Universität Rostock

Traditio et Innovatio

Vergleich von Methoden zur kombinierten Struktur- und Parameterinferenz von Reaktionsmodellen

Name:	Robin van der Wall
Matrikelnummer:	218 203 492
Abgabedatum:	21.3.2025
Betreuer und Gutachter:	M.Sc. Justin Kreikemeyer Universität Rostock Fakultät für Informatik und Elektrotechnik
Gutachter:	Prof. Dr. Adelinde Uhrmacher Universität Rostock Fakultät für Informatik und Elektrotechnik

Bachelorarbeit
Fakultät für Informatik und Elektrotechnik

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

1. Einleitung

Mit dem Abschluss des Human Genome Project im Jahr 2003 wurde in den Biowissenschaften ein Wandel eingeleitet: Während biologische Phänomene zuvor häufig durch die Reduktion auf einzelne Bestandteile untersucht wurden, eröffneten die umfangreichen Sequenzdaten des menschlichen Genoms neue Möglichkeiten zur Analyse von Interaktionen und Dynamiken biologischer Systeme. Diese Entwicklungen wurden maßgeblich durch den raschen Fortschritt der Bioinformatik unterstützt, was den Aufstieg der Systembiologie vorantrieb (Chuang et al., 2010).

Ein Problembereich, der mit den neu entwickelten Werkzeugen der Bioinformatik untersucht werden konnte, ist die Rekonstruktion von Chemisches Reaktionsnetzwerk (CRN) aus molekularen Konzentrationsdaten. Ein tiefgehendes Verständnis chemischer Prozesse, die innerhalb von Zellen ablaufen, ist von großem Interesse, insbesondere für die Prozessoptimierung oder die Krebsforschung, um gezielt Prozesse in mutierten Zellen zu beeinflussen oder zu stoppen (Abramovitch et al., 2004).

Historisch wurden Reaktionsgleichungen manuell von Biologen aus den Konzentrationsdaten durch Anwendung von Expertenwissen und heuristischen Schätzungen extrahiert (Unsleber und Reiher, 2020). Erste Methoden zur Automatisierung ermöglichten es dann, die Parameter der gefundenen Reaktionsgleichungen automatisch zu bestimmen, erforderten jedoch weiterhin einen Biologen, der manuell die Struktur der Reaktionsgleichung vorgibt.

Eine neuere Entwicklung in der Bioinformatik ist die automatische Rekonstruktion der Struktur von Reaktionsgleichungen. Dabei wurden bereits verschiedene Methoden aus der Mathematik und Informatik eingesetzt, darunter maschinelles Lernen (Ji und Deng, 2021), lineare Programmierung (Taylor et al., 2021) und genetische Algorithmen (Rausanu et al., 2015). Diese Ansätze unterscheiden sich sowohl in der Art der Ein- und Ausgabe als auch in ihrer Genauigkeit und Skalierbarkeit.

Trotz der Vielfalt der verwendeten Ansätze und ihrer stark variierenden Eigenschaften wurde bislang kein umfassender Vergleich dieser Methoden durchgeführt. Als eines der Haupthindernisse für einen solchen Vergleich wurde das Fehlen geeigneter Werkzeuge zur quantitativen Bewertung dieser Methoden identifiziert.

Das Ziel dieser Arbeit ist es, einen explorativen Vergleich von Ansätzen zur gleichzeitigen Struktur- und Parameterinferenz durchzuführen und auf dessen Grundlage ein flexibles *Benchmarking-Tool* zu entwickeln. Die theoretischen Grundlagen werden in Kapitel 2 erläutert. In Kapitel 3 *Literaturrecherche* erfolgt anschließend ein qualitativer Vergleich mithilfe einer strukturierten Literaturrecherche, der systematisch ausgewertet wird.

In Kapitel 4 wird auf Grundlage der qualitativen Auswertung die Entwicklung eines *Benchmarking-Tools* vorgestellt. Zunächst werden die Anforderungen an das Programm definiert. Im Mittelpunkt steht die Anpassbarkeit des Programms, die es ermöglichen

soll, eine möglichst große Anzahl verschiedener Methoden zu vergleichen. Zudem wird festgelegt, anhand welcher Parameter ein Vergleich der Ansätze sinnvoll ist und durch welche Metriken diese beschrieben werden können. Anschließend wird die prototypische Implementierung des Programms beschrieben. Abschließend erfolgt ein quantitativer Vergleich anhand von drei Beispielmethoden: Reactmine (Martinelli et al., 2023), Coupled SINDy (Burrage et al., 2024) und Evolving SINDy (Kreikemeyer et al., 2024).

In Kapitel 5 wird das *Benchmarking-Tool* ausgewertet und mögliche zukünftige Entwicklungsrichtungen des Programms identifiziert.

2. Chemische Reaktionen

Als chemische Reaktion wird laut dem *Gold Book* der International Union of Pure and Applied Chemistry, 2025 die Interkonvertierung chemischer Spezies definiert. Eine chemische Spezies bezeichnet eine Gruppe chemisch identischer molekularer Einheiten, die im Verlauf eines Experiments nicht zu unterscheiden sind. Auch Moleküle mit identischer Atomzusammensetzung können als verschiedene Spezies betrachtet werden, wenn sich ihre Struktur unterscheidet und sie im Experiment differenziert werden können.

2.1. Chemische Reaktionsnetzwerke

$$R_j : k_{1j}S_1 + k_{2j}S_2 + \cdots + k_{ij}S_i \xrightarrow{r} k'_{1j}S_1 + k'_{2j}S_2 + \cdots + k'_{ij}S_i \tag{2.1}$$

Formal werden Reaktionen oft als CRN Gleichung 2.1 festgehalten (Gratie et al., 2013), wobei die Mengenverhältnisse der Spezies – auch als Stöchiometrie der Reaktion bezeichnet – definiert sind.

Die Menge der an einem Reaktionssystem beteiligten Spezies wird durch $S_1, \ldots, S_i$ dargestellt, während $R_1, \ldots, R_j$ die einzelnen Reaktionen beschreibt. Der Ausdruck k_{im} gibt an, in welcher Quantität Spezies S_i an Reaktion R_m beteiligt ist.Spezies, die auf der linken Seite einer Reaktionsgleichung vorkommen, d. h. für die gilt $k > 0$, werden als Reaktanten bezeichnet, während Spezies auf der rechten Seite, d. h. $k' > 0$, als Produkte bezeichnet werden. Die Differenz der Quantitäten einer Spezies zwischen der rechten und der linken Seite, $n = k - k'$ ist der stöchiometrische Koeffizient dieser Spezies in der Reaktion.

Mit der Reaktionsrate r wird die Geschwindigkeit beschrieben, mit der Spezies umgewandelt werden.

2.2. Mathematische Modelle

2.2.1. Reaktionsratengleichung

Um Reaktionen zu simulieren und zu analysieren, ist es notwendig, sie mit einem mathematischen Modell zu verknüpfen. Historisch wurde dies meist durch ein System gewöhnlicher Differenzialgleichungen realisiert, wie von Gillespie, 1977 beschrieben.

$$\frac{d[S_i]}{dt} = \sum_{j=1}^{n} n_{i,j} v_j \tag{2.2}$$

Jede Spezies S eines Reaktionsmodells wird als Funktion $[S_i] : \mathbb{R}_+ \to \mathbb{R}_+$ modelliert, die die zeitliche Entwicklung ihrer Konzentration $[S]$ beschreibt. Die Abhängigkeiten

zwischen den Spezies können dann, wie in Gratie et al., 2013 beschrieben, als System von Differentialgleichungen in Gleichung 2.2 dargestellt werden. Dabei bezeichnet v_j die Reaktionsrate der Reaktion j und $n_{i,j}$ den stöchiometrischen Koeffizienten der Spezies S_i in der Reaktion r_j.

Die direkte Modellierung mit nichtlinearen gewöhnlichen Differentialgleichungen, wie in Gleichung 2.2, wird als *Reaktionsratengleichung* (RRG) bezeichnet (Higham, 2008).

Hierbei wird die Annahme getroffen, dass es sich bei einer chemischen Reaktion um einen deterministischen, kontinuierlichen Prozess handelt. Diese Annahme kann allerdings nicht allgemeingültig getroffen werden, denn sie entspricht nicht der physikalischen Realität, wie unter anderem Gillespie, 1977 festhält: Zum einen sind Konzentrationen von Spezies keine kontinuierlichen Einheiten, sondern verändern sich nur in ganzzahligen, diskreten Schritten. Des Weiteren müsste man, um deterministische Vorhersagen über ein Reaktionssystem zu treffen – selbst wenn man davon ausgeht, dass die klassische Mechanik ausreicht, um es zu beschreiben und quantenphysikalische Eigenschaften ignoriert werden –, die Positions- und Geschwindigkeitsinformationen jedes Moleküls im System kennen.

Wenn also Systeme betrachtet werden, die Spezies mit sehr geringen Konzentrationen enthalten, wie es beispielsweise bei der Genregulation der Fall ist, können solche Annahmen nicht immer mit gutem Gewissen getroffen werden.

Aufgrund dieser Einschränkungen ist es sinnvoll, alternative Modellierstrategien zu betrachten, die die physikalische Realität genauer rekonstruieren, wenn dies erforderlich ist.

2.2.2. Chemische Mastergleichungen

$$[\vec{S}](t) = \begin{bmatrix} [S_1](t) \\ [S_2](t) \\ \vdots \\ [S_i](t) \end{bmatrix} \tag{2.3}$$

Sei Gleichung 2.3 ein Zustand einer Reaktion zum Zeitpunkt t, wobei $[S_j](t)$ hier wieder die Konzentration der Spezies S_j zum Zeitpunkt t repräsentiert. Angenommen, die Reaktion r_i befindet sich in einem solchen Zustand, anstatt wie in der RRG angenommen als deterministischer Prozess, so kann der Zustandsübergang genauer als Zufallsprozess modelliert werden.

$$\frac{dP([\vec{S}],t)}{dt} = \sum_{j=1}^{M} \left(a_j([\vec{S}] - \vec{v}_j) P([\vec{S}] - \vec{v}_j, t) - a_j([\vec{S}]) P([\vec{S}], t) \right). \tag{2.4}$$

Diese Annahme führt zur *chemische Mastergleichung* (CMG) in Gleichung 2.4, bei der die Reaktion anhand der zeitlichen Entwicklung der Wahrscheinlichkeitsverteilung $P([\vec{S}], t)$ beschrieben wird. Dabei ist $P([\vec{S}], t)$ die Wahrscheinlichkeit, dass sich die Reaktion zum Zeitpunkt t im Zustand $[S]$ befindet.

$a_j([\vec{S}])$ bezeichnet hier die Wahrscheinlichkeit, dass die Reaktion j im Zustand $[\vec{S}]$ stattfindet, wobei $\vec{v}_j$ der Stöchiometrievektor ist, der sich aus den stöchiometrischen

Koeffizienten der Reaktion j ergibt. Somit ist $a_j([\vec{S}] - \vec{v}_j)$ die Wahrscheinlichkeit, dass die Reaktion j den Zustand $[\vec{S}]$ aus dem Zustand $[\vec{S}] - \vec{v}_j$ erreicht.

Die Summe in Gleichung 2.4 beschreibt folglich die *Wahrscheinlichkeitsflüsse* zwischen Zuständen, wobei, wie beschrieben, der erste Term $a_j([\vec{S}] - \vec{v}_j)P([\vec{S}] - \vec{v}_j, t)$ den Zustrom in den Zustand $[\vec{S}]$ durch eine Reaktion j angibt und der zweite Term $a_j([\vec{S}])P([\vec{S}], t)$ den Abfluss, also die Wahrscheinlichkeit, dass aus diesem Zustand die Reaktion j ausgelöst wird.

Im Allgemeinen ist es nicht möglich, die CMG direkt zu berechnen oder analytisch zu lösen, da reale Reaktionen oft sehr große bis unendliche Zustandsräume haben (Higham, 2008). Gillespie, 1977 schlägt daher den *stochastischer Simulationsalgorithmus* (SSA) vor, auch als *Gillespie-Algorithmus* in der Literatur bekannt, bei dem die CMG numerisch durch stichprobenartige Annäherung gelöst wird.

Die CMG entspricht einem kontinuierlichen Markov-Prozess, da der Übergang in den nächsten Zustand nur vom aktuellen Zustand abhängt und nicht von der Vergangenheit. Der SSA nach Gillespie kann also als Monte-Carlo-Simulation eines zeitkontinuierlichen Markov-Prozess interpretiert werden, da er Zustandsübergänge zufällig auf Grundlage der Reaktionswahrscheinlichkeiten bestimmt.

2.2.3. Inverses Problem

Das Problem, aus gegebenen experimentellen Daten, insbesondere zeitaufgelösten Konzentrationsdaten, die zugrunde liegenden Reaktionsmechanismen und deren Reaktionsraten zu bestimmen, nennt sich das inverse Problem der chemischen Kinetik (Santosa und Weitz, 2011). Mathematisch lässt es sich als Lösung eines linearen Systems beschreiben, das überbestimmt ist, also mehr Gleichungen als Unbekannte besitzt, und schlecht konditioniert ist, insbesondere aufgrund der Kopplung der Konzentrationsdaten.

Konzentrationsdaten allein beschreiben die zugrunde liegende Reaktion jedoch nicht eindeutig, da verschiedene Reaktionsmechanismen zu denselben beobachteten Konzentrationsverläufen führen können. Dies liegt unter anderem daran, dass nicht alle Zwischenprodukte oder Nebenreaktionen experimentell erfasst werden können und dass verschiedene Kombinationen von Reaktionsraten ähnliche Dynamiken erzeugen können. Santosa und Weitz, 2011 beschreibt zwei Zeitreihendaten als ununterscheidbar, wenn sie dieselbe RRG erzeugen. Das so formulierte inverse Problem ist also nicht eindeutig.

Aus Konzentrationsverläufen allein ist es außerdem nicht möglich, aussagekräftige Schlüsse über die stochastischen Momente der Reaktion zu ziehen, da sie nur als Mittelwert interpretiert werden können. Um aus experimentellen Daten auf ein der CMG (bzw. einer zeitkontinuierlichen Markov-Kette) zugrunde liegendes **CRN!** (**CRN!**) zu schließen, ist es notwendig, auch Informationen höherer Momente zu berechnen, beispielsweise durch Wiederholung der Messungen, wie in Klimovskaia et al., 2016 gezeigt.

3. Literaturrecherche

3.1. Methodik

Die Entwicklung eines möglichst breit anwendbaren Benchmarking-Werkzeugs erfordert einen qualitativen Vergleich der vorhandenen *Struktur-Inferenzmethoden*. Dadurch können die Faktoren identifiziert werden, nach denen die Benchmark-Ergebnisse verglichen werden, wie mögliche Schnittstellen gestaltet sein müssen sowie die Bestimmung der erforderlichen Eingabedaten und der erzielbaren Ergebnisse.

Eine umfassende Literatursammlung ist notwendig, um einen solchen Vergleich sinnvoll durchzuführen. Zu diesem Zweck wurde eine strukturierte Literaturrecherche durchgeführt, die sich an der in Templier und Paré, 2015 beschriebenen *Narrative Review* orientiert.

Sie beschreiben generelle Prozeduren zur Durchführung einer Literaturrecherche, und kategorisieren dann vier Formen einer solchen: *Narrative*, *Developmental*, *Cumulative* und *Aggregative Review*. Diese unterscheiden sich primär in ihren Zielen und Grad des notwendigen Rigor bei der Durchführung.

3.1.1. Allgemeine Prozedur

Die von Templier und Paré, 2015 vorgestellte allgemeine Strategie zur Durchführung einer Literaturrecherche ist in sechs Schritte aufgeteilt: (1) Problemformulierung, (2) Sammlung von Quellen, (3) Filtern der Quellen nach Inklusion, (4) Qualitätsbewertung der Quellen, (5) Auslesen von Daten, (6) Auswertung und Synthese der Daten.

Schritt (1) bildet die Grundlage für die Planung einer Review. In diesem ersten Schritt werden das Ziel der Literaturrecherche sowie die zu erfassenden Informationen festgelegt.

Im (2) Schritt erfolgt die Suche nach Literatur und Informationsquellen, die zur definierten Problemstellung passen. Dazu muss eine Suchstrategie definiert und begründet werden. Im Rahmen einer *Cumulative Review*, einer Kategorie von Literaturrecherche bei der versucht wird, Daten aus verschiedenen, ähnlichen Studien zu sammeln, um allgemeine Schlussfolgerungen zu ziehen, ist in diesem Schritt beispielsweise darauf zu achten, dass auch unveröffentlichte Studien in die Literatursammlung aufgenommen werden, um *Publication bias* entgegenzuwirken.

Nachdem eine Sammlung an Literaturquellen gefunden wurde, erfolgt in Schritt (3) eine Filterung nach Inklusion. Das bedeutet, dass eine Prüfung stattfindet, welche der Quellen das klar definierte Ziel der Review erfüllen und die erforderlichen Informationen oder Daten beinhalten.

Nachdem alle relevanten Quellen gesammelt sind, müssen diese im (4) Schritt noch

nach Qualität bewertet werden. In diesem Zusammenhang ist beispielsweise zu evaluieren, ob die Methodik der untersuchten Studie als sinnvoll zu erachten ist oder wie die Publikation von anderen Quellen bewertet wurde.

Abschließend können jetzt die gesuchten Daten extrahiert (5) und analysiert (6) werden.

Im Vergleich zur von Page et al., 2021 beschriebenen oft verwendeten PRISMA[1]-Strategie, deren Ziel die Realisierung einer möglichst nachvollziehbaren und wiederholbaren Literaturrecherche ist, erweist sich die von Templier und Paré, 2015 vorgestellte allgemeine Strategie zur Durchführung einer Literaturrecherche als deutlich flexibler. So hängt der Grad des geforderten Rigor, und welche Schritte in welcher Reihenfolge durchgeführt werden sollten von der definierten Problemstellung ab.

3.1.2. Narrative Review

Als *Narrative Review* wird in Templier und Paré, 2015 eine Literaturrecherche beschrieben, die bereits veröffentlichte Forschungsarbeiten zu einem Thema zusammenfasst. Ziel ist es, einen umfassenden Bericht über den aktuellen Wissensstand zum betrachteten Thema zu erstellen.

Generell besteht bei einer *Narrative Review* keine Notwendigkeit, Auswahlverfahren für Literaturquellen detailliert darzulegen. Ziel ist nicht die vollständige Zusammenstellung und Auswertung aller Literaturquellen zum gewählten Thema, sondern die Selektion einer repräsentativen Auswahl. Die angewandten Auswahlkriterien müssen lediglich insofern gerechtfertigt werden, als dass sie sich für die angestrebte Zielsetzung als sinnvoll erweisen.

Als ein Kernkriterium für eine *Narrative Review* beruft sich Templier und Paré, 2015 auf die Definition in Sandelowski et al., 2006 und Sandelowski et al., 2012, laut der sich die Datenauswertung in einer Literaturrecherche in zwei Kategorien einteilen lässt: Konfiguration und Assimilation. Die meisten *Narrative Reviews* führen nur eine Konfigurationsanalyse durch, das bedeutet es wird nach logischen Zusammenhängen in dem gesammelten Literaturbestand gesucht und versucht daraus eine schlüssige Argumentation oder These zu formen. Die Assimilationsauswertung versucht empirische Aussagen durch Zusammenfassung von Beweisen und Daten zu erreichen. In einer *Narrative Review* kann das zum Beispiel bedeuten, quantitativ festzuhalten, wie oft bestimmte Kategorien im gesammelten Literaturbestand vorkommen.

3.2. Durchführung

Bei der Durchführung der Literaturrecherche wurde der Strukturvorschlag von Templier und Paré, 2015 berücksichtigt. In der Aufgabenstellung wurden bereits mehrere Quellen als Beispiele gegeben, die als Grundlage für die Literatursuche dienten.

[1] *Preferred Reporting Items for Systematic reviews and Meta-Analyses*

3.2.1. Zieldefinition

Für das Design eines *Benchmarking-Tools* ist es notwendig, eine Sammlung von Implementierungen von Methoden zur Struktur- und Parameterinferenz zu haben. Das Primärziel der Literaturrecherche ist es folglich, Literatur zu identifizieren, die Quellcode zur Verfügung stellt oder beschreibt, wie die Methode implementiert werden kann. Sekundär ist es hilfreich, wenn aus der gefundenen Literatursammlung eine Kategorisierung von Ansätzen, die in der Struktur- und Parameterinferenz genutzt werden, extrahiert werden kann. Dadurch kann bei der Planung des *Benchmarking-Tools* einfacher entschieden werden, welche Parameter der Methoden verglichen werden sollen. Dabei sollten auch Ansätze berücksichtigt werden, die noch keine Implementierung besitzen, damit beim Entwurf des *Benchmarking-Tools* auf Zukunftssicherheit Rücksicht genommen werden kann.

3.2.2. Quellensammlung

Für die Literatursuche wurde *Google Scholar*[2] genutzt, eine Suchmaschine, die Literatur aus Journals, akademischen Büchern und Konferenzen indiziert, einschließlich Vorveröffentlichungen sowie Thesen, Dissertationen und mehr.

Vor Beginn der Recherche wurde eine Liste von Suchtermen definiert. Hierfür wurden die Quellen aus der Aufgabenstellung nach häufig verwendeten Begriffen durchsucht, die automatische Methoden zur Struktur- und Parameterinferenz von Reaktionsmodellen beschreiben. Bereits in den Beispielquellen konnten mögliche Kategorien identifiziert werden, zu denen weitere Quellen gesucht werden konnten.

Ein Problem mit *Google Scholar* ist, dass Suchterme wortgenau gesucht werden. Das bedeutet, dass eine Suche mit dem Suchterm *Reaction Network* keine Literatur mit *Reaction Networks* im Titel findet. *Google Scholar* bietet jedoch die Möglichkeit, Suchterme mit **OR** oder **AND** logisch zu verknüpfen. Diese Funktion wurde genutzt, um mit einer Suche gleichzeitig nach der Singular- und Pluralform der Suchterme zu suchen.

Ein weiteres nützliches Werkzeug, das *Google Scholar* anbietet, ist die Verwendung von Anführungszeichen. Dadurch werden nur Quellen gefunden, deren Titel genau die Wörter im Suchterm in der gegebenen Reihenfolge enthalten.

Für die ersten Suchen wurde dies genutzt, um die Anzahl der Suchergebnisse einzuschränken. Es stellte sich jedoch nach wenigen Suchen heraus, dass diese Einschränkung zu stark war und mehrere Suchterme gar keine Ergebnisse lieferten. Daher wurden dieselben Suchen ohne Anführungszeichen wiederholt. Hierbei trat das gegenteilige Problem auf: Einige Suchen lieferten tausende Ergebniseinträge. Deshalb wurde entschieden, nur die ersten 10 Seiten jeder Suche nach relevanten Quellen zu durchsuchen. Eine Auswahl der genutzten Suchterme findet sich im Anhang Anhang A.

Bei dieser initialen Quellensammlung wurden Quellen ausschließlich anhand ihres Titels als relevant eingeordnet und zur Literatursammlung hinzugefügt. Zur Verwaltung dieser wurde *Zotero*[3], ein Literaturverwaltungsprogramm, verwendet. Eine quantitative Auswertung der gefundenen Quellen findet sich in Abschnitt 3.3.

[2] https://scholar.google.de
[3] https://www.zotero.org

Name	Status	Eingabe...	Daten/Benc...	V...	Kategorie...	Implementati...	Note
Ji and Deng, 2021	Skimmed	Zeitreihen		☐	Neural Net...	Verfügbar	Autoencoder Impl.
Galagali and Marzouk, 2015	Skimmed	Verteilun...		☐	Bayesian In...	Implementierbar	Hat Pseudocode, scheint fu
Oates et al., 2014	Didn't read	Zeitreihen		☐	Bayesian In...	Verfügbar	nonlinear?
Camacho et al., 2007	Didn't read			☐			Gene Regulatory
Rausanu et al., 2015	Skimmed	Zeitreihen		☐	genetic pro...	Implementierbar	Simulated Annealing releva
Bortolussi et al., 2023	Skimmed			☐	Neural Net...		1 Zitierung, keine Implemer
Yang et al., 2019	Didn't read			☐			Eher optimierung/verkleine
Kontos et al., 2014	Skimmed	Zeitreihen		☐		Implementierbar	Gute Einleitung, def worth :
Kreikemeyer et al., 2024	Skimmed	Zeitreihen		☐	genetic pro...	Verfügbar	GOAL
Nobile and Mauri, 2013	Skimmed	Zeitreihen		☐	genetic pro...		Gibt es ne implementation?
Kiehl, 2009	Didn't read	Zeitreihen		☐	genetic pro...		Erzeugt netzwerke die best
Galagali and Marzouk, 2019	Skimmed	Zeitreihen	Reale	☐	Bayesian In...	Verfügbar	
Luzano	Skimmed	Zeitreihen	Reale	☑	symbolic re...	Implementierbar	Anwendungsbeispiel

Abbildung 3.1.: Ausschnitt aus der *Notion* Datenbank

3.2.3. Filterung nach Inklusion

Nach dem Abschluss der Literatursammlung wurden die Quellen in eine durch *Notion*[4] realisierte Datenbank übertragen (Abbildung 3.1). *Notion* ist ein Programm zur strukturierten Wissensverwaltung und bietet flexible Möglichkeiten zur Verwaltung von Datenbanken. Durch Erweiterungen wird außerdem eine *Zotero*-Integration angeboten, wodurch die Literatursammlung automatisch übertragen und synchronisiert wird.

Jedem Literatureintrag wurde das Feld *Relevanz* zugeordnet, in dem festgehalten werden kann, wie relevant die Quelle für das Thema ist. Als relevant wurden alle Literaturquellen eingestuft, die Methoden zur Struktur- und Parameterinferenz von Reaktionsmodellen beschreiben.

Die meisten gefilterten Quellen wurden aus folgenden Gründen ausgeschlossen: (1) Ältere Quellen, die sich nur mit der Theorie und Lösbarkeit des Problems beschäftigen, aber keine Methode zur Lösung beschreiben, (2) Quellen, die sich mit Strukturoptimierung befassen, um beispielsweise effizientere Simulationen zu ermöglichen, anstatt mit Strukturinferenz, (3) Literaturquellen, deren Titel zwar relevant erscheint, die sich jedoch nur mit tangential relevanten Themen befassen.

Literatur, die aufgrund von (1) ausgeschlossen wurde, wurde gesondert markiert. Auch wenn sie nicht genau das Thema erfüllt, kann sie dennoch für das Schreiben der finalen Arbeit relevant sein.

3.2.4. Qualitätsbewertung der Quellen

Die wissenschaftliche Qualität der Quellen war kein ausschlaggebendes Kriterium für die Inklusion, weshalb keine Qualitätsbewertung der Methodik in der Literatur vorgenommen wurde. Eine Einschätzung der Qualität wurde lediglich in Bezug auf die Relevanz vorgenommen: Quellen mit vielen Zitierungen, aber auch neuere Quellen mit

[4] https://www.notion.so

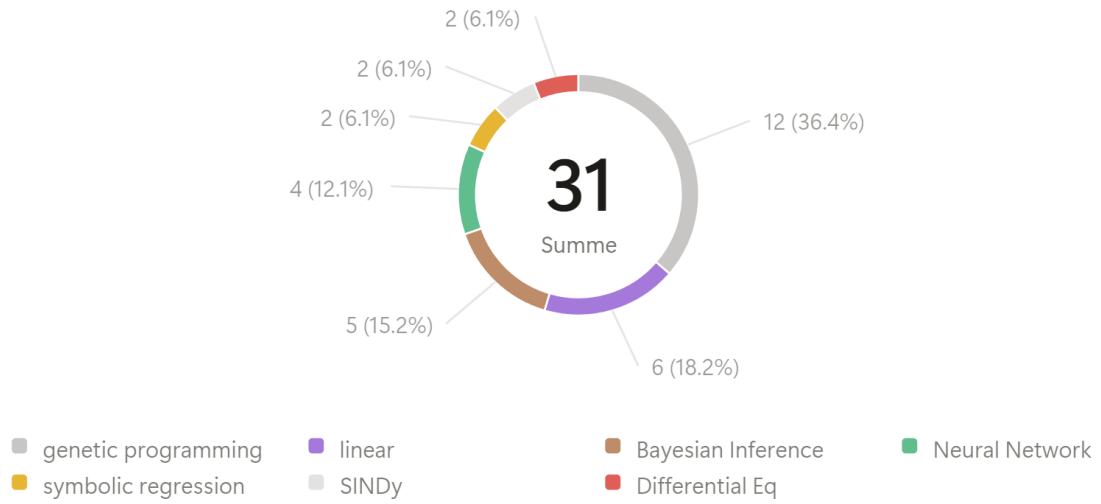


Abbildung 3.2.: Frequenz der Kategorien

wenigen Zitierungen, haben eine höhere Wahrscheinlichkeit, repräsentativ für das Feld zu sein.

3.2.5. Auslesen und Auswertung der Daten

Ein initiales Auslesen der Quellen wurde bereits während der Filterung (Unterabschnitt 3.2.3) durchgeführt. Dabei wurde festgehalten, welche der Quellen Implementationen zur Verfügung stellen oder detailliert beschreiben, sodass eine eigene Implementierung möglich ist. Nachdem die finale Literatursammlung definiert wurde, wurde eine genauere Konfigurationsanalyse durchgeführt. Der Hauptfokus lag auf den Methoden, die Implementationen zur Verfügung stellen, da diese die Basis für das Design des *Benchmarking Tools* darstellen. Es wurde versucht, Kategorien zu identifizieren, in die sich diese Methoden einteilen lassen. Anschließend wurde in den Quellen ohne Implementation nach Methoden gesucht, die sich in diese definierten Kategorien einordnen ließen. Methoden ohne Implementation, die keiner implementierten Kategorie zugeordnet werden konnten, sind weniger relevant für das Design des *Benchmarking Tools*, wurden jedoch nicht aus der Konfigurationsanalyse ausgeschlossen.

Das Ergebnis der Konfigurationsanalyse wird im qualitativen Vergleich (Abschnitt 3.4) beschrieben.

3.3. Quantitative Resultate

Die initiale Literaturrecherche ergab 178 Quellen, die auf 58 relevante Arbeiten gefiltert wurden. Nach einer systematischen Auswertung wurden diese Quellen in einer Tabelle zusammengefasst und anhand der Parameter *Kategorie* und *Verfügbarkeit einer Implementierung* klassifiziert.

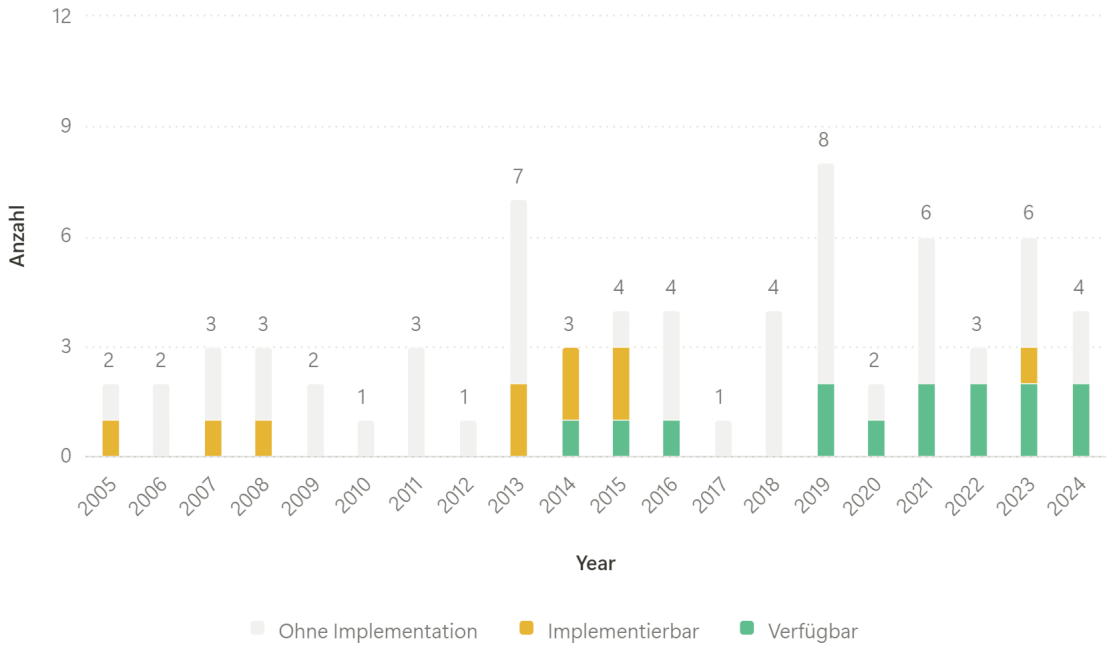


Abbildung 3.3.: Literatursammlung nach Veröffentlichungsdatum aufgeschlüsselt. Als implementierbar wurden Quellen kategorisiert, die entweder genau beschreiben, wie sie zu implementieren sind, oder detaillierten Pseudocode angeben.

Von den 58 analysierten Methoden konnten 31 einer spezifischen Kategorie zugeordnet werden. Die übrigen Methoden ließen sich keiner Kategorie mit einer verfügbaren Implementierung zuweisen.

Bei der Definition des Ziels der *Narrative Review* wurde sich darauf beschränkt, einen explorativen Überblick über das Feld mithilfe einer Konfigurationsanalyse zu erstellen. Im Folgenden wird eine Auswertung der Literaturrecherche gegeben, allerdings ist zu beachten, dass aufgrund der auf der Zieldefinition basierenden Suchstrategie nur schwer Schlüsse aus den *quantitativen* Ergebnissen geschlossen werden können.

Wenn eine solche Assimilationsauswertung gewünscht gewesen wäre, hätte man bei der Suchstrategie gezielt vermeiden müssen, dass bestimmte Suchterme umfangreicher als andere durchsucht wurden, und es hätte eine strenge Auswahl der Ursprungsliteratur (hier die aus der Aufgabenstellung gegebenen Literaturbeispiele) erfordert. Dies hätte allerdings die Konfigurationsanalyse beeinträchtigt, da durch eine solche engere Literaturauswahl die Größe der endgültigen Literatursammlung eingeschränkt wäre und viele Nischenentwicklungen nicht berücksichtigt worden wären.

Es ist folglich zu beachten, dass aus den gesammelten quantitativen Daten nur Aussagen über die Literatursammlung, die aus diesem spezifischen Literaturrechercheprozess entstanden ist, getroffen werden können und nicht über das gesamte Forschungsthema der kombinierten Struktur- und Parameterinferenz.

3.3.1. Auswertung

Die Kategorien, in die die Methoden eingeteilt wurden, sind ad hoc definiert worden und wurden primär als Orientierungshilfe für den qualitativen Vergleich genutzt. Man kann direkt an Abbildung 3.2 erkennen, warum eine allgemeingültige quantitative Analyse für die gefundene Literatur nicht sinnvoll wäre. *Genetic Programming* war beispielsweise in mehreren der Suchterme direkt erwähnt, während nie direkt nach Sparse Identification of Nonlinear Dynamical systems (SINDy) gesucht wurde. Darum ist es nicht überraschend, dass *Genetic Programming* überproportional vertreten ist.

Das Kriterium der Suche war es, möglichst viele Methoden mit verfügbaren Implementationen zu finden. Anhand von Abbildung 3.3 ist zu erkennen, dass dieses Ziel mit der genutzten Suchstrategie gut erfüllt werden konnte. Zum einen wurde Literatur aus einem großen Zeitraum gefunden, von 2005 bis 2024, sowie eine nicht dargestellte Quelle aus dem Jahr 1995. Zum anderen wurden mehr Quellen aus den letzten zehn Jahren identifiziert, in denen eine höhere Wahrscheinlichkeit für das Vorhandensein von Implementationen vermutet wurde, was sich anhand der Resultate bestätigen ließ.

3.4. Qualitativer Vergleich

Durch den qualitativen Vergleich, auch Konfigurationsanalyse in Templier und Paré, 2015, sollen Methoden zur kombinierten Struktur- und Parameterinferenz kategorisiert und analysiert werden. Das Ziel hierbei ist es vor allem, Eigenschaften der Methoden zu identifizieren, die für die Konzeption des *Benchmarking Tools* relevant sind. Dazu gehören insbesondere die Eingabedaten, auf denen die Methoden arbeiten, die Ausgabedaten, die aus der Methode erhalten werden, sowie die Form der Ausführung.

3.4.1. Methoden

Unter den Methoden, die Implementationen zur Verfügung stellen – d.h. vor allem Methoden aus dem Zeitraum 2019 bis 2024 – ist ein Trend hinsichtlich der Erweiterung der SINDy-Methode von Brunton et al., 2016 zu erkennen. SINDy ist eine Strategie zur symbolischen Regression, um dynamische Systeme an einen Datensatz anzupassen. SINDy verwendet dazu eine Kombination aus numerischer Differentiation, einer Bibliothek möglicher nichtlinearer Funktionen und sparsamen Regressionsverfahren, um eine möglichst kompakte Gleichung zu identifizieren. Ein typischer Workflow besteht darin, zunächst die zeitlichen Ableitungen der beobachteten Zustandsvariablen zu approximieren und dann eine Auswahl relevanter Terme aus einer vorgegebenen Funktionsbibliothek zu treffen. Hierbei kommen Techniken wie die Lasso-Regression (Roth, 2004) zum Einsatz, um irrelevante Terme zu eliminieren. Dadurch entsteht ein einfaches, interpretierbares Modell, das die zugrunde liegende Dynamik mit hoher Genauigkeit beschreibt. Dies ist vor allem bei der kombinierten Struktur- und Parameterinferenz von CRN von Vorteil, da ein Ziel, das viele Ansätze verfolgen, insbesondere die Generierung von menschlich interpretierbaren Modellen ist. Aus diesem Grund lassen sich direkte *Blackbox*-Ansätze aus dem maschinellen Lernen, wie neuronale Netze, nicht ohne Weiteres nutzen (Ji und Deng, 2021).

Mangan et al., 2016 ist die erste Quelle, die versucht, mit SINDy CRN zu rekonstruieren. Dazu entwickeln sie *implicit*-SINDy, bei dem im Gegensatz zu SINDy nicht nach einer Differentialgleichung in expliziter Form $\dot{X} = f(X)$, sondern in der impliziten Form $f_N(X) - f_D(X)\dot{X} = 0$ gesucht wird, wobei f_N und f_D Polynome sind. Hierdurch können Gleichungen abgebildet werden, die nicht direkt durch SINDy erfassbar sind. Mangan et al., 2016 argumentieren, dass hierdurch vor allem Abhängigkeiten wie das Massenerhaltungsgesetz abgebildet werden können.

Bhatt et al., 2023 setzt *implicit*-SINDy fort. Hier wird nun auch versucht, die Anzahl der Zustandsvariablen – im Kontext von CRN Spezies – zu reduzieren. Sie argumentieren, dass es in CRN oft mehr Spezies als tatsächlich unabhängige Reaktionsprozesse gibt und dass Spezies häufig in stöchiometrischer Abhängigkeit miteinander stehen. Hierfür nutzen sie die Singular Value Decomposition, eine Optimierungsmethode zur Bestimmung der minimalen Anzahl an Zustandsvariablen.

Hoffmann et al., 2019 mit *reactive*-SINDy und Burrage et al., 2024 mit *coupled*-SINDy haben unabhängig voneinander Ansätze entwickelt, bei denen die Kopplung der Variablen direkt bei der Lösung der linearen Gleichung berücksichtigt wird, indem die Funktionsbibliothek nicht aus Polynomen, sondern direkt aus gekoppelten Termen besteht, die sich als Reaktionen interpretieren lassen.

Durch diesen Ansatz zur Erzeugung der Funktionsbibliothek wächst diese jedoch kombinatorisch mit der Anzahl der Spezies, was die Skalierbarkeit limitiert. Kreikemeyer et al., 2024 versucht dieses Problem zu begrenzen, indem die Funktionsbibliothek durch Anwendung einer evolutionären Optimierungsstrategie reduziert wird.

Ein ähnlicher Ansatz wie bei Hoffmann et al., 2019 und Burrage et al., 2024 findet sich bereits in Santosa und Weitz, 2011, also fünf Jahre vor Brunton et al., 2016, jedoch unter einem anderen Namen. Als Nachteil der SINDy-Methoden identifiziert Martinelli et al., 2023, dass diese Schwierigkeiten bei der Identifikation aus *wild-type*-Daten haben. Damit sind experimentelle Bedingungen gemeint, bei denen alle molekularen Spezies in ihrer natürlichen, unveränderten Form vorliegen, ohne gezielte genetische oder chemische Manipulationen. Um dieses Problem zu adressieren, stellt Martinelli et al., 2023 *Reactmine* vor, einen statistischen Suchalgorithmus, bei dem die Sparsität durch eine gezielte Begrenzung des Suchbaums erzwungen wird.

SINDy-Ansätze rekonstruieren eine RRG, da sie ausschließlich auf den Ergebnissen eines einzelnen Experiments basieren. Dadurch können sie keine Informationen über die zugrunde liegenden stochastischen Momente liefern, die eine CRN präziser beschreiben.

Im Gegensatz dazu verfolgt Klimovskaia et al., 2016 mit *ReactioNet Lasso* einen anderen Ansatz: Statt nur auf Zeitreihendaten eines einzelnen Experiments zu arbeiten, nutzt diese Methode statistische Momente, die aus mehrfachen Wiederholungen desselben Experiments gewonnen werden. Dadurch wird eine genauere Rekonstruktion der CMG ermöglicht.

Einen anderen Ansatz verfolgen Ji und Deng, 2021. Hier wird ein neuronales Netz auf zeitaufgelösten Konzentrationsdaten trainiert, wobei während des Trainings bestimmte Eigenschaften erzwungen werden, die eine Interpretation des resultierenden Netzwerks als CRN ermöglichen sollen. Zudem schlagen Ji und Deng, 2021 Varianten dieser Methode vor, bei denen physikalisches Vorwissen eingebracht werden kann, um den benötigten Umfang an Trainingsdaten zu reduzieren.

Die meisten der beschriebenen Methoden – mit Ausnahme von *ReactioNet Lasso* – liefern als Ergebnis eine Punktschätzung der möglichen Reaktionsgleichung. Das bedeutet, dass sie entweder die „beste“ Reaktionsgleichung oder eine Liste der „besten“ Gleichungen ausgeben, jedoch keine Informationen über die Unsicherheit der einzelnen Parameter bereitstellen.

Hier setzen Bayes'sche Methoden wie Jiang et al., 2022 und Nieves et al., 2024 an, die zusätzlich eine Quantifizierung der Unsicherheit der Modellparameter ermöglichen.

4. Benchmarking Tool

4.1. Konzeption

Das *Benchmarking Tool* soll Wissenschaftlern ermöglichen, Methoden der kombinierten Struktur- und Parameterinferenz wiederholbar zu vergleichen. Um dies zu gewährleisten, muss das *Benchmarking Tool* möglichst flexibel gestaltet werden, sodass Nutzer es an ihre individuellen Anforderungen anpassen können.

Es ist davon auszugehen, dass Nutzer das *Benchmarking Tool* sowohl um eigene Implementierungen neuer Inferenzmethoden erweitern als auch bestehende Methoden – wie beispielsweise in Kreikemeyer et al., 2024 beschrieben – ohne erneute Implementierung vergleichen möchten. Daraus ergibt sich eine klare Anforderung an die Modularität des *Benchmarking Tool* .

Da diese Anpassung einen grundlegenden Nutzungsfall des *Benchmarking Tool* darstellt, sollte hierfür kein tiefgehendes Wissen über die interne Programmstruktur erforderlich sein. Stattdessen sollte ein pluginbasiertes Interface zur Verfügung stehen, das die Eingaben des *Benchmarking Tool* in ein für die jeweilige Methode gültiges Format überträgt und deren Ausgaben in eine kanonische Repräsentation des *Benchmarking Tool* umwandelt.

Dabei ist zu beachten, dass viele Methoden neben den Eingabedaten auch eine Vielzahl von Hyperparametern besitzen – beispielsweise die Baumtiefe in Martinelli et al., 2023. Zudem ist es von Interesse, dieselbe Methode unter verschiedenen Parametereinstellungen zu vergleichen. Um zu vermeiden, dass für jede Kombination von Hyperparametern ein eigenes Plugin erstellt werden muss, sollte das *Benchmarking Tool* die Möglichkeit bieten, diese Parameter direkt zu definieren und an das jeweilige Plugin zu übergeben.

Zum Vergleich der Inferenzmethoden ist es notwendig, quantifizierbare Eigenschaften dieser Methoden zu bestimmen. Zum einen sollten diese widerspiegeln, wie gut das erzeugte CRN die übergebenen Daten erklärt – hierfür eignet sich beispielsweise der mittlere quadratische Fehler. Zum anderen sollte bewertet werden, wie nahe das rekonstruierte Modell am Referenzmodell liegt, etwa durch das Zählen identischer Terme im CRN. Da ein zentrales Ziel vieler Methoden die Generierung interpretierbarer CRN ist (Ji und Deng, 2021), sollten zusätzlich Metriken berücksichtigt werden, die diese Eigenschaft widerspiegeln – beispielsweise die Anzahl der Reaktionen im erzeugten CRN.

Damit solche Vergleiche mit dem Referenzmodell durchgeführt werden können, muss dieses bekannt sein. Zu diesem Zweck kann eine minimale Beschreibungssprache definiert werden, um Modelle in externen Dateien zu spezifizieren.

Um maximale Reproduzierbarkeit und Genauigkeit zu gewährleisten, sollten die Konzentrationsdaten idealerweise durch Simulation des Modells generiert werden. Gleich-

Abbildung 4.1.: Top-Down Architektur des *Benchmarking Tool*

zeitig muss das zuvor definierte Flexibilitätskriterium gewahrt bleiben. Daher sollte es keine Einschränkungen geben, wenn stattdessen nur anhand vorhandener Konzentrationsdaten getestet werden soll oder Zeitreihendaten und Modell getrennt eingegeben werden. Dies ist insbesondere relevant, wenn beispielsweise echte Experimentaldaten vorliegen und ein geschätztes Modell durch Expertenwissen ergänzt wird oder wenn die Simulation des Modells zu zeitaufwendig wäre.

Um die Wiederholbarkeit von Experimenten zu gewährleisten, sollten die Ausführungsparameter des *Benchmarking Tool* in einer externen Datei gespeichert werden. Dazu gehören beispielsweise Informationen darüber, welche Daten oder Modelle für den Vergleich verwendet wurden, welche Plugins zum Einsatz kamen und welche Hyperparameter diesen übergeben wurden.

4.2. Architektur

> "This is the Unix philosophy: Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."
>
> — Doug McIlroy, zitiert in Mayer, 2022

Beim Definieren der Anforderungen wird deutlich, dass viele der im *Benchmarking Tool* erzeugten Daten auch außerhalb des Tools von Nutzen sind. Beispielsweise soll das *Benchmarking Tool* in der Lage sein, für in Textdateien definierte Modelle Konzentrationsdaten zu simulieren – eine potenziell rechenintensive Aufgabe. Diese generierten Konzentrationsdaten könnten jedoch auch in anderen Programmen oder Tools weiterverwendet werden. Zudem ist es nicht immer erforderlich, eine Inferenzmethode direkt im *Benchmarking Tool* auszuführen. Stattdessen könnten Nutzer ausschließlich an der Berechnung quantifizierbarer Metriken und der Erstellung eines zugehörigen Berichts interessiert sein, während die eigentliche Ausführung der Methoden in externen Werkzeugen erfolgt.

Aufgrund dieser Eigenschaften, insbesondere des Flexibilitätskriteriums, bietet es sich an, die Architektur nach einem Unix-inspirierten Ansatz zu gestalten. Das bedeutet, dass das Programm in drei separate Komponenten unterteilt wird: Datengenerierung, Pluginausführung und Evaluation. Jede dieser Komponenten sollte als eigenständiges Programm funktionsfähig sein und über Textschnittstellen kommunizieren können.

4.2.1. Datengenerierung

Die Komponente **Datengenerierung** ist dafür verantwortlich, ein Modell aus einer Textdatei in eine interne Datenstruktur für Reaktionsmodelle zu überführen und darauf basierend Daten zu simulieren. Diese Daten sollten nicht nur intern gespeichert, sondern auch in einer Datei abgelegt werden, sodass sie von externen Programmen genutzt werden können.

4.2.2. Pluginausführung

Die **Pluginausführung** ist dafür verantwortlich, alle für den Vergleich vorgesehenen Plugins mit ihren jeweiligen Hyperparametern zu initialisieren und auszuführen, wobei auch Messungen wie beispielsweise die Laufzeit erfasst werden. Nach der Ausführung werden alle Ergebnisse zusammen mit den Eingabedaten weitergegeben. Sowohl die Eingabedaten als auch die erzeugten Ausgabemodelle sollten zusätzlich in Dateien gespeichert werden.

Die Plugins selbst sollten nicht direkt Teil des Programms sein, sondern in einem externen Verzeichnis abgelegt werden, um sie vom Hauptprogrammcode zu trennen.

4.2.3. Evaluation

Im **Evaluationsmodul** werden die Ausführungsergebnisse der Plugins analysiert. Dazu müssen die von den Plugins erzeugten CRN mit denselben Parametern simuliert werden wie die Daten des Referenzmodells. Die daraus resultierenden Simulationsergebnisse sollten extern gespeichert werden, sodass sie beispielsweise für Visualisierungen genutzt werden können. Anschließend werden verschiedene Metriken berechnet und in einem finalen Bericht ausgegeben. Nutzer sollten zudem die Möglichkeit haben, eigene Metriken zu definieren und in die Analyse zu integrieren.

4.3. Implementation

Um die Nützlichkeit des beschriebenen *Benchmarking Tool* zu demonstrieren, wurde eine erste prototypische Implementierung entwickelt. Ziel dieser Implementierung ist es, die zentralen Funktionalitäten aufzuzeigen und anschließend in Abschnitt 4.4 eine Fallstudie an drei Methoden zur Struktur- und Parameterinferenz von Reaktionsnetzwerken durchzuführen.

Für die Implementierung wurde die Programmiersprache *Python*[1] in Version 3 gewählt. *Python* ist die am häufigsten genutzte Programmiersprache in der Wissenschaft Guo, 2021 und eignet sich daher besonders gut für ein erweiterbares Programm, das Wissenschaftlern eine einfache Anpassung ermöglichen soll. Zudem nutzen viele der in Kapitel 3 betrachteten Implementierungen ebenfalls Python. Ein weiterer Vorteil ist die große Anzahl wissenschaftlicher Bibliotheken, die insbesondere für die Evaluation und Simulation nützlich sind.

[1] python.org

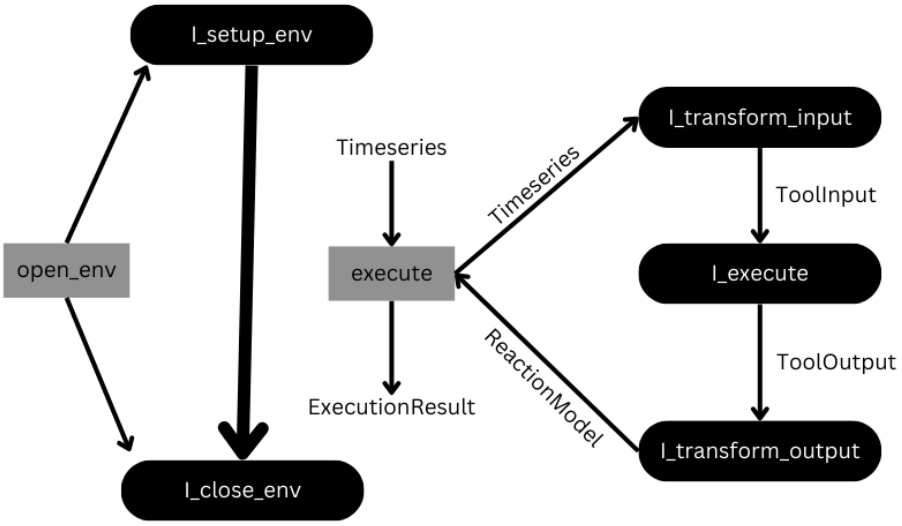


Abbildung 4.2.: Ausführungsdiagram von **BenchmarkPlugin**

4.3.1. Datenstrukturen

Die zwei zentralen Datenstrukturen dieser Implementierung sind **ReactionModel** und **Timeseries**. **Timeseries** repräsentiert die internen Konzentrationsdaten und besteht aus einer Liste der Speziesnamen sowie einer Liste von **timestamp**-Objekten. Ein **timestamp** ist eine einfache Datenklasse, die einen Zeitpunkt als **float** sowie eine zugehörige Liste von Konzentrationswerten, ebenfalls als **float**, speichert.

ReactionModel ist die zentrale Datenstruktur des *Benchmarking Tool* . Sie dient sowohl als Eingabemodell für die Datengenerierung und Evaluation als auch als Ausgabemodell der Pluginausführung. **ReactionModel** besteht aus einer Liste der Speziesnamen und einer Liste von **Reaction**-Objekten.

Ein **Reaction**-Objekt ist eine Datenklasse mit einem Feld für die Reaktionsrate als **float** sowie zwei Listen von **Reactant**-Objekten – jeweils für Reaktanten und Produkte. Jedes **Reactant**-Objekt speichert die zugehörige Spezies und deren Stöchiometrie.

Zusätzlich bietet **ReactionModel** Methoden zur Serialisierung und Deserialisierung im **.json**-Format sowie zur formatierten Ausgabe als **String**.

4.3.2. Pluginsystem

Plugins werden als *Python*-Dateien im Verzeichnis **/plugins** abgelegt. Sie implementieren die abstrakte Klasse **BenchmarkPlugin**.

BenchmarkPlugin besitzt zwei nicht überschreibbare Methoden: **execute** und **open_env**. Nutzer müssen jedoch die Methoden **I_execute**, **I_transform_input** und **I_transform_output** überschreiben.

(1) **I_execute** ist für die Ausführung der implementierten Methode zuständig. Diese muss nicht direkt im Plugin enthalten sein, sondern kann auch ein externes Programm aufrufen. Die genaue Implementierung ist dem Nutzer überlassen. (2) **I_transform_input** nimmt eine **Timeseries** als Eingabe und wandelt sie in das für **I_execute** benötigte Format um. Dadurch kann die bevorzugte Datenstruktur der implementierten Methode genutzt werden. (3) **I_transform_output** ist das Gegenstück zu **I_transform_input** und konvertiert die Ausgabe von **I_execute** in ein **ReactionModel**.

Durch diese Aufteilung ist es möglich, die Berechnungszeit von **I_execute** zu messen, ohne die Zeit für die Umwandlung der Ein- und Ausgabedaten zu berücksichtigen.

Die Ausführung eines Plugins im Modul *Pluginausführung* erfolgt über die Methode **execute**. Diese gibt als Ergebnis ein **ExecutionResult**-Objekt zurück, das die Eingabe, Ausgabe, Ausführungsdauer und den Zeitpunkt der Ausführung speichert.

Bevor **execute** aufgerufen werden kann, muss sichergestellt werden, dass die Methode **I_setup_env** bereits ausgeführt wurde. In dieser Methode können Nutzer zeitaufwendige Initialisierungen implementieren, die nicht bei jedem Aufruf von **execute** erneut ausgeführt werden sollen.

Die Implementierung von **I_setup_env** ist optional. Falls sie jedoch genutzt wird, sollte auch **I_close_env** implementiert werden, um die Umgebung ordnungsgemäß zu deinitialisieren.

Die Verwaltung von **I_setup_env** und **I_close_env** erfolgt über die Methode **open_env**. Diese gibt einen Kontextmanager zurück, sodass das Plugin mit dem **with**-Keyword in *Python* genutzt werden kann. Dadurch wird sichergestellt, dass das Plugin automatisch deinitialisiert wird, sobald es den aktuellen Kontext verlässt.

Falls ein Plugin Hyperparameter benötigt, sollten diese als Parameter im Konstruktor angegeben werden. Jeder Hyperparameter sollte dabei einen **default value** besitzen.

Zusätzlich muss die **property info** überschrieben werden. Diese gibt ein **Info**-Objekt zurück, das Metainformationen über die im Plugin implementierte Methode enthält, wie beispielsweise deren Namen.

Eine Beispielimplementierung eines Plugins findet sich in Anhang B.

Das Laden der Plugins erfolgt über die Funktion **load_plugins**, jedoch werden sie dabei noch nicht initialisiert. Diese Funktion durchsucht den Ordner **/plugins** nach allen *Python*-Dateien, die **BenchmarkPlugin** implementieren, und lädt sie als Module mithilfe des **importlib**-Moduls.

4.3.3. Datengenerierung

Im Programmteil **Datengenerierung** werden Modelle, die in einer Form wie in Abbildung 4.3 als Textdateien definiert sind, in ein **ReactionModel** geladen.

```
1  S I R
2  1980 20 0
3  S + I > 2 I @ 0.02
4  I > R @ 5
```

Abbildung 4.3.: Textbeschreibung eines Reaktionsmodels

```
model:predatorprey.txt 1.0 1000
plugin:evolib-csindy.py
plugin:evolib-evolving_sindy.py reaction_number=5
plugin:evolib-evolving_sindy.py reaction_number=2
plugin:reactmine.py gamma=6 beta=7 delta_max=3
plugin:reactmine.py gamma=3 delta_max=3
```

Abbildung 4.4.: Textbeschreibung eines Experiment

Die erste Zeile der Datei enthält die Namen der Spezies, während die zweite Zeile die Initialpopulationen für die Simulation angibt.

Das **ReactionModel** wird anschließend numerisch mithilfe der Bibliothek **scipy**[2] simuliert. Die Simulationsergebnisse werden sowohl als **.csv**-Datei gespeichert als auch als **Timeseries** weitergegeben.

4.3.4. Evaluator

Das Programm **Evaluator** erhält als Eingabe das erzeugte **ReactionModel**, die Eingabe-**Timeseries**, das Referenz-**ReactionModel**, aus dem die Eingabedaten generiert wurden, sowie alle Ausführungsdaten des Plugins.

Zum Vergleich wird das gleiche Verfahren wie in der **Datengenerierung** zur Simulation des erzeugten **ReactionModel** verwendet. Anschließend können die beiden **Timeseries** mithilfe einfacher Metriken direkt verglichen werden. In dieser prototypischen Implementierung wurden der R^2**-Wert** sowie der **mittlere quadratische Fehler** berechnet. Das Evaluationsprogramm wurde bewusst einfach gehalten, sodass es um beliebige weitere Metriken erweitert werden kann.

Zusätzlich wird das erzeugte **ReactionModel** direkt mit dem Referenzmodell verglichen. Dabei werden zwei Metriken verwendet: die Anzahl der überlappenden Reaktionsterme sowie das Verhältnis der Anzahl der generierten Reaktionsterme zur Referenz.

Abschließend gibt der Evaluator einen Bericht sowohl als **.json**-Datei als auch als Textstream aus.

4.3.5. Benchmarking Tool

Die Aufgabe des *Benchmarking Tool* besteht darin, die zu vergleichenden Plugins mit den korrekten Hyperparametern zu initialisieren, dem Datengenerierungsprogramm die

[2] scipy.org

richtige Modelldatei zu übergeben und anschließend die generierte **Timeseries** an die Plugins weiterzuleiten. Danach werden die Ergebnisse dem Evaluator in der korrekten Form übergeben. Abschließend wird ein Bericht erstellt, der alle im Verlauf des Programms generierten Eingaben und Ausgaben enthält. Dieser Bericht wird als **.json**-Datei gespeichert, um die Reproduzierbarkeit zu gewährleisten.

Die Eingaben werden in einer Textdatei beschrieben, die eine Struktur ähnlich der in Abbildung 4.4 dargestellten Form aufweist. Zeilen, die mit **model:** beginnen, geben die Datei an, in der das Modell beschrieben ist, sowie den Zeitpunkt, bis zu dem es simuliert werden soll, und die Anzahl der gewünschten Beobachtungen. Zeilen, die mit **plugin:** beginnen, enthalten den Namen der Plugins sowie die zu übergebenden Hyperparameter. Falls mehr als ein Modell definiert ist, führt das *Benchmarking Tool* für jedes Modell einen vollständigen Durchlauf durch.

4.4. Fallstudie

Um eine Fallstudie durchzuführen, wurden die von Kreikemeyer et al., 2024 implementierten Methoden *Coupled SINDy* und *Evolving SINDy* mit der in Martinelli et al., 2023 beschriebenen Implementierung von *Reactmine* verglichen.

Für die Implementierung der Plugins wurde die Methode **I_execute** genutzt, um die bereits vorhandenen Varianten mithilfe des **subprocess**-Moduls aus der *Python*-Standardbibliothek auszuführen. Der Kommandozeilen-Output dieser Programme wurde dann in der Methode **I_transform_output** direkt in ein **ReactionModel** umgewandelt.

Für *Coupled SINDy* und *Evolving SINDy* erfolgte die Eingabe über einen Dateipfad, da diese Programme **.csv**-Dateien als Eingabe erwarten. Hierfür wurde eine temporäre Datei mit den Konzentrationsdaten erzeugt. *Reactmine* hingegen konnte die **Timeseries** direkt über die Kommandozeile übergeben.

Bei *Coupled SINDy* waren keine relevanten Hyperparameter zu berücksichtigen. Für *Evolving SINDy* wurde jedoch die maximale Anzahl der Reaktionen als einstellbarer Parameter definiert. Im *Reactmine*-Plugin konnten zudem die maximale Baumtiefe und die Anzahl der Kandidaten angepasst werden.

Das durchgeführte Experiment ist in Abbildung 4.4 dargestellt. Hierbei wurde ein simples *Predator-Prey*-Modell bis zum Zeitpunkt $t = 1.0$ simuliert, wobei 1000 Beobachtungen erfasst wurden. Zudem wurden mehrere Durchläufe für *Evolving SINDy* und *Reactmine* mit unterschiedlichen Hyperparametereinstellungen verglichen.

4.4.1. Resultate

Aus den Ergebnissen in Tabelle 4.1 ist ersichtlich, dass lediglich *Coupled SINDy* und *Evolving SINDy* mit fünf Reaktionstermen die Daten adäquat modellieren. Bei den anderen Methoden ist anzunehmen, dass ungeeignete Hyperparameter gewählt wurden. Diese Schlussfolgerung wird durch die Zeit-Konzentrationsdiagramme in Anhang C weiter gestützt.

Methode	**MSE**	**R^2**	**Reaktionsverhältnis**
Coupled SINDy	21.25	0.99999	3.00
Evolving SINDy 1	17.24	0.99999	1.67
Evolving SINDy 2	8619542.46	-2.40	0.67
Reactmine 1	1484627.70	0.37	1.33
Reactmine 2	1915425.50	0.19	1.00

Tabelle 4.1.: Ergebnisse des Experiments. Overlapping Reactions war immer 0

5. Auswertung und zukünftige Entwicklung

Wie bereits vor Beginn der Literaturrecherche vermutet, konnte bestätigt werden, dass ein Mangel an vergleichenden Studien zu Methoden der kombinierten Struktur- und Parameterinferenz besteht. Selbst in relativ spezialisierten Bereichen, wie der Anwendung von SINDy zur Inferenz, wurden innerhalb weniger Jahre parallele, nahezu identische Ansätze entwickelt. Dies deutet darauf hin, dass Wissenschaftlern umfassende Literaturübersichten fehlen. Obwohl die Literaturrecherche in dieser Arbeit mit einem sehr spezifischen Fokus durchgeführt wurde, konnten dennoch zahlreiche Methoden identifiziert werden, die eine genauere Betrachtung verdienen.

Das *Benchmarking Tool* konnte prototypisch entsprechend den Anforderungen implementiert werden. Es ermöglicht die wiederholbare Durchführung von Experimenten mit einer Vielzahl an Methoden. Aufgrund seines hohen Maßes an Flexibilität lässt es sich problemlos an verschiedene Inferenzansätze anpassen. Dadurch ist es in Zukunft auch möglich, Methoden zu testen, die mit anderen Eingabeformaten als direkten Konzentrationsverläufen arbeiten, wie beispielsweise in Klimovskaia et al., 2016.

Obwohl bereits einige Metriken prototypisch implementiert wurden, fehlen derzeit noch wichtige Vergleichsmerkmale. Beispielsweise behaupten viele Methoden, darunter Ji und Deng, 2021, eine besondere Robustheit gegenüber Rauschen aufzuweisen. Diese Eigenschaft wurde bisher nicht systematisch untersucht. Zudem wäre es von Interesse zu analysieren, wie sich die Methoden mit zunehmender Größe der Eingabedaten verhalten – sowohl hinsichtlich ihrer Genauigkeit als auch der erforderlichen Rechenzeit.

Der nächste Schritt für das *Benchmarking Tool* wäre die Durchführung eines groß angelegten Vergleichs der populärsten Methoden zur Struktur- und Parameterinferenz.

A. Auswahl an Suchtermen aus der Literatursuche

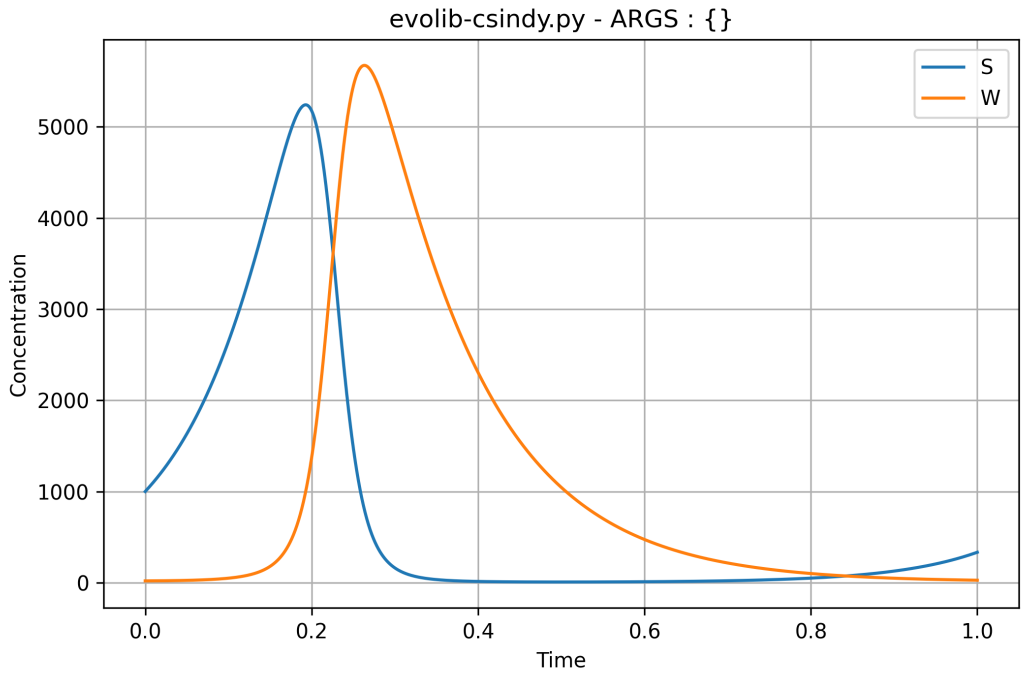
“Reaction Network Synthesis” OR “Reaction Networks Synthesis”
“Reaction Network Inference” OR “Reaction Networks Inference”
“Reaction Network Reverse Engineering” OR “Reaction Networks Engineering”
“Reaction Network Discovery In Silico” OR “Reaction Networks Discovery In Silico”
“Inverse Modeling Reaction Networks”
“Structural Calibration Reaction Network” OR “Structural Calibration Reaction Networks”
“Symbolic Regression Reaction Networks” OR ““Symbolic Regression Reaction Network”
“Data-Driven Modeling Reaction Networks” OR “Data-Driven Modeling Reaction Network”
Reaction Network Inference OR Reaction Networks Inference
Reaction Network Reverse Engineering OR Reaction Networks Engineering
Reaction Network Discovery In Silico OR “Reaction Networks Discovery In Silico
Inverse Modeling Reaction Networks
Identification "Reaction Networks" OR Identification "Reaction Network”
Structural Calibration Reaction Network OR Structural Calibration Reaction Networks
Genetic Programming “Reaction Network” OR Genetic Programming “Reaction Networks”
Symbolic Regression Reaction Networks OR Symbolic Regression Reaction Network
Data-Driven Modeling Reaction Networks OR Data-Driven Modeling Reaction Network
“Chemical Kinetic Model Synthesis” OR “Chemical Kinetic Models Synthesis”
“Chemical Kinetic Model Inference” OR “Chemical Kinetic Models Inference”
“Chemical Kinetic Model Reverse Engineering” OR “Chemical Kinetic Models Engineering”
“Inverse Modeling Chemical Kinetic Models”
“Structural Calibration Chemical Kinetic Model” OR “Structural Calibration Chemical Kinetic Models”
“Genetic Programming Chemical Kinetic Model” OR “Genetic Programming Chemical Kinetic Models”
“Symbolic Regression Chemical Kinetic Models” OR “Symbolic Regression Chemical Kinetic Model”

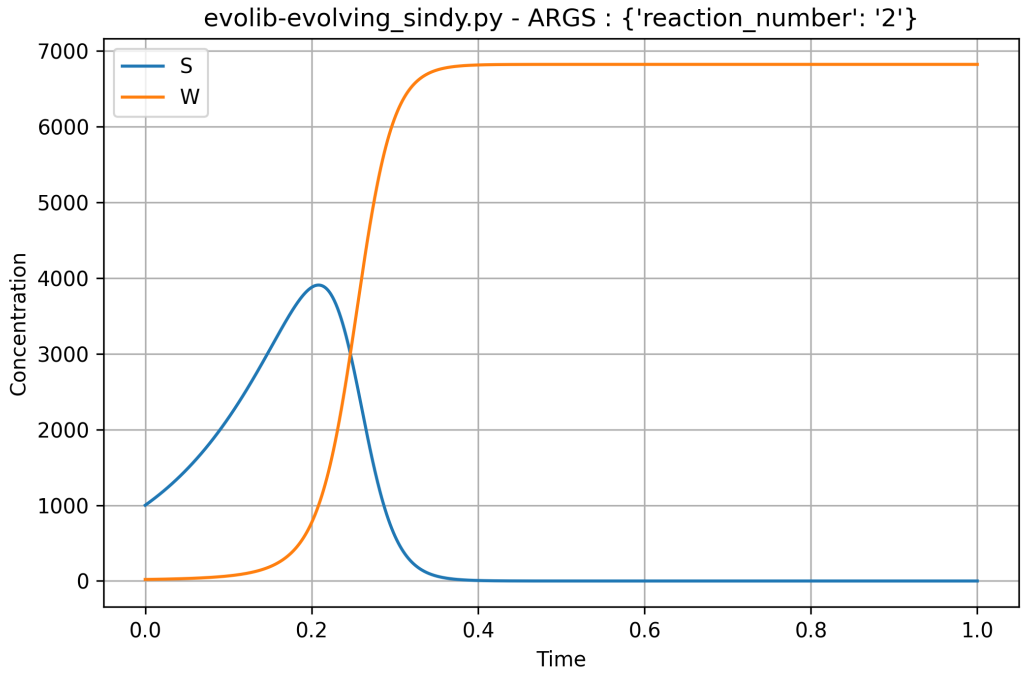
B. Coupled SINDy Plugin

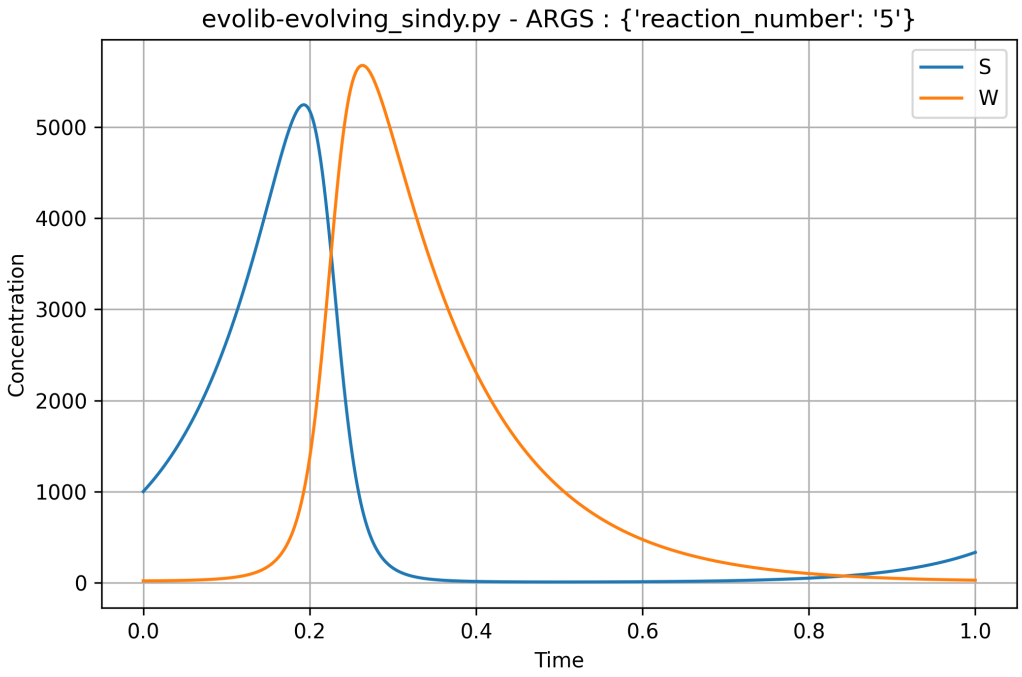
```python
from benchmarker import BenchmarkPlugin, ReactionModel
from common import util
from subprocess import CompletedProcess
import os


TOOL_FOLDER = 'csindy-library-evolution'
class EvolibCSindy(BenchmarkPlugin):
    @property
    def info(self):
        return BenchmarkPlugin.Info(method       = "CSINDy",
                                    tool_name    = "Evolib",
                                    tool_version = "1.0.0")

    @staticmethod
    def I_transform_input(input) -> str:
        return (input.file, input.species)
    @staticmethod
    def parse_reaction(reaction_str : str):
        reactants_str, rest = reaction_str.split("->")
        products_str, rate_str = rest.split("@")
        rate = float(rate_str.strip().replace("[Hz];", ""))

        def parse_side(side_str):
            side = []
            for part in side_str.split("+"):
                part = part.strip()
                if part:
                    if part[0].isdigit():
                        stoichiometry = ""
                        species = ""
                        for char in part:
                            if char.isdigit():
                                stoichiometry += char
                            else:
                                species += char
                        side.append(ReactionModel.Reactant(species.
                            strip(), int(stoichiometry.strip())))
                    else:
                        side.append(ReactionModel.Reactant(part, 1))
            return side

        reactants = parse_side(reactants_str)
        products = parse_side(products_str)
        return ReactionModel.Reaction(reactants, products, rate)
```

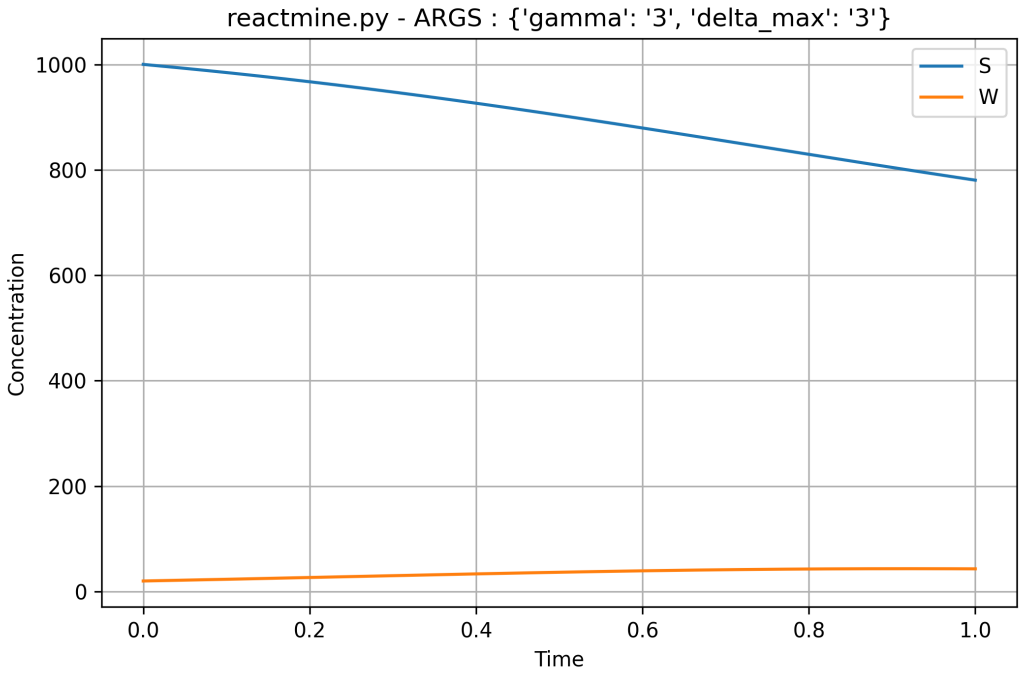
```
45
46      @staticmethod
47      def I_transform_output(output) -> str:
48          model_string = output[1]
49          model = ReactionModel(output[0])
50          for reaction_str in model_string.split("\n"):
51              if reaction_str.strip():
52                  reaction = EvolibCSindy.parse_reaction(reaction_str)
53                  model.reactions.append(reaction)
54          return model
55
56      def I_execute(self, transformed_input):
57          species_names = ",".join(transformed_input[1])
58          tooloutput = util.run_in_venv(f"evolib.sindy.sindy_runner␣{
                species_names}␣{transformed_input[0]}", flag="-m",
                venv_path=self.venv_path)
59          if tooloutput.stderr: print(tooloutput.stderr)
60          return (transformed_input[1], tooloutput.stdout)
61      def I_setup_env(self):
62          self.original_dir = os.getcwd()
63          util.autoset_workdir(TOOL_FOLDER)
64          self.venv_path = os.path.join(os.getcwd(), '.venv')
65          #os.chdir(os.path.join(os.getcwd(), 'src'))
66          os.chdir(os.path.relpath("./src"))
67          return True
68      def I_close_env(self):
69          os.chdir(self.original_dir)
70          return True
```

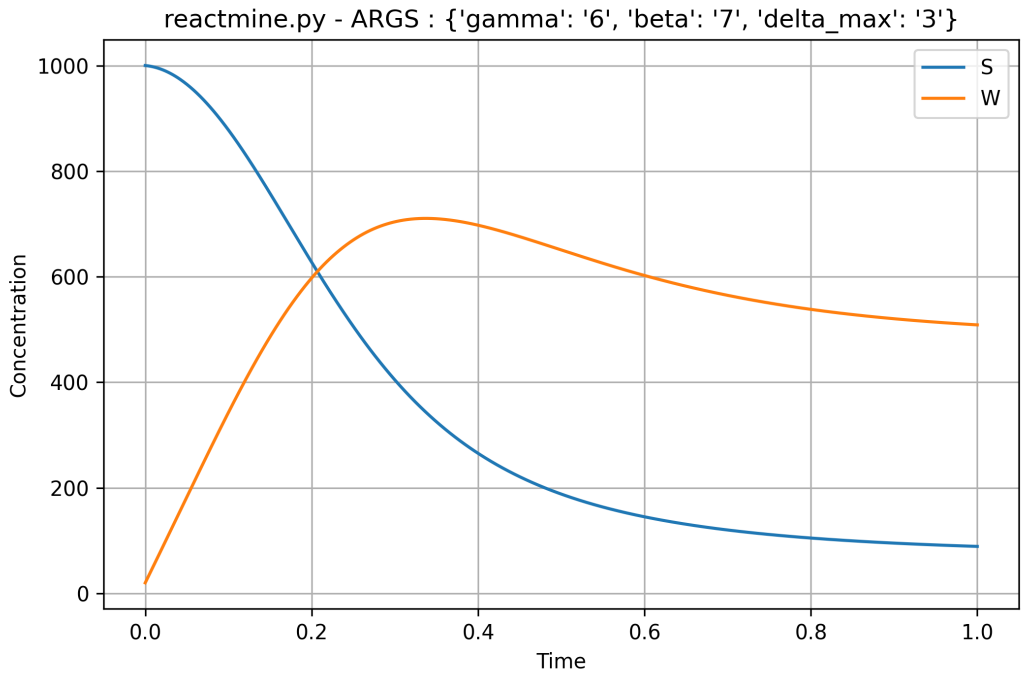
C. Fallstudie Zeit-Konzentrationsgraphen


evolib-csindy.py - ARGS : {}
S
W
5000
4000
3000
2000
1000
0
Concentration
0.0
0.2
0.4
0.6
0.8
1.0
Time


evolib-evolving_sindy.py - ARGS : {'reaction_number': '2'}
S
W
7000
6000
5000
4000
3000
2000
1000
0
Concentration
0.0
0.2
0.4
0.6
0.8
1.0
Time


evolib-evolving_sindy.py - ARGS : {'reaction_number': '5'}
S
W
Concentration
0
1000
2000
3000
4000
5000
0.0
0.2
0.4
0.6
0.8
1.0
Time


reactmine.py - ARGS : {'gamma': '3', 'delta_max': '3'}
S
W
Concentration
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0
Time


reactmine.py - ARGS : {'gamma': '6', 'beta': '7', 'delta_max': '3'}
S
W
1000
800
600
400
200
0
Concentration
0.0
0.2
0.4
0.6
0.8
1.0
Time

Literatur

Abramovitch, R., Tavor, E., Jacob-Hirsch, J., Zeira, E., Amariglio, N., Pappo, O., Rechavi, G., Galun, E., & Honigman, A. (2004). A Pivotal Role of Cyclic AMP-Responsive Element Binding Protein in Tumor Progression. *Cancer Research*, *64*(4), 1338–1346. https://doi.org/10.1158/0008-5472.CAN-03-2089

Bhatt, N., Jayawardhana, B., & Plaza, S. S.-E. (2023). SINDy-CRN: Sparse Identification of Chemical Reaction Networks from Data [ISSN: 2576-2370]. *2023 62nd IEEE Conference on Decision and Control (CDC)*, 3512–3518. https://doi.org/10.1109/CDC49753.2023.10384032

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, *113*(15), 3932–3937. https://doi.org/10.1073/pnas.1517384113

Burrage, P. M., Weerasinghe, H. N., & Burrage, K. (2024). Using a library of chemical reactions to fit systems of ordinary differential equations to agent-based models: a machine learning approach. *Numerical Algorithms*, *96*(3), 1063–1077. https://doi.org/10.1007/s11075-023-01737-0

Chuang, H.-Y., Hofree, M., & Ideker, T. (2010). A Decade of Systems Biology. *Annual Review of Cell and Developmental Biology*, *26*, 721–744. https://doi.org/10.1146/annurev-cellbio-100109-104122

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, *81*(25), 2340–2361. https://doi.org/10.1021/j100540a008

Gratie, D.-E., Iancu, B., & Petre, I. (2013). ODE Analysis of Biological Systems [Series Title: Lecture Notes in Computer Science]. In M. Bernardo, E. De Vink, A. Di Pierro & H. Wiklicky (Hrsg.), *Formal Methods for Dynamical Systems* (S. 29–62). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-38874-3_2

Guo, P. (2021). Ten million users and ten years later: Python tutor's design guidelines for building scalable and sustainable research software in academia. *The 34th Annual ACM Symposium on User Interface Software and Technology*, 1235–1251.

Higham, D. J. (2008). Modeling and Simulating Chemical Reactions. *SIAM Review*, *50*(2), 347–368. https://doi.org/10.1137/060666457

Hoffmann, M., Fröhner, C., & Noé, F. (2019). Reactive SINDy: Discovering governing reactions from concentration data. *The Journal of Chemical Physics*, *150*(2), 025101. https://doi.org/10.1063/1.5066099

International Union of Pure and Applied Chemistry. (2025). *Compendium of Chemical Terminology, 5th ed.* [Online version 5.0.0, 2025. Available at https://goldbook.iupac.org/]. IUPAC.

Ji, W., & Deng, S. (2021). Autonomous Discovery of Unknown Reaction Pathways from Data by Chemical Reaction Neural Network. *The Journal of Physical Chemistry A*, *125*(4), 1082–1092. https://doi.org/10.1021/acs.jpca.0c09316

Jiang, R., Singh, P., Wrede, F., Hellander, A., & Petzold, L. (2022). Identification of dynamic mass-action biochemical reaction networks using sparse Bayesian methods [Publisher: Public Library of Science]. *PLOS Computational Biology*, *18*(1), e1009830. https://doi.org/10.1371/journal.pcbi.1009830

Klimovskaia, A., Ganscha, S., & Claassen, M. (2016). Sparse Regression Based Structure Learning of Stochastic Reaction Networks from Single Cell Snapshot Time Series (D. A. Beard, Hrsg.). *PLOS Computational Biology*, *12*(12), e1005234. https://doi.org/10.1371/journal.pcbi.1005234

Kreikemeyer, J. N., Burrage, K., & Uhrmacher, A. M. (2024). Discovering Biochemical Reaction Models by Evolving Libraries. In R. Gori, P. Milazzo & M. Tribastone (Hrsg.), *Computational Methods in Systems Biology* (S. 117–136). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-71671-3_10

Mangan, N. M., Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Inferring Biological Networks by Sparse Identification of Nonlinear Dynamics [Conference Name: IEEE Transactions on Molecular, Biological, and Multi-Scale Communications]. *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, *2*(1), 52–63. https://doi.org/10.1109/TMBMC.2016.2633265

Martinelli, J., Grignard, J., Soliman, S., Ballesta, A., & Fages, F. (2023). Reactmine: a statistical search algorithm for inferring chemical reactions from time series data [arXiv:2209.03185 [q-bio, stat]]. https://doi.org/10.48550/arXiv.2209.03185

Mayer, C. (2022). *The art of clean code: best practices to eliminate complexity and simplify your life*. No Starch Press.

Nieves, E., Dandekar, R., & Rackauckas, C. (2024). Uncertainty quantified discovery of chemical reaction systems via Bayesian scientific machine learning. *Frontiers in Systems Biology*, *4*. https://doi.org/10.3389/fsysb.2024.1338518

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*, *372*. https://doi.org/10.1136/bmj.n71

Rausanu, S., Grosan, C., Wu, Z., Parvu, O., Stoica, R., & Gilbert, D. (2015). Computational models for inferring biochemical networks. *Neural Computing and Applications*, *26*(2), 299–311. https://doi.org/10.1007/s00521-014-1617-x

Roth, V. (2004). The generalized LASSO [Conference Name: IEEE Transactions on Neural Networks]. *IEEE Transactions on Neural Networks*, *15*(1), 16–28. https://doi.org/10.1109/TNN.2003.809398

Sandelowski, M., Voils, C. I., & Barroso, J. (2006). Defining and designing mixed research synthesis studies. *Research in the schools: a nationally refereed journal sponsored by the Mid-South Educational Research Association and the University of Alabama*, *13*(1), 29.

Sandelowski, M., Voils, C. I., Leeman, J., & Crandell, J. L. (2012). Mapping the mixed methods–mixed research synthesis terrain. *Journal of mixed methods research*, *6*(4), 317–331.

Santosa, F., & Weitz, B. (2011). An inverse problem in reaction kinetics. *Journal of Mathematical Chemistry*, *49*(8), 1507–1520. https://doi.org/10.1007/s10910-011-9835-2

Taylor, C. J., Booth, M., Manson, J. A., Willis, M. J., Clemens, G., Taylor, B. A., Chamberlain, T. W., & Bourne, R. A. (2021). Rapid, automated determination of reaction models and kinetic parameters. *Chemical Engineering Journal*, *413*, 127017. https://doi.org/10.1016/j.cej.2020.127017

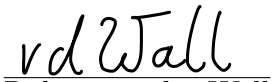
Templier, M., & Paré, G. (2015). A Framework for Guiding and Evaluating Literature Reviews. *Communications of the Association for Information Systems*, *37*(1). https://doi.org/10.17705/1CAIS.03706

Unsleber, J. P., & Reiher, M. (2020). The Exploration of Chemical Reaction Networks. *Annual Review of Physical Chemistry*, *71*(Volume 71, 2020), 121–142. https://doi.org/https://doi.org/10.1146/annurev-physchem-071119-040123

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt und ohne fremde Hilfe verfasst habe. Dazu habe ich keine außer den von mir angegebenen Hilfsmitteln und Quellen verwendet und die den benutzten Werken inhaltlich und wörtlich entnommenen Stellen habe ich als solche kenntlich gemacht. Ich versichere, dass die eingereichte elektronische Fassung mit den gedruckten Exemplaren übereinstimmt.

Rostock, den 21.3.2025



Robin van der Wall