Nachbildung einer Schwellenwertfunktion mit einem chemischen Analogcomputermodell

Friedrich-Schiller-Universität Jena.
Fakultät Mathematik und Informatik.
Modul Molekulare Algorithmen 2020

Tobias Jung, Ferdinand Schreck

13. Juli 2020

Inhaltsverzeichnis

1 Einführung und Problemstellung

Die Veranstaltung „Molekulare Algorithmen“ im Sommersemester 2020 an derFriedrich-Schiller-Universität Jena beschäftigte sich mit Chemischem Rechnen und den dabei relevanten Spezialbereichen der Chemischen Analogcomputermodelle, Chemischen Digitalcomputermodelle und DNA-Computing. Das Projekt, welches den Inhalt dieses Dokumentes bestimmt, stammt aus dem ersten der drei genannten Bereiche.

Ziel ist die Entwicklung und Simulation eines chemischen Analogcomputermodells zur Nachbildung einer konkreten Schwellenschwertfunktion der Form

$$f(x) = \begin{cases} 0 & x \leq 3 \\ 1 & \text{sonst} \end{cases} \tag{1.1}$$

Dabei muss die Funktion zunächst geeignet angenähert werden, sodass die absolute Abweichung der definierten Funktion gegenüber f an den Stellen $x = 2,9$ und $x = 3,1$ kleiner als 10^{-4} bleibt (a). Im Anschluss soll die Funktion als chemisches Analogcomputermodell implementiert (b) und ihr Funktionsverlauf für $0 \leq x \leq 6$ dargestellt werden (c).

Wir folgen dem Hinweis, der mit der Ausgabe der Projektthemen gegeben wurde und nähern die Funktion 1.1 mit Hilfe einer Sigmoidfunktion an.

2 Umsetzung mit einer einfachen Sigmoidfunktion

2.1 Ansatz mit einer einfachen Sigmoidfunktion

2.1.1 Funktion

Die Berechnung der Schwellenwertfunktion bzw. Schwanenhalsfunktion soll nun auf Basis einer einfachen Sigmoidfunktion erfolgen. Der Blog [1] beschreibt, wie solch eine Formel hergeleitet werden kann. Aus den dort getroffenen Erörterungen ergibt sich folgende Formel

$$h(x) = \frac{-k' \cdot |x|}{-k' - |x| + 1} \cdot \frac{x}{|x|} \tag{2.1}$$

k ist dabei als Parameter zu verstehen, durch den die Steilheit des Schwanenhalses eingestellt werden kann.

2.1.2 Parameteranpassung

Die Wahl des k' hängt an Bedingung (a). Die Funktion 2.1 ist punktsymmetrisch wie man mit

$$f(-x) = \frac{-k' \cdot |-x|}{-k' - |-x| + 1} \cdot \frac{-x}{|-x|} = \frac{-k' \cdot |x|}{-k' - |x| + 1} \cdot \frac{-x}{|x|} = -\frac{-k' \cdot |x|}{-k' - |x| + 1} \cdot \frac{x}{|x|} = -f(x) \tag{2.2}$$

sieht. Die Funktion weicht allerdings noch in zwei Punkten von den Anforderungen ab: Zunächst „springt“ sie nicht wie gefordert zwischen Null und Eins, sondern zwischen minus k' und k'. Außerdem „springt“ sie an der Stelle $x = 0$, nicht wie gefordert an der Stelle $x = 3$. Integriert man diese Anforderungen in die Formel, so erhält man nach Vereinfachung:

$$h(x) = 0,5 - \frac{0,5(x-3)}{-(1-k') - |x-3|} \tag{2.3}$$

Nun genügt es festzustellen, für welches k' $h(3,1)$ um weniger als 10^{-4} von dem tatsächlichen Wert 1 (nach 1.1) abweicht.

$$1 - (0.5 - \frac{0,5(3,1-3)}{-(1-k') - |3,1-3|}) \leq 10^{-4} \tag{2.4}$$

$$0,5 + \frac{0,05}{k'-1,1} \leq 10^{-4} \qquad | -0,5 \tag{2.5}$$

$$\frac{0,05}{k'-1,1} \leq 10^{-4} - 0,5 \qquad | : 0,05 \tag{2.6}$$

$$\frac{1}{k'-1,1} \leq \frac{10^{-4} - 0,5}{0,05} \tag{2.7}$$

$$\Rightarrow k' < 1,1 \tag{2.8}$$

$$\tag{2.9}$$

$$\frac{1}{k'-1,1} \leq \frac{10^{-4} - 0,5}{0,05} \qquad | \cdot (k'-1,1) \tag{2.10}$$

$$\frac{1}{k'-1,1} \geq -9,998 \cdot (k'-1,1) \qquad | : -9,998 \tag{2.11}$$

$$\frac{1}{-9,998} \leq k'-1,1 \qquad | + 1,1 \tag{2.12}$$

$$\frac{1}{-9,998} + 1,1 \leq k' \tag{2.13}$$

$$k' \geq 0,99998 \tag{2.14}$$

$$\tag{2.15}$$

Die Formeln 2.8 und 2.14 können zusammengesetzt werden zu

$$0,99998 \leq k' < 1,1 \tag{2.16}$$

Man beachte den Relationszeichenwechsel in 2.11 und 2.12 aufrgund von Division einer negativen Zahl. Das Ergebnis in 2.14 ist aufgerundet, so dass die resultierende Ungleichung in 2.16 tatsächlich gilt. Wir wählen $k' = 0,99999$ um Bedingung (a) sicherzustellen und erhalten

$$h(x) = 0,5 - \frac{0,5(x-3)}{-0,00001 - |x-3|} \tag{2.17}$$

Forthin bezeichne k den Faktor $0,00001$.

2.2 Implementierung der einfachen Sigmoidfunktion h

2.2.1 Allgemeines

Zur Implementierung der in Abschnitt 2.1 ermittelten Funktion wird die Simulationssoftware COPASI [2] verwendet. Die entsprechende Datei finden Sie als simpleSigmoid.cps (mit COPASI 4.28).

Die Implementierungen sollen sich zum Großteil auf bereits ausgearbeitete Reaktionsnetzwerke aus [3] stützen, da ihre Funktionsweise dort bereits erklärt und belegt ist. Allerdings ergeben sich weitere Herausforderungen, auf die, neben der Beschreibung der generellen Implementierung, speziell eingegangen wird.

Desweiteren werden alle Werte wie Funktionsvariablen, Zwischenergebnisse und konstante Werte als Spezies im biochemischen Reaktionsnetzwerk implementiert. Deren Konzentrationen spiegeln direkt mathematische Werte wieder. Da vorwiegend Grundoperationen aus [3] benutzt werden, bleiben Eingabewerte für Teilrechnungen erhalten. Somit müssen konstante Werte nicht als „fixed“ markiert werden.

2.2.2 Vollständige Subtraktion

Beim Betrachten der Formel 2.17 fällt zunächst auf, dass Zahlen der reellen Achse für die Berechnung relevant werden können. Insbesondere ist damit gemeint, dass negative und positive Zahlenwerte betrachtet werden müssen. Um dies zu realisieren, wurde die Betrag-Vorzeichen-Darstellung, wie Hendrik Happe sie in [4] beschreibt, herangezogen. Dabei wird jede Variable durch drei Spezies repräsentiert: eine Spezies, die den Betrag der Zahl definiert (im Folgenden werden diese Variablen mit dem Suffix _amount gekennzeichnet) sowie zwei als binär angenommene Variablen mit den Suffixen _positive und _negative, die jeweils ein positives oder negatives Vorzeichen der Variable kennzeichnen.

In der oben definierten Formel taucht die vollständige Subtraktion an zwei Stellen auf: Im Nenner für den Term $-k - |Diff|$, sowie im Zähler und im Nenner für den Term $x–3 = Diff$. Da im Rahmen von [3] lediglich die nicht negative Subtraktion definiert wurde, soll hier eine Möglichkeit der Berechnung der vollständigen Subtraktion aufgezeigt werden. In diesem Dokument sollen dabei nur die für diesen Ansatz relevanten Kombinationen (positiv - positiv und negativ - positiv) betrachtet werden.

Für den Fall $-k–|Diff|$ ergibt sich dabei die Addition der Beträge der beiden Variablen, versehen mit einem negativen Vorzeichen.

Der Fall $x–g$ beziehungsweise $x - 3$ gestaltet sich schwieriger. Für Die Werte $x > 3$ würde die nicht negative Subtraktion zur Berechnung ausreichen, für Werte $x < 3$ würden damit lediglich Ergebnisse nahe Null aus der Berechnung resultieren. Allerdings fällt auf, dass das Ergebnis der vollständigen Berechnung stets denjenigen Betrag erhält, der durch die Subtraktion der kleineren von der größeren Variable errechnet werden kann. Wenn die Berechnung also immer in der Form „Betrag der größeren Variable – Betrag der kleineren Variable“ definiert wird, dann wird der Betrag des Ergebnisses stets korrekt berechnet.

Um dies umzusetzen, werden die Beträge der beiden Eingabevariablen x und g in zwei weitere Hilfsspezies *Minuend* und *Subtrahend* überführt. Die eigentliche Subtraktion erfolgt dann stets mithilfe dieser Variablen in der Form $Diff = Minuend - Subtrahend$. Um diese Überführung in korrekter Weise durchzuführen, muss die Information vorhanden sein, welche der Eingabevariablen größer ist. Diese Information wird in den beiden als binär angenommenen Variablen $x < g$ und $x > g$ codiert. Für den Spezialfall $x = g$ konvergieren beide Variablen gegen $0,5$. Dadurch werden beide Variablen „zur Hälfte“ in den Subtrahenden beziehungsweise Minuenden überführt. Da die Konzentrationen aber identisch sind, ergeben diese beiden Teilreaktionen zusammen die erwünschte Überführung

und die Konzentrationen landen in den korrekten „Zielvariablen“. Mit der Berechnung des Inhaltes dieser Variablen wird sich der nächste Abschnitt beschäftigen. Hier soll zunächst davon ausgegangen werden, dass diese Variablen korrekt belegt sind.

Aus den Informationen der Positivität beziehungsweise Negativität der Variablen ergibt sich mit der Information, welche der beiden Eingabevariablen größer ist, die Hilfsvariable, in die der jeweilige Betrag überführt werden muss.

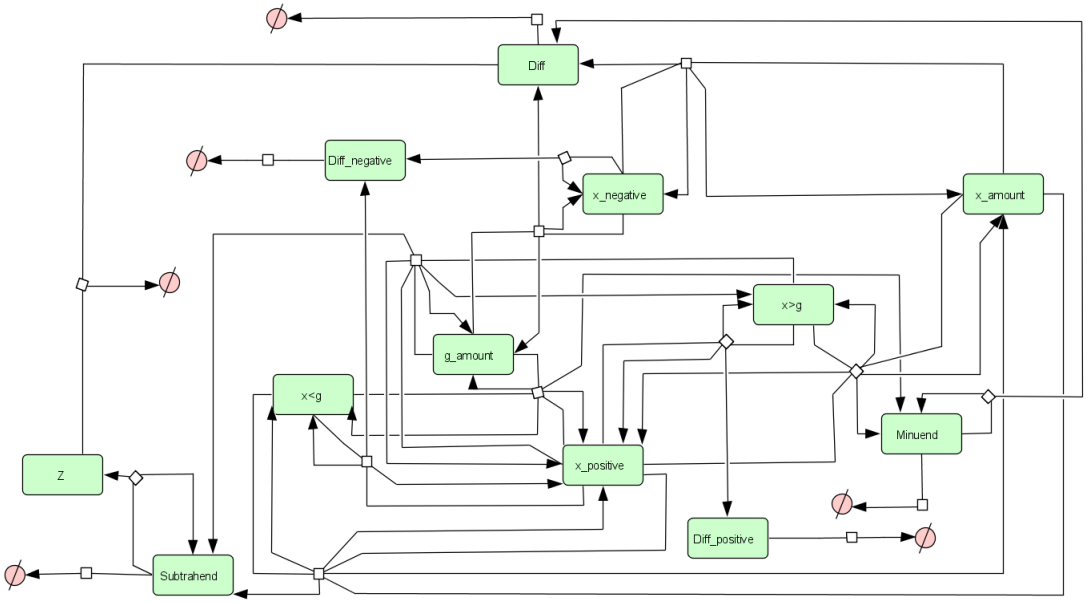

Abbildung 2.1: Reaktionsnetzwerk zur Berechnung potentiell negativer Subtraktionen

Zu guter Letzt muss nun noch das Vorzeichen der Differenz berechnet werden. Im Fall der Berechnung von $-k–|Diff|$ wurde bereits definiert, dass die Differenz dieser beiden Variablen auf jeden Fall negativ ist. Im Fall $x–g$ hängt das Vorzeichen der Differenz vom größer-kleiner-Vergleich der beiden Variablen ab. Ist der (ursprüngliche) Subtrahend, also das g, größer, so wird das Ergebnis negativ sein. Aus dem umgekehrten Fall resultiert dann entsprechend ein positives Ergebnis. Zusätzlich wurde hier noch der Fall definiert, dass der Minuend (also in diesem Falle x) negativ ist. Dieser ist allerdings nur relevant für die Subtraktion von $-k–|Diff|$ und findet für $x–g$ keine Anwendung.

2.2.3 Vergleich zweier Zahlen

Wie im vorigen Abschnitt bereits erörtert, basiert die vollständige Subtraktion auf einem größer-kleiner-Vergleich der Beträge zweier Eingabevariablen, im Konkreten ist die Rede von x und k. Das Ergebnis dieses Vergleiches soll in die Spezies $x < k$ und $x > k$ überführt werden, auf Basis derer die vollständige Subtraktion ausgeführt werden kann. In Bezugnahme auf die Korrespondenz vom 25.06.2020 lautet die Formel, die dabei zur

Anwendung kommt:

$$z = \frac{(g - x) \cdot a + (x - g) \cdot b}{(g - x) + (x - g)} \tag{2.18}$$

Diese ist in der Form if $(x < g)$ then $z = a$ else $z = b$ zu interpretieren. Da hier ein binärer Vergleich angestellt wird, wird a also auf 1 und b auf 0 eingestellt. Damit fällt der Term $(x - g) \cdot b$ weg. Für den umgekehrten Vergleich sind lediglich die Variablen miteinander zu vertauschen.

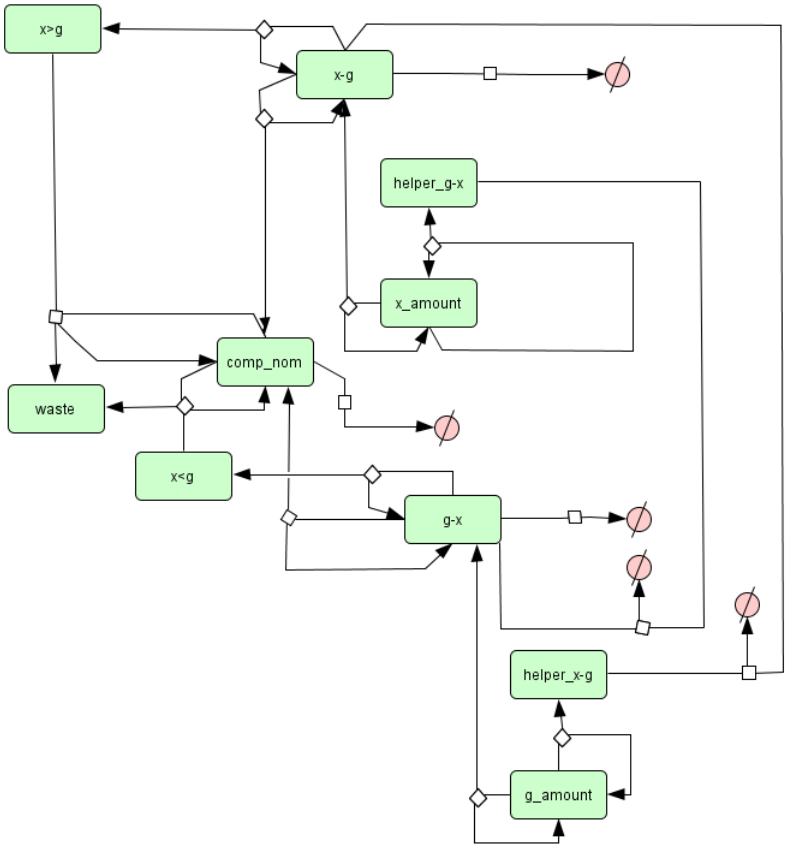

Abbildung 2.2: Reaktionsnetzwerk zum größer-kleiner Vergleich zweier Eingabevariablen x und g.

Zuletzt muss noch der Vergleich identischer Zahlen betrachtet werden. Mit dem vorliegenden Modell konvergieren die beiden Ausgabespezies $x > g$ und $x < g$ für $x = g$ gegen $0,5$. Wie bereits beschrieben, ist dies auch für die Berechnung der negativen Subtraktion zielführend und soll daher so belassen werden.

2.2.4 Berechnung der gesamten Gleichung

Nachdem das benötigte Handwerkszeug behandelt ist, soll zunächst der Bruch in der Gleichung berechnet werden. Dazu werden Zähler und Nenner isoliert betrachtet und schließlich durcheinander geteilt. Hier soll zuerst der Zähler berechnet werden. Zur Erinnerung: dessen Form lautet $j \cdot (x–g)$ beziehungsweise $0,5 \cdot (x–3)$

Die Berechnung von $x–3$ wurde bereits ausführlich im Abschnitt 2.2.2 behandelt und resultiert in der Spezies *Diff*. Für die Berechnung der Multiplikation wird nun zunächst die Berechnung aus [3] herangezogen. Diese muss im Folgenden allerdings noch um die Berechnung des Vorzeichens der Ergebnisses erweitert werden. Diese ist bei Happe [4] wie folgt definiert:

$$\begin{aligned} [Y\_positive] &= [X1\_positive] \cdot [X2\_positive] + [X1\_negative] \cdot [X2\_negative] \\ [Y\_negative] &= [X1\_positive] \cdot [X2\_negative] + [X1\_negative] \cdot [X2\_positive] \end{aligned} \tag{2.19}$$

Setzt man nun für $X1$ j ein und für $X2$ *Diff*, dann ist die Formel entsprechend adaptiert. Das Ergebnis wird in der Spezieskonzentration von *numerator* (-_amount, -_positive, -_negative) gespeichert.

Als nächstes soll der Nenner berechnet werden. Er berechnet sich nach der Formel: $-0.000001-|x-3|$ beziehungsweise $-k-|x-g|$. Wie bereits im Abschnitt zur vollständigen Subtraktion beschrieben, ist die Subtraktion von k und der Differenz von x und g lediglich eine Addition dieser Komponenten mit einem Ergebnis mit negativem Vorzeichen. Die Addition ist nach der in [3] definierten Addition ausgeführt und wird in der Spezies nominator_amount gespeichert. Das negative Vorzeichen ist durch das initiale Setzen der Stoffkonzentration von nominator_negative auf 1 umgesetzt. Nun sind also Zähler und Nenner berechnet. Als nächstes soll der gesamte Bruch berechnet werden. Zur Berechnung des Betrages wird wieder die Formel aus [3] herangezogen und auf die Spezies *nominator_amount* und *numerator_amount* angewendet. Auch hier muss wieder das resultierende Vorzeichen berechnet werden. Erneut werden dazu die Formeln nach Happe [4] herangezogen. Die Positivität und Negativität des Ergebnisses wird auf die selbe Weise berechnet, wie dies zuvor bei der Multiplikation geschehen ist. Nun wird allerdings die Spezies $X1$ durch die Spezies *numerator* und die Spezies $X2$ durch die Spezies *nominator* substituiert. Zum Schluss bleibt noch die Subtraktion $0,5–\frac{numerator}{nominator}$. Dabei kommt es der Berechnung zu Gute, dass das Ergebnis der Funktion minimal Null werden kann. Der Betrag des Bruches kann also nicht kleiner werden als $0,5$. Mit dieser Erkenntnis kann die aufwändige vollständige Subtraktion umgangen werden, da die nichtnegative Subtraktion zur Berechnung ausreicht.

2.3 Bewertung des einfachen Ansatzes

Mit diesem Ansatz lässt sich der y-Wert für ein gegebenes x prinzipiell berechnen. Die chemisch berechnete Funktion soll hier mit $cAlt(x)$ bezeichnet werden, um deren Ergebnisse von den „realen“ Werten der Funktion h unterscheiden zu können.

Prinzipiell ergibt sich bei diesem Ansatz der chemischen Berechnung ein Problem: Die Subtraktion zweier Spezies konvergiert nur sehr langsam gegen das tatsächliche Ergebnis. Diese wird in diesem Modell an vielen Stellen eingesetzt: Im Zähler, im Nenner, die Subtraktion des Bruches von 0,5 , sowie im Vergleich der beiden Variablen x und g. Aus dieser vielfachen Verwendung der Subtraktion ergibt sich eine recht große Ungenauigkeit für kleine Zeitintervalle der Berechnung. Im Umkehrschluss muss das Intervall zur Berechnung von y also sehr groß eingestellt werden.

Um eine annähernd ausreichende Genauigkeit zu erreichen, wurde die Laufzeit des Modells zunächst auf 10 Millionen Sekunden mit einer Intervallgröße von 1 Sekunde eingestellt. Dies entspricht in etwa einer Modelllaufzeit von 57 Tagen. An der Stelle $x = 2,9$ ergibt sich damit eine Abweichung der Funktion $cAlt(x)$ gegenüber der Funktion $f(x)$ um etwa $7,581 \cdot 10^{-4}$. An der Stelle $x = 3,1$ ergibt sich eine Abweichung um etwa $3,49 \cdot 10^{-4}$. Damit wird die geforderte Genauigkeit von weniger als $1 \cdot 10^{-4}$ also noch nicht erreicht. Um dieser Anforderung näher zu kommen muss also, aufgrund der „Trägheit“ der Subtraktion, die Modelllaufzeit erhöht werden. Im Folgenden sollen die Ergebnisse für eine entsprechende Laufzeit von 20 Millionen Sekunden diskutiert werden. Dies entspricht in etwa 115 Tagen. An der Stelle $x = 2,9$ ergibt sich damit eine Abweichung der Stoffkonzentration vom Wert der Funktion $f(x)$ um gerundete $4,951 \cdot 10^{-4}$ und um $2 \cdot 10^{-2}$ an der Stelle $x = 3,1$. Auch diese Modelllaufzeit reicht also nicht aus, um die geforderte Genauigkeit zu erreichen.

x	f(x)	h(x)	cAlt(x), 10 Mio. S	cAlt(x), 20 Mio S
2	0	$0,049999500005 \cdot 10^{-4}$	$5,036442061 \cdot 10^{-4}$	$3,562178267 \cdot 10^{-4}$
2,9	0	$0,4999500049995 \cdot 10^{-4}$	$7,58139 \cdot 10^{-4}$	$4,95103 \cdot 10^{-4}$
3	0	0,5	0,4999997479	0,4999998735
3,1	1	0,9999500049995	0,999651	0,9998
4	1	0,999995	0,9999920001	0,9999935001

Tabelle 2.1: Funktionswerte der verschiedenen Funktionen und Modelllaufzeiten

x	cAlt(x), 10 Mio. S	cAlt(x), 20 Mio S
2	$5,036442061 \cdot 10^{-4}$	$3,562178267 \cdot 10^{-4}$
2,9	$7,58139 \cdot 10^{-4}$	$4,95103 \cdot 10^{-4}$
3	0,4999997479	0,4999998735
3,1	$3,49 \cdot 10^{-4}$	$2 \cdot 10^{-2}$
4	$0,079999 \cdot 10^{-4}$	$0,064999 \cdot 10^{-4}$

Tabelle 2.2: Absolute Abweichungen des chemisch berechneten Wertes gegenüber f

x	cAlt(x), 10 Mio. S	cAlt(x), 20 Mio S
2	$4,986442560995 \cdot 10^{-4}$	$3,512178766995 \cdot 10^{-4}$
2,9	$7,0814399950005 \cdot 10^{-4}$	$4,4513499950005 \cdot 10^{-4}$
3	$0,002521 \cdot 10^{-4}$	$0,001265 \cdot 10^{-4}$
3,1	$2,990049995 \cdot 10^{-4}$	$1,500049995 \cdot 10^{-4}$
4	$0,029999 \cdot 10^{-4}$	$0,014999 \cdot 10^{-4}$

Tabelle 2.3: Absolute Abweichungen des chemisch berechneten Wertes gegenüber h

Jedoch konvergiert der Wert für y immer langsamer gegen das exakte Ergebnis. Daher kann die gewünschte Genauigkeit in COPASI nicht erreicht werden, denn bei Einstellung eines entsprechenden hohen Simulationsdauer-Parameters stürzt die Simulation vor Beendigung ab.

An den oben aufgezeigten Werten lässt sich klar die Schwierigkeit des Modells ablesen, Werte nahe Null korrekt zu berechnen, da die Werte für 3,1 eine wesentlich kleinere Abweichung vom Wert $f(x)$ aufweisen als die für 2,9. Dieses Problem „erbt“ das Modell aus den vielen Subtraktionen, auf denen es basiert.

Als weiterer Nachteil dieses Modells stellt sich heraus, dass das Modell zu „träge“ ist, um die Funktion im simulierten Zeitverlauf darzustellen. Wird also die Stoffkonzentration der Eingabevariable x über die Zeit erhöht, so konvergiert die Konzentration von Y nur sehr langsam gegen Eins und nähert sich dieser erst bei $x = 6$ an. Dies ist damit zu begründen, dass die relevanten Parameterkonzentrationen im Intervall 0 bis etwa 2,8 bereits in einen nahezu stationären Zustand übergegangen sind und das System somit viel Zeit braucht, um diese Stationarität zu überwinden. Daher wurden zum Plotten der Funktion 64 einzelne Funktionswerte im Intervall zwischen 0 und 6 berechnet und mit R dargestellt. Dabei wurde x jeweils um 0,1 erhöht, an der kritischen Stelle der Funktion, also um $x = 3$ herum, wurde dieses Intervall auf 0,05 verringert, um den „Sprung“ der Funktion besser beschreiben zu können. Für die Berechnung wurde eine „verträgliche“ Laufzeit von 100 000 Sekunden (etwa 28 Stunden) mit einer Intervallgröße von 0,1 Sekunden gewählt. Das resultierende Diagramm ist im Folgenden aufgezeigt.

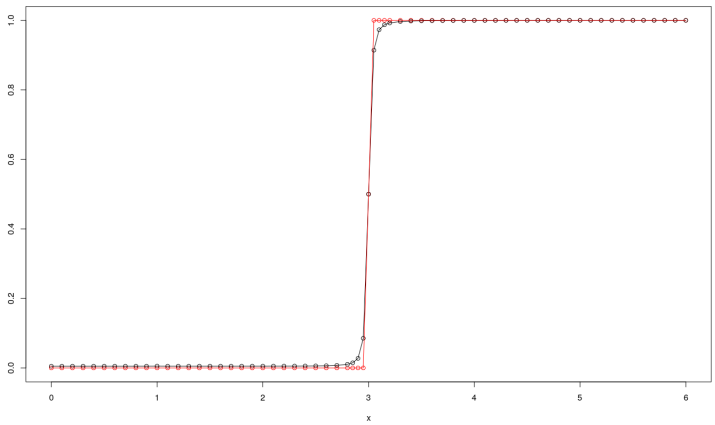

Abbildung 2.3: cAlt(x) in schwarz, h(x) in rot

Die Tabelle 2.1 zeigt einen weiteren Nachteil dieses Ansatzes auf: Zwar wurde im ersten Abschnitt zur Herleitung der Formel definiert, dass die Funktion $h(x)$ gegen Eins beziehungsweise gegen Null konvergiert, dies geschieht allerdings erst in der Unendlichkeit. Für Werte zwischen 0 und 2,9, beziehungsweise 3,1 und 6 werden nur näherungsweise die Ergebnisse Eins beziehungsweise Null erzielt.

Somit ist das Modell zwar theoretisch geeignet, die gewünschte Funktion zu berechnen, praktisch aufgrund der zu hohen Ungenauigkeit aber nicht auf eine Weise durchführbar, die den Anforderungen genügen würde. Aufgrund der vielseitigen Probleme, mit welchen dieser Ansatz aufwartet, namentlich die hohe Berechnungsdauer respektive hohe Ungenauigkeit, Konvergenz gegen Null und Eins erst in der Unendlichkeit und Trägheit im Zeitverlauf, soll im Folgenden ein alternativer Ansatz erörtert werden, der diese Probleme angehen soll.

3 Umsetzung mit einer logistischen Funktion

Wie in 2.3 angemerkt, sind die Laufzeiten an jener Methode das größte Problem. Damit wird der Einsatz eines solchen chemischen Modells in der Praxis uninteressanter. Es soll nun versucht werden ein Modell zu finden, dass in der Laufzeit deutlich besser abschneidet und ebenfalls Werte von vertretbaren Genauigkeiten (im Idealfall nach Bedingung (a)) liefert. Dazu wird ein Ansatz, der die Exponentialfunktion beinhaltet und an die logistische Funktion angelehnt ist, gewählt.

3.1 Ansatz mit einer logistischen Funktion

3.1.1 Funktion

Für die logistische Funktion wird folgende Form gewählt (nach[5])

$$\frac{1}{1+e^{-x}} \tag{3.1}$$

Hierbei fehlen noch zwei wichtige Adaptionen. Zum einen muss die Funktion in Richtung der positiven x-Achse verschoben werden. Dies kann bewerkstellig werden, indem statt x im Exponenten der e - Funktion $x-3$ verwendet wird. Desweiteren wird ein Parameter k benötigt, der die Steigung der Funktion derart beeinflussen kann, dass Bedingung (a) eingehalten werden kann. Dieser Paramter findet sich im Exponenten vor *x-3*, wie der allgemeinen logistischen Funktion (nach [6]) entnommen werden kann.

$$G \cdot \frac{1}{1+e^{-k \cdot G \cdot t}(\frac{G}{f(0)}-1)} \tag{3.2}$$

Die Zeit wird mit der Eingabegröße x identifiziert. G steht für die obere Schranke von $f(t)$ und ist für diesen Fall 1. $f(0)$ hingegen $\frac{1}{2}$. Dann bleibt:

$$\frac{1}{1+e^{-k \cdot (x-3)}} \tag{3.3}$$

Der einzige freie Paramter ist k, der im nächsten Abschnitt angepasst wird, um Bedingung (a) der Aufgabe in 1 zu erfüllen.

3.1.2 Parameteranpassung

Da die logistische Funktion nur eine verschobene Tangens - hyperbolicus Funktion ist, erbt die Funktion 3.3 ihre Punktsymmetrie. Damit genügt es festzustellen, für welches k $g(3,1)$ um weniger als 10^{-4} von dem tatsächlichen Wert 1 (nach 1.1) abweicht. Es gilt die folgende Ungleichung zu lösen:

$$1 - \frac{1}{1+e^{-k\cdot(3,1-3)}} \leq 10^{-4} \qquad | -1 \tag{3.4}$$

$$-\frac{1}{1+e^{-k\cdot 0,1}} \leq -0,9999 \qquad | : (-1) \tag{3.5}$$

$$\frac{1}{1+e^{-k\cdot 0,1}} \geq 0,9999 \qquad | \cdot (1+e^{-k\cdot 0,1}) \tag{3.6}$$

$$1 \geq 0,9999 \cdot (1+e^{-k\cdot 0,1}) \qquad | : 0,9999 \tag{3.7}$$

$$\frac{1}{0,9999} \geq 1+e^{-k\cdot 0,1} \qquad | -1 \tag{3.8}$$

$$\frac{1}{0,9999} - 1 \geq e^{-k\cdot 0,1} \qquad | \, ln() \tag{3.9}$$

$$ln(\frac{1}{0,9999} - 1) \geq -k \cdot 0,1 \qquad | : -0,1 \tag{3.10}$$

$$ln(\frac{1}{0,9999} - 1) \cdot -10 \leq k \tag{3.11}$$

$$k \geq 92,1024 \tag{3.12}$$

Man beachte den Relationszeichenwechsel in 3.6 und 3.11 aufrgund von Division einer negativen Zahl. In 3.7 passiert dies nicht, da $1+e^{-k\cdot 0,1} > 0$, wegen $e^z > 0, \forall z \in \mathbb{R}$. Das Ergebnis in 3.12 ist nur eine Näherung. Daher wird $k = 93$ gewählt, um die Bedingung (a) sicherzustellen. Damit ist die zu implementierende logistische Funktion

$$g(x) = \frac{1}{1+e^{-93\cdot(x-3)}} \tag{3.13}$$

3.2 Implementierung der logistischen Funktion g

3.2.1 Allgemeines

Zur Implementierung der in Abschnitt 3.1 ermittelten Funktion wird wieder die Simulationssoftware COPASI [2] vewendet. Die entsprechende Datei finden Sie als logisticSigmoid (mit COPASI 4.25).

Die Implementierungen sollen sich ebenfalls zum Großteil auf bereits ausgearbeitete Reaktionsnetzwerke aus [3] stützen, da ihre Funktionsweise dort bereits erklärt und belegt ist. Allerdings ergeben sich abermals weitere Herausforderungen, auf die, neben der Beschreibung der generellen Implementierung, speziell eingegangen wird.

Desweiteren werden alle Werte wie Funktionsvariablen, Zwischenergebnisse und konstante Werte als Spezies im biochemischen Reaktionsnetzwerk implementiert. Deren

Konzentrationen spiegeln direkt mathematische Werte wieder. Da Grundoperationen aus [3] benutzt werden, bleiben Eingabewerte für Teilrechnungen erhalten. Somit müssen konstante Werte nicht als „fixed“ markiert werden.

3.2.2 Vermeidung unbekannter Operationen

Für diesen Ansatz wird zunächst die Funktion aufgespalten, um die in [3] nicht erwähnten arithmetischen Operationen zu vermeiden.

$$\frac{1}{1 + e^{-93 \cdot (x-3)}} \tag{3.14}$$

Da der Nenner des Bruches stets größer 0 bleibt, ist die Division unproblematisch. Ebenso die Addition von 1 in ebenjenem Nenner. Allerdings läuft die Funktion Gefahr für Bedingung (c) bei $x - 3$ einen negativen Wert zu erhalten. Jedoch hilft hier die Eigenheit von Potenzen aus. Der Exponent wird ausmultipliziert zu $-93x + 3 \cdot 93$ und die Potenz in $e^{-93x} \cdot e^{3 \cdot 93}$ gespalten. Die erste Potenz besitzt nun einen potenziell negativen Exponenten, der allerdings aufgrund der Potenzgesetze umgeschrieben werden kann in $1/e^{93x}$. Somit verbleiben nur noch bereits ausgearbeitete Operationen, da Potenzen als mehrfache Hintereinanderausführung von Multiplikationen einer Zahl mit sich selbst aufgefasst werden können.

3.2.3 Exponentialfunktion mit Taylorpolynom

Als nächstes wird die Exponentialfunktion e^t im System aufgenommen. Der naive Ansatz ist die eulersche Zahl als Konstante einzuführen und entsprechend mit t Hintereinanderausführungen von Multiplikationen e^t zu ermitteln. Die Berechnung von e^t benötigt durch variable Werte t variable Mengen von Multiplikationen/Reaktionen, was in einem festen System wie diesem nicht umsetzbar ist. Auch die Möglichkeit e^t mit einer Reaktion der Art $t \cdot e \to t \cdot e + e^t$ zu beschreiben, ist nicht praktikabel, da t nur als fester Wert in die Reaktion eingegeben werden kann und wieder die Variabilität in t stets zu einer neuen Reaktion und damit einem neuen System für verschiedene t führen würde. Das Ziel ist jedoch **ein** System für die Funktion 3.13, da eine Umsetzung durch t Systeme, $t \in \mathbb{R}$ eben nicht möglich ist.

Das bedeutet die Funktion e^t muss angenähert werden, durch eine Funktion, die nur variabel in der Eingangsgröße und nicht in der Anzahl benötigter Reaktionen ist. Eine bekannte Möglichkeit ist die Taylorentwicklung und die Annährung an die Funktion durch ein Taylorpolynom N-ten Grades (mit Entwicklungspunkt a).

$$\sum_{n=0}^{N} \frac{f^{(n)}(a)}{n!} \cdot (t - a)^n \tag{3.15}$$

Die Exponentialfunktion kann dann durch das folgenden Taylorpolynom N-ten Grades um den Entwicklungspunkt $a = 0$ genähert werden (siehe [7])

$$e^t = 1 + t + \frac{t^2}{2!} + \ldots + \frac{t^N}{N!} \tag{3.16}$$

Für die Implementierung wird zunächst Grad 6 gewählt, dem Hinweis aus der Korrespondenz vom 25.06.2020 folgend.

Insgesamt werden folgende Konstanten benötigt:

1. one: für +1 im Nenner
2. three: für $3 \cdot k$
3. 2! - 6!: für die Nenner der Terme im Taylorpolynom

Die Initialkonzentrationen werden entsprechend der mathematischen Werte der Ausdrücke gewählt; k wird mit 93 initialisiert. Die vorkommenden Potenzen in den Taylortermen werden durch Hintereinanderausführung der Multiplikation berechnet. Für $(3k)^2$ wird zweimal $3k$ als Eingangsspezies verwendet und für weitere $(3k)^j$,$j = 3\ ,\ \ldots 6$ werden $(3k)^{j-1}$ und $3k$ als Eingangspezies verwendet. Die durch Division entstandenen Taylorterme werden im System mit dem Suffix *taylor_i* , $i = 0 \ldots 6$ versehen.

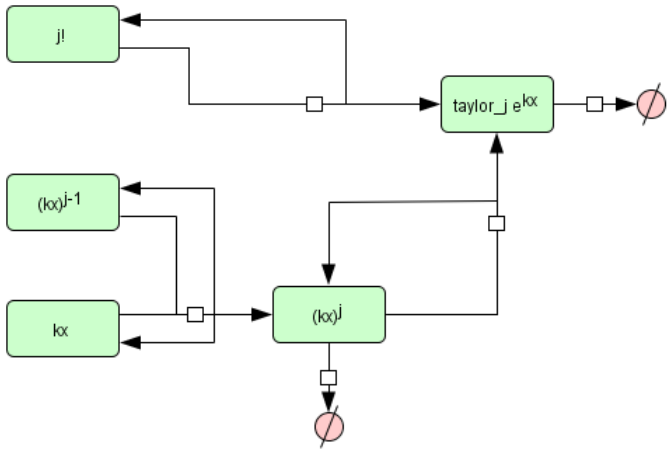

Abbildung 3.1: Reaktionsnetzwerk zur Berechnung des j-ten Taylorterm für e^{kx} nach [3]. Zunächst wird $(kx)^j$ berechnet, bevor mit Division durch j! der tatsächliche Term erzeugt hat.

Da einmal e^{kx} und einmal e^{3k} angenähert werden müssen, muss zweimal ein entsprechendes Taylorpolynom berechnet werden.

3.2.4 Mehrfachaddition

Um das Zusammenfügen der Tayloterme zu erleichtern wird Addition mit mehr als zwei Eingabewerten realisiert. Dafür wird der Fall aus [3] verallgemeinert. Für die Addition von n Eingabespezies wird folgendes Reaktionsnetzwerk benutzt

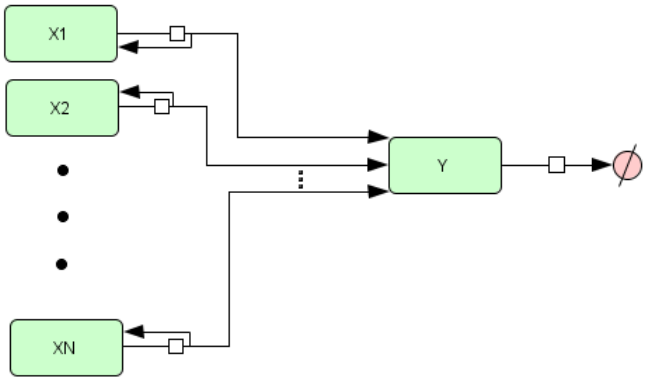

Abbildung 3.2: Reaktionsnetzwerk zur Berechnung einer Summe Y aus n Operanden Xi, $i = 1, \cdots, n$

mit den folgenden Differentialgleichungen

$$\begin{aligned} [\dot{X1}] &= 0 \\ [\dot{X2}] &= 0 \\ &\vdots \\ [\dot{XN}] &= 0 \\ [\dot{Y}] &= k_1[X1] + k_2[X2] + \ldots + k_n[XN] - k_{n+1}Y \end{aligned} \tag{3.17}$$

Die Lösung des DGS ergibt sich dann als

$$\begin{aligned} [X1] &= c_1 \\ [X2] &= c_2 \\ &\vdots \\ [XN] &= c_n \\ [Y] &= k_1c_1 + k_2c_2 + \ldots + k_nc_n \end{aligned} \tag{3.18}$$

beziehungsweise

$$[Y] = k_1[X1] + k_2[X2] + \ldots + k_n[XN] \tag{3.19}$$

Für $k_1 = k_2 = \ldots = k_{n+1} > 0$ gilt weiterhin $[\mathrm{Y}](\infty) = \lim_{t\to\infty}((1 - e^{-k_1 t}) \cdot ([X1](t) + [X2](t) + \ldots + [XN](t))) = [X1](0) + [X2](0) + \ldots + [XN](0)$. Somit kann zur Berechnung von e^{3k} und e^{kx} das obige Reaktionsnetzwerk mit $n = 6$ benutzt werden.

3.2.5 Berechnung der gesamten Gleichung

Zum Schluss wird entsprechend den Systemen aus [3] $1/e^{kx}$, $e^t = e^{3k} \cdot 1/e^{kx}$ und der Nenner $denom = e^t + 1$ berechnet. Eine abschließende Division liefert das Ergebnis in der Konzentration der Spezies *result* Somit ergibt sich für $x = 6$ das $result = 0,98445$.

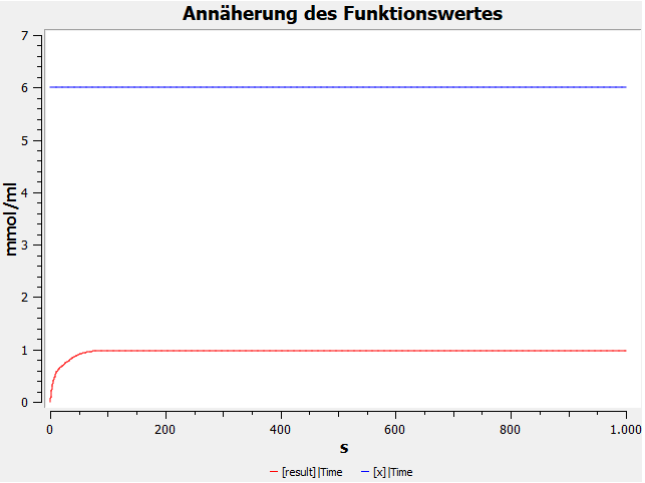

Abbildung 3.3: Annäherung der Stoffkonzentration result für Eingangsspezieskonzentration 6.

Die Überprüfung durch manuelles Ausrechnen der einzelnen Funktionsbestandteile liefert ebenfalls einen Wert von gerundet $0,98445$.

3.3 Bewertung des logistischen Ansatzes

Um die Genauigkeit des Ansatzes zu diskutieren, soll die chemisch berechnete Funktion mit $cLog(x)$, die Funktion $g(x)$ bei der e^t durch Taylorpolynome angenähert wird, wird mit $taylorG(x)$ bezeichnet werden. Zunächst ist $cLog(x)$ für $0 \leq x \leq 6$ simuliert wurden, um Anforderung (c) aus 1 zu erfüllen.

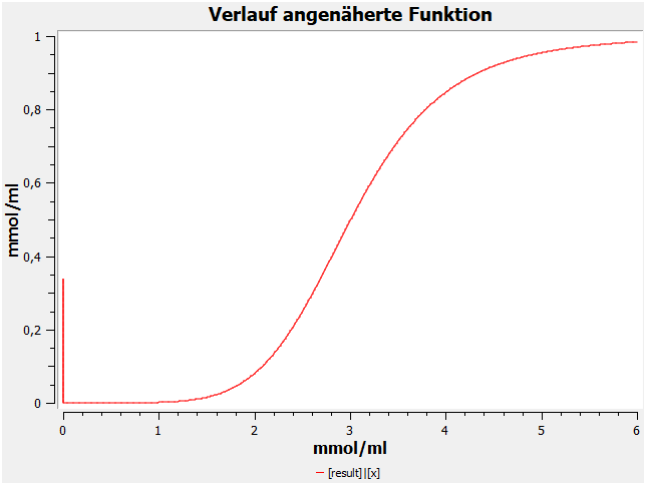

Abbildung 3.4: Verlauf von cLog(x) für Eingabewerte $0 \leq x \leq 6$

Der Auschlag bei $x = 0$ kommt simulationswerkzeugbedingt vor, da x mit $\frac{aktuelle\ Simulationszeit}{\frac{1}{6} \cdot\ Gesamtlaufzeit}$ berechnet wird.

Wie bereits im Abschnitt 3.2.5 angedeutet wird, berechnet $clog(x)$ tatsächlich $taylorG(x)$. Die folgende Tabelle stellt die Funktionswerte für $0 \leq x \leq 6$ für $cLog(x)$, $taylorG(x)$, $g(x)$ und $f(x)$ gegenüber.

x	cLog	taylorG	g	f
0	$1,49383 \cdot 10^{-12}$	$1,49383 \cdot 10^{-12}$	$6,78953 \cdot 10^{-122}$	0
1	0,00143176	0,00143176	$1,66427989 \cdot 10^{-81}$	0
2	0,0815221	0,0815221	$4,07955867 \cdot 10^{-41}$	0
2,9	0,449508	0,449508	0,000091	0
3	0,5	0,5	0,5	0
3,1	0,548852	0,548852	0,999909	1
4	0,848212	0,848212	0,999999	1
5	0,955052	0,955052	1,000000	1
6	0,98445	0,98445	1,00000	1

Tabelle 3.1: Vergleich der Funktionsergebnisse

Die Werte für clog(x) wurden dabei mit einer Simulationslaufzeit von 1000s erhoben. Die Werte für $taylorG(x)$ und $g(x)$ sind in der gleichen Genauigkeit angegeben wie die chemisch berechneten Werte, um eine bessere Vergleichbarkeit zu gewährleisten. Damit fällt Folgendes auf: Bei der Berechnung von $cLog(x)$ nähert sich das Ergebnis seinem tatsächlichen mathematischen Pendant im Rahmen der Simulationsumgebungsgenauigkeit genau an und verändert sich danach nicht mehr wesentlich. Im konkreten kann sich der Endwert nur noch in der Umgebung von $1 \cdot 10^{-6}$ vom tatsächlichen Wert bewegen. Dies liegt jedoch mit hoher Wahrscheinlichkeit am numerischen Vorgehen des Simulationswerkzeuges, da die theoretische Korrektheit für unendliche Laufzeiten in [3]

und 3.2.4 festgestellt wurden. Allerdings ist ersichtlich, dass für alle Werte eine relative große Diskrepanz zur logistischen Funktion $g(x)$ existiert. Deutlich wird das zum Beispiel bei $x = 3,1$: $cLog(x) = 0,548852; g(x) = 0,999909$. Damit liegt $cLog(x)$ nicht mehr im Rahmen der Bedingung (a). Da der Wert allerdings mit $taylorG(x)$ übereinstimmt, wird geschlussfolgert, dass diese Diskrepanz systematischer Natur ist. Also vor allem die Annäherung von e^t durch ein Taylorpolynom 6. Grades mit Entwicklungspunkt $a = 0$ ist der Verursacher dieser Ungenaunigkeit. Im folgenden Diagramm ist nochmals der Unterschied beider Funktionen zu sehen. Die Steigung der tatsächlichen Funktion im kritischen Bereich ist dabei deutlich steiler.

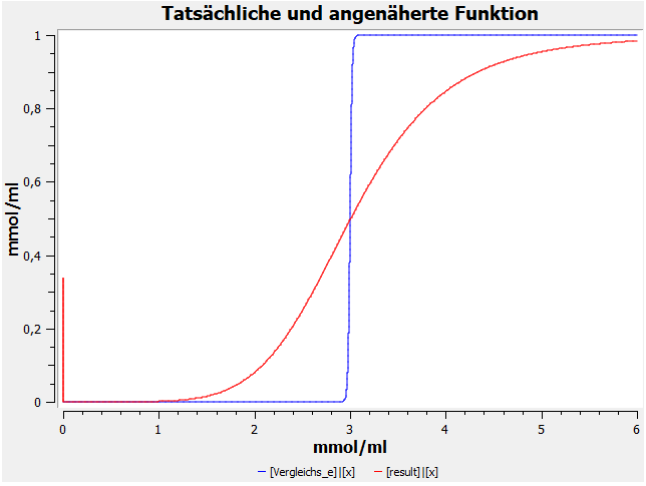

Abbildung 3.5: Verlauf von cLog(x) (rot) und g(x) (blau) für Eingabewerte $0 \leq x \leq 6$

Bezüglich der Laufzeiten veranschaulicht folgende Tabelle die Zeitpunkte, zu denen $cLog(x)$ $taylorG(x)$ erreicht und danach maximal im Bereich $1 \cdot 10^{-6}$ vom tatsächlichen Wert abweicht.

x	Zeitpunkt in s
0	239,199
1	263,445
2	250,820
2,9	104,842
3	164,002
3,1	187,302
4	221,747
5	237,299
6	217,535

Tabelle 3.2: Laufzeiten für das chemische Modell

Diese Ergebnisse liegen wiederum bei 1000s Simulationszeit mit $0,001$ Intervallen vor. Die Reaktionen verlaufen dabei alle mit dem gleichen Reaktionsparamter $v = 0,1$. Die

erhaltenden Laufzeiten von um 200 Sekunden erscheinen lang. Da nach dieser Zeit allerdings ein Wert vorliegt, der nicht weiter geändert wird, ist das Modell deutlich schneller im Erreichen seiner Ergebnisse als der erste Ansatz in 2.

Damit wurde eine Laufzeitverbesserung erreicht, jedoch ist die hohe Ungenauigkeit ein kritischer Punkt der weiter untersucht werden muss.

3.4 Verbesserungsversuche

Die Genauigkeitsgrenze für den logistischen Ansatz ergibt sich aus dem Taylorpolynom (siehe 3.3). Damit setzen die Experimente zur Verbesserung an ebendiesem an. Genauer gesagt kann einerseits der Grad des Polynoms erhöht werden oder ein anderer Entwicklungspunkt gewählt werden. Auf beide Varianten wird mit kurzer Beschreibung der Ansatzentwicklung, speziellen Implementierungsproblematiken, sowie einer Bewertung eingegangen.

3.4.1 Polynomgraderhöhung

In der Formel 3.15 wird der Grad über die Wahl des Paramters N bestimmt. Im Experiment wurde $N = 6$ in Zweierschritten bis auf $N = 12$ erhöht. An dieser Stelle ist beispielhaft die Formel für den finalen Grad $N = 12$, $a = 0$ aufgezeigt:

$$e^t = 1 + t + \frac{t^2}{2!} + \ldots + \frac{t^{12}}{12!} \tag{3.20}$$

Die fehlenden Taylorterme im Vergleich zu 3.2.3 werden in der gleichen Weise implementiert wie bisherige Terme.

Die folgende Tabelle zeigt die Resulte für die kritischen Stellen 2,9 und 3,1 für die Grade 6 bis 12 zur Annäherung der Exponentialfunktion um den Entwicklungspunkt $a = 0$. $TaylorG_N(x)$ beschreibt den Wert der chemischen Berechnung für Grad N an der Stelle x. Da klar ist, dass die chemische Berechnung den tatsächlichen Polynomwert im Rahmen der numerischen Rechenweise und Genauigkeit der Simulationsumgebung erreicht,wird der Ausdruck $TaylorG_N(x)$ in diesem Fall synonym zu $clog(x)$ aus der Bewertung des Ansatzes 3.1 benutzt.

x	$TaylorG_6$	$TaylorG_8$	$TaylorG_{10}$	$TaylorG_{12}$
2,9	0,449508	0,432847	0,416358	0,400044
3,1	0,548852	0,564983	0,580958	0,596774

Tabelle 3.3: Ergebnisse im kritischen Bereich bei Polynomgraderhöhung

Die Genauigkeit für höhere Polynomgrade nimmt also zu. Allerdings an den kritischen Punkten sehr langsam mit etwa 0,017 pro Graderhöhung um 2. Die Laufzeiten (Zeitpunkt bis der Zielwert erreicht wird) erhöhen sich ebenfalls moderat mit erhöhtem Polynomgrad. z.B. $x = 2,9$, mit 1000s und 0,01 Intervall):

	$TaylorG_6$	$TaylorG_8$	$TaylorG_{10}$	$TaylorG_{12}$
Laufzeit in s	104,82	286,61	284,25	362,64

Tabelle 3.4: Laufzeiten für $x = 2,9$ bei Polynomgraderhöhung

Es soll nun die Variante mit Grad 12 betrachtet werden. Folgende Ergebnisse bei 1000s Simulationszeit werden erreicht

x	$TaylorG_{12}$
0	$2,06094 \cdot 10^{-21}$
1	$2,06419 \cdot 10^{-6}$
2	0,00782111
2,9	0,400044
3	0,5
3,1	0,596774
4	0,968964
5	0,997789
6	6 0,99975

Tabelle 3.5: Chemische Ergebnisse für Grad 12

und folgenden Laufzeiten bei Simulationszeiten von 1000s und 0,001 Intervallen, zur besseren Vergleichbarkeit zu 3.3

x	Laufzeit in s
0	431,234
1	356,875
2	352,289
2,9	362,905
3	312,408
3,1	287,255
4	286,894
5	241,417
6	199,685

Tabelle 3.6: Laufzeiten für Grad 12

Die Polynomgraderhöhung ist zum Erreichen einer besseren Genauigkeit durchaus zielführend. Allerdings ist mit einem sehr hohem Polynomgrad zu rechnen, um die Genauigkeit für Bedingung (a) zu erfüllen. Ebenfalls verschlechtert sich jedoch die durchschnittliche Laufzeit. Diese Variante zur Verbesserung des logistischen Ansatzes wird in diesem Falle als nicht praktikabel verworfen.

3.4.2 Wahl des Entwicklungspunktes

Alternativ kann der Entwicklungspunkt geändert werden, um die Werte nahe dieses Wertes genauer anzunähern. Nach 3.15 und $N = 6$, $a \neq 0$ ergibt sich eine Formel, die potenziell negative Werte enthält. Zudem benötigt man im Vorfaktor eines jeden Terms e^a. Dieser Wert wird als Konstante in das System gegeben. Für $n = 0$ ergeben sich keine neuen Probleme. Für $(x-a)^2$ wird die Klammer nach binomischer Formel ausmultipliziert und der negativen Term ans Ende der Berechnung von $(x-a)^2$ gestellt. Da $(x-a)^2 > 0$ muss es sich dann auf jeden Fall um eine nichtnegative Subtraktion handeln und es kann die Subtraktion aus [3] verwendet werden. Für alle anderen geraden n gilt $(x-a)^n = (x-a)^2) \cdot (x-a)^{n-2}$, so dass diese Terme als Potenzen bzw. konkret Mehrfachhintereinanderausführung von Multiplikationen implementiert werden können. Für ungerade n gilt $(x-a)^n = (x-a)^{n-1} \cdot (x-a) = (x-a)^{n-1} \cdot x - (x-a)^{n-1} \cdot a$, wobei $n-1$ gerade und $(x-a)^{n-1}$ wie eben erwähnt berechnet werden kann. Die Terme $(x-a)^{n-1} \cdot a$ werden hingegen gesammelt, mit dem Vorfaktor multipliziert und der zweite Term ganz zum Schluss abgezogen. Da das Taylorpolynom als Annäherung für e^t eingesetzt wird, muss der Endwert positiv sein. Das heißt, wenn diese Terme zum Schluss abgezogen werden, verbleibt die Rechnung in der nichtnegativen Subtraktion. Die restlichen Terme für gerade Exponenten müssen ebenfalls jeweils mit ihren Vorfaktoren verarbeitet werden und werden vor Abzug der negativen Terme in einer Summe zusammengefasst. Da $0 \leq x \leq 6$ bleibt der Rest im Bereich bekannter Operationen.

Welcher Entwicklungspunkt ist nun sinnvoll? Für den Ansatz mit 3.13 ergibt sich $k = 93$. Damit berechnen wir Werte zwischen e^{kx} und e^{3k}. Mit der Wahl der Mitte des x-Intervalls 3, ergibt sich ein möglichst sinnvoller Entwicklungspunkt $a = 3k = 3 \cdot 93 = 279$. Damit müssen wir e^{279} in das System geben. Dies liegt im Bereich von 10^{121} und bringt die Simulationssoftware an ihre Grenzen. Ganz konkret werden alle Konzentrationen bei der Initialkonzentration belassen. Bereits bei $a = 18$ meldet die Simulationssoftwarefehler jedoch Fehler an. Mit der Annäherung des Exponentialterms mit einem Taylorpolynom $N = 6$, $a = 17$ ergibt sich für $x = 3.1$ 0.551934. Damit ist diese Variante nur leicht besser als $N = 6$, $a = 0$ und deutlich schwächer als $N = 12$, $a = 0$. Man kann nun für $a = 17$ weitere Terme hinzufügen, aber erhält bis $N = 12$, $a = 17$ nur einen Wert von etwa 0.6 für $x = 3,1$ und erreicht damit nicht die geforderte Genauigkeit (a).

Damit wird auch dieser Verbesserungsansatz als inpraktikabel verworfen, zum Einen aufgrund der sehr geringen Verbesserung im Vergleich zur Graderhöhung zum Anderen aufgrund der Grenzen der Simulationssoftware.

3.4.3 Weitere Möglichkeiten

Der behindernde Faktor für beide Verbesserungen und des Ansatzes allgemein ist die Annäherung der Exponentialfunktion, deren Größenordnung vor allem vom letzten Taylorterm dominiert wird. Die erreichten Annäherungen sind bei allen bisherigen Verbesserungsversuchen um mehr als 10^{100} abweichend bei $x = 3.1 \rightarrow e^{93 \cdot 3,1}$. Eine theoretische Möglichkeit ist die Verwendung der Exponentialfunktion mit geringeren Exponenten. Dazu muss vor allem der Paramater klein sein. Sowohl die hier verwendete intuitive

Funktion, als auch eine abgewandelte Version (nach der Korrespondenz vom 07.07.2020) in der Form $\frac{e^{k\cdot(x-3)}}{1+e^{k\cdot(x-3)}}$ benötigen Parameter größer 90. Die Möglichkeit diesen Ansatz zu verbessern, bleibt damit im Auffinden einer andersartigen Funktion als Näherung einer Schwellenwertfunktion unter Verwendung einer Exponentialfunktion.

4 Fazit

Beide Ansätze haben ihre Vor- und Nachteile, die im Folgenden gegenübergestellt werden sollen. Mit dem Ansatz der einfachen Sigmoidfunktion lassen sich die Funktionswerte der Funktionen prinzipiell sehr viel genauer berechnen als mit dem logistischen Ansatz. Dies ist vor allem für Funktionswerte nahe Eins richtig. Der Kompromiss, der dabei eingegangen werden muss, ist allerdings eine sehr lange Laufzeit des Modells, die mitunter die Simulationen von COPASI zum Absturz bringt. Dieses Problem lässt sich ebenfalls in der Trägheit des Modells über den Zeitverlauf beobachten: Bei einer Wachstumssimulation der Eingabevariable x von 0 bis 6 konvergiert die Konzentration der Ausgabespezies Y erst im Bereich von $x = 6$ annähernd gegen Eins. Auch eine Diskretisierung der x-Werte, also eine Treppenstufenfunktion für die Konzentration von x bringt keine wesentlichen Verbesserungen. Zudem konvergiert die Funktion h erst im Unendlichen gegen Null beziehungsweise Eins.

Daher wurde ein zweiter Ansatz mit der Exponentialfunktion und deren Annäherung durch ein Taylor-Polynom eruiert. Hier wurde schnell deutlich, dass wesentlich größere Abweichungen von der geforderten Genauigkeit erzielt werden. Im Gegensatz zur ersten beschriebenen Funktion wird diese Stoffkonzentration allerdings schon nach sehr geringer Simulationslaufzeit erreicht. Die Verbesserung der Genauigkeit kann hier allerdings nicht praktikabel erwirkt werden. Mit diesem Ansatz lässt sich jedoch der „Sprung“ der Funktion wesentlich besser im Verlaufsdiagramm der Simulation visualisieren.

Beide Ansätze haben also ihre Stärken und Schwächen: Während die einfache Funktion die Werte (vor allem für $x > 3$) sehr genau berechnen kann, kann der zweite Ansatz mit einer sehr guten Visualisierbarkeit und sehr viel geringerer Modelllaufzeit aufwarten. Die einfache Sigmoidfunktion hingegen benötigt sehr große Modelllaufzeiten, die COPASI mitunter nicht simulieren kann; der Ansatz mit dem Taylorpolynom erreicht wesentlich schlechtere Genauigkeiten des Berechnungsergebnisses.

Das Ziel, an den Stellen $x = 2,9$ und $3,1$ lediglich eine Abweichung von 10^{-4} aufzuweisen, konnte allerdings mit keinem der beiden Ansätze für alle x im Intervall von 0 bis 6 erreicht werden.

Literaturverzeichnis

[1] D. Dini, "Normalized tunable sigmoid functions," 2010 (Zugriff: 28.06.2020). https://dinodini.wordpress.com/2010/04/05/normalized-tunable-sigmoid-functions/.

[2] COPASI-Project, "Copasi," (Zugriff: 29.06.2020). http://copasi.org/.

[3] T. Hinze, *Computer der Natur. Ausgewählte molekulare Prinzipien der biologisch inspirierten Inforamtionsverarbeitung.* Verlag bookboon.com, London, 2013, PDF-File.

[4] H. Happe, "Chemical Analog Computing with Negative Numbers," in *the 19th International Conference, CMC 2018, Dresden, Germany, September 4–7*, 2018, PDF-File.

[5] Wikipedia, "Logistische Funktion," (Zugriff: 29.06.2020). https://de.wikipedia.org/wiki/Logistische_Funktion.

[6] Wikipedia, "Sigmoidfunktion," (Zugriff: 29.06.2020). https://de.wikipedia.org/wiki/Sigmoidfunktion.

[7] A. Iske, "Kapitel 5: Weiterer Ausbau der Differentialrechnung." Lecture Slides, 2006/2007.