Entwicklung und Simulation eines chemischen Analogcomputermodells zur (näherungsweisen) Berechnung der Wurfweite beim schrägen Wurf

Mark Umnus
Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik
Modul Molekulare Algorithmen 2020

11. Juli 2020

Inhaltsverzeichnis

1. **Problembeschreibung** 1
2. **Herleitung** 2
3. **Umsetzung** 4
4. **Ergebnisse** 6
5. **Fazit** 7
A. **Anhang** 9

1. Problembeschreibung

Die Flugbahn eines Balles beim schrägen Wurf beschreibt idealisiert eine Parabel, die als parametrisiertes quadratisches Polynom $h(x)$ mit Abwurfwinkel α, der Ballanfangsgeschwindigkeit v und der Anfangshöhe h_0 dargestellt werden kann. Dieser Zusammenhang wird in Abbildung 1 dargestellt. Die nichtnegative reelle Nullstelle dieses Polynoms verkörpert die Wurfweite w. In dieser Arbeit soll zunächst die Formel für $h(x)$ aus den Gesetzen der Newton'schen Mechanik hergeleitet werden. Die Formel soll für beliebige $\alpha \in \left[0, \frac{\pi}{2}\right]$, $v > 0$ und $h_0 \geq 0$ gültig sein. Anschließend soll ein Reaktionsnetzwerk mit der zugehörigen Massenwirkungskinetik erstellt werden, sodass die Eingabeparameter α, h_0, v und die Erdbeschleunigung g als Stoffkonzentratio-

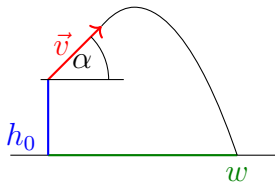

Abbildung 1: Modell des Wurfs mit Starthöhe h_0, Abwurfwinkel α und Startgeschwindigkeit v

nen ausgedrückt werden und sich die Stoffkonzentration einer Ausgabespezies asymptotisch der gesuchten Wurfweite w annähert.

2. Herleitung

Die mathematische Modellierung erfolgt in drei Schritten. Im ersten Schritt wird die Funktionsgleichung der Parabel gesucht, die die Höhe des geworfenen Objektes in Abhängigkeit von der Zeit beschreibt. Die nichtnegative Nullstelle dieses Polynoms, die im zweiten Schritt bestimmt wird, ist der Zeitpunkt t_n, zu dem das Objekt den Boden berührt. Im dritten Schritt wird die horizontale Entfernung in Abhängigkeit der Zeit seit dem Abwurf modelliert und mithilfe des eben bestimmten Zeitpunktes des Aufkommens die letztendlich gesuchte Wurfweite w berechnet.

Die Höhe soll wie in Formel (1) vereinfacht als quadratische Funktion modelliert werden. Im Folgenden werden die einzelnen Parameter mithilfe der Ableitungen bestimmt.

$$\begin{aligned} h(t) &= at^2 + bt + c \\ \dot{h}(t) &= 2at + b \\ \ddot{h}(t) &= 2a \end{aligned} \tag{1}$$

Die zweite Ableitung der Höhe wird als konstant angenommen und ist im Wesentlichen durch die Erdbeschleunigung gegeben, wie Formel (2) zeigt.

$$\ddot{h}(t) = -g \Rightarrow a = -\frac{g}{2} \tag{2}$$

Da die erste Ableitung von h die Geschwindigkeit angibt, mit der sich die Höhe zu jedem Zeitpunkt ändert, ist $h(0)$ durch v und α gegeben:

$$\dot{h}(0) = b \stackrel{!}{=} vf(\alpha) \tag{3}$$

v ist ein Skalar, nämlich gerade die Länge des Richtungsvektors $\vec{v}$, der auf Abbildung 1 zu sehen ist. Dieser Richtungsvektor hat jedoch eine vertikale und eine horizontale Komponente, wobei für h lediglich die vertikale von Bedeutung ist. Aus diesem Grund muss v noch durch eine Funktion f skaliert werden, die vom Startwinkel α abhängt. Durch elementare Geometrie ist bekannt, dass f die Sinusfunktion ist. Würde des Objekt senkrecht nach oben geworfen, also $\alpha = \frac{\pi}{2}$, dann ist $\sin(\alpha) = 1$ und der volle Betrag von $\vec{v}$ wird der vertikalen Geschwindigkeit angerechnet. Die Konstante c aus Formel (1) ist bekanntermaßen durch h_0 gegeben, sodass insgesamt gilt:

$$h(t) = -\frac{g}{2}t^2 + v\sin(\alpha)t + h_0 \tag{4}$$

Als Nächstes gilt es, eine nichtnegative Nullstelle von h zu bestimmen, sofern eine solche existiert, da diese den Zeitpunkt des Landens repräsentiert. Anhand der Diskriminante

$$D = b^2 - 4ac = \underbrace{(v\sin\alpha)^2}_{\geq 0} + 2g\underbrace{h_0}_{\geq 0} \geq 0 \tag{5}$$

lässt sich ablesen, dass h mindestens eine reelle Nullstelle besitzt; außer im Fall $(v = 0 \vee \alpha = 0) \wedge h_0 = 0$ besitzt h sogar zwei reelle Nullstellen. Da der Scheitelpunkt der symmetrischen h-Funktion bei $t = v\sin(\alpha)g^{-1} \geq 0$ liegt, hat h eine nichtnegative Nullstelle. Sie liegt unter Nutzung der allgemeinen Lösungsformel für quadratische Polynome sowie (5) bei

$$t_n = \frac{v\sin\alpha + \sqrt{D}}{g} \tag{6}$$

und muss anschließend in die Formel für die Wurfweite in Abhängigkeit von der Zeit eingesetzt werden, welche im Folgenden entwickelt wird.

Das Vorgehen zum Finden der Weitenfunktion $w(t)$ ist analog zu (1). Dieses Mal jedoch wird angenommen, dass $\dot{w}(t)$ konstant ist, weil der Luftwiderstand in dieser Approximation vernachlässigt wird. Weiterhin gilt, wiederum analog zu (3), dass die horizontale Geschwindigkeit zum Startzeitpunkt durch die horizontale Komponente von $\vec{v}$ gegeben ist:

$$\begin{aligned}\dot{w}(0) &= v\cos(\alpha)\\ \Rightarrow w(t) &= v\cos(\alpha)t\end{aligned}$$

Um die gesuchte Wurfweite zu erhalten, muss nun $w(t_n)$ mit dem in (6) gefundenen Landezeitpunkt bestimmt werden. Die trigonometrischen Funktionen können nicht auf eine einfache Art durch Massenwirkungskinetik berechnet werden [3]. Stattdessen wird im Folgenden eine Annäherung durch Taylorpolynome im Entwicklungspunkt $x_0 = 0$ verwendet. Für diese Anwendung würden Taylorpolynome dritten Grades ausreichen. Allerdings können negative Werte in der späteren Umsetzung zu Problemen führen. Bei Eingabewerten von $\alpha \in [0, \frac{\pi}{2}]$ kann die Taylorapproximation der Cosinusfunktion beim Grad drei allerdings negativ werden (für $\alpha > \sqrt{2}$), weswegen hierfür der vierte Grad verwendet wird, der dieses Problem nicht aufweist. In Formel (7) wird die Approximation des Cosinus direkt so geschrieben, dass von links nach rechts gelesen keine negativen Werte auftreten.

$$\begin{aligned}\sin\alpha &\approx T^3_{\sin}\alpha = \alpha - \frac{\alpha^3}{6}\\ \cos\alpha &\approx T^4_{\cos}\alpha = 1 + \frac{\alpha^4}{24} - \frac{\alpha^2}{2}\end{aligned} \tag{7}$$

Zusammen ergibt dies die finale Gleichung:

$$\begin{aligned}w(t_n) &= v\cos\alpha \cdot t_n\\ &\overset{(6)}{=} v\cos\alpha\frac{v\sin\alpha + \sqrt{(v\sin\alpha)^2 + 2gh_0}}{g}\\ &\overset{(7)}{\approx} vT^4_{\cos}\alpha\frac{vT^3_{\sin}\alpha + \sqrt{[vT^3_{\sin}\alpha]^2 + 2gh_0}}{g}\end{aligned} \tag{8}$$

Im Folgenden soll eine grobe Fehlerabschätzung für den durch die Taylorentwicklung entstehenden Approximationsfehler durchgeführt werden. Eine analytische Herangehensweise ist in diesem Fall sehr ungenau. Es gilt

$$f(x) = T^n_f(x) + R^{n+1}_f(x)$$

wobei für das Lagrange'sche Restglied bei Entwicklung im Punkt x_0 gilt:

$$\exists\xi \in [x_0, x] : R^{n+1}_f(x) = \frac{f^{n+1}(\xi)}{(n+1)!}(x - x_0)^{n+1}$$

In dieser Arbeit wurde der Entwicklungspunkt $x_0 = 0$ verwendet und Winkel $\alpha \in [0, \frac{\pi}{2}]$ zugelassen, weswegen für die Taylorapproximation des Sinus folgende obere Schranke gilt:

$$\begin{aligned}\left|\sin(x) - T^3_{\sin}x\right| &= \left|R^4_{\sin}(x)\right| = \left|\frac{\sin^{(4)}(\xi)}{24}x^4\right|\\ &= \frac{\sin(\xi)}{24}x^4 \le \frac{1}{24}x^4 \le \frac{\pi^4}{384} < 0{,}2537\end{aligned}$$

Analog gilt für die Abschätzung des Cosinus folgende Schranke:

$$\left|R^5_{\cos}(x)\right| \le \frac{\pi^5}{3840} < 0{,}0797$$

Dabei ist zu beachten, dass diese Fehler nicht zwangsweise von den Taylorpolynomen erreicht werden müssen. Insbesondere der Fehler der Sinusfunktion ist viel zu

groß bemessen. Auf numerischem Wege erhält man als obere Schranke für den Sinus mit 64-Bit-Arithmetik etwa $8 \cdot 10^{-2}$. Weiterhin zu beachten ist außerdem, dass der Approximationsfehler durch die numerischen Eigenschaften der anderen Schritte beeinflusst wird. So wird die jeweilige trigonometrische Funktion zunächst mit v multipliziert. Ist dieser Wert größer als 1, wird der Fehler verstärkt, ist er kleiner, wird auch der Fehler abgeschwächt. Die genauen Auswirkungen der Taylorapproximation werden später noch einmal in der Diskussion der Ergebnisse aufgegriffen.

3. Umsetzung

Die Formel zur Berechnung der Wurfweite beim schrägen Wurf soll nun als Reaktionsnetzwerk modelliert werden. Es ist bekannt, dass Reaktionsnetzwerke Addition, nichtnegative Subtraktion, Multiplikation, Division und die Quadratwurzeloperation abbilden können [1]. Dies sind all jene Operationen, die auch für Formel (8) benötigt werden. Sie werden im Folgenden *grundlegende Operationen* genannt. Für eine chemische Modellierung muss jede grundlegende Operation in mehrere Reaktionen übersetzt werden. Dies ist beispielhaft auf Abbildung 2 für eine Division gezeigt; die anderen können in [1] nachgelesen werden.

Implementiert und simuliert wird das Reaktionsnetzwerk mithilfe der freien Software COPASI [2]. Diese ermöglicht es über ein graphisches Benutzerinterface, Spezies und Reaktionsgleichungen zu definieren und anschließend eine Simulation für eine definierte, virtuelle Zeit laufen zu lassen. Beispielsweise kann eine gewünschte Dauer von drei Minuten in wenigen Sekunden simuliert werden, was die Effizienz von COPASI zeigt.

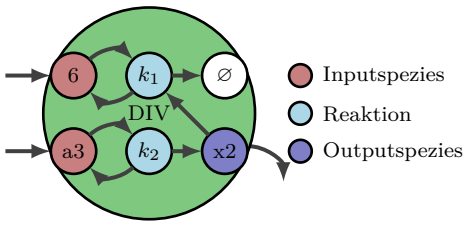

Abbildung 2: Reaktionsnetzwerk für die Spezies `x2` ausgehend von `a3` und `6`. Die untere Reaktion ist beispielsweise als $\texttt{a3} \rightarrow^{k_2} \texttt{a3} + \texttt{x2}$ zu lesen. Für die Parameter wurde $k_1 = k_2 = 0{,}1$ gewählt.

Listing 1: SSA-Instruktionen der Wurfweitenformel in Präfixnotation

```
a2 = MUL alpha alpha
a4 = MUL a2 a2
x0 = DIV a4 TWENTYFOUR
x1 = DIV a2 TWO
Tcos_ = ADD ONE x0
Tcos = SUB Tcos_ x1
a3 = MUL a2 alpha
x2 = DIV a3 SIX
Tsin = SUB alpha x2
b = MUL v Tsin
x3 = MUL b b
x4 = MUL g h0
x5 = MUL TWO x4
D = ADD x3 x5
x6 = SQRT D
x7 = ADD b x6
tn = DIV x7 g
vTcos = MUL v Tcos
wn = MUL vTcos tn
```

Der größte Nachteil von COPASI für die Aufgabe dieser Arbeit ist, dass es unübersichtlich ist, die Vielzahl an Spezies und Operationen zu verwalten. Aufgrund der vergangenen Projektarbeiten und dem Umfang der Formel (8) konnte die Menge der benötigten Spezies auf etwa 30 und die der

Reaktionen auf etwa 60 für eine naive Implementierung geschätzt werden. Würde eine Änderung der zugrundeliegenden Formel notwendig werden oder müsste im Fehlerfall die zu korrigierende Stelle gefunden werden, wäre der manuelle Aufwand hoch. Aus diesem Grund werden zwei zentrale Eigenschaften genutzt, um die Modellierung zu vereinfachen. Die erste Eigenschaft ist die, dass die Konzentrationen der Eingabespezies bei den Reaktionen der grundlegenden Operationen konstant bleiben. Die Reaktion zu k_2 in Abbildung 2 beispielsweise verbraucht ein `a3`, produziert jedoch auch eines. Dadurch ist es möglich, Formel (8) in eine SSA-Form zu überführen, um Berechnungen zu sparen und Zwischenergebnisse wie die Taylorpolynome wiederzuverwenden [4]. Dies ist in Listing 1 dargestellt. Jede Konstante auf der linken Seite einer Gleichung entspricht einem Wert, der sich aus der grundlegenden Operation angewendet auf eine oder zwei andere Konstanten ergibt. So werden sukzessive komplexere Werte errechnet. Die Abbildung 3 zeigt beispielhaft, durch welchen Berechnungsgraph die Konzentration der Spezies `Tsin` bestimmt wird. Jede dieser Gleichungen kann in entsprechende Reaktionen übersetzt werden und jede Konstante in eine Spezies.

Das Problem der negativen Werte in der Cosinusapproximation hätte auch durch Fallunterscheidungen gelöst werden können. Um die Komplexität gering zu halten, wurde allerdings die Lösung durch einen höheren Grad der Taylorapproximation bevorzugt. Dadurch entfällt die Notwendigkeit von ϕ-Instruktionen in der SSA-Form. Anhand des Listings 1 wird deutlich, wie die Reduktion der Spezies und Reaktionen wirkt. Zum Beispiel wird α^2 in Formel (8) zwar mehrere Male durch die Taylorpolynome verwendet, muss in der SSA-Form

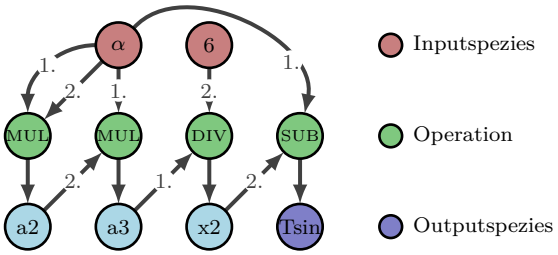

Abbildung 3: Berechnungsgraph für die Spezies `Tsin`, wobei jeder „Operation“-Knoten für mehrere Reaktionen steht wie auf Abbildung 2 gezeigt. Die Zahlen an den Kanten zeigen an, welches Argument die jeweilige Spezies für die Operation ist; `x2` ergibt sich also durch `x2 = DIV a3 6`.

jedoch nur einmal berechnet werden. Dadurch kann die Anzahl der für das Reaktionsnetzwerk benötigten Spezies minimal gehalten werden. Zusammen mit den Daten aus [1] lässt sich die Gesamtanzahl der benötigten Spezies und Reaktionen bestimmen. Insgesamt werden Spezies für die Eingabewerte, für die Zahlen wie g oder 1, als Zwischenspezies für die Subtraktionen und als eine `null`-Spezies, die zerfallene Spezies repräsentiert, benötigt. Hinzu kommt eine Spezies pro Zwischenergebnis. In der Summe werden 30 Spezies im Modell verwendet. Für die Anzahl der Reaktionen ergibt sich ein Wert von 45 und damit eine Einsparung um etwa 25%.

Die zweite wichtige Eigenschaft, die sich in dieser Arbeit zunutze gemacht wurde, ist die Verwendung eines einfachen XML-Dateiformats durch COPASI. Das ermöglicht es, COPASI-Dateien aus einem eigenen Beschreibungsformat zu erzeugen, wodurch die fehleranfällige, manuelle Arbeit im GUI entfällt. Es wurde daher ein Py-

thonprogramm entwickelt, das ein deklaratives YAML-Format in eine für COPASI lesbare Datei übersetzt und dabei bei Bedarf Spezies und Reaktionen erzeugt.[1] Das hat den Vorteil, dass die YAML-Datei viel leichter zu warten und anzupassen ist. Die komplette YAML-Datei, die als Basis für das COPASI-Modell dieser Arbeit dient, ist aus Platzgründen im Anhang 2 ab Seite 11 aufgeführt.

Im Detail sind für die grundlegenden Operationen `ADD`, `SUB`, `MUL`, `DIV` und `SQRT` Vorlagen für Reaktionen hinterlegt, die [1, S. 33-38] entnommen sind. Insbesondere sind dabei auch die Parametrisierungen entsprechend gewählt. Werden Zwischenspezies benötigt, werden diese automatisch miterzeugt. Schließlich wird mithilfe der Python-Bibliothek jinja2 die eigentliche COPASI-Datei befüllt.

4. Ergebnisse

Für die Simulation und Erzeugung der Ergebnisse wurde die Python-Schnittstelle von COPASI verwendet.[2] Diese erlaubt es wie das GUI, einen *Time course* zu definieren und zu parametrisieren, insbesondere mit der Laufzeit in Sekunden. Als Ausgabe wird eine CSV-Datei generiert, die die Stoffkonzentrationen aller Spezies zu jedem Zeitpunkt enthält. Auf Grundlage dieser konnten abschließend die Graphiken erzeugt werden, die der Übersichtlichkeit halber im Anhang A.1 ab Seite 9 aufgeführt sind.

Dass das implementierte Reaktionsnetzwerk im Allgemeinen in der Lage ist, die Wurfweitenfunktion sehr genau nachzubilden, zeigt Tabelle 1. Gezeigt sind hier die Werte, die für $w(t_n)$ durch COPASI berechnet werden, wobei mit Ausnahme der Spalte zu $\alpha = 1{,}57$ jeweils für 200 Sekunden simuliert wurde. Um die Ergebnisse einordnen zu können, sind zusätzlich die Abweichungen der analog berechneten Werte von einer „exakten“ Berechnung mit Fließkommaarithmetik aufgeführt. Als weiterer Bewertungsmaßstab wird der Fehler aufgezeigt, der allein durch die Verwendung der Taylorapproximation auftritt. So wird deutlich, dass die Abweichungen meist die gleiche Größenordnung haben, was bedeutet, dass der durch COPASI gemachte Fehler zu großen Teilen auf die Ersetzung der trigonometrischen Funktionen zurückzuführen ist. Eine Ausnahme davon ist der Fall $\alpha = 0$, welcher der Entwicklungspunkt der Taylorpolynome ist und dadurch mathematisch exakt rekonstruiert werden kann, was für das Reaktionsnetzwerk aufgrund des asymptotischen Rechenverfahrens nicht möglich ist. Erwartungskonform kann beobachtet werden, dass große h_0 den Approximationsfehler vermindern. Das liegt daran, dass der bei konstantem Abwurfwinkel ebenfalls konstante Fehler neben dem mit $2g$ skalierten h_0 weniger ins Gewicht fällt. Ebenfalls ist zu erkennen, wie der Fehler überproportional durch v skaliert wird, was bereits in den Vorbetrachtungen angekündigt wurde. Eine Erhöhung von v um den Faktor 10 hat eine Erhöhung des Fehlers um den Faktor 100 zur Folge. Bei den Experimenten sind vor allem Besonderheiten für Werte von α nahe an $\frac{\pi}{2}$ (Abbildung 8) aufgefallen. Hier sollte für die Wurfweite idealerweise $w(t_n) = 0$ gelten, aber durch die Annäherung von $\frac{\pi}{2}$ durch 1,57, die Annäherung des Cosinus durch ein Taylorpolynom und die approximative Berechnung durch COPASI ist dieser Wert nicht zu erreichen. Zusätzlich braucht es mehrere Größenordnungen mehr Zeit, um halbwegs gute Annäherun-

[1] `CopasiTool` on GitHub
[2] `python-copasi` on pypi.org

Tabelle 1: Berechnete Wurfweiten für verschiedene Parameter, dazu absolute Abweichungen von Vergleichswerten. „exakt“ und „Taylor“ wurden mit 64-Bit-Fließkommaarithmetik und Winkel- und trigonometrischen (für Ersteres) Funktionen der Python-Standardbibliothek berechnet. Für die Zeile zu $\alpha = 1{,}57$ wurde die Simulation über 200000 Sekunden verwendet.

α	h_0	v	$w_{\text{COPASI}}(t_n)$	$\Delta_{\text{exakt – COPASI}}$	$\Delta_{\text{exakt – Taylor}}$
0	1	5	2,25759	0,0000282	0,0000000
0,785	10	5	6,48228	0,0003942	0,0025475
0,785	0	5	2,54243	0,0059892	0,0076719
0,785	0	50	254,241	0,6009172	0,7671920
1,57	1	5	0,116439	0,1116985	0,1116525

gen zu erreichen. Während bei den meisten Experimenten eine simulierte Zeit von 120 Sekunden ausreichte, um einen stabilen Zustand zu erreichen, waren in diesem Sonderfall selbst 200000 Sekunden kaum genug.

5. Fazit

In dieser Arbeit wurde gezeigt, wie eine komplexe Funktion zur näherungsweisen Berechnung der Wurfweite beim schrägen Wurf mittels eines chemischen Analogcomputermodells implementiert werden kann. Dazu wurde die entsprechende Funktion zunächst mathematisch hergeleitet und die benötigten grundlegenden Operationen identifiziert. Anschließend wurde ein Programm entwickelt, das die grundlegenden Operationen in COPASI-lesbare Modelle mit Spezies und Reaktionen übersetzt. Durch die danach durchgeführten Simulationen konnte bestätigt werden, dass das chemische Analogcomputermodell in der Lage ist, qualitativ gute Ergebnisse zu erzielen. 120 Sekunden bis zu einem stabilen Zustand sind jedoch zu lang für eine Berechnung, die selbst auf alter Hardware nur wenige Mikrosekunden benötigt. Natürlich kann das Modell mit einem Digitalcomputer simuliert werden, so wie in dieser Arbeit gezeigt, allerdings muss dann auf Vorteile von Analogcomputern wie numerische Sicherheit verzichtet werden. Insgesamt kann das hier erarbeitete Modell daher nicht als praxistauglich eingestuft werden.

Literatur

[1] Thomas Hinze. *Computer der Natur*. Hrsg. von Thomas Hinze. Verlag bookboon.com, ISBN 978-87-403-0378-0, 2013.

[2] Stefan Hoops u. a. „COPASI—a COmplex PAthway SImulator“. In: *Bioinformatics* 22.24 (Okt. 2006), S. 3067–3074. DOI: 10.1093/bioinformatics/btl485. eprint: https://academic.oup.com/bioinformatics/article-pdf/22/24/3067/546592/btl485.pdf.

[3] Issa Katime u. a. „Harmonic oscillations in a polymerization“. In: *International Journal of Chemical Kinetics* 41 (8 2009), S. 507–511.

[4] B. K. Rosen, M. N. Wegman und F. K. Zadeck. „Global Value Numbers and Redundant Computations“. In: *Pro-*

ceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL '88. San Diego, California, USA: Association for Computing Machinery, 1988, S. 12–27. DOI: 10.1145/73560.73562.

A. Anhang

A.1. Graphiken

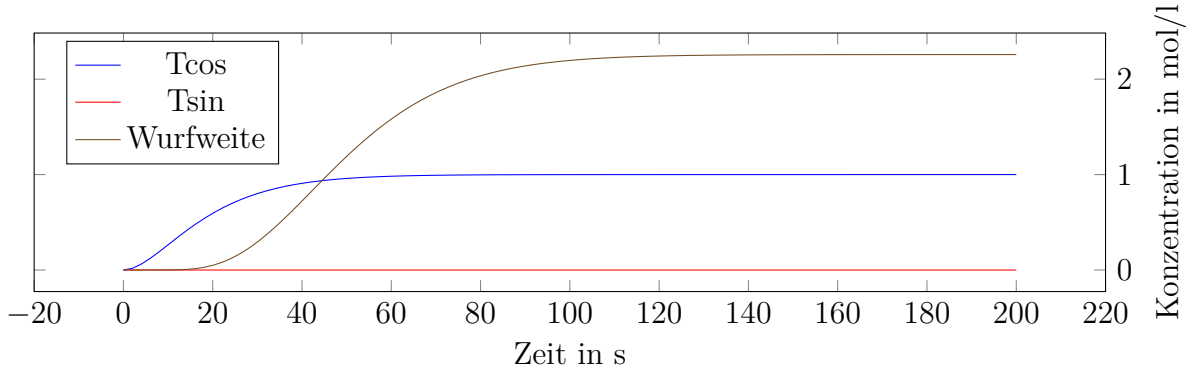

Abbildung 4: Wurfweite mit $\alpha = 0$, $h_0 = 1$ und $v = 5$; Die Taylorapproximationswerte sind sehr genau, da αdem Entwicklungspunkt entspricht. Die Werte werden gemäß Listing 1 berechnet.

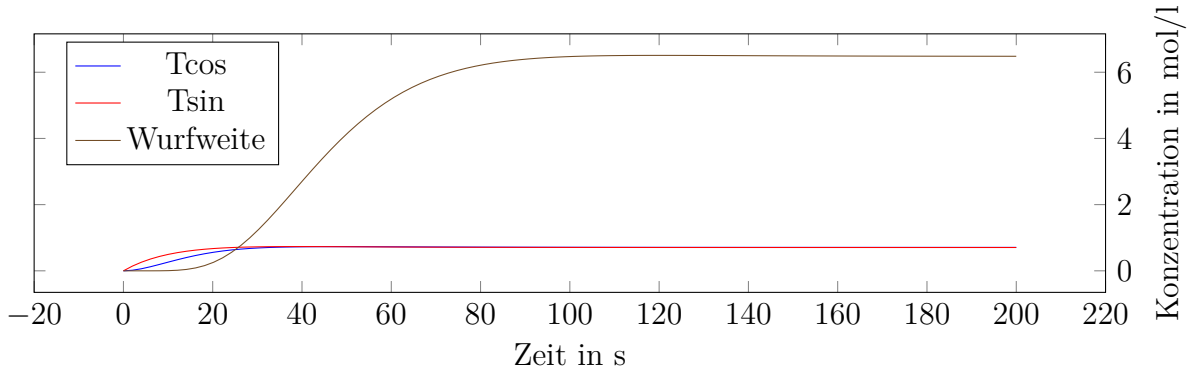

Abbildung 5: Wurfweite mit $\alpha \approx \frac{\pi}{4}$, $h_0 = 10$ und $v = 5$; Taylorapproximationswerte zur Überprüfung

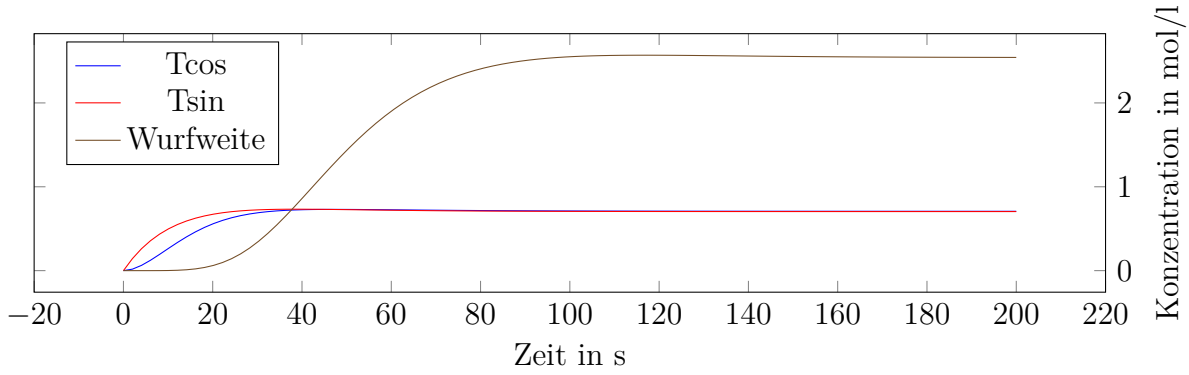

Abbildung 6: Wurfweite mit $\alpha \approx \frac{\pi}{4}$, $h_0 = 0$ und $v = 5$; Taylorapproximationswerte zur Überprüfung

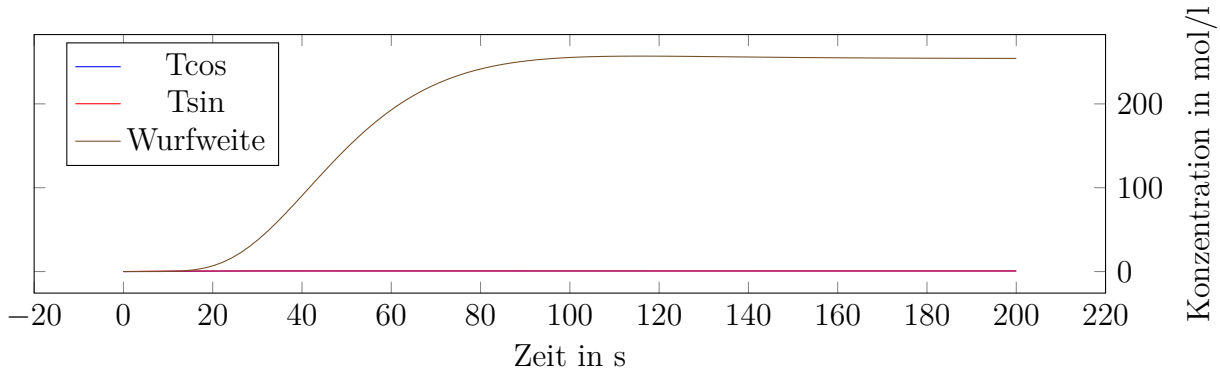

Abbildung 7: Wurfweite mit $\alpha \approx \frac{\pi}{4}$, $h_0 = 0$ und $v = 50$

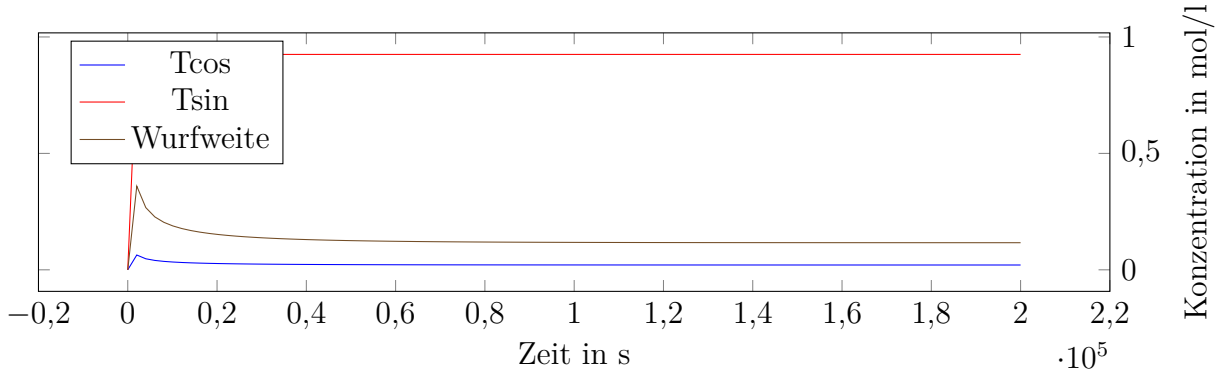

Abbildung 8: Wurfweite mit $\alpha \approx \frac{\pi}{2}$, $h_0 = 1$ und $v = 5$; Es ist zu sehen, dass die Taylorapproximationen große Fehler aufweisen, selbst nach langer Simulationszeit.

A.2. Implementierungsdetails

Listing 2: YAML-Datei, die in eine COPASI-Datei übersetzt wird

```
name: Throw distance approximation
duration: 200
input:
  - name: alpha
    initial_value: 0
  - name: h0
    initial_value: 1
  - name: v
    initial_value: 5
  - name: g
    initial_value: 9.81
  - name: ONE
    initial_value: 1
  - name: TWO
    initial_value: 2
  - name: SIX
    initial_value: 6
  - name: TWENTYFOUR
    initial_value: 24

functions:
  - a2 = MUL alpha alpha
  - a4 = MUL a2 a2
  - x0 = DIV a4 TWENTYFOUR
  - x1 = DIV a2 TWO
  - Tcos_ = ADD ONE x0
  - Tcos = SUB Tcos_ x1

  - a3 = MUL a2 alpha
  - x2 = DIV a3 SIX
  - Tsin = SUB alpha x2

  - b = MUL v Tsin
  - x3 = MUL b b
  - x4 = MUL g h0
  - x5 = MUL TWO x4
  - D = ADD x3 x5
  - x6 = SQRT D
  - x7 = ADD b x6
  - tn = DIV x7 g

  - vTcos = MUL v Tcos
  - wn = MUL vTcos tn
```