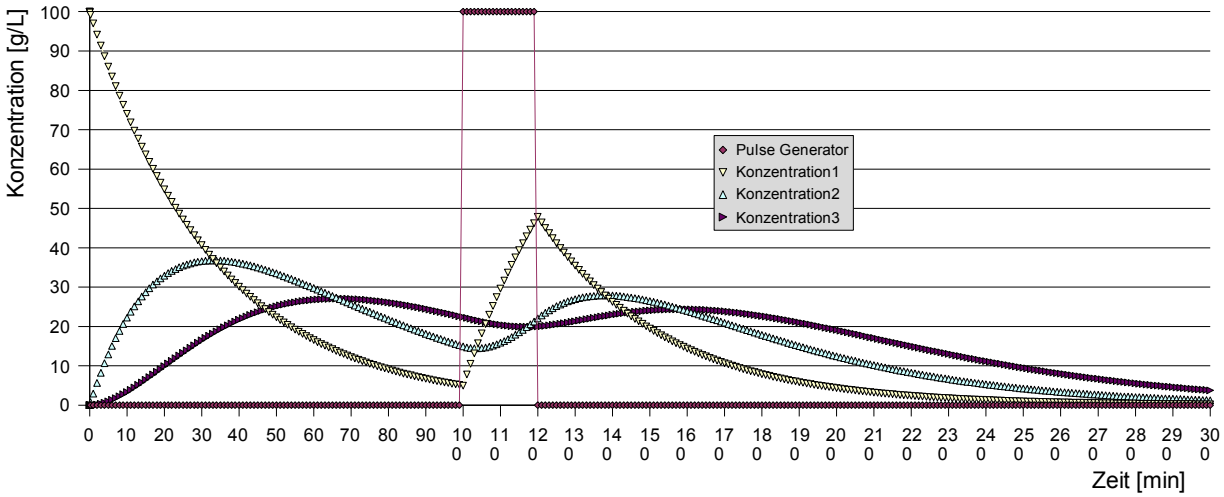


Diagramm 4: Konzentrationsverlauf in einer Behälterkaskade

4.6 Behälterausfluss nach Toricelli

Aus einem offenen Gefäß fließt aus einer Boden- oder Seitenöffnung Flüssigkeit aus. Die Öffnungen liegt um *h* unterhalb des Flüssigkeitsspiegels. Mit der vereinfachenden Annahme, dass der Gefässquerschnitt oben wesenlich größer sein soll als die untere Austrittsöffnung und die Reibung vernachlässigt wird, lässt sich die Austrittsgeschwindigkeit nach **(12)** formulieren. Dies besagt, dass die Flüssigkeit genauso schnell ausströmt wie ein Körper, der aus der Höhe *h* fallen gelassen wird.

$$v = \sqrt{2 \cdot g \cdot h} \qquad (12)$$

Was nun von Interesse sein kann, ist die Veränderung der Füllhöhe im Laufe der Zeit. Dazu wissen wir, dass der Volumenstrom $\dot{V}$ ein Produkt aus Geschwindigkeit *v* und Ausströmquerschnitt *A* ist. Das Volumen *V* des Behälters ist das Produkt seiner Grundfläche *G* und Höhe *h*. Damit lässt sich die Differentialgleichung **(13)** formulieren.

$$\frac{dh}{dt} = -\frac{A}{G} \cdot \sqrt{2g \cdot h} \qquad (13)$$

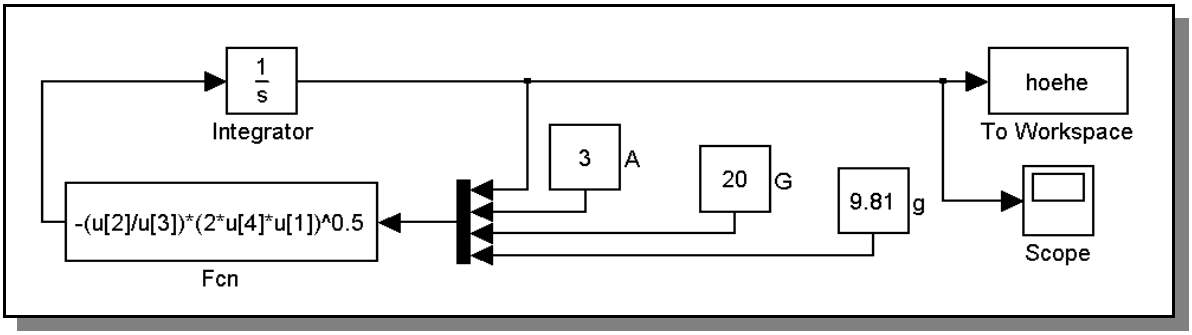


Abb. 18: Modell zur Pegelbestimmung

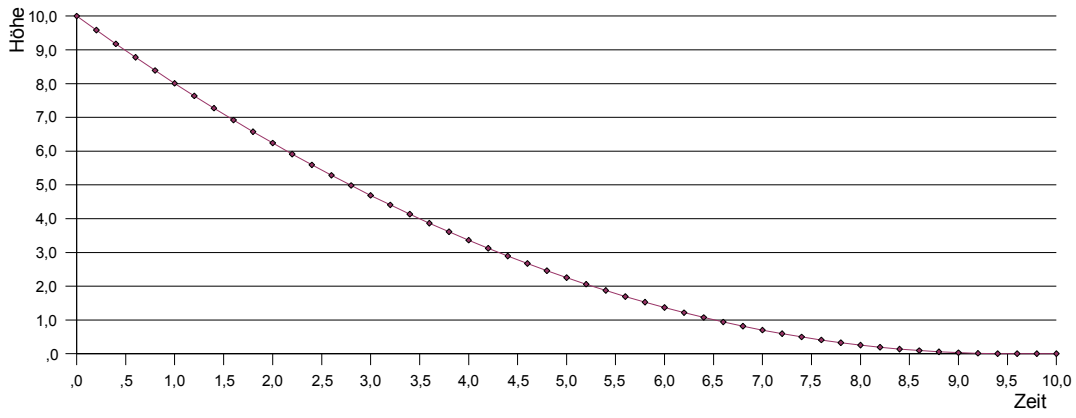


Diagramm 5: Flüssigkeitspegel (qualitativ)

4.7 Natürliche Populationsentwicklung

In einem funktionierenden Ökosystem stellt sich im Laufe der Zeit meist ein natürliches Gleichgewicht zwischen Jäger und Gejagtem ein, welches gewissen periodische Schwankungen unterliegt. In einem sehr vereinfachten System wollen wir hier das Verhältnis zwischen dem Hasen und seinem Rivalen Fuchs untersuchen.

Die Population der Hasen ist direkt proportional zur Geburtenrate, ebenso nimmt sie auch direkt proportional mit der natürlichen Todesrate wieder ab. Hasen sterben aber nicht nur an Altersschwäche, und es ist umso wahrscheinlicher, dass ein Hase von einem Fuchs gefressen wird je mehr es von beiden gibt. Mathematisch formuliert führt das zu Gleichung **(14)**.

$$\frac{dH}{dt} = a \cdot H - b \cdot H - c \cdot F \cdot H \qquad (14)$$

a: Geburtenrate Hasen
b: Natürliche Todesrate Hasen
c: Todesrate Hasen durch Füchse

Füchse wiederum bekommen umso mehr Nachwuchs, je besser das Nahrungsangebot in Form von Hasenfleisch ist und je weniger sie mit ihren Artgenossen teilen müssen. Und zu guter Letzt sterben auch mehr Füchse wenn es mehr Füchse gibt. Die Entwicklung der Füchse lässt sich somit mit **(15)** beschreiben.

$$\frac{dF}{dt} = d \cdot \frac{H}{F} - e \cdot F \qquad (15)$$

d: Geburtenrate Füchse
e: Todesrate Füchse

Im Simulink-Modell (**Abb. 19**) sind Hasen auf der linken und Füchse auf der rechten Seite angeordnet. Die Faktoren, die auf ihre Populationsentwicklung gegenseitig Einfluss nehmen, werden in der Mitte ausgetauscht und gleichzeitig zum Scope weitergeleitet. Im Wesentlichen handelt es sich dabei um den Ausgang der beiden Integratoren, welcher den jeweiligen Populationsstand liefert. Die Konstanten werden über beschriftete `Constant`-Blöcke in die Funktionen eingebracht, was nachträgliche Änderungen wesentlich vereinfacht.

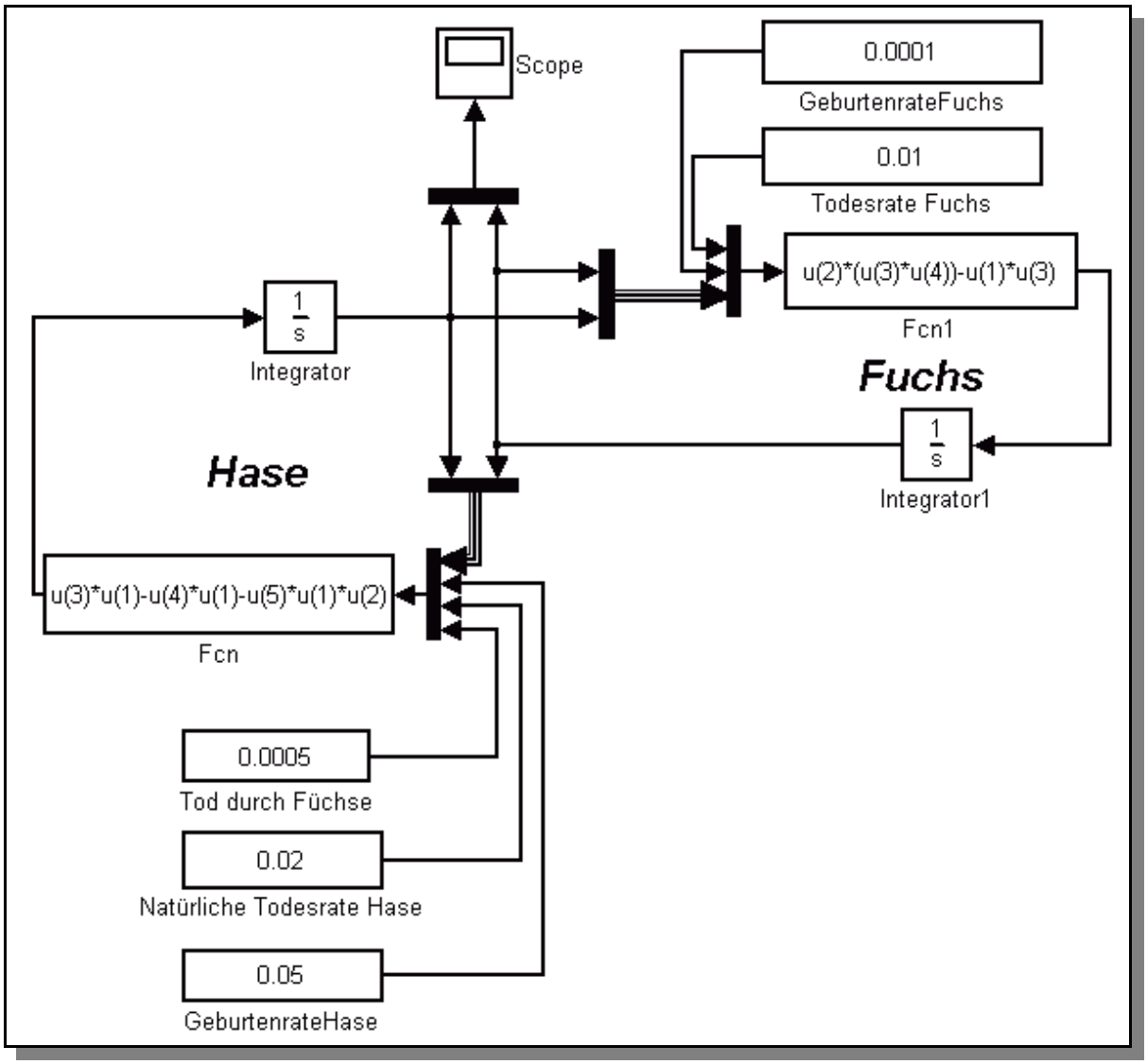


Abb. 19: Modell Populationsentwicklung Hase-Fuchs

Bei einem ersten Testversuch wird das Ergebnis recht ernüchternd sein: Man sieht lediglich zwei waagrechte Geraden. Da in den beiden Termen **(14)** und **(15)** *H* und *F* immer in einem Produkt vorkommen, benötigen wir eine Startbedingung des Integrators > 0. Dies deckt sich auch mit den Erfahrungen aus der Natur: Wo es keine Füchse gibt, können sie auch nicht einfach so aus dem nichts auftauchen. Um der Natur auf die Sprünge zu helfen, setzen wir bei dem Integrator der Hasen `Initial Condition` auf 100 und bei den Füchsen auf 20. Das Ergebnis ist mit dem Matlab-eigenen Plotter in **Diagramm 6** dargestellt.

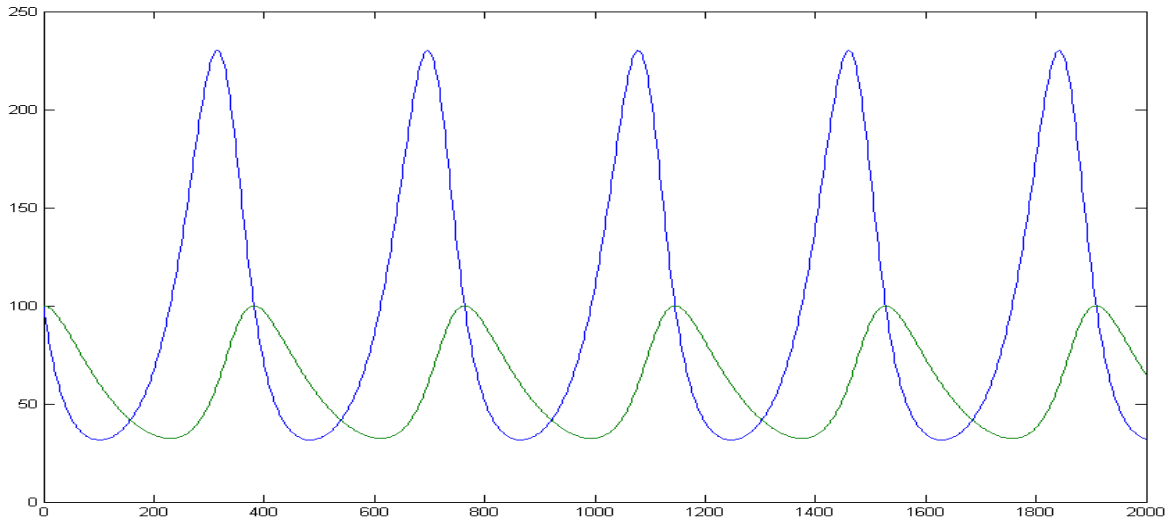


Diagramm 6: natürliche Populationsentwicklung

Was passiert, wenn es nur Hasen und keine Füchse gibt, erfährt man wenn man *Initial Condition* nur bei den Hasen ungleich null setzt. Dies gibt dann in etwa die Situation in Australien zur großen Hasenschwemme wieder.

5 Fazit

Am 26. Oktober 2005 machte sich die Raumsonde Venus Express auf eine 135 Millionen Kilometer lange Reise. Testlauf gab es dafür keinen, lediglich einige hunderttausend Simulationsläufe mit Matlab Simulink. Das macht deutlich, dass das Potential dieses Programms mit dem Lösen einfacher Differentialgleichungen noch lange nicht erschöpft ist.

Dennoch erfüllt es diesen Zweck sehr gut, und ist erst die Differentialgleichung aufgestellt, ist es nur noch eine Frage eines möglichst sauberen, strukturierten Vorgehens um das passende Diagramm für seine Technische Dokumentation zu erhalten. Es liegt im Sinne des Autors, dass das vorliegende Dokument dazu ein wenig behilflich war.

Christian Mayr
Matr.Nr.: 277719

vub3 ws 2005/06
Verfahrens- und Umwelttechnik
FH Konstanz