Technische Universität Wien

Vienna University of Technology

Diplomarbeit

Analyse der Kettenbruch-Regression und dessen Anwendung in der Werkstoffmodellierung

zur Erlangung des akademischen Grades

Diplom-Ingenieur

ausgeführt am

Austrian Institute of Technology
Leichtmetallkompetenzzentrum Ranshofen

eingereicht an der

Fakultät für Mathematik der Technischen Universität Wien

im Rahmen des Studiums

Technische Mathematik

unter der Anleitung von

FH-Prof. DI Dr. Gabriel Kronberger
und
Dr. mont. Carina Schlögl

eingereicht durch

Alois Christian Ott, B.Sc.

Ranshofen, am 23. November 2022

Zusammenfassung

Die Continued Fraction Regression (kurz CFR) verwendet analytische Kettenbrüche zur Modellierung physikalischer Daten. In der Diplomarbeit wird untersucht, wie gut sich dieser Algorithmus für die Heißkompressionsdaten einer AA6082 Legierung eignet. In der künstlichen Intelligenz, spezieller dem maschinellen Lernen, ist die symbolische Regression dem Teilgebiet der genetischen Programmierung zuzuordnen. Nach der Behandlung des Grundkonzeptes wird der CFR-Algorithmus damit in Verbindung gebracht und ausführlich vorgestellt.
Der Erfolg des Verfahrens beruht auf der sogenannten Padé-Approximation. So lässt sich eine holomorphe Funktion über rationale Funktionen oft besser beschreiben als über eine klassische Reihenentwicklung. Wann sich diese rationalen Funktionen dann als verallgemeinerte Kettenbrüche darstellen lassen, ist dabei der springende Punkt. Der mathematische Hintergrund ist hierfür Schritt für Schritt abgebildet, bis man schließlich ein Kriterium von Baker erhält, welche analytischen Funktionen in verallgemeinerte Kettenbrüche entwickelbar sind und dessen Kontinuanten eine Stufenfolge in der Padé-Tabelle bilden. Auch meromorphe Funktionen lassen sich in Kettenbrüche entwickeln, wobei man hier keine verallgemeinerten Kettenbrüche mehr erhält. Baker liefert weiters ein Theorem, wann eine Folge von Padé-Approximanten einer analytischen Funktion auch außerhalb des Konvergenzradius konvergiert.
Anschließend folgt in einem empirischen Teil, ein Test über das im Algorithmus verwendete lokale Suchverfahren (Downhill Simplex). Es stellt sich dabei heraus, dass dieses für Kettenbrüche von geringer Tiefe einen hohen Einfluss hat. Hier und für folgende Implementierungen des CFR-Algorithmus kommt die Softwareumgebung HeuristicLab zum Einsatz.
Im letzten Kapitel werden die Stauchversuchsdaten bereinigt, skaliert, gesampled und in Trainings- und Testdaten aufgeteilt. Die Trainingsdaten werden anschließend für die Parameteroptimierung und das Erlernen von Modellen verwendet. Der originale Algorithmus wird mit einer eigenen Implementierung der Kettenbruch-Regression verglichen. Bei Modellen mit hoher Kettenbruchtiefe zeigt der originale Algorithmus ein schlechtes Konvergenzverhalten und es tritt ein Underfitting ein. Erst durch eine Anpassung der lokalen Suche lässt sich ein Underfitting ausschließen. So wird mit Hilfe dieser Modifikation in der eigenen Implementierung erreicht, dass in den ersten Generationen Kettenbrüche mit niedriger Tiefe erzeugt werden. Erst im weiteren Verlauf werden durch Mutation und Rekombination Kettenbrüche höherer und schließlich auch maximaler Tiefe gebildet. Bei einem Vergleich mit anderen Modellen (antrainierbar im HeuristicLab) stellt sich heraus, dass die Kettenbruch-Regression zwar nicht so genaue Vorhersagen treffen kann, dafür aber sehr kompakte und leicht interpretierbare Modelle liefert.

Abstract

The Continued Fraction Regression (CFR for short), uses analytical continued fractions to model physical data. The thesis investigates how well the algorithm can be applied to the hot compression data of an AA6082 alloy. In artificial intelligence, especially in machine learning, symbolic regression belongs to the subfield of genetic programming. After dealing with the basic concept, the CFR algorithm is related to it and presented in detail.
The success of the method is based on the so-called Padé approximation. Thus, a holomorphic function can often be better described via rational functions than via a classical series expansion. The conditions under which these rational functions can be represented as generalized continued fractions are of interest. The mathematical background for this is illustrated step by step until a criterion from Baker is shown. It reveals the conditions when analytic functions can be expanded into generalized continued fractions whose continuants form a staircase sequence in the Padé table. Meromorphic functions can also be expanded into continued fractions. In general, these are no longer generalized continued fractions. Baker also provides a theorem for when sequences of Padé approximants of an analytic function converge outside the radius of convergence.
This is followed by an empirical part that tests the local search method (downhill simplex) used in the algorithm. It turns out that this has a high influence on continued fractions of low depth. Here and for the following implementations of the CFR algorithm, the software environment HeuristicLab is used.
In the last chapter, the hot compression test data is cleaned, scaled, sampled and divided into training and test data. The training data is then used for parameter optimization and model learning. The original algorithm is compared with a custom implementation of the continued fraction regression. For models with a higher depth of continued fractions, the original algorithm shows poor convergence behaviour and underfitting occurs. An adjustment of the local search method can exclude the latter. By using this modification of the own implementation, continued fractions with low depth are generated in the first generations. In the further course continued fractions of higher and finally also maximum depth are formed through mutation and recombination. In a comparison with other models (trainable in HeuristicLab), it turns out that the continued fraction regression cannot make such precise predictions, but it provides very compact and easily interpretable models.

Inhaltsverzeichnis

1 Einleitung

1.1 Forschungsthema

In einer Veröffentlichung wurde ein Algorithmus (Continued Fraction Regression, CFR) für multivariate Regression beschrieben, der auf dem Prinzip der Padé-Approximation beruht [Sun und Moscato, 2019], [Moscato et al., 2021]. So soll im Allgemeinen ein Padé-Approximant der Ordnung n und m (Grad des Zähler- und Nennerpolynoms) einer Funktion eine bessere Annäherung liefern als eine Taylorreihenentwicklung der selben Ordnung $n + m$ (die nötige Differenzierbarkeit sei vorausgesetzt). Selbst außerhalb des Konvergenzradius der Taylorreihe soll der Padé-Approximant gute Dienste leisten.

Ein Padé-Approximant ist eine rationale Funktion und lässt sich als Kettenbruch darstellen. Die Glieder können unterschiedlich konstruiert werden und sich je nach betrachteter Funktion besser eignen. So verwendet der CFR-Algorithmus eine generalisierte Form der regulären Kettenbruchdarstellung.

Je höher die vorgegebene Ordnung eines Padé-Approximanten ist, desto mehr Möglichkeiten gibt es diesen zu bilden (Zähler + Nennergrad). Laut [Baker et al., 1996] kann durch die Darstellung des Modells als Kettenbruch eine bessere Inter- und Extrapolation erreicht werden. Der CFR-Algorithmus verwendet einen evolutionären Ansatz um durch geeignete Mutations- und Kreuzungsoperatoren eine passende Modellstruktur zu identifizieren (welche Variablen werden in welcher Ebene des Kettenbruchs verwendet). Die numerischen Parameter werden lokal optimiert (Nelder-Mead-Methode).

Der CFR-Algorithmus wurde mit einer umfangreichen Menge von Modellen getestet und mit gängigen Algorithmen verglichen [Moscato et al., 2020]. Der Vergleich zeigt eine bessere Laufzeit des CFR-Algorithmus und einen geringeren Prognosefehler (mean squared error - MSE). Die so erhaltenen Modelle - analytische Kettenbrüche - sind noch dazu sehr kompakt.

1.2 Anwendungsbezug

In der Leichtmetallforschung spielen Werkstoffeigenschaften und dessen Beschreibung eine bedeutende Rolle. So sollen für das Leichtmetallkompetenzzentrum Ranshofen (LKR) die Spannungs-Dehnungskurven einer AA6082-Legierung modelliert werden. Hierbei stehen die Messergebnisse eines Abschreck- und Umformdilatometers DIL 805A/D von TA Instruments zur Verfügung.

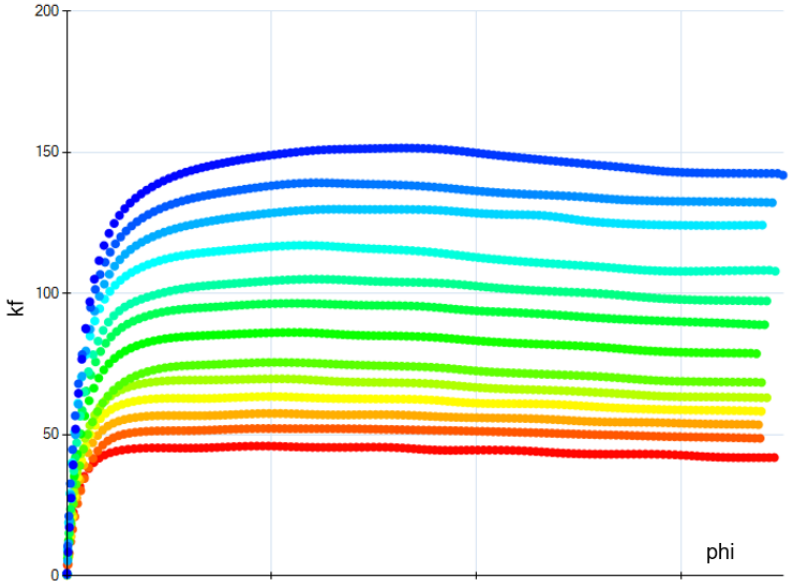


Abbildung 1: Spannungs-Dehnungs-Kurven

Ein Teil der Messdaten ist in Abbildung 1 dargestellt. Diese stammen von Stauchversuchen, bei denen die Proben zuvor auf 200° (blau) bis zu 500°(rot) erhitzt wurden. Aufgetragen ist die Zugfestigkeit (in MPa) über der Stauchung (in 20% Schritten der ursprünglichen Probenlänge). Es soll eine Funktion in Form eines kurzen mathematischen Ausdrucks ermittelt werden, welche die Messdaten möglichst genau beschreibt. Der CFR-Algorithmus scheint dafür besonders geeignet, da die Messdaten darauf hindeuten, dass diese mit einer rationalen Funktion gut beschrieben werden können (ev. Pol in 0). In Vorarbeiten [Kabliman et al., 2019] wurden verschiedene Regressionsmethoden verwendet und gute Resultate erzielt. Allerdings zeigten diese Methoden schlechte Extrapolationseigenschaften.

1.3 Zielsetzung und Erkenntnisinteresse

Ziel dieser Arbeit ist es herauszufinden, ob sich Kettenbrüche als Modellstruktur zur Prognosemodellierung von Spannungs-Dehnungskurven eignen. Speziell soll hier auch darauf eingegangen werden, weshalb eine Nelder-Mead-Methode und die genetischen Operatoren im CFR-Algorithmus eine gute Konvergenzrate liefern. Eine eigene Implementierung des Algorithmus soll einen empirischen Teil der Diplomarbeit ermöglichen. Die Kollegen im LKR Ranshofen sind an einem Modell mit möglichst niedrigem MSE und mit einer geringen Anzahl an Parametern interessiert.

1.4 Forschungskonzept

Die zugrundeliegende These lautet:

Der CFR-Algorithmus besitzt ein gutes Konvergenzverhalten bezüglich des Mean Squared Errors, da er auf der Variation von Padé-Approximanten beruht.
Folgende Fragen sollen dabei beantwortet werden:

- Wie funktioniert der CFR-Algorithmus?
- Was sind Padé-Approximanten einer holomorphen Funktion?
- Weshalb ist eine Variation von Padé-Approximanten einer meromorphen Funktion (bis auf endlich viele Polstellen als Potenzreihe darstellbar) für Inter- und Extrapolation besonders gut geeignet? (Main-Part I)
- Warum sind analytische Kettenbrüche geeignet, um die Padé-Approximanten einer meromorphen Funktion darzustellen? (Main-Part II)
- Weshalb kann man die Vorgehensweise des CFR-Algorithmus als Variation und Rekombination von Padé-Approximanten sehen?
- Wie schneidet der CFR-Algorithmus bei der Anwendung auf Spannungs-Dehnungs-Kurven ab?
- Wo liegen die Vorteile (und Nachteile) des CFR-Algorithmus im Vergleich zu anderen Algorithmen, die bisher für die Modellierung der Spannungs-Dehnungs-Kurven verwendet wurden?

Um den Main-Part beantworten zu können, kann man sich großteils in [Baker et al., 1996] informieren. Dies ist ein umfassendes Werk über Padé-Approximanten und kann angefangen von den Grundlagen bis hin zur Anwendung weiterhelfen.

1.5 Eigene Motivation

Es gab bisher keine öffentlich zugängliche Implementierung des Algorithmus. Deshalb wurde hierfür eine eigene Implementierung geschrieben, die als ein Ergebnis dieser Masterarbeit auch veröffentlicht wurde. Diese soll die Resultate von [Moscato et al., 2020] reproduzierbar machen.

2 Maschinelles Lernen in der Werkstoffmodellierung

Hier sollen Disziplinen, die mit dem maschinellen Lernen verwandt sind, vorgestellt werden. Dies wird im Weiteren das Grundverständnis und die Zuweisung des CFR-Algorithmus vereinfachen.

2.1 Künstliche Intelligenz

Was ist eigentlich Künstliche Intelligenz (KI) bzw. gibt es eine allgemein anerkannte Definition dafür? Diese Frage ist leider nicht so einfach zu beantworten. Einigkeit gibt es in Fachkreisen nur darüber, dass hier Uneinigkeit herrscht. Ins Leben gerufen wurde Artificial Intelligence (AI), als ein akademisches Fachgebiet, in einer Konferenz 1955 am Darthmouth College [McCarthy et al., 1955]. Dies war ein von der Rockefeller-Stiftung geförderter Workshop unter der Leitung von John McCarthy, dem Erfinder der Programmiersprache LISP. Für seine folgenden Beiträge in der KI erhielt er auch den Turing Award 1971 und einen Kyoto Preis 1988 [Borchers, 2011]. Hier fällt der Name des Logikers und Informatikers Alan Mathison Turing, was wohl niemanden wundert. Denn dieser war es, der mit seinem Imitation-Game (heutzutage Turing-Test genannt) zur Feststellung von Intelligenz den Grundstein legte [Turing, 1950]. Bei dem Test soll ein menschlicher Beobachter an einen Computer gesetzt werden und mit zwei Gesprächspartnern (ohne Hör- und Sichtkontakt) kommunizieren. Einer davon ist eine Maschine, der andere ein Mensch. Kann der Beobachter nach einer umfangreichen Befragung nicht zwischen den beiden unterscheiden, so ist der Maschine dasselbe Denkvermögen wie das des Menschen zuzuschreiben. Turing hoffte, dass damit auch Bewusstsein nachgewiesen werden könne, der Philosoph John Searle lehnte dies mit seinem Gedankenexperiment des Chinesischen Zimmers allerdings ab [Searle, 1992]. Alan Turing entwickelte auch eines der ersten Schachprogramme (1953). Vollständig ausgereift und als Meilenstein der KI zu betrachten war dann wohl der Sieg des Schachspiel-Computers Deep Blue (von IBM) über den Weltmeister Garry Kasparov (1997) [Scherk et al., 2017]. Der Gedanke einer künstlichen Intelligenz reicht allerdings noch viel weiter zurück. Man denke nur an den Laplace'schen Dämon, also ein geschlossenes mathematisches Weltgleichungssystem, mit dem sich alle vergangenen und zukünftigen Zustände vorhersagen lassen. Also eine Art „mechanische Maschine", die dann automatisch auch den Menschen und seinen Geist miteinschließt. Ebenso hat sich bereits Pierre-Simon Laplace 1814 damit beschäftigt [Höfling, 1994].
Im Folgenden befindet sich eine moderne Beschreibung von künstlicher Intelligenz:

Künstliche Intelligenz ist eine Wissenschaft und eine Reihe an Computertechnologien, die von der Art und Weise inspiriert sind, wie die Menschen ihr Nervensystem und ihren Körper nutzen, um zu spüren, zu lernen, zu denken und Maßnahmen zu ergreifen – jedoch meist recht unterschiedlich wirken. [Stone et al., 2016]

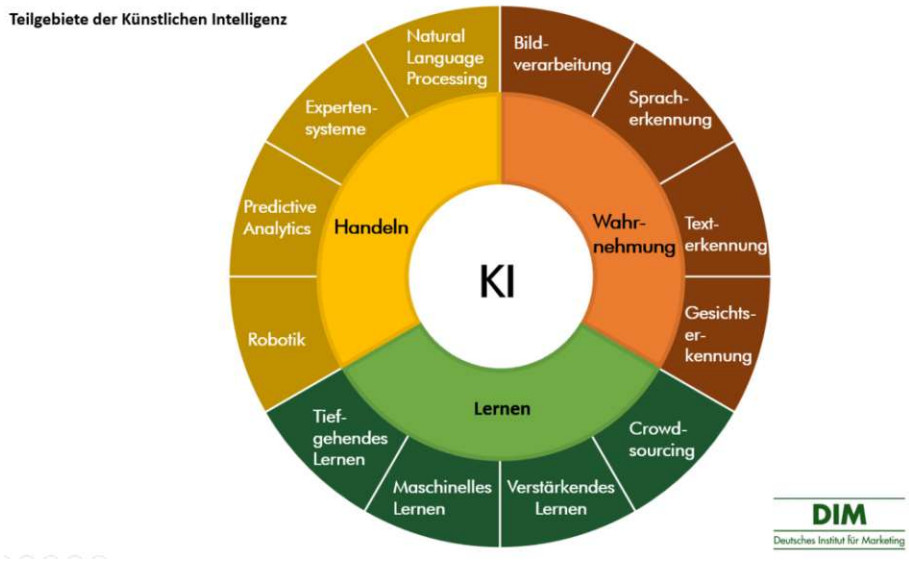


Abbildung 2: Unterteilung der künstlichen Intelligenz in drei Teilgebiete [Bernecker, 2019]

Diese Definition von KI beinhaltet direkt und indirekt die drei Begriffe Wahrnehmung (Spüren), Lernen und Handeln (Maßnahmen ergreifen).

Ebenfalls befindet das Deutsche Institut für Marketing eine Unterteilung in diese drei Teilgebiete als praktisch. Zur Veranschaulichung siehe Abbildung 2. Eine weitere geachtete Darlegung von KI ist die folgende von Nilsson:

Künstliche Intelligenz ist die Aktivität die Maschinen gewidmet wird, um ihnen Intelligenz zu geben und Intelligenz ist die Qualität, die es einem Objekt ermöglicht, in seiner Umgebung angemessen und vorausschauend zu funktionieren. [Nilsson, 2009]

Die künstliche Intelligenz überschneidet sich auch mit vielen anderen Disziplinen (Abbildung 3). Darunter fallen auch die Mathematik und deren Teildisziplin die Statistik. Big Data ist der KI übergeordnet und das maschinelle Lernen fällt zwar vollständig in Big Data, aber nicht (wie man zuerst vermuten würde) in die KI. Dies liegt vor allem daran, dass sich der Drang zur KI bzw. zum maschinellen Lernen erst ergibt, wenn sich gegebene Daten nicht mehr überschauen lassen.

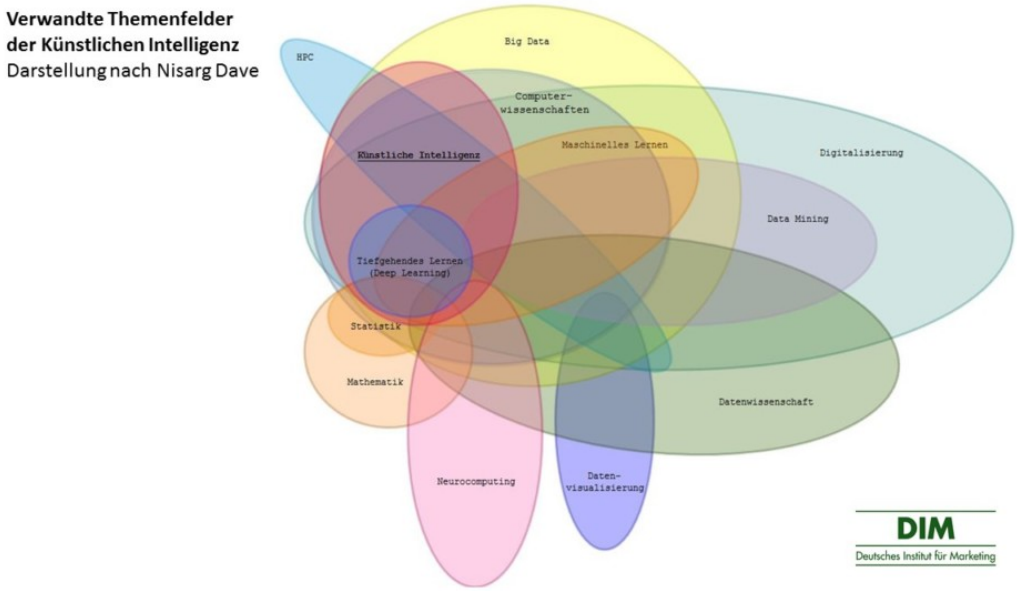


Abbildung 3: Verwandte Themenfelder der KI [Bernecker, 2019]

Diese Arbeit bewegt sich ausschließlich im KI-Teilbereich des Lernens, genauer gesagt mit dem maschinellen Lernen. Was darunter verstanden wird und wie dieses für die Materialmodellierung genutzt werden kann, steht im Weiteren.

2.2 Maschinelles Lernen

Ziel des Maschinellen Lernens (ML) ist es, aus Umweltdaten (Trainingsdaten/ Produktivdaten) Muster und Gesetzmäßigkeiten zu erkennen, um mit Hilfe dessen Vorhersagen treffen und auch ein automatisches Handeln ermöglichen zu können. Das System verbessert sich durch die Qualität seiner Aktionen von selbst. Der Fokus liegt dabei auf der Konstruktion und der Entwicklung von Algorithmen, die dies ermöglichen. Hier werden keine streng statischen Programmanweisungen befolgt, sondern meist statistische Modelle aufgebaut. Über diese sollen dann unbekannte Daten beurteilt werden. Ein Auswendiglernen ist hier durch schlechte Qualität der Vorhersagen für neue noch unbekannte Daten oder Zustände erkennbar. Eine sogenannte Überanpassung hat dann stattgefunden. [Reitmaier, 2015]

Gebrauch findet das maschinelle Lernen vor allem bei Problemen, die sich physikalisch nur schwer beschreiben lassen. Existieren zwar physikalische Modelle, aber keine expliziten Algorithmen zur Lösungsfindung, so kann ein maschineller Ansatz ebenfalls hilfreich sein. Eng verbunden ist ML auch mit der Computerstatistik, die sich ebenfalls mit der Vorhersageerstellung befasst. Desweiteren liefert die mathematische Optimierung eine brauchbare Theorie, Methoden und eine Anwendungsdomäne, die hier oft von Nutzen ist. [Wernick et al., 2010]

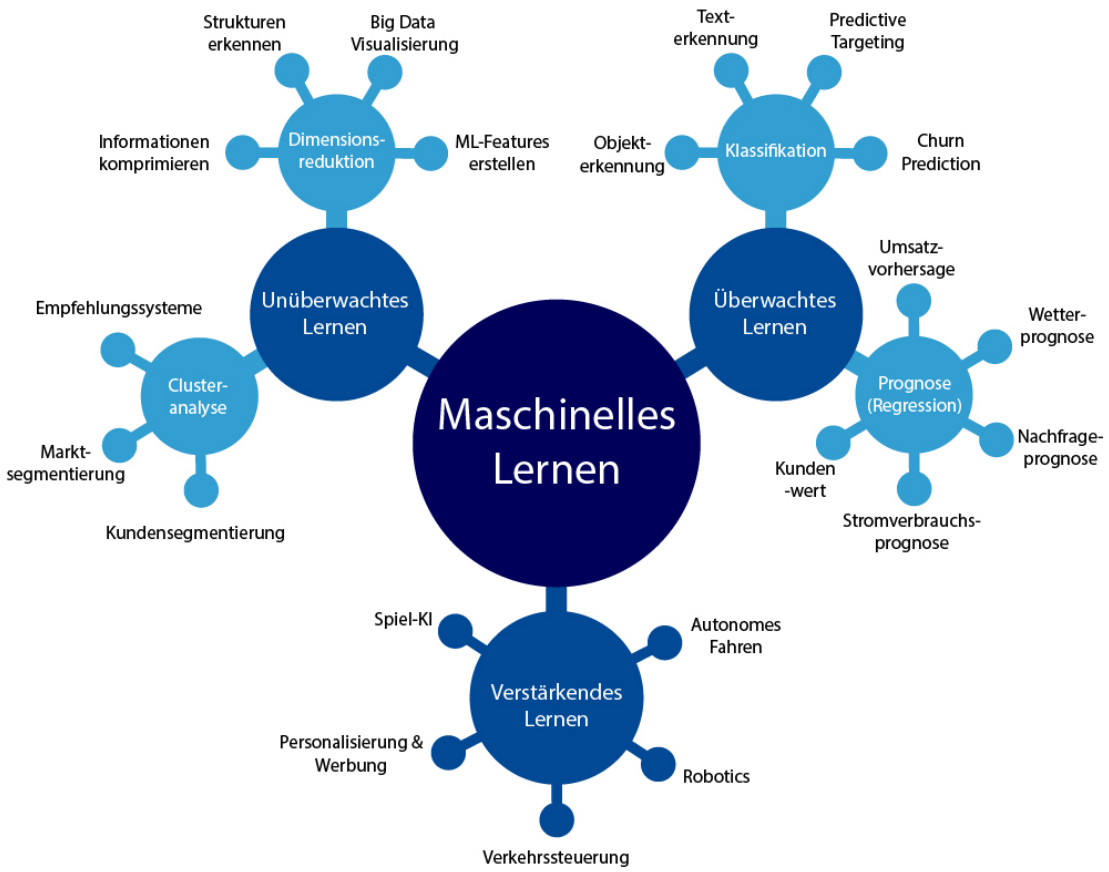


Abbildung 4: Maschinelles Lernen im Überblick: Anwendungsbeispiele nach Arten [Wuttke, 2020]

Aus Abbildung 4 ist ersichtlich, dass die Texterkennung und die Robotik Teildisziplinen des maschinellen Lernens sind. Im Vergleich zu Abbildung 3 fällt aber die Robotik in das Gebiet Handeln, die Texterkennung in die Wahrnehmung und nicht wie vermutet in das Lernen. Das Problem liegt darin, dass sich hier die einzelnen Disziplinen überlappen und nicht scharf voneinander trennen lassen. Deshalb kommt es auch immer auf die Betrachtungsweise des Datenwissenschaftlers an.

2.2.1 Überwachtes-, unüberwachtes- und verstärkendes Lernen

Das maschinelle Lernen wird allgemein in überwachtes-, unüberwachtes- und verstärkendes Lernen unterteilt. Der Unterschied liegt hauptsächlich darin, in welchem Ausmaß das Training stattfindet. Die Kettenbruch-Regression ist Ersterem zuzuweisen.

Überwachtes Lernen (Supervised Learning)

Beim überwachten Lernen gibt es zu jedem Eingabewert einen dazugehörigen Ausgabewert. Dies erlaubt also die Beurteilung eines Eingabewertes mit dem tatsächlichen Ausgabewert zu vergleichen. Bevor nun mit dem Training begonnen werden kann, müssen die Daten in eine Trainings- und in eine Testmenge unterteilt werden, siehe Abbildung 5.

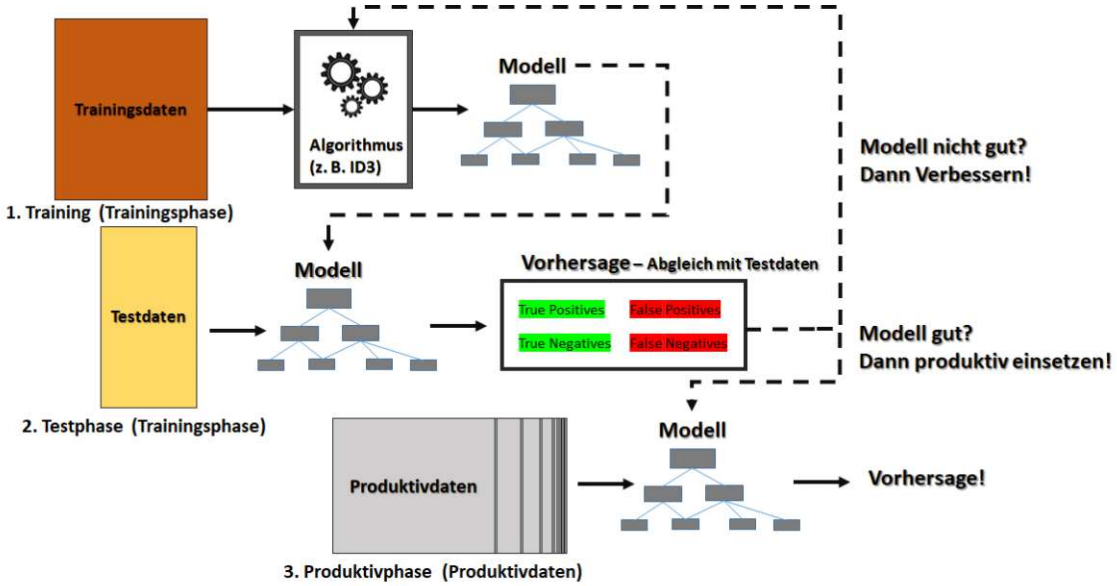


Abbildung 5: Überwachtes vs unüberwachtes maschinelles Lernen [Aunkofer, 2017]

Die Trainingsdaten dürfen anschließend vom Algorithmus verwendet werden, um ein Modell anzulernen. Liefert dieses schließlich genügend richtige/genaue Vorhersagen (dies muss im Voraus festgelegt werden), so wird es einer Testphase übergeben. Weist das Modell auch hier ausreichend Genauigkeit auf und zwar in dem Sinne, dass es ähnliches Verhalten wie mit den Trainingsdaten zeigt, so wird es produktiv eingesetzt. Weichen allerdings die Vorhersagen im Test merkbar ab, so hat eine Überanpassung stattgefunden. Das Modell ist nicht in der Lage zu verallgemeinern und muss in die Trainingsphase zurückkehren. Somit hat die Testphase genau genommen auch zum Training beigetragen. Erst in der Produktivphase hat man keinen Einfluss mehr darauf.

2.3 Evolutionäre Algorithmen

Die Kettenbruch-Regression ist ein evolutionärer Algorithmus und deshalb wird in diesem Unterkapitel die Entstehung und Funktionsweise solcher behandelt. Es gibt eine Reihe an Kriterien auf die bei einer Implementierung zu achten ist und auch dies wird anhand einiger Beispiele verdeutlicht.

2.3.1 Einführung

Bei evolutionären Algorithmen handelt es sich um stochastische, metaheuristische Optimierungsverfahren. Das heißt ein Problem wird zufallsgetrieben und näherungsweise gelöst. Das Finden einer optimalen Lösung ist hier nicht sichergestellt, aber der Vorteil liegt darin, dass solche Verfahren auf sehr vielseitige Problemstellungen angewendet werden können. Wie bereits der Name vermuten lässt, basiert deren Funktionsweise auf dem von Charles Darwin ins Leben gerufenen Prinzip der natürlichen Evolution. Für die Herleitung des Grundkonzeptes evolutionärer Algorithmen dienen drei Hypothesen. [De Jong, 2016]
Die Abstammungshypothese besagt, dass alle Arten an Lebewesen (Individuen) auf nur wenige Urformen zurückzuführen sind (und von diesen abstammen). Es wird gezeigt, dass evolutionäre Algorithmen eine initiale Population bilden und aus dieser weitere und bessere Populationen abgeleitet werden.
Desweiteren folgt die Allmählichkeitshypothese und deren Aussage, wonach sich Evolution nur langsam vollzieht und nicht etwa in Sprüngen. Für folgende Algorithmen bedeutet dies nur kleine Änderungen der Individuen von Generation zu Generation.
Hinzu kommt auch noch die Selektionshypothese. Lebewesen, die besser für das Überleben geeignet sind, zeugen auch mehr Nachkommen. Dieses Prinzip soll auch in der Diplomarbeit verwendet werden und es werden großteils nur Individuen in einer neuen Population aufgenommen, falls sie näher am Ergebnis liegen als ihre Konkurrenten.

Evolutionäre Algorithmen (EA) oder auch bekannt als Evolutionary Computation (EC), ist der Überbegriff vierer Hauptströmungen, die sich ab den 60er Jahren gebildet haben: [Nissen, 1998]

1. Evolutionäre Strategien (ES) - Ingo Rechenberg [Rechenberg, 1973], Hans-Paul Schwefel [Schwefel, 1977]
2. Genetische Algorithmen (GA) - John H. Holland [Holland, 1992]
3. Evolutionäre Programmierung (EP) - Lawrence J. Fogel [Fogel et al., 1966]
4. Genetische Programmierung (GP) - John R. Koza [Koza, 1992]

Zwischen diesen vier Paradigmen wird meist nicht so streng unterschieden. Es sei aber angemerkt, dass bei den genetischen Algorithmen die Struktur der Programme, die man optimieren möchte, fest gewählt wird. Es handelt sich also um eine reine Parameteroptimierungsmethode. Bei der genetischen Programmierung, die als Erweiterung der genetischen Algorithmen zu sehen ist, wird auch über die Struktur optimiert. Die Lösungskandidaten werden als Bäume repräsentiert (Baumstruktur).

2.3.2 Grundkonzept

Das Grundkonzept ist bei allen Paradigmen gleich:

1. Initialisierung: Zu Beginn wird eine festgelegte Anzahl von meist zufälligen Lösungskandidaten/Programmen/Individuen erzeugt. Dies ist die initiale Population und wird auch die erste Generation genannt.
2. Bewertung: Anschließend folgt eine Bewertung der einzelnen Lösungskandidaten auf deren Eignung mit Hilfe einer Fitnessfunktion/Evaluierungsfunktion.
3. Schleife: Wiederhole solange, bis eine Terminierungsbedingung erfüllt ist (das heißt ein Programm ist ausreichend gut oder zu viele Schritte wurden durchgeführt):

(a) Paarungsselektion: Wähle Individuen für Rekombination aus der Population aus (meist mit gutem Fitnesswert)

(b) Rekombination: Kombiniere Individuen miteinander (Eltern) und erzeuge Neue (Kinder)

(c) Mutation: Anschließend kann ein Kind auch noch zufällig verändert werden

(d) Bewertung: Es erfolgt eine erneute Bewertung aller Individuen der Generation

(e) Umweltselektion: Wähle Kandidaten zur Bildung einer neuen Generation aus

Ohne dabei näher auf die Gestalt eines Programmes und der darauf definierten Verfahren einzugehen, ist ein natürlicher Bezug ersichtlich. Für eine graphische Darstellung des Ablaufes siehe auch Abbildung 6:

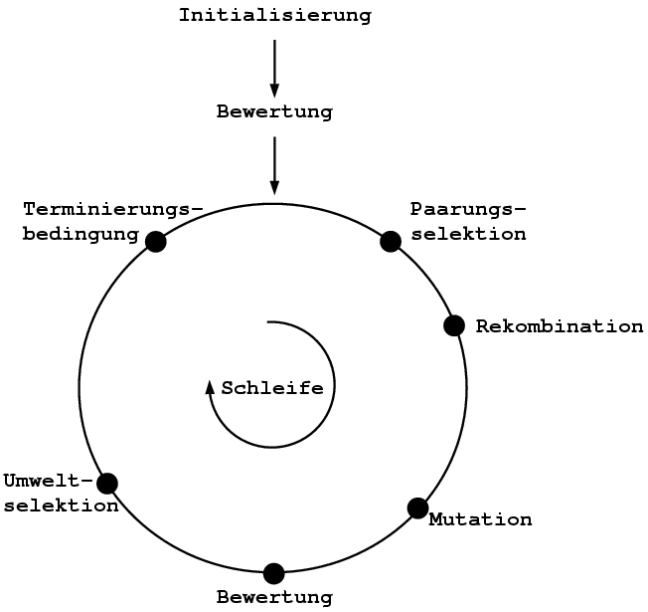


Abbildung 6: Evolutionäre Algorithmen [Rudolph, 2015]

2.3.3 Genetische Programmierung

Hier soll ein Überblick über das vierte Paradigma, nämlich der genetischen Programmierung, gegeben werden. Die zugrundeliegende Theorie dahinter stammt von John R. Koza. Als gute Übersicht dienen im Folgenden [Poli et al., 2008] und auch [Musch, 2004].

Programme/Individuen

Wird in GP von einem Programm gesprochen, so ist dies kein klassischer Befehlssatz einer Programmiersprache. Ein Programm wird hier durch eine weitaus eingeschränktere und domänenspezifischere Sprache formuliert. Am Besten ist es vorerst an einen mathematischen Ausdruck zu denken. Zu dessen Bildung benötigt man zwei Mengen, nämlich eine Menge an Terminalen und eine an Funktionen.
Die Terminalmenge besteht aus:

- Variablen: Diese sind problemabhängig. Als Beispiel können dies bei der Positionierung eines Roboters die Koordinaten $x_1, x_2, \ldots$ und die Zeit t sein oder bei einem Kartenspiel die Variable Trumpf, die immer für eine der vier Spielfarben steht.

- Funktionen ohne Argument: Diese können bei verschiedenen Programmaufrufen unterschiedliche Werte annehmen und sind dennoch von keiner Variablen abhängig. Man denke an den Abstand eines Roboters zu einem Hindernis oder an den Zufallszahlengenerator rand(), der eine zufällige Zahl aus einem Intervall zurückliefert.

- Konstanten: Bereits zu Beginn können benötigte Konstanten wie die Kreiszahl π oder die Gravitationskonstante g festgelegt werden. Ebenfalls dürfen Konstanten auch zufällig initialisiert und im Weiteren durch genetische Operatoren verändert werden.

Die Funktionenmenge hingegen besteht aus Funktionssymbolen mit mehreren Argumenten wie zum Beispiel:

- Arithmetische Operatoren: Als typisches Beispiel können hier die vier Grundrechenarten $+,-,*$ und / angegeben werden. Diese werden meist als zweistellige Funktionssymbole aufgefasst, dürfen aber auch beliebig modifiziert werden.
- Winkelfunktionen: Als Beispiele dienen die bekannten Funktionen Sinus, Kosinus oder auch die Exponentialfunktion.
- Logische Operatoren: Wie das logische AND, OR, $\Longrightarrow$ oder auch $\leq$. Bevorzugt werden solche Symbole bei logischen Schaltungen eingesetzt.
- Schleifen: Auch eine for-Schleife, die mehrere Argumente beinhaltet, kann verwendet werden. Dies ermöglicht zum Beispiel ein Anlernen von kleineren Computerprogrammen.

Die beiden Mengen an Terminalen und Funktionen zusammen werden auch als Primitive Menge bezeichnet. Um eine effiziente GP Implementierung zu ermöglichen, muss bei der Auswahl an primitiven Elementen auf zwei Dinge geachtet werden:

- Abgeschlossenheit: Können alle Rekombinations- und Mutationsoperatoren (müssen auch zu Beginn festgelegt werden) auf alle möglichen Programme angewendet werden ohne Schwierigkeiten zu bekommen? Unter Schwierigkeiten versteht man:
  - Typenkonsistenz: Jede Funktion hat einen gewissen Definitionsbereich. Bekommt die Funktion ein Argument außerhalb dessen zugewiesen, so kann auch kein Wert zurückgeliefert werden. Zum Beispiel kann die Addition reelle Zahlen (double) und ganze Zahlen (integer) addieren, jedoch nicht eine reelle Zahl und TRUE (boolean). Aushilfe kann man sich hier über Konvertierung von TRUE (boolean) auf 1 (integer) (und analog für FALSE und 0) verschaffen - sozusagen eine Erweiterung des Definitionsbereichs.
  - Auswertungssicherheit: Probleme können auch entstehen obwohl der richtige Typ an eine Funktion übergeben wird. Man denke hier an die Logarithmusfunktion und eine negative reelle Zahl oder an eine Division durch 0. Ein Roboter könnte auch bei einer Bewegung mit einem Objekt kollidieren. Hier schaffen geschützte Versionen der Funktionen Abhilfe. Es wird im Voraus abgefragt, ob eine Division durch 0 erfolgt und falls ja einfach 1 zurückgeliefert oder beim Logarithmieren immer der Betrag einer reellen Zahl genommen. Bei einer möglichen Kollision darf sich ein Roboter nicht mehr bewegen. Eine weitere Möglichkeit wäre die Rückgabe von schlechten Fitnesswerten. Könnten Probleme wie ein Überlauf des Wertebereichs entstehen, kann auch eine Ausgabe von Exceptions nötig sein.
- Angemessenheit: Ist man in der Lage in seiner Sprache (erlaubte Kombination der primitiven Elemente) eine (angemessene) Lösung zu formulieren? Hierfür muss ein Vorwissen über das zu behandelnde Problem gegeben sein. Nimmt man als terminale Menge $\{x, 0, 1, 2, \ldots\}$ und als Funktionenmenge $\{+, -, *, /\}$, so ist man zwar in der Lage die Funktion $\exp(x) := \sum_{n=0}^{\infty} \frac{x^n}{n!}$ beliebig zu approximieren, allerdings nicht exakt (über rationale Funktionen) darzustellen. Jedoch kann eine solche Annäherung oft schon ausreichen und es erspart uns "teure Funktionen" in die primitive Menge miteinzubeziehen. Ebenfalls zu beachten ist, dass die Aufnahme von unnötigen Elementen die Performance bei GP verschlechtern kann.

In genetischer Programmierung werden die Programme als Bäume dargestellt. Als Beispiel für einen sogenannten Syntaxbaumes, welcher die zweistelligen Funktionen $+$, $*$ und max und die Terminale x, y und 3 verwendet, siehe Abbildung 7:

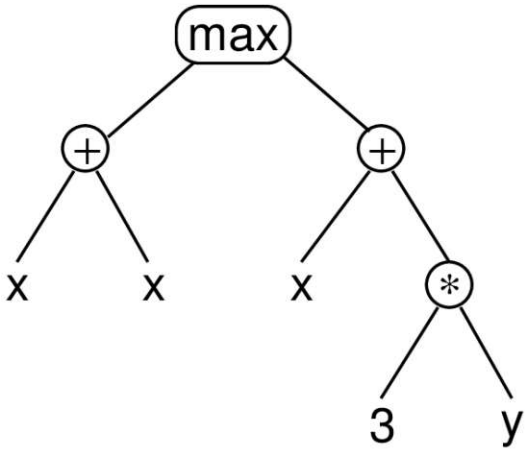


Abbildung 7: Syntax Tree [Poli et al., 2008]

Auf Grund der Baumstruktur in GP eignen sich auch besonders Programmiersprachen, die eine solche Darstellung unterstützen. Gerne wird hierfür Lisp (List Processing) verwendet, eine der ältesten noch gebräuchlichen Programmiersprachen nach Fortran. Egal ob von einem Programm oder von Daten gesprochen wird, beides wird hier über Listen und Klammern realisiert. Dies ermöglicht auch eine einfache Verschachtelung von Bäumen. In Präfixnotation und dementsprechend eine natürliche Schreibweise für Listen, erhält man für den Baum in Abbildung 7 die Form: $(\max(+xy)(+x(*3y)))$. Eine Berechnung erfolgt dann von Innen nach Außen.

Initialisierung

Zu Beginn eines jeden Laufes muss eine initiale Population gebildet werden. Dazu müssen die Parameter der Populationsgröße und der maximalen Baumhöhe bekannt sein. Die Populationsgröße gibt hier die Anzahl der Bäume/Programme an, die erzeugt werden sollen. Unter der Höhe eines Baumes versteht man die maximale Anzahl an Kanten, die von der Wurzel zu einem Blatt erforderlich sind. Der Baum in Abbildung 7 hat als Wurzel die Funktion max und als Blätter die fünf Terminale $x, x, x, 3$ und y. Funktionen sind immer interne Knoten. Der Pfad von max nach y hat die Länge 3. Da dies einer der zwei längsten Pfade ist, hat der Baum auch eine Höhe von 3. Wären also $\{\max, +, *, x, 3, y\}$ eine Teilmenge der primitiven Menge (wovon hier ausgegangen wird) und die maximale Baumhöhe ≥ 3, so könnte der Baum in Abbildung 7 zufällig erzeugt werden.

Es gibt unterschiedliche Methoden, um einen Baum zu erzeugen. Eine der Einfacheren, ist die sogenannte grow-Methode. Hierbei wird der Wurzelknoten mit einem primitiven Element belegt. Ist der Wurzelknoten ein Terminal, so ist ein Baum mit Höhe 0 angelegt und wird der initialen Population hinzugefügt. Ansonsten sind für eine n-äre Funktion n neue Knoten (Kinder) entstanden, die analog wieder mit primitiven Elementen belegt werden. Bei der Belegung eines Knotens ist immer darauf zu achten, ob dieser die maximale Baumhöhe bereits erreicht hat. Falls ja, dürfen nur noch Terminale verwendet werden, da diese den Pfad schließen.

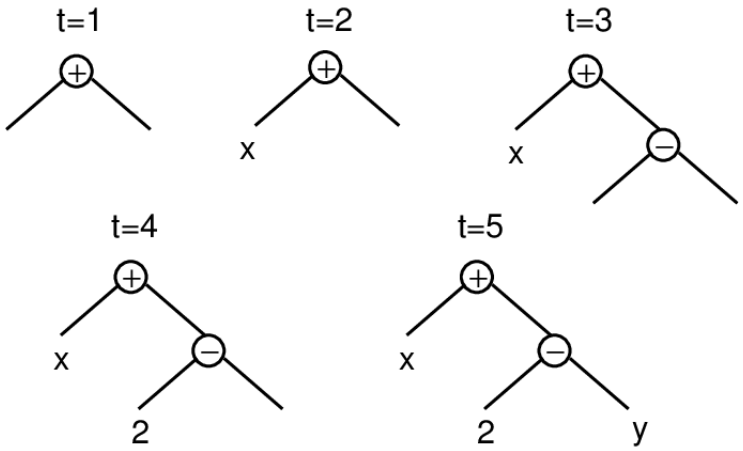


Abbildung 8: Grow Initialisation Method [Poli et al., 2008]

Als Beispiel hierfür siehe Abbildung 8, bei der ein Baum mit einer maximalen Höhe von 2 gebildet wird. In Schritt 1 (time t=1) wird der Wurzelknoten mit der zweistelligen Addition (zufällig aus der primitiven Menge gewählt) belegt. Es entstehen also zwei neue Knoten (Kinder). Für den Linken wird rekursiv wieder ein primitives Element ausgewählt. Dies ist für t=2 das Terminal x und dadurch wird der linke Teilbaum geschlossen. Im dritten Schritt kommt für den rechten Teilbaum erneut eine zweistellige Funktion hinzu. Da die Subtraktionsargumente einen Baum der Höhe 2 bilden, dürfen diese nur noch aus der terminalen Menge gewählt werden. Die Blätter 2 und y (in t=4 und t=5) schließen den Baum wieder.
Besteht die primitive Menge hauptsächlich aus Terminalen, so entstehen Bäume, die kaum die maximal erlaubte Baumhöhe erreichen. Sind hingegen sehr viele Funktionen enthalten, so haben alle erzeugten Bäume nahezu dieselbe (maximale) Höhe. Ähnliche Baumstrukturen bekommt man auch, wenn viele Funktionen dieselbe Stelligkeit aufweisen. Es muss also bereits bei der Bildung von Terminal- und Funktionenmengen auf dies geachtet werden. Abhilfe können auch Methoden schaffen, die gezielt und dennoch teils zufallsbasiert, auf die primitive Menge zugreifen.

Bewertung/Fitness

Es ist nun also eine initiale Population an Programmen/Bäumen gegeben. Meist ist unter diesen noch kein Kandidat anzufinden, der eine akzeptable Lösung darstellt. Dennoch müssen alle bewertet werden und es muss herausgefunden werden, welche sich für eine weitere Behandlung (Rekombination und Mutation) eignen. Dies erfolgt mit Hilfe einer sogenannten Fitnessfunktion/Evaluierungsabbildung, die zu jedem Programm dessen Qualität zurückliefert. Wie ein Rückgabewert aussieht ist problemabhängig. So könnte zum Beispiel ein Programm ein Logikgatter darstellen, welches zu gegebenen Eingangssignalen ein Ausgangssignal liefert. Hat man nun mehrere Testfälle (Paare von Eingangs- und Ausgangssignalen) gegeben, so kann die Fitnessfunktion überprüfen wie oft die logische Schaltung richtig liegt und diese Anzahl zurückgeben. Natürlich ist hier ein hoher Fitnesswert eines Programms gefragt. Wir haben also ein Maximierungsproblem vorliegen. Neben der Funktionalität könnte die Fitness aber auch noch von der Komplexität abhängen. So ergäbe es Sinn ein Logikgatter schlechter zu beurteilen, falls es viele logische Bauteile beinhaltet.
Möchte man hingegen eine Datenmenge über eine Funktion beschreiben (Abbildung 7 wäre eine mit zwei Eingangsvariablen), so werden Funktionen mit kleinen Abweichungen gegenüber welchen mit großen bevorzugt. Oft wird hierfür ein Datenpunkt über die Funktion ausgewertet und das Resultat mit dem tatsächlichen Wert subtrahiert und quadriert. Summiert man diese quadratischen Fehler über alle Datenkpunkte auf, so erhält man den sogenannten Mean-Squared-Error MSE. Dies ist ein Minimierungsproblem, bei dem sich ebenfalls wieder die Anzahl der verwendeten primitiven Elemente als Maß der Komplexität miteinbeziehen lässt.

Paarungsselektion/Umweltselektion

Sind nun alle Programme der ersten Generation bewertet, so kann an die Auswahl für die Rekombination gegangen werden. Die meisten Selektionsverfahren wählen dabei bevorzugt die Individuen mit der höchsten Fitness aus. Es kann/darf auch vorkommen, dass ein und dasselbe Programm für mehrere Nachfolger dient. Um hier nicht an Diversität zu verlieren, da ein Individuum die nächste Generation mit Kindern überflutet, müssen Selektionsverfahren auch genügend mittelmäßige Programme auswählen. Ein sehr gebräuchliches Verfahren ist die Turnierselektion. Zu Beginn muss hierfür eine Turniergröße (kleiner als Populationsgröße) festgelegt werden. Genau so viele Teilnehmer werden nun zufällig aus der Population gewählt. Der Gewinner dieses Turniers ist der mit dem besten Fitnesswert und wird entweder für Reproduktion, Mutation oder Rekombination freigegeben. Für Rekombination muss analog ein weiteres Turnier gestartet werden, um einen Partner zu bestimmen. Es werden so lange Turniere durchgeführt, bis sich eine neue Population gebildet hat. Wichtig anzumerken ist, dass eine kleine Turniergröße (relativ betrachtet zur Populationsgröße) mehr schwache und mittelmäßige Kandidaten erlaubt. Dies liegt daran, dass bei Turnieren mit vielen Teilnehmern auch die Wahrscheinlichkeit der Teilnahme von guten Individuen höher ist. Man sagt bei dieser Variante der Turnierselektion bleibt der Selektionsdruck konstant. Es gibt auch Varianten, bei denen der Selektionsdruck steigt (Erhöhung der Populationsgröße) oder fällt (Senkung der Populationsgröße), um einer vorzeitigen Konvergenz (Annäherung an ein lokales Optimum) vorzubeugen. Eine genauere Behandlung dazu findet man in [Miller und Goldberg, 1995].

Genetische Operatoren

In GP wird zwischen drei verschiedenen genetischen Operatoren unterschieden. Die bei einem GP Lauf am häufigsten ausgeführte ist die Rekombination. Wir bezeichnen dessen Wahrscheinlichkeit mit p_K, die meist so zwischen $0,7$ und $1,0$ liegt. Hierfür benötigt man zwei Elternprogramme (wie bereits erwähnt wurde), für die jeweils eine Kopie angelegt wird und welche anschließend für ein Nachwuchsprogramm kombiniert werden. Der Mutationsoperator hingegen fordert nur einen Elter, der kopiert und manipuliert wird. Die Wahrscheinlickeit hierfür ist p_M, die so um $0,0$ bis $0,3$ liegt. Zu guter Letzt gibt es noch die Reproduktion, die ein ausgewähltes Programm (mit einer Wahrscheinlichkeit p_P) kopiert und dessen Kopie ohne Veränderung in die nächste Population gibt. Diese wird in GP meist nicht verwendet und ansonsten liegt p_P nahe 0. Zusammen muss sich also ergeben, dass $p_K + p_M + p_P = 1$.

Rekombination/Crossover

Es gibt eine Vielzahl an Rekombinationsoperatoren. Dies liegt vor allem am zu behandelnden Problem und auch daran, dass sich daraus dann viele Variatonen gebildet haben. Ein grundlegender Operator ist das sogenannte Sub Tree Crossover, welches zwei Programmbäume vermischt:

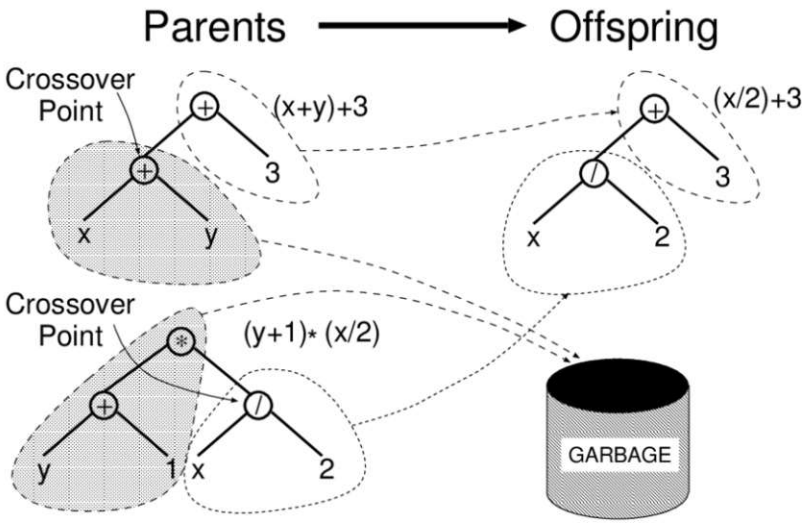


Abbildung 9: Sub Tree Crossover [Poli et al., 2008]

Über ein Selektionsverfahren werden dabei zwei Programme ausgewählt. Kopien davon sind auf der linken Seite in Abbildung 9 als Bäume dargestellt. Das Sub Tree Crossover wählt nun weiters einen beliebigen Knoten im ersten Parent und einen weiteren im Zweiten aus (gekennzeichnet als Crossover Point). Diese zwei Knoten betrachtet man nun als Wurzel zweier Unterbäume (grau eingezeichnet). Im ersten Elter wird nun der Unterbaum verworfen und an Stelle des Crossover Points der Unterbaum aus Elter 2 angehängt. Der Komplementärbaum aus Parent 2 kann auch wieder verworfen werden. Was als Resultat bleibt, ist das Kind/Offspring auf der rechten Seite.
Da in einem Baum die Anzahl an Blättern/Terminalen immer die der internen Knoten/Funktionen übersteigt, erfolgt bei Sub Tree Crossover meist nur ein "kleiner Austausch" unter den Parents. Oft nur welcher mit Blatt und Blatt. Um dem vorzubeugen gibt es auch hierfür wieder eine Vielzahl an Varianten. Eine Möglichkeit wäre, bei der Wahl für den Crossover Point zwischen internen und externen Knoten zu unterscheiden. Die Wahrscheinlichkeit für ein Terminal soll dabei höher liegen [Koza, 1992].

Mutation

Um die Funktionsweise eines Mutationsoperators näher zu bringen, eignet sich die sogenannte Sub Tree Mutation gut. Diese wird in der GP Software oft als Standardoperator verwendet. Es wird gezeigt werden, dass sie Ähnlichkeit zu der oben beschriebenen Sub Tree Crossover Methode aufweist.

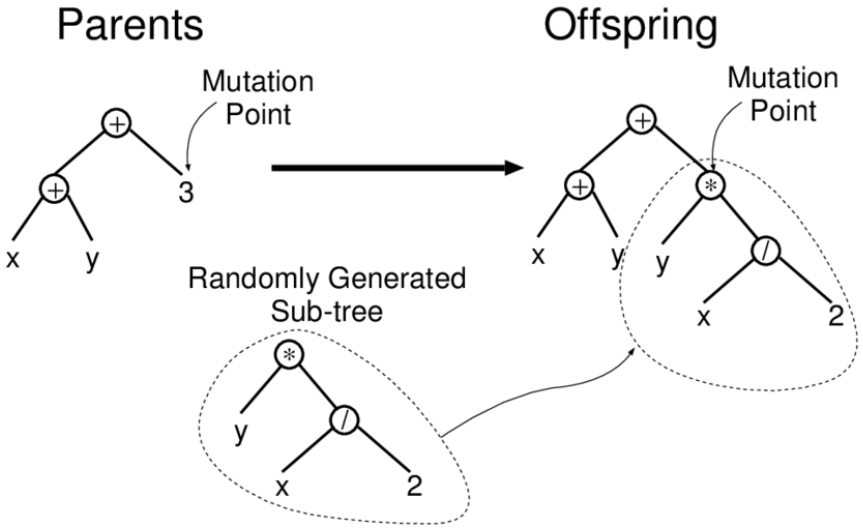


Abbildung 10: Sub Tree Mutation [Poli et al., 2008]

Weiters wird davon ausgegangen, dass über Selektion ein Programm ausgewählt und kopiert wurde. Veranschaulicht sei diese Kopie nun der Baum links oben in Abbildung 10. Wie bei Sub Tree Crossover wird auch hier ein zufälliger Knoten (Mutation Point) ausgewählt. In diesem Fall das Blatt 3. Anschließend wird ein zufälliger Baum generiert. Der sich im Parent befindende Teilbaum, mit Mutationspunkt als Wurzel, wird verworfen und durch den neuen Baum ersetzt.
Wurde nun das Sub Tree Crossover bereits in der GP implementiert, so lässt sich diese durch Modifikation als eine Sub Tree Mutation nutzen. Hierbei muss nur der zweite Parent, nicht wie gewöhnlich ausgewählt und übergeben, sondern nur zufällig generiert werden. Als Crossover Point im zweiten Parent wird einfach die Wurzel verwendet.

Abbruchbedingung

Wann und ob ein Algorithmus terminiert ist problemspezifisch. So kann ein erzeugtes Individuum, welches das vorgegebene Problem löst, zurückgegeben und der Ablauf abgebrochen werden. Meist wird jedoch keine exakte Lösung gefunden und vom Anwender muss ein Grenzwert für die ausreichende Fitness eines Individuums festgelegt werden. Da das Erreichen einer solchen Fitness jedoch nicht sichergestellt ist, wird oft noch zusätzlich eine maximale Anzahl an Generationen festgelegt. Eine Zeitbegrenzung des Suchvorgangs wäre auch noch möglich.

2.3.4 Genetische Algorithmen

Wie bereits in der Einführung zu den evolutionären Algorithmen angemerkt wurde, können die genetischen Algorithmen als ein Teilgebiet der genetischen Programmierung aufgefasst werden. Der grobe Unterschied liegt sozusagen nur in einer vordefinierten Baumlänge der Individuen. Im folgenden Kapitel wird ein spezieller genetischer Algorithmus diskutiert, der als Repräsentanten Kettenbrüche festgewählter Tiefe verwendet. Wir dürfen also an den obigen Aufbau der genetischen Programmierung denken, wenn wir von einem genetischen Algorithmus sprechen.

2.3.5 Memetische Algorithmen

Wird ein genetischer Algorithmus um einen lokalen Suchmechanismus erweitert, so spricht man von einem Memetischen Algorithmus (MA). Eine lokale Suche wählt nun einen Lösungskandidaten und vergleicht ihn in seiner Nachbarschaft (leichte Abwandlungen des ursprünglichen Kandidaten), um damit eine Optimierung zu erreichen. Für eine ausführliche Behandlung memetischer Algorithmen und für Anwendungsbeispiele siehe [Hart et al., 2004] und [Moscato, 1989].

2.3.6 Symbolische Regression

Darunter versteht man ein systematisches Durchsuchen eines Funktionenraumes nach einem brauchbaren Kandidaten. Dieser soll eine gegebene Datenmenge so gut wie möglich/nötig beschreiben. Hierbei soll bereits vor der Regressionsanalyse der Raum so klein wie möglich gewählt werden und unter den (genetischen) Operatoren abgeschlossen sein. Hierfür wird meist genetische Programmierung verwendet.

3 Algorithmus zur Kettenbruch-Regression

Wie der Name bereits vermuten lässt, handelt es sich hier um ein symbolisches Regressionsverfahren. Dabei wird der Funktionenraum der analytischen Kettenbrüche nach geeigneten Kandidaten durchsucht. Der Anreiz zur Verwendung und Untersuchung der Continued Fraction Regression (kurz CFR) kommt aus der Publikation [Moscato et al., 2020]. Dort wird ein Algorithmus vorgestellt und auf viele physikalische Datensätze (Schaffer's data collection) erfolgreich angewendet. Verglichen wurde er dabei auch mit 10 Algorithmen aus der scikit-learn software collection und ging dabei in Trainings- und Testphasen meist als Sieger hervor. Die vorausgehende Veröffentlichung [Sun und Moscato, 2019] war der Anlass für diese umfassende Untersuchung.
Im vorausgehenden Kapitel wurde die genetische Programmierung genauer behandelt, weil nun mit Hilfe dessen dieses Regressionsproblem gelöst wird. Beginnend mit der Definition des Problems wird anschließend analog zum GP Kapitel der CFR Algorithmus aufgebaut. Für den Pseudocode und einer ausführlichen Beschreibung siehe auch [Moscato et al., 2021].

3.1 Problemstellung

Gegeben sei eine Menge an Datenpunkten $(x_1, x_2, \ldots, x_n, y) \in \mathbb{R}^{n+1}$ und $n \in \mathbb{N}$, die eine unbekannte multivariate Zielfunktion darstellen. Diese soll nun durch einen analytischen Kettenbruch approximiert werden. Gesucht wird also eine Funktion $f : \mathbb{R}^n \to \mathbb{R}$ der Form

$$f(\boldsymbol{x}) = g_0(\boldsymbol{x}) + \cfrac{h_0(\boldsymbol{x})}{g_1(\boldsymbol{x}) + \cfrac{h_1(\boldsymbol{x})}{g_2(\boldsymbol{x}) + \cfrac{h_2(\boldsymbol{x})}{g_3(\boldsymbol{x}) + \ddots}}}$$

wobei die g_i und h_i lineare Funktionen von $\mathbb{R}^n \to \mathbb{R}$ sind

$$g_i(\boldsymbol{x}) = \boldsymbol{a_i}\boldsymbol{x} + \alpha_i$$
$$h_i(\boldsymbol{x}) = \boldsymbol{b_i}\boldsymbol{x} + \beta_i$$

mit den Koeffizientenvektoren $\boldsymbol{a_i}, \boldsymbol{b_i} \in \mathbb{R}^n$ und den Konstanten $\alpha_i, \beta_i \in \mathbb{R}$ für alle $i \geq 0$.

3.2 Programme/Individuen

Als Terminale dienen die Eingangsvariablen $x_1, x_2, \ldots, x_n$ zusammen mit sogenannten kurzlebigen Zufallskonstanten (ephemeral random constants) $\mathcal{R}$. Diese werden nur bei der Initialisierung eines Individuums zufällig generiert und bleiben im Weiteren unverändert (im Gegensatz zum rand-Befehl). Dabei werden die Konstanten $\mathcal{R}$ aus dem Intervall (-3,3) gewählt.
Die Funktionenmenge wird aus den vier arithmetischen Grundoperatoren $\{+, -, *, /\}$ gebildet.
Was die Abgeschlossenheit betrifft, ist Vorsicht geboten. Die Typenkonsistenz ist zwar immer erfüllt, da ja die arithmetischen Operationen auf reelle Zahlen (double) angewendet werden dürfen, jedoch nicht die Auswertungssicherheit. Durch die Auswertung an einem Datenpunkt kann schnell eine Division durch 0 (oder Nahe 0) erfolgen. Solche Situationen sind vor der Division abzufragen und bei Eintritt ist unmittelbar eine schlechte Fitness des Programmes zurückzuliefern.
Die Angemessenheit wird im Kapitel 4 - Mathematische Analyse noch ausführlich diskutiert. Es sei hier nur mal festgestellt, dass sich jede rationale Funktion als Kettenbruch darstellen lässt. Da sich nun jede meromorphe Funktion (z.B. Gammafunktion, Riemannsche-Zeta-Funktion, Tangens, etc.) durch rationale Funktionen approximieren lässt, ist dies auch über Kettenbrüche möglich.

3.3 Initialisierung

Bevor eine erste Population gebildet werden kann, muss die Kettenbruchtiefe definiert werden. Diese gibt die Anzahl der Glieder eines Kettenbruches an. Im Kapitel 4, über die Mathematische Analyse, wird sie auch als Konvergenten bezeichnet (die Notation ist nur etwas anders).

Ohne hier bereits eine exakte Definition anzugeben, sehen Kettenbrüche der Tiefe 1 bis 4 folgendermaßen aus:

Tiefe 1:

$$f(\boldsymbol{x}) = g_0(\boldsymbol{x})$$

Tiefe 2:

$$f(\boldsymbol{x}) = g_0(\boldsymbol{x}) + \frac{h_0(\boldsymbol{x})}{g_1(\boldsymbol{x})}$$

Tiefe 3:

$$f(\boldsymbol{x}) = g_0(\boldsymbol{x}) + \frac{h_0(\boldsymbol{x})}{g_1(\boldsymbol{x}) + \frac{h_1(\boldsymbol{x})}{g_2(\boldsymbol{x})}}$$

Tiefe 4:

$$f(\boldsymbol{x}) = g_0(\boldsymbol{x}) + \frac{h_0(\boldsymbol{x})}{g_1(\boldsymbol{x}) + \frac{h_1(\boldsymbol{x})}{g_2(\boldsymbol{x}) + \frac{h_2(\boldsymbol{x})}{g_3(\boldsymbol{x})}}}$$

Grob gesagt wird also immer ein Zähler- und ein Nennerpolynom angehängt. Eine Tiefe von $n \in \mathbb{N}$ besagt somit also $h_{n-1} = 0$. Man sieht auch sofort, dass es sich bei Kettenbruchtiefe 1 um Lineare Regression handelt. Für Tiefen ≥ 1 ist Lineare Regression also immer implizit in der Kettenbruch-Regression enthalten.
Im Gegensatz zur GP üblichen Initialisierung mit Bäumen, sehen hier alle erzeugten Kandidaten in Struktur und Tiefe gleich aus. Jedoch wird bei der ersten Generation darauf geachtet weniger komplexe Kettenbrüche zu erzeugen. Dies erfolgt dadurch, dass jede Variable (in jedem Glied) nur mit einer Wahrscheinlichkeit von $p = 1/3$ verwendet wird. Die Koeffizienten (für verwendete Variablen) und Konstanten bestehen alle aus kurzlebigen Zufallskonstanten $\mathcal{R}$, die anschließend durch Rekombination und Mutation optimiert werden. Es handelt sich also um reine Parameteroptimierung, weshalb bei dem CFR Algorithmus auch von einem genetischen Algorithmus gesprochen wird.
Es gibt hier keine variable Populationsgröße. Die erzeugten Kettenbrüche werden in einem Trinärbaum der Tiefe 2 (siehe Abbildung 11) abgespeichert:

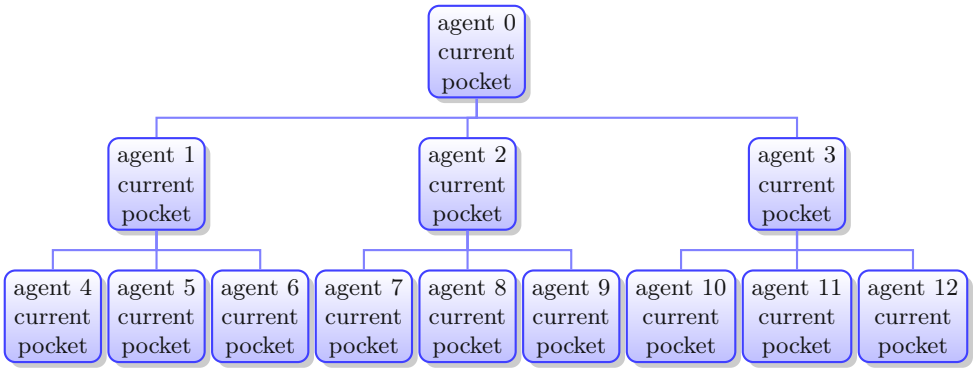


Abbildung 11: Struktur der Population bei CFR Algorithmus

Die Knoten werden dabei als Agenten bezeichnet und beinhalten jeweils zwei Lösungskandidaten. Einer davon ist der sogenannte current und der andere der pocket. Insgesamt sind also 26 Lösungskandidaten zu erzeugen. Bis zur Bewertung sind die Kettenbrüche noch beliebig angeordnet.
Nach erfolgter Bewertung findet eine Sortierung innerhalb des Baumes statt und die Populationsstruktur hat nach der Initialisierung und im Weiteren auch in jeder Generation zwei Invarianten zu erfüllen:

1. In jedem Agenten hat der pocket-Lösungskandidat einen besseren Fitnesswert als der current-Lösungskandidat. Ist dies nicht der Fall, muss pocket mit current vertauscht werden.

2. Jeder interne Knoten/Agent (kein Blattknoten) hat einen pocket Lösungskandidat mit besserem Fitnesswert als die pocket Lösungskandidaten der drei Nachfolger/Kinder. Ansonsten wird der Agent mit bester pocket Fitness nach oben sortiert. Dies impliziert auch, dass die beste Lösung im Wurzelknoten des Trinärbaums zu finden ist.

3.4 Bewertung/Fitness

Die Bewertung eines Kettenbruchs f erfolgt anhand von Trainingsdaten $D = \{(\boldsymbol{x}_i, y_i)$, für $i \leq N\}$, wobei hier N der Anzahl an Datenpunkten entspricht. Daraus wird der Mean-Squared-Error gebildet:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (f(\boldsymbol{x}_i) - y_i)^2$$

Um im Weiteren eine Balance zwischen Genauigkeit und Modellkomplexität zu erreichen, verwendet man eine angepasste Variante des MSE und zwar

$$\text{angepasster MSE} = MSE \times (1 + \Delta \times \#\text{ Anzahl an verwendeten Variablen})$$

dabei ist $\Delta > 0$ das Gewicht einer verwendeten Variablen.

3.5 Mutations-/Paarungsselektion

Die Auswahl der Kandidaten für Mutation und speziell für Rekombination erfolgt bei CFR immer gleich. Zuerst wird für alle current solution im Trinärbaum die Mutation durchgeführt. Diese erfolgt mit einer gewissen Wahrscheinlichkeit p_M für jeden Kandidaten. Wurde nun ein Kettenbruch für Mutation ausgewählt, gibt es hierfür zwei mögliche Operatoren. Welcher angewendet werden soll, hängt von der Differenz seiner Fitness zu der seiner pocket solution ab. Sollte einer der beiden Bedingungen

$$\begin{aligned} \text{current\_fitness} &< 1,2 * \text{ pocket\_fitness} \\ \text{current\_fitness} &> 2,0 * \text{ pocket\_fitness} \end{aligned}$$

erfüllt sein, so findet eine sogenannte Große Mutation statt, andernfalls nur eine Kleine Mutation. Deren Definition wird im folgenden Kapitel behandelt.
Wurden nun alle current Lösungskandidaten der Mutation unterzogen, so erfolgt anschließend die Rekombinationsphase. Hierbei werden nach einem festen Schema pocket- und current Lösungskandidaten (in Unterbäumen) kombiniert:

1. current(E)=Rekombination (pocket(E), current(K_1))
2. current(K_3)=Rekombination(pocket(K_3), current(E))
3. current(K_1)=Rekombination(pocket(K_1), current(K_2))
4. current(K_2)=Rekombination(pocket(K_2), current(K_3))

Man sieht, dass auch hier wieder nur die current Lösungskandidaten überschrieben werden. Es wird auch jeder Kandidat noch einer lokalen Optimierung unterzogen. Die genaue Beschreibung dazu findet sich im folgenden Unterkapitel 3.7 - Lokale Suche. Sind alle Rekombinationsschritte durchgeführt (inklusive Optimierung), wird der Trinärbaum wieder sortiert.
Sollte nach fünf Generationen keine Verbesserung der Wurzel (pocket) stattfinden, wird diese durch einen zufällig generierten Kettenbruch ersetzt. Dies soll helfen einer frühzeitigen Konvergenz vorzubeugen.

3.6 Genetische Operatoren

Bis jetzt war es noch nicht nötig von der digitalen Repräsentation eines Kettenbruchs zu sprechen. Die hier vorgestellten genetischen Operatoren benötigen allerdings eine spezielle Struktur. So besteht ein Kandidat aus einem

1. boolean vector variables, der allgemein angibt, ob eine Variable im Kettenbruch enthalten sein darf. Seine Länge entspricht also der Anzahl an Eingangsvariablen.

2. boolean vector fractionvars, der speziell alle im Kettenbruch verwendeten Variablen in jedem Glied (ohne Konstanten) anspricht. Diese werden dann bei True eingeblendet und bei False wieder aus.

3. double vector coefficients, welcher einen jeden Koeffizienten (inklusive Konstanten) beinhaltet.

3.6.1 Genetische Mutation

In CFR wird nun zwischen zwei zum Einsatz kommenden Mutationsoperatoren unterschieden.

Kleine Mutation

Zu Beginn wird eine beliebige und aktivierte Variable aus variables gewählt. Anschließend wird auch noch ein beliebiges Glied bestimmt und die dazugehörige Variable in fractionvars negiert. Es werden also zwei Fälle unterschieden:

1. Die Variable war an und wird abgeschaltet. Dann wird der dazugehörende Koeffizient (in coefficients) auf 0 gestellt.

2. Die Variable war aus und wird eingeschaltet. Anschließend wird der betroffene Koeffizient in coefficients mit einem Wert zwischen -3 und 3 belegt ($\mathcal{R}$).

Große Mutation

Hier wird eine zufällige Variable aus variables (kann im Gegensatz zu der Kleinen Mutation auch ausgeschaltet sein) gewählt. Es wird also wieder unterschieden:

1. Die Variable war an und wird nun abgeschaltet. Für alle korrespondierenden Variablen in fractionvars setze den Wert auf False und für die dazugehörigen Koeffizienten in coefficients wird der Wert zu 50% beibehalten oder 0 gesetzt.

2. Die Variable war aus und wird nun eingeschaltet. Anschließend wird für alle betroffenen Variablen in fractionvars entweder

   (a) von False auf True gestellt und der korrespondierende Koeffizient zu 50% auf 0 oder auf einen zufälligen Wert zwischen -3 und 3 gestellt ($\mathcal{R}$).

   (b) bei True wird nichts verändert.

3.6.2 Genetische Rekombination

Hier wird ein zufälliger Mengenoperator aus der Vereinigung $\cup$, dem Schnitt $\cap$ und der symmetrischen Differenz $\triangle$ gewählt. Dieser wird dann auf die Variablenmenge (sozusagen auf variables) beider Eltern A und B angewendet und damit die Variablenmenge des Kindes C gebildet.
Es wird also ein neues Kindprogramm initialisiert und in coeffvars alles auf False und in coefficients alles auf 0 gesetzt. Für Variablen aus fractionvars, für welche die zugehörige Variable in variables True ist, wird folgendermaßen vorgegangen:

1. Ist die Variable für beide Eltern True (ebenfalls in fractionvars), so setze die Variable auf True und den dazugehörenden Koeffizienten $\text{cof}_C = \text{cof}_A + \text{rand}(-1, 4) * (\text{cof}_B - \text{cof}_A)/3$.

2. Ist nur eine Variable eines Elter True, so setze die Variable auf True und den dazugehörenden Koeffizienten $cof_C = cof_{\text{Elter}}$.

Anschließend müssen noch alle Konstanten (bezüglich der Kettenbruchglieder als lineare Funktionen aufgefasst) mit $const_C = const_A + (-1, 4) * (const_B - const_A)/3$ initialisiert werden.

3.7 Lokale Suche

Hat bereits Mutation und Rekombination stattgefunden, wird auf den erzeugten Nachfolgekandidaten eine Nelder-Mead-Methode angewendet. Der Vorteil dieser lokalen Suche ist, dass nicht wie bei Optimierungsverfahren üblich, der Gradient einer Funktion (hier des Kettenbruchs) betrachtet werden muss. Die Funktion darf auch nichtlinear sein. Als Anregung, es gibt auch eine vereinfachte und auch effizientere Variante des Nelder-Mead-Verfahren. Für deren Entwicklung wurde die genetische Programmierung verwendet [Fajfar et al., 2017].
Sei also nun $N \in \mathbb{N}$ die Anzahl an Koeffizienten (und Konstanten), die im Kettenbruch eingeblendet sind. So können diese Koeffizienten zusammengefasst (einen Vektor bilden) und als Element aus $\mathbb{R}^N$ betrachtet werden. Durch Variieren dieses Vektors soll also der MSE Wert des Kettenbruches verbessert werden. Anders betrachtet kann man dies auch als Minimierungsproblem über eine Funktion $f : \mathbb{R}^N \to \mathbb{R}$ auffassen.
Ähnlich zum Simplex-Verfahren von George Dantzig [Dantzig, 1990], muss auch hier ein Simplex gebildet werden. Dazu werden die kanonischen Vektoren des $\mathbb{R}^N$ verwendet und auf den Koeffizientenvektor addiert. Zusammen mit dem ursprünglichen Vektor ergibt sich also ein Simplex mit $N + 1$ Punkten, welches in keiner Hyperebene eines Teilraums des $\mathbb{R}^N$ liegt. Iterativ wird nun zu jedem dieser Vektoren der MSE-Wert bestimmt. Der Vektor mit dem schlechtesten Mean-Squared-Error wird neu gebildet, um im Weiteren eine Verbesserung zu erzielen. Analog wird dann wieder der neue "schlechteste Vektor" manipuliert. Die hierfür notwendigen Schritte sind für die Ebene in Abbildung 12 dargestellt

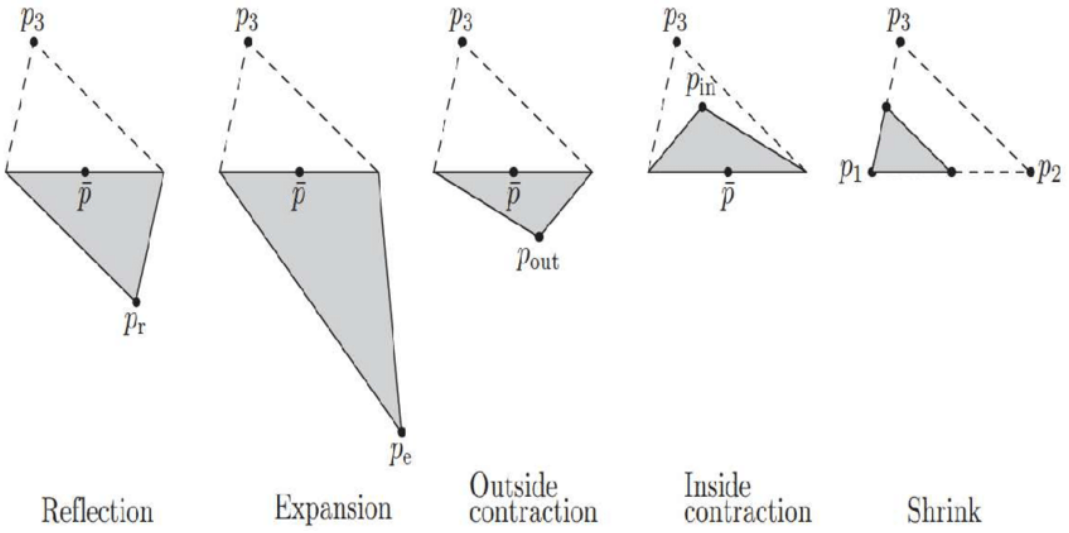


Abbildung 12: Nelder-Mead-Methode [Wright, 2010]

und unterliegen einer genauen Reihenfolge, die nun kurz besprochen wird. Vorbemerkt sei schon mal, dass mit p_3 immer (in der Ebene) der schlechteste Punkt vorliegt. Der in der Graphik eingezeichnete Punk $\bar{p}$ ist der Mittelwert über alle Punkte, ausgenommen dem Schlechtesten, also über p_1 und p_2.
Die allgemeine Formulierung des Algorithmus lautet:

1. Sortiere die Punkte $x_0, x_1, \ldots, x_N$ des Simplex so, dass $f(x_N)$ den schlechtesten-, $f(x_{N-1})$ den zweitschlechtesten- und $f(x_0)$ den besten Funktionswert aufweisen.
2. Bestimme nun über alle Punkte, ausgenommen x_N, den Mittelwert $\bar{x} := \frac{1}{N} \sum_{n=0}^{N-1} x_n$.
3. Anschließend wird x_N um $\bar{x}$ reflektiert und die Reflektion $r := \bar{x} + \alpha(\bar{x} - x_N)$ erhalten.
4. Sollte nun r einen besseren Wert als x_0 haben, so wird versucht x_N verstärkt zu spiegeln. Die sogenannte Expansion $e := \bar{x} + \gamma(\bar{x} - x_N)$ wird mit r verglichen und x_N durch den besseren der beiden ersetzt. Daraufhin folgt wieder Schritt 1.
5. Ansonsten wird überprüft, ob die Reflektion r zumindest besser als x_{N-1} ist. Trifft dies zu, so wird x_{N-1} durch r ersetzt und danach auch hier wieder zu Schritt 1 zurückgegangen.

6. Andernfalls wird der bessere der beiden Punkte aus r und x_N gewählt, schlicht bezeichnet als h und damit wird der kontrahierte Punkt $c := h + \beta(\bar{x} - h)$ gebildet.

7. Der erhaltene Punkt c wird wieder mit x_N verglichen und sollte er besser sein, wird x_N ersetzt. Hier auch wieder zurück zu Schritt 1.

8. Sollten die vorigen Schritte nicht helfen, wird das Simplex komprimiert und alle x_i durch $x_i + \sigma(x_0 - x_i)$, wobei $i \in \{1, 2, \ldots, N\}$ (x_0 bleibt also gleich) ersetzt.

9. Gehe zurück zu Schritt 1.

Der Algorithmus wird so lange ausgeführt, bis eine Terminierungsbedingung erfüllt ist. Dies kann zum Beispiel eine fest vorgegebene Anzahl an Iterationen sein, oder auch wenn die Funktionswerte aller Punkte kaum voneinander abweichen - sprich, falls sich das Simplex eng genug um ein lokales Optimum geschnürt hat. Nun muss noch näher auf die in den Operatoren verwendeten Konstanten/Parameter eingegangen werden. Typisch gewählte Werte sind hierfür:

- $\alpha = 1$ (Reflektion)
- $\gamma = 2$ (Expansion)
- $\beta = 1/2$ (Kontraktion)
- $\sigma = 1/2$ (Komprimierung)

Jedoch können die Parameter auch allgemein unter den Bedingungen $0 < \alpha < \gamma$ und $\beta, \sigma \in (0, 1)$ gewählt werden. Achtung, in Abbildung 12 wird auch noch zwischen einer Kontraktion nach Innen und Außen unterschieden. Dies kann durch eine geeignete Wahl des Vorzeichens und einem zusätzlichen Schritt im Algorithmus eingebunden werden.

3.8 Parameter und Terminierungsbedingungen

Die in den Publikationen verwendeten Paramater und Bedingungen auf einen Blick:

Parameter	**Wert**
Bestrafung durch die Anzahl an verwendeten Variablen Δ	0,10
Mutationswahrscheinlichkeit p_M	0,10
Kettenbruchtiefe	4
Populationsgröße (Trinärbaum)	13
Anzahl der Generationen	200
Anzahl der Generationen ohne Verbesserung bis die Wurzel ersetzt wird	5
Feature Skalierung / Normalisierungsmethode	MinMax
Anzahl der Nelder Mead Instanzen	4
Anzahl der Iterationen in Nelder-Mead	250
Anzahl der Iterationen ohne Verbesserung bis Abbruch von Nelder-Mead	10
Prozentsatz der Trainingsdaten die Nelder-Mead verwendet	20%

Tabelle 1: Übersicht der im CFR Algorithmus verwendeten Parameter

4 Mathematische Analyse

Im Folgenden soll der mathematische Hintergrund der Kettenbruch-Regression untersucht werden.

4.1 Komplexe Analysis und algebraische Grundlagen

In diesem Unterkapitel werden grundlegende Definitionen und Sätze aus der komplexen Analysis und der Algebra aufgefrischt. Diese werden in den folgenden Kapiteln benötigt, um die Theorie der Padé-Approximation aufzubauen und anschließend auch mit der Kettenbruchtheorie verbinden zu können. Obwohl man meinen würde, dass hier ein reeller Aufbau reicht (die Daten der Spannungs-Dehnungskurven sind ja reell), wird dennoch alles im Komplexen durchgeführt. Dies bringt einiges an Vorteilen mit sich, man denke nur daran, dass Nennerpolynome von rationalen Funktionen mit komplexen Koeffizienten vollständig in Linearfaktoren zerfallen. Außerdem werden Konvergenzsätze (von Padé-Approximanten) wie in [Cuyt, 1990], [Nuttall, 1970] und [Guillaume und Huard, 2000] so anwendungsreich wie möglich formuliert. An Tatsachen wie z.B., dass sich meromorphe Funktionen lokal als Laurent-Reihe (mit endlichem Hauptteil) darstellen lassen, können dabei nicht außer Acht gelassen werden. Selbst eine Betrachtung von holomorphen Funktionen f würde nicht reichen, da für die Kettenbruchbildung die Inverse $1/f$ notwendig ist. Lokal nur als Potenzreihe betrachtet gibt es hier nicht immer Inverse dazu.

Definition 4.1. Um im Folgenden Verwechslungen zu vermeiden, eine **komplexe Funktion** ist eine Funktion $f : D \to \mathbb{C}$ mit $D \subset \mathbb{C}$ beliebig.
Eine **komplexwertige Funktion** hingegen bildet nach $\mathbb{C}$ ab und hat eine beliebige Definitionsmenge.

Im Unterkapitel 4.2 über die Padé-Approximation einer Funktion in einer Variablen wird hauptsächlich über komplexe Funktionen gesprochen. Im Folgenden wird auch der Begriff des Grenzwertes benötigt. Der Abstandsbegriff, der hierbei auf $\mathbb{C}$ betrachtet wird, ist wie gewohnt die aus der euklidischen Norm induzierte Metrik.

Definition 4.2. Sei D eine offene Teilmenge der Menge der komplexen Zahlen $\mathbb{C}$. Eine Funktion $f : D \to \mathbb{C}$ heißt **(komplex) differenzierbar** in $z_0 \in D$, wenn folgender Grenzwert existiert

$$\lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0}.$$

Diesen bezeichnet man auch als $f'(z_0)$. Wenn f in allen $z \in D$ differenzierbar ist, so ist f **holomorph** in D.

Viele Rechenregeln, die aus dem Reellen bekannt sind, lassen sich auch hier übertragen. Als wichtige Beispiele seien die Ableitung von Summen, Ketten-, Produkt- und Quotientenregel erwähnt. Besonders praktisch ist im komplexen auch der Satz von Goursat, der besagt, dass eine holomorphe Funktion auch gleich beliebig oft reell und komplex differenzierbar ist.
Es folgt ein weiterer und zu dem der komplexen Differenzierbarkeit verwandter Begriff.

Definition 4.3. Sei D wieder eine offene Teilmenge von $\mathbb{C}$. Eine komplexe Funktion $f : D \to \mathbb{C}$ wird als **analytisch** in $z_0 \in D$ bezeichnet, falls sich f um z_0 in eine Potenzreihe mit Konvergenzradius $R > 0$ entwickeln lässt. Das bedeutet

$$f(z) = \sum_{n=0}^{\infty} c_n (z - z_0)^n \qquad \forall z \in \{\omega \in \mathbb{C} : |\omega - z_0| < R\}$$

wobei hier die Koeffizienten c_n auch wieder komplexe Zahlen sind. Falls f auf ganz D analytisch ist, sagt man auch kurz f ist analytisch.

Hauptsächlich wird in den folgenden Kapiteln die Eigenschaft einer Funktion in 0 analytisch zu sein benötigt. Es sollen auch noch einmal folgende Eigenschaften bewusst gemacht werden.

Lemma 4.4. Sei f (wie in Definition 4.3) eine in z_0 analytische Funktion.

(1) Dann ist f auch in z_0 holomorph

(2) Die Koeffizienten lassen sich darstellen als

$$c_n = \frac{f^{(n)}(z_0)}{n!} \qquad \text{für alle } n \geq 0 \qquad \text{(Potenzreihe = Taylorreihe)}$$

(3) Die Ableitung von f ist lokal um z_0 gegeben als

$$f'(z) := \sum_{n=1}^{\infty} n c_n (z - z_0)^{n-1} \qquad \forall z \in \{\omega \in \mathbb{C} : |\omega - z_0| < R\}.$$

Das bedeutet Potenzreihen dürfen gliedweise differenziert werden.

Beweis. Für einen vollständigen Beweis siehe [Ferus, 2010]. ■

Eine weitere Folgerung liefert der Identitätssatz. Wenn f lokal als eine Potenzreihe g und einer weiteren Potenzreihe h dargestellt werden kann, so gilt stets $h = g$ (alle Koeffizienten stimmen überein), also die Eindeutigkeit.

Satz 4.5. Gegeben sei eine offene und zusammenhängende Menge (ein Gebiet) $G \subset \mathbb{C}$ und eine Funktion $f : G \to \mathbb{C}$. Die folgenden Aussagen sind äquivalent:

(1) f ist holomorph auf G

(2) f ist analytisch auf G

Beweis. (1) $\implies$ (2) :
Folgt aus dem Potenzreihenentwicklungssatz (für einen Beweis siehe auch [Jänich, 2004])
(2) $\implies$ (1) :
Folgt aus der Tatsache, dass eine Potenzreihe gliedweise differenzierbar ist (Lemma 4.4). ■

Es ist also egal, ob von einer holomorphen oder einer analytischen Funktion f gesprochen wird.

Bei beiden Definitionen darf davon ausgegangen werden, dass für alle $z_0 \in G$ eine lokale Darstellung als Potenzreihe existiert. Im nächsten Kapitel wird dies noch des Öfteren benötigt. Für eine Entwicklung in $z_0 = 0 \in G$ spricht man auch von einer MacLaurin Reihe. Dies wird noch näher behandelt und vorerst folgt der Ausbau des Begriffs der Holomorphie zur Meromorphie.

Definition 4.6. Sei $f : D \to \mathbb{C}$ eine komplexe Funktion und D offen. Ein Punkt $z_0 \in D$ wird als **isolierte Singularität** bezeichnet, falls f auf $D \setminus \{z_0\}$ holomorph ist.
Hierbei unterscheidet man wiederum 3 Fälle:

1. z_0 ist eine **hebbare Singularität**, falls die Funktion f auf D holomorph fortgesetzt werden kann.
2. z_0 wird als **Pol** bezeichnet, wenn es keine hebbare Singularität ist und ein $n \in \mathbb{N}_{\neq 0}$ existiert, sodass $(z-z_0)^n f(z)$ in z_0 hebbar ist. Das kleinste n mit dieser Eigenschaft wird als Ordnung des Poles bezeichnet.
3. z_0 wird andernfalls eine **wesentliche Singularität** genannt.

Für eine auf ganz D differenzierbare Funktion ist jedes $z_0 \in D$ natürlich auch eine isolierte Singularität und f darin auch hebbar. Klassische Beispiele für Polstellen sind bei gebrochenrationalen Funktionen (exakte Definition im Folgenden) gegeben. Die einfachste hierbei ist mit Sicherheit $1/z$ mit Polstelle in $z_0 = 0$. Hier muss nur $n = 1$ gesetzt werden und man erhält $(z-0)^1 \frac{1}{z} = 1$ die konstante 1-Funktion, die durch $0 \mapsto 1$ holomorph fortgesetzt werden kann. Die Funktion $sin(1/z)$ hingegen hat eine wesentliche Singularität in $z_0 = 0$.

Definition 4.7. Eine Funktion $f : D \to \mathbb{C}$ mit $D \subset \mathbb{C}$ offen, wird als eine **meromorphe Funktion** auf D bezeichnet, falls sie auf einer Menge $D \setminus P_f$ holomorph ist und P_f dabei die diskrete Menge an Polstellen von f ist.

Jede meromorphe Funktion ist lokal darstellbar als Quotient zweier holomorpher Funktionen (siehe Faktorisierungssatz von Weierstrass bzw. Approximationssatz von Runge). Ersichtlich ist auch, dass mit dieser Erweiterung des Begriffs der Holomorphie die gebrochenrationalen Funktionen hinzugenommen werden:

Definition 4.8. Als eine **gebrochenrationale Funktion** von $D(\subset \mathbb{C}$ offen$) \to \mathbb{C}$ bezeichnen wir den Quotienten zweier Polynome $p(z)$ und $g(z)$ der Gestalt

$$\frac{p(z)}{g(z)} = \frac{p_0 + p_1 z + \ldots + a_n z^n}{g_0 + g_1 z + \ldots + g_m z^m} \qquad \text{mit } n, m \in \mathbb{N}.$$

Im Gegensatz zu Polynomen muss die so definierte Funktion nicht auf ganz $\mathbb{C}$ definiert und somit auch nicht überall differenzierbar sein. Probleme bereiten hier die Nullstellen des Nennerpolynoms.

Die Nullstellen des Nennerpolynoms sind allesamt Polstellen, was eine einfache Folgerung aus dem Hauptsatz der Algebra (Nennerpolynome zerfallen vollständig in Linearfaktoren) ist.
In diesem Kapitel und auch in den darauf aufbauenden, wird immer wieder der Begriff eines Ringes benötigt. Deshalb wird auch dieser kurz wiederholt:

Definition 4.9. Ein Ring $(R, +, \cdot)$ besteht aus einer Grundmenge R und den zweistelligen Operationen $+$ und $\cdot$ mit den Eigenschaften

- $(R, +)$ ist eine kommutative Gruppe. Also ist $+$ assoziativ, kommutativ, desweiteren gibt es auch ein neutrales Element und zu jedem Element aus R auch ein inverses
- Die Halbgruppe $(R \setminus \{0\}, \cdot)$ muss hier nur assoziativ sein
- Die Operation $\cdot$ ist distributiv über $+$

Ist in $(R, +, \cdot)$ die Operation $\cdot$ noch dazu kommutativ, spricht man von einem kommutativen Ring.
Ein **kommutativer Ring mit Einselement** ist dann ein kommutativer Ring, wo es bzgl. der Operation $\cdot$ auch noch ein neutrales Element gibt.

Die Definition lässt schon ahnen, dass Schwierigkeiten mit der multiplikativen Inversenbildung auftreten werden. Deshalb soll hier auch noch gleich der aus der Algebra (siehe auch [Goldstern und Winkler, 2018]) bekannte Begriff der formalen Potenzreihen angesprochen werden. Diese finden in den folgenden Kapiteln immer wieder Anwendung.

Definition 4.10. Gegeben sei ein kommutativer Ring mit Einselement R (meist der Körper $\mathbb{C}$). Die Menge aller Folgen

$$(a_n)_{n\in\mathbb{N}} =: \sum_{n\in\mathbb{N}} a_n x^n$$

mit $a_n \in R$ für alle $n \in \mathbb{N}$ wird als **Menge der formalen Potenzreihen** $R[[x]]$ bezeichnet. Man sagt auch a_n ist der n-te Koeffizient der Potenzreihe $(a_n)_{n\in\mathbb{N}}$.
Die Menge aller $(b_n)_{n\in\mathbb{N}} \in R[[x]]$ für die ab einem Index $m \in \mathbb{N}$ für alle $m > n$ gilt, dass $b_n = 0$ ist, kann als Einbettung des Polynomrings $R[x]$ aufgefasst werden. Ebenfalls als Einbettung von R kann die Abbildung $\iota : R \to R[[x]]$ mit $r \mapsto (r, 0, 0, \ldots)$ betrachtet werden. Bezüglich der Unteralgebrenbeziehung ist sogar

$$R \leq R[x] \leq R[[x]].$$

Es gilt $R[[x]]$ ist zusammen mit den darauf definierten Operationen der Addition

$$(a_n)_{n\in\mathbb{N}} + (b_n)_{n\in\mathbb{N}} := (a_n + b_n)_{n\in\mathbb{N}},$$

mit der als diskreten Faltung definierten Multiplikation

$$(a_n)_{n\in\mathbb{N}} \cdot (b_n)_{n\in\mathbb{N}} := \left(\sum_{n=0}^{k} a_n b_{k-n}\right)_{k\in\mathbb{N}},$$

mit dem neutralen Element der Addition $(0)_{n\in\mathbb{N}}$ und dem neutralen Element der Multiplikation $(1, 0, 0, \ldots)$, ebenfalls wieder ein kommutativer Ring mit Eins.

Die Tatsache, dass sich jedes Element $r \in R$ mittels $(r, 0, 0, \ldots)$ als formale Potenzreihe auffassen lässt, spielt auch im Beweis von Satz 4.17 eine Rolle. Dort wird eine quadratische Matrix $(a_{ij})_{i=1,\ldots,n;j=1,\ldots,n}$ mit Einträgen $a_{ij} \in R[[x]]$ betrachtet, denn die wie üblich definierte Determinante auf $\mathbb{C}^{n\times n}$ lässt sich auch auf $R[[x]]^{n\times n}$ bestimmen. Die Determinante ist hier mit der auf $R[[x]]$ definierten Addition und Multiplikation wohldefiniert und eine Abbildung $\det : R[[x]]^{n\times n} \to R[[x]]$. Es gelten auch die gewohnten Regeln für Determinanten.

Es folgt ein Wechsel von den klassischen Potenzreihen zu den formalen Potenzreihen und somit von der Analysis in die Algebra. Ansonsten müssten immer Betrachtungen durchgeführt werden ob und auf welcher Definitionsmenge Konvergenz vorliegt. Mit dem Ring der formalen Potenzreihen wird eine Auflockerung geschaffen, denn alle analytischen Funktionen können mit einer Teilmenge davon identifiziert werden.
Für den nächsten Schritt wird eine weitere Definition eingeführt:

Definition 4.11. Die Menge R^* aller $r \in R$, zu denen ein multiplikatives inverses Element existiert, bezeichnet man auch als **Einheiten** von R.

Was hier im speziellen noch betrachtet werden muss und auch im Kapitel 4.3 über die Darstellung einer rationalen Funktion als Kettenbruch Bedeutung findet, ist die Existenz und Darstellung eines multiplikativ inversen Elements aus $R[[x]]$.

Lemma 4.12. Sei R ein kommutativer Ring mit Eins und $(a_n)_{n\in\mathbb{N}} \in R[[x]]$, dann gilt $(a_n)_{n\in\mathbb{N}}$ ist genau dann eine Einheit aus $R[[x]]^*$, wenn a_0 eine Einheit aus R^* ist.
Für alle $(p_n)_{n\in\mathbb{N}} \in R[[x]]^*$ lässt sich $(p_n)^{-1}_{n\in\mathbb{N}} =: (q_n)_{n\in\mathbb{N}}$ rekursiv darstellen als

$$q_0 := p_0^{-1}$$
$$q_n := -q_0 \sum_{i=1}^{n} p_i q_{n-i} \qquad \text{für alle } n \geq 1.$$

Beweis. Um die erste Aussage zu zeigen, wird mit dessen Implikation begonnen:
$\Rightarrow$
Sei also $(a_n)_{n\in\mathbb{N}}$ eine Einheit in $R[[x]]^*$ und $(b_n)_{n\in\mathbb{N}}$ das multiplikativ inverse Element dazu. Damit $(a_n)_{n\in\mathbb{N}} \cdot (b_n)_{n\in\mathbb{N}} = (1, 0, 0, \ldots) =: (c_n)_{n\in\mathbb{N}}$ erfüllt ist, muss gelten

$$c_0 := \sum_{n=0}^{0} a_n b_{0-n} = a_0 b_0 = 1$$

was genau dann der Fall ist, wenn $b_0 = a_0^{-1}$ ist und somit a_0 eine Einheit in R.
Nun zu der Replikation:
$\Leftarrow$
Sei also $a_0 \in R^*$ und $b_0 := a_0^{-1}$ das multiplikativ inverse Element dazu. Dies erlaubt im Weiteren die Definition von

$$b_n := -b_0 \sum_{i=1}^{n} a_i b_{n-i} \qquad \text{für alle } n \geq 1$$

da hier dann auch alle Summanden wohldefiniert sind. Es lässt sich also eine formale Potenzreihe $(b_n)_{n\in\mathbb{N}} \in R[[x]]$ bilden. Multipliziert man nun $(a_n)_{n\mathbb{N}}$ und $(b_n)_{n\in\mathbb{N}}$ miteinander und betrachtet für $n \geq 1$ den n-ten Koeffizienten

$$c_n := \sum_{i=0}^{n} a_i b_{n-i} = a_0 b_n + \sum_{i=1}^{n} a_i b_{n-i} = \underbrace{a_0(-b_0)}_{=(-1)} \sum_{i=1}^{n} a_i b_{n-i} + \sum_{i=1}^{n} a_i b_{n-i} = 0$$

so ist ersichtlich, dass die Multiplikation das neutrale Element ergibt und somit ist die erste Aussage bewiesen.

Die zweite Aussage des Lemmas wurde bereits gezeigt. Hier braucht man nur $(p_n)_{n\in\mathbb{N}}$ als $(a_n)_{n\in\mathbb{N}}$ und $(q_n)_{n\in\mathbb{N}}$ als $(b_n)_{n\in\mathbb{N}}$ definieren. ■

Korollar 4.13. Alle $\sum_{n=0}^{\infty} c_n z^n \in \mathbb{C}[[z]]$ mit $c_0 \neq 0$ sind (multiplikativ) invertierbar.

Beweis. Da der Körper der komplexen Zahlen $\mathbb{C}$ auch ein kommutativer Ring mit Einselement ist und dessen Menge an Einheit $\mathbb{C}^*$ genau die Menge $\mathbb{C} \setminus \{0\}$ ist, folgt die Behauptung unmittelbar aus Lemma 4.12. ■

4.2 Padé Approximation

Allgemein bezeichnet man als Padé-Approximation die Darstellung einer (komplexen) Funktion über die beste gebrochenrationale Funktion. Was dabei als beste rationale Funktion zu verstehen ist, wird hier ausführlich behandelt.
Betrachtet werden Funktionen mit nur einer komplexen Eingangsvariablen. Obwohl man später eine multivariate Approximationstheorie benötigt, ist dieser Aufbau nötig. Denn manche Resultate lassen sich von der Padé Approximation auf den multivariaten Fall verallgemeinern. Außerdem sind Padé Approximanten immer auch teil der Approximanten höherer Ordnung. Ein guter und ausführlicher Aufbau findet sich hierfür in der Enzyklopädie [Baker et al., 1996].
Zu Beginn wieder eine grundlegende Definition:

Definition 4.14. Sei $D \subset \mathbb{C}$ offen und $0 \in D$. Die Taylorreihenentwicklung einer auf D holomorphen Funktion f mit Entwicklungsstelle $z_0 = 0$:

$$f(z) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} z^n = f(0) + f'(0)z + \frac{1}{2!} f''(0)z^2 + \dots$$

wird als **MacLaurin-Reihe** von f bezeichnet.

In Kapitel 4.1 wurde bereits angesprochen, dass sich jede in $z_0 = 0$ holomorphe Funktion auch gleich beliebig oft in 0 differenzieren lässt, sie ebenfalls in 0 auch analytisch ist und deren Darstellung als Potenzreihe mit der Taylorreihenentwicklung übereinstimmen muss. Die Auffassung von f als Potenzreihe um 0 und deren Einbettung in den Ring der formalen Potenzreihen benötigt man auch für das Nächste:

Definition 4.15. Gegeben sei eine in $z_0 = 0$ holomorphe Funktion f (auf einer offenen Umgebung die 0 enthält) und eine gekürzte (von gemeinsamen Linearfaktoren) rationale Funktion

$$r(z) = \frac{a'_0 + a'_1 z + \dots + a'_L z^L}{b'_0 + b'_1 z + \dots + b'_M z^M} = \frac{a_0 + a_1 z + \dots + a_L z^L}{1 + b_1 z + \dots + b_M z^M} := \frac{A^{[L/M]}(z)}{B^{[L/M]}(z)}.$$

wobei um Eindeutigkeit zu erhalten, Nenner und Zähler durch $b'_0 \neq 0$ dividiert wurden.
(Der Fall $b'_0 = 0$ wird später noch ausführlich behandelt.)
Um die Tatsache hervorzuheben, dass die Ordnung von Zähler- und Nennerpolynom voneinander abhängen und bereits festgelegt sind, schreibt man auch $A^{[L/M]}(z)$ und $B^{[L/M]}(z)$.
Es wird nun r als **Padé-Approximant** der Ordnung (L, M) von f bezeichnet, falls die MacLaurin-Reihe von r mit der von f bis zur Ordnung $L + M$ übereinstimmt. Ist dies der Fall, so schreibt man für $r(z)$ auch kurz $[L/M](z)$.
Betrachtet als formale Potenzreihen soll also gelten, dass

$$f(z) = \sum_{n=0}^{\infty} \underbrace{\frac{f^{(n)}(0)}{n!}}_{:=c_n} z^n = \frac{a_0 + a_1 z + \dots + a_L z^L}{1 + b_1 z + \dots + b_M z^M} + \mathcal{O}(z^{L+M+1}), \tag{1}$$

wobei hier $\mathcal{O}(z^{L+M+1})$ für eine formale Potenzreihe steht, bei der alle Koeffizienten bis zur Ordnung $L + M$ gleich 0 sind.

Aus Korollar 4.13 und $b_0 \neq 0$ folgt, dass sich die Inverse des Nennerpolynoms als formale Potenzreihe darstellen lässt. Also darf dieses mit dem Zählerpolynom (aufgefasst als Potenzreihe) multipliziert werden und man erhält wiederum eine Potenzreihe, weshalb obrige Definition (1) mit

der klassischen Definition (über die Taylorreihe) übereinstimmt. Den Koeffizientenvergleich erlaubt ja Lemma 4.4 und die gliedweise Ableitung.

Bei Multiplikation der Gleichung (1) mit dem Nennerpolynom von r, resultiert

$$(1 + b_1 z + \ldots + b_M z^M)(c_0 + c_1 z + \ldots) = (a_0 + a_1 z + \ldots + a_L z^L) + \mathcal{O}(z^{L+M+1}).$$

Durch einen Koeffizientenvergleich bis zur Ordnung L sieht man also, dass eine rationale Funktion r folgendes Gleichungssystem

$$\begin{aligned} c_0 &= a_0 \\ c_1 + b_1 c_0 &= a_1 \\ c_2 + b_1 c_1 + b_2 c_0 &= a_2 \\ &\vdots \\ c_L + b_1 c_{L-1} + \ldots + b_L c_0 &= a_L \end{aligned} \tag{2}$$

erfüllen muss, um als Kandidat in Frage zu kommen. Für alle $j > M$ sind hier alle $b_j = 0$ (nur im Falle, dass $L > M$ gilt). Somit sind also schon mal $L + 1$ Gleichungen gegeben. Des weiteren muss für die Koeffizienten der Ordnung $L + 1$ bis $L + M$ auch

$$\begin{array}{ccccccccccc} c_{L+1} & + & b_1 c_L & + & b_2 c_{L-1} & + & \ldots & + & b_M c_{L+1-M} & = & 0 \\ c_{L+2} & + & b_1 c_{L+1} & + & b_2 c_L & + & \ldots & + & b_M c_{L+2-M} & = & 0 \\ & & & & & & & & \vdots & & \\ c_{L+M} & + & b_1 c_{L+M-1} & + & b_2 c_{L+M-2} & + & \ldots & + & b_M c_L & = & 0 \end{array} \tag{3}$$

erfüllt sein. Dieses homogene Gleichungssystem (3) ist frei von den Koeffizienten a_j. Kann es gelöst und somit die Koeffizienten b_i bestimmt werden, so muss man nur noch in das erste Gleichungssystem (2) einsetzen und erhält den Padé-Approximanten.
Bringt man die Konstanten auf die rechte Seite, so folgt in Matrixschreibweise:

$$\begin{pmatrix} c_L & c_{L-1} & \ldots & c_{L+1-M} \\ c_{L+1} & c_L & \ldots & c_{L+2-M} \\ c_{L+2} & c_{L+1} & \ldots & c_{L+3-M} \\ \vdots & \vdots & & \vdots \\ c_{L+M-1} & c_{L+M-2} & \ldots & c_L \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_M \end{pmatrix} = - \begin{pmatrix} c_{L+1} \\ c_{L+2} \\ c_{L+3} \\ \vdots \\ c_{L+M} \end{pmatrix} \tag{4}$$

Es wird hier für alle $j < 0$ und auch im Weiteren $c_j = 0$ gesetzt. Nun lässt sich auch leicht charakterisieren, wann dieses $M \times M$ Gleichungssystem eine eindeutige Lösung besitzt und somit auch der Padé-Approximant der Ordnung (L, M) existiert und eindeutig ist. Denn dies ist genau dann der Fall, wenn die Determinante der Matrix aus (4) ungleich 0 ist. Die Eigenschaft, dass der Padé-Approximant gekürzt ist, muss erfüllt sein. Ansonst wäre man durch Kürzen von Nenner und Zähler in der Lage einen weiteren Padé-Approximanten der Ordnung (L,M) zu erzeugen, bei dem die beiden ursprünglichen Führungskoeffizienten zu 0 werden, was im Widerspruch zur Eindeutigkeit aller Koeffizienten wäre.

Definition 4.16. Vertauscht man in (4) die Spalten (beziehungsweise die Reihenfolge der Unbekannten) und setzt die letzte an die erste Stelle, die vorletzte an die zweite Stelle usw. so erhält man eine **Hankel-Matrix**. Das heißt die Einträge der Gegendiagonalen sind alle gleich. Dessen Determinante

$$C(L/M) := \begin{vmatrix} c_{L+1-M} & c_{L+2-M} & c_{L+3-M} & \ldots & c_L \\ c_{L+2-M} & c_{L+3-M} & c_{L+4-M} & \ldots & c_{L+1} \\ c_{L+3-M} & c_{L+4-M} & c_{L+5-M} & \ldots & c_{L+2} \\ \vdots & \vdots & \vdots & & \vdots \\ c_L & c_{L+1} & c_{L+2} & \ldots & c_{L+M-1} \end{vmatrix}$$

wird im Folgenden auch kurz als **Hankel-Determinante** bezeichnet.

Wie in [Jacobi, 1846] beschrieben, gibt es eine Darstellung des Padé-Approximanten über die Koeffizienten der MacLaurin-Reihe. Dabei werden Nenner- und Zählerpolynom als Determinante geeigneter Matrizen beschrieben:

Satz 4.17. Gegeben sei eine in $z_0 = 0$ analytische Funktion $f(z) = \sum_{i=0}^{\infty} c_i z^i$ mit einer von 0 verschiedenen Hankel-Determinante $C(L/M)$. Es folgt, dass sich der Padé-Approximant der Ordnung (L, M) eindeutig darstellen lässt als

$$[L/M] = \frac{A^{[L/M]}(z)}{B^{[L/M]}(z)}$$

mit

$$A^{[L/M]}(z) := \frac{P^{[L/M]}(z)}{Q^{[L/M]}(0)} \qquad \text{und} \qquad B^{[L/M]}(z) := \frac{Q^{[L/M]}(z)}{Q^{[L/M]}(0)}$$

wobei

$$Q^{[L/M]}(z) := \begin{vmatrix} c_{L+1-M} & c_{L+2-M} & \dots & c_{L+1} \\ c_{L+2-M} & c_{L+3-M} & \dots & c_{L+2} \\ \vdots & \vdots & & \vdots \\ c_L & c_{L+1} & \dots & c_{L+M} \\ z^M & z^{M-1} & \dots & 1 \end{vmatrix} \tag{5}$$

und weiters

$$P^{[L/M]}(z) := \begin{vmatrix} c_{L+1-M} & c_{L+2-M} & \dots & c_{L+1} \\ c_{L+2-M} & c_{L+3-M} & \dots & c_{L+2} \\ \vdots & \vdots & & \vdots \\ c_L & c_{L+1} & \dots & c_{L+M} \\ \sum_{i=0}^{L-M} c_i z^{M+i} & \sum_{i=0}^{L-M+1} c_i z^{M+i-1} & \dots & \sum_{i=0}^{L} c_i z^i \end{vmatrix} \tag{6}$$

sind. Es lässt sich auch jeder Eintrag in der C-Tabelle (Definition 4.18) darstellen als $C(L/M) = Q^{[L/M]}(0)$. Schließlich gibt es auch eine praktische Darstellung des Fehlers der MacLaurin-Reihe von $[L/M]$:

$$\sum_{i=0}^{\infty} c_i z^i Q^{[L/M]}(z) - P^{[L/M]}(z) = \sum_{i=1}^{\infty} z^{L+M+i} \begin{vmatrix} c_{L+1-M} & c_{L+2-M} & \dots & c_{L+1} \\ c_{L+2-M} & c_{L+3-M} & \dots & c_{L+2} \\ \vdots & \vdots & & \vdots \\ c_L & c_{L+1} & \dots & c_{L+M} \\ c_{L+i} & c_{L+i+1} & \dots & c_{L+i+M} \end{vmatrix} \tag{7}$$

Beweis. Um für die Existenz der zwei definierten Determinanten nicht über den Konvergenzradius der MacLaurin-Reihe argumentieren zu müssen, wird wieder formal vorgegangen. Dazu fasst man sämtliche Matrix-Einträge als formale Potenzreihen auf. Die aus der linearen Algebra bekannte

Identität

$$\lambda \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & \dots & \vdots \\ a_{j1} & a_{j2} & \dots & a_{jn} \\ \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & \dots & \vdots \\ \lambda a_{j1} & \lambda a_{j2} & \dots & \lambda a_{jn} \\ \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}$$

mit $n \in \mathbb{N}$, $\lambda \in \mathbb{C}$ und $a_{ij} \in \mathbb{C}$ für alle $i, j \in \{1, 2, \dots, n\}$, um eine beliebige Zeile $j \leq n$ zu behandeln, darf auch hier benutzt werden. Angewendet auf die letzte Zeile von $Q^{[L/M]}(z)$ folgt

$$\sum_{i=0}^{\infty} c_i z^i Q^{[L/M]}(z) = \begin{vmatrix} c_{L+1-M} & c_{L+2-M} & \dots & c_{L+1} \\ c_{L+2-M} & c_{L+3-M} & \dots & c_{L+2} \\ \vdots & \vdots & & \vdots \\ c_L & c_{L+1} & \dots & c_{L+M} \\ \sum_{i=0}^{\infty} c_i z^{i+M} & \sum_{i=0}^{\infty} c_i z^{i+M-1} & \dots & \sum_{i=0}^{\infty} c_i z^i \end{vmatrix}.$$

Betrachtet man im Weiteren die Differenz

$$\underbrace{\sum_{i=0}^{\infty} c_i z^i Q^{[L/M]}(z)}_{=:Q} - \underbrace{P^{[L/M]}(z)}_{=:P}$$

und entwickelt beide Matrizen Q und P nach der letzten Zeile, so resultiert

$$\sum_{j=1}^{M+1} (-1)^{M+j} \left(\sum_{i=0}^{\infty} c_i z^{M+i+1-j} |Q_j| \right) - \sum_{j=1}^{M+1} (-1)^{M+j} \left(\sum_{i=0}^{L-M+j-1} c_i z^{M+i+1-j} |P_j| \right),$$

wobei hier Q_j bzw. P_j die Matrizen sind, die durch Streichen der j-ten Spalte und der letzten Zeile von Q bzw. P hervorgehen. Man sieht sofort, dass $Q_j = P_j$ für alle $j \in \{1, \dots, M+1\}$ gilt. Beide Summen zusammengefasst und mit einem Indexshift um $L - M + j$ ergibt

$$\sum_{j=1}^{M+1} (-1)^{M+j} \left(\sum_{i=L-M+j}^{\infty} c_i z^{M+i+1-j} |Q_j| \right) = \sum_{j=1}^{M+1} (-1)^{M+j} \left(\sum_{i=0}^{\infty} c_{L-M+j+i} z^{i+L+1} |Q_j| \right).$$

Anschließend lässt sich durch eine Rückentwicklung wieder schreiben

$$Q - P = \begin{vmatrix} c_{L+1-M} & c_{L+2-M} & \dots & c_{L+1} \\ c_{L+2-M} & c_{L+3-M} & \dots & c_{L+2} \\ \vdots & \vdots & & \vdots \\ c_L & c_{L+1} & \dots & c_{L+M} \\ \sum_{i=0}^{\infty} c_{L-M+i+1} z^{L+i+1} & \sum_{i=0}^{\infty} c_{L-M+i+2} z^{L+i+1} & \dots & \sum_{i=0}^{\infty} c_{L+i+1} z^{L+i+1} \end{vmatrix}.$$

Als weiterer Schritt wird nun das z^{L+1}-fache der ersten Zeile von der letzten subtrahiert. Ebenfalls wird auch das z^{L+2}-fache der zweiten Zeile von der letzen subtrahiert. Analog fährt man so bis zur vorletzten Zeile fort und erhält:

$$Q - P = \begin{vmatrix} c_{L+1-M} & c_{L+2-M} & \dots & c_{L+1} \\ c_{L+2-M} & c_{L+3-M} & \dots & c_{L+2} \\ \vdots & \vdots & & \vdots \\ c_L & c_{L+1} & \dots & c_{L+M} \\ \sum_{i=M}^{\infty} c_{L-M+i+1} z^{L+i+1} & \sum_{i=M}^{\infty} c_{L-M+i+2} z^{L+i+1} & \dots & \sum_{i=M}^{\infty} c_{L+i+1} z^{L+i+1} \end{vmatrix}$$

Abschließend wird noch ein Indexshift aller Summationsindizes der letzten Zeile um $M - 1$ durchgeführt und durch einen Koeffizientenvergleich ergibt sich die Gleichung (7). Man sieht also, dass

die Differenz $Q - P$ aus $\mathcal{O}(z^{L+M+1})$ ist, was schließlich gezeigt werden sollte.

Weiters ist ersichtlich, dass die für $Q^{[L/M]}(z)$ verwendete Matrix aus der weiter oben definierten Hankel-Matrix (Definition 4.16) um die Erweiterung einer zusätzlichen Zeile und Spalte hervorgeht. Durch das Setzen von $z = 0$ und entwickeln nach der letzten Zeile erhält man $Q^{[L/M]}(0) = C(L/M) = b'_0$. Dass zu Beginn die Hankel-Determinante ungleich 0 gefordert wurde, stellt also auch die Eindeutigkeit von $Q^{[L/M]}(z)$ sicher und weiters auch die von $[L/M]$. ■

Dieser Satz gibt Auskunft, wann ein eindeutiger Padé-Approximant einer analytischen Funktion existiert und wie er aus dessen MacLaurin-Reihe gebildet werden kann. Die Frage, die nun noch gestellt werden muss, ist: Gibt es für $Q^{[L/M]}(0) = 0$ keine, genau eine oder mindestens eine Lösung? Macht man sich zuerst noch einmal klar, dass $Q^{[L/M]}(0) = 0$ auch bedeutet $b'_0 = 0$. Somit kommt man zu dem in Definition 4.15 versprochenen Sonderfall. Hierfür erst einmal folgende Definition:

Definition 4.18. Seien im Folgenden $C[L/M] := Q^{[L/M]}(0)$ die bereits weiter oben definierten Hankel-Determinanten einer in 0 analytischen Funktion f. Die sogenannte **C-Tabelle**:

M\L	0	1	2	…
0	$C[0/0]$	$C[1/0]$	$C[2/0]$	…
1	$C[0/1]$	$C[1/1]$	$C[2/1]$	…
2	$C[0/2]$	$C[1/2]$	$C[2/2]$	…
⋮	⋮	⋮	⋮	⋱

soll es nun vereinfachen die Existenz von Padé-Approximanten von f festzustellen.

Die erste Spalte und die ersten beiden Zeilen lassen sich einfach bestimmen, woraus sich dann später rekursiv die restliche Tabelle befüllen lässt.

Lemma 4.19. Folgende Aussagen sind in der C-Tabelle gültig:

(1) Die erste Zeile ist konstant 1. Das heißt $C[L/0] = 1$ für alle $L \geq 0$.

(2) Für die zweite Zeile gilt für alle $L \geq 0$, dass $C[L/1] = c_L$.

(3) Die Einträge der ersten Spalte sind $C[0/M] = (-1)^{M(M-1)/2} c_0^M$ für alle $M \geq 0$.

Beweis.

(1) Es ist ersichtlich, dass für $M = 0$ und $L \geq 0$ das Gleichunggsystem (3) wegfällt und sich Gleichungssystem (2) vereinfacht zu

$$\begin{aligned} c_0 &= a_0 \\ c_1 &= a_1 \\ &\vdots \\ c_L &= a_L \end{aligned}$$

also für $b'_0 = 1$ die Bedingung (1) eines Padé-Approximanten erfüllt ist.

(2) Betrachtet man die zweite Zeile und ein beliebiges L, so folgt aus

$$Q^{[L/1]}(0) = \begin{vmatrix} c_L & c_{L+1} \\ 0 & 1 \end{vmatrix} = c_L$$

dass $C[L/1] = c_L$ für alle $L \geq 0$ gilt.

(3) Der Beweis wird über vollständige Induktion geführt. Für $M = 0$ wurde schon gezeigt, dass $C[0/0] = 1$, was mit $(-1)^0 c_0^0$ übereinstimmt. Der Induktionsanfang für $M = 1$ sei gegeben durch

$$C[0/1] = |c_0| = c_0$$

was mit $(-1)^0 c_0^1$ und dem für Zeile 2 gezeigten übereinstimmt.
Sei nun die Voraussetzung für die $(M-1)$-te Zeile erfüllt:

$$C[0/M-1] = \begin{vmatrix} c_{-(M-1)+1} & c_{-(M-1)+2} & \cdots & c_0 \\ c_{-(M-1)+2} & c_{-(M-1)+3} & \cdots & c_1 \\ \vdots & \vdots & & \vdots \\ c_{-1} & c_0 & \cdots & c_{M-3} \\ c_0 & c_1 & \cdots & c_{M-2} \end{vmatrix} = (-1)^{(M-1)(M-2)/2} c_0^{M-1}$$

Für den Induktionsschritt nach M wird auf $C[0/M]$ der Laplace'sche Entwicklungssatz nach der letzten Spalte angewendet

$$C[0/M] = \begin{vmatrix} c_{-M+1} & c_{-M+2} & \cdots & c_0 \\ c_{-M+2} & c_{-M+3} & \cdots & c_1 \\ \vdots & \vdots & & \vdots \\ c_{-1} & c_0 & \cdots & c_{M-2} \\ c_0 & c_1 & \cdots & c_{M-1} \end{vmatrix} = \begin{vmatrix} 0 & & & \\ \vdots & & C[0/M-1] & \\ 0 & & & \\ c_0 & c_1 & \cdots & c_{M-2} \end{vmatrix}$$
$$= (-1)^{M-1} c_0 C[0/M-1] = (-1)^{M(M-1)/2} c_0^M$$

und man hat somit auch den letzten Punkt bewiesen.

■

Es gibt weiters eine praktische Formel, die wir zum Befüllen der C-Tabelle verwenden können. Die sogenannte Frobenius Formel

$$C(L/M+1) = \frac{C(L+1/M)C(L-1/M) - C(L/M)^2}{C(L/M-1)} \tag{8}$$

ist immer gültig, sofern für $C(L/M-1) \neq 0$ gilt. Diese geht unmittelbar aus einem Satz von Sylvester hervor, der die Determinante einer quadratischen Matrix A über die Determinanten mehrerer Teilmatrizen über das Löschen zweier Zeilen und Spalten aus A angibt. Veranschaulicht kann die Frobenius Formel als eine Stern-Identität

$$\begin{bmatrix} & * & \\ * & * & * \\ & * & \end{bmatrix}$$

betrachtet werden. Interessant ist nun auch, dass sich mit deren Hilfe die 0 Einträge in einer C-Tabelle als ganz spezielle Blöcke identifizieren lassen. Dies soll im Folgenden zuerst an zwei kleinen Blöcken (mit 3×3 und 4×4) veranschaulicht und schließlich dann auch verallgemeinert werden:

$C[\lambda/\mu]$ $\neq 0$	$C[\lambda+1/\mu]$ $\neq 0$	$C[\lambda+2/\mu]$ $\neq 0$
$C[\lambda/\mu+1]$ $\neq 0$	0	$C[\lambda+2/\mu+1]$ $\neq 0$
$C[\lambda/\mu+2]$ $\neq 0$	$C[\lambda+1/\mu+2]$ $\neq 0$	$C[\lambda+2/\mu+2]$ $\neq 0$

Nimmt man an $C[\lambda+1/\mu] \neq 0$, $C[\lambda/\mu+1] \neq 0$ und $C[\lambda+1,\mu+1] = 0$ seien bereits in obiger C-Tabelle bestimmt worden. Ein weiterer Eintrag, wie zum Beispiel der, dass auch $C[\lambda+1/\mu+2] \neq 0$ sei, gibt nun die Möglichkeit einer ersten Anwendung der Frobenius Formel:

$$\begin{bmatrix} & * & \\ * & 0 & ? \\ & * & \end{bmatrix}$$

Diese liefert schließlich, dass $C[\lambda+2/\mu+1] = C[\lambda+1/\mu+2]C[\lambda+1/\mu]/C[\lambda/\mu+1]$ sein muss und da alle drei Werte auf der rechten Seite ungleich 0 sind gilt auch $C[\lambda+2/\mu+1] \neq 0$.

Es wird weiters angenommen, dass die drei oben definierten Einträge bekannt sind. Dieses Mal jedoch geht man von $C[\lambda+2/\mu+1] \neq 0$ aus und möchte das untere Element bestimmen:

$$\begin{bmatrix} & * & \\ * & 0 & * \\ & ? & \end{bmatrix}$$

Die Formel von Frobenius liefert schließlich wieder $C[\lambda+1/\mu+2] = C[\lambda+2/\mu+1]C[\lambda/\mu+1]/C[\lambda+1/\mu] \neq 0$.
Ersichtlich ist bereits jetzt, dass eine mögliche 0er Zeile oder 0er Spalte immer nur aus einem 0 Element bestehen kann. Genau genommen müssen sogar alle Einträge um diese 0 ungleich 0 sein, aber dies wird noch behandelt. Eine Bemerkung soll hier auch noch erfolgen, nämlich dass für einen einzelnen 0 Eintrag niemals ein Padé-Approximant dieser Ordnung existiert.

Betrachtet man nun folgenden Ausschnitt einer C-Tabelle:

$C[\lambda/\mu]$ $\neq 0$	$C[\lambda+1/\mu]$ $\neq 0$	$C[\lambda+2/\mu]$ $\neq 0$	$C[\lambda+3/\mu]$ $\neq 0$
$C[\lambda/\mu+1]$ $\neq 0$	0	0	$C[\lambda+3/\mu+1]$ $\neq 0$
$C[\lambda/\mu+2]$ $\neq 0$	0	0	$C[\lambda+3/\mu+2]$ $\neq 0$
$C[\lambda/\mu+3]$ $\neq 0$	$C[\lambda+1/\mu+3]$ $\neq 0$	$C[\lambda+2/\mu+3]$ $\neq 0$	$C[\lambda+3/\mu+3]$ $\neq 0$

Eine interessante Tatsache ist, dass alle Padé-Approximanten der ersten Spalte des Blocks mit rechts angrenzender 0 (also bei $C[\lambda/\mu+1]$ und $C[\lambda/\mu+2]$) mit dem Padé-Approximanten der Ordnung (λ/μ) übereinstimmen. Analoges gilt auch für die erste Zeile des Blocks mit darunter angrenzenden Nullen und dass sie mit dem Padé-Approximanten der Ordnung (λ/μ) übereinstimmen. Obwohl die Hankel-Matrix für $C[\lambda+1/\mu+1]$ singulär ist, gibt es dennoch einen Padé-Approximanten dieser Ordnung, der ebenfalls mit dem von (λ/μ) übereinstimmt. Für die anderen drei Nulleinträge existieren jedoch keine.

Satz 4.20. Die 0 Einträge in der C-Tabelle sind als quadratische Blöcke vertreten. Um diesen Block sind alle angrenzenden Elemente ungleich 0, was veranschaulicht wieder einen quadratischen Block bildet.
Allerdings können auch beliebig große Blöcke von 0 Einträgen auftreten, die somit nur von links und von oben begrenzt werden.

Beweis. Der klassische Beweis verwendet großteils die Sternidentität (8). Ähnlich zu den oben betrachteten Beispielen wird damit in der C-Tabelle von links oben nach rechts unten vorgegangen. [Baker et al., 1996] ■

Hier ist darauf zu achten, dass die Randeinträge ungleich 0 sind. Es kann zum Beispiel vorkommen, dass die linke Spalte bereits mit Nullen initialisiert wird und dennoch kann nicht auf die Einträge rechts davon geschlossen werden. Siehe als Beispiel auch Kapitel 5.1 Tabelle 3.

Für einen 3×3 und 4×4 Block ist bereits erwähnt worden, für welche 0 Einträge dennoch ein Padé-Approximant existiert. Für einen Block beliebiger Größe wird dies im Folgenden nun ganz allgemein dargestellt:

$C[\lambda/\mu]$ $\neq 0$	$C[\lambda+1/\mu]$ $\neq 0$	$C[\lambda+2/\mu]$ $\neq 0$	$\cdots$	$C[\lambda+r-1/\mu]$ $\neq 0$	$C[\lambda+r/\mu]$ $\neq 0$
$C[\lambda/\mu+1]$ $\neq 0$	0 konsistent	$\ddots$	0 konsistent	0 inkonsistent	$C[\lambda+r/\mu+1]$ $\neq 0$
$C[\lambda/\mu+2]$ $\neq 0$	$\ddots$	0 konsistent	0 inkonsistent	0 inkonsistent	$C[\lambda+r/\mu+2]$ $\neq 0$
$\vdots$	0 konsistent	0 inkonsistent	0 inkonsistent	$\ddots$	$\vdots$
$C[\lambda/\mu+r-1]$ $\neq 0$	0 inkonsistent	0 inkonsistent	$\ddots$	0 inkonsistent	$\vdots$
$C[\lambda/\mu+r]$ $\neq 0$	$C[\lambda+1/\mu+r]$ $\neq 0$	$C[\lambda+2/\mu+r]$ $\neq 0$	$\cdots$	$\cdots$	$C[\lambda+r/\mu+r]$ $\neq 0$

Tabelle 2: Ausschnitt aus C-Tabelle mit Block an Nullen

Konsistent bedeutet hierbei, obwohl die Koeffizientenmatrix aus Gleichungssystem (4) singulär ist, existiert eine Lösung und der resultierende Padé-Approximant stimmt mit dem von (λ/μ) überein. Für eine inkonsistente Koeffizientenmatrix existiert auch kein Padé-Approximant.
Der folgende Satz von Baker beschreibt dies noch genauer und erweitert den vorigen Satz 4.20 und somit auch den in [Padé, 1892].

Satz 4.21. Gegeben sei eine in $z = 0$ analytische Funktion $f(z) = \sum_{i=0}^{\infty} c_i z^i$ und ein in dessen C-Tabelle liegender $(r-1) \times (r-1)$ Block an Nullen mit $r > 1$ (wie in Tabelle 2). Für die Hankel-Determinanten des umschließenden $(r+1) \times (r+1)$ Blocks soll also gelten

$$\begin{aligned} C[\lambda/\mu] &\neq 0 \quad && \text{mit } \lambda, \mu \in \mathbb{N} \text{ fest gewählt} && \text{(1. Element)} \\ C[\lambda + i/\mu] &\neq 0 \quad && \text{für alle } i \in \{1, \ldots, r\} && \text{(1. Zeile)} \\ C[\lambda/\mu + j] &\neq 0 \quad && \text{für alle } j \in \{1, \ldots, r\} && \text{(1. Spalte)} \end{aligned}$$

im Inneren

$$C(\lambda + i/\mu + j) = 0 \qquad \text{für alle } i, j \in \{1, \ldots, r-1\}$$

und für ein fehlendes Randelement

$$C[\lambda + i/\mu + r] \neq 0 \qquad \text{für ein } i \in \{1, \dots, r\}$$
$$\text{oder}$$
$$C[\lambda + r/\mu + j] \neq 0 \qquad \text{für ein } j \in \{1, \dots, r\}.$$

Dann folgt für die zu dem $(r-1) \times (r-1)$ Null-Block gehörenden Padé-Approximanten

$[L/M] = [\lambda/\mu]$	für alle $L + M \leq \lambda + \mu + (r-1)$	(Gegendiagonale exklusiv)
$[L/M]$ existiert nicht	für alle $L + M > \lambda + \mu + (r-1)$	(Gegendiagonale inklusiv).

Beweis. Der Beweis erfolgt in zwei Teilen:
1.Teil:
Zu Beginn wird gleich gezeigt, dass

$$C[\lambda/\mu] = [\lambda + \ell/\mu] \qquad \text{für alle } \ell \in \{1, 2, \dots, r-1\}$$
$$\text{und}$$
$$C[\lambda/\mu] = [\lambda/\mu + m] \qquad \text{für alle } m \in \{1, 2, \dots, r-1\}$$

für die erste Zeile und Spalte (ohne Randelemente) gilt. Dies soll über vollständige Induktion nach dem Spaltenindex m bzw. Zeilenindex ℓ erfolgen.
Sei also $m' \in \{1, 2, \dots, r-1\}$ beliebig gewählt. So erlaubt es die Gleichung (5) aus Satz 4.17 den Nenner $Q^{[\lambda/\mu+m']}$ des Padé-Approximanten $[\lambda/\mu + m']$ (dieser existiert ja da per Definition $C[\lambda/\mu + m'] \neq 0$ ist) darzustellen als

$$Q^{[\lambda/\mu+m']}(z) = \begin{vmatrix} c_{\lambda-(\mu+m')+1} & c_{\lambda-(\mu+m')+2} & \cdots & c_{\lambda+1} \\ c_{\lambda-(\mu+m')+2} & c_{\lambda-(\mu+m')+3} & \cdots & c_{\lambda+2} \\ \vdots & \vdots & & \vdots \\ c_{\lambda} & c_{\lambda+1} & \cdots & c_{\lambda+(\mu+m')} \\ z^{(\mu+m')} & z^{(\mu+m')-1} & \dots & 1 \end{vmatrix}.$$

Eine Laplace Entwicklung nach der letzten Zeile liefert (Notation wie im Beweis von Satz 4.17)

$$Q^{[\lambda/\mu+m']}(z) = \sum_{i=0}^{\mu+m'} \pm z^i \left| Q_{(\mu+m')+1-i} \right|,$$

wobei es hier nicht auf das Vorzeichen ankommt. Da alle $\left|Q_{(\mu+m')+1-i}\right|$ mit $i \in \{0, 1, \dots, \mu + m'\}$ konstant sind folgt, dass sich der $(\mu+m')$-te Koeffizient ergibt zu $\pm |Q_1|$. Betrachtet man Definition 4.16 und daraus folgend

$$C(\lambda + 1/\mu + m') = \begin{vmatrix} c_{(\lambda+1)+1-(\mu+m')} & c_{(\lambda+1)+2-(\mu+m')} & \cdots & c_{(\lambda+1)} \\ c_{(\lambda+1)+2-(\mu+m')} & c_{(\lambda+1)+3-(\mu+m')} & \cdots & c_{(\lambda+1)+1} \\ \vdots & \vdots & \vdots & \vdots \\ c_{(\lambda+1)} & c_{(\lambda+1)+1} & \cdots & c_{(\lambda+1)+(\mu+m')-1} \end{vmatrix},$$

so ist sofort ersichtlich, dass $|Q_1|$ mit $\pm C(\lambda + 1/\mu + m')$ übereinstimmt. Da per Definition gilt, dass $C(\lambda+1/\mu+m') = 0$ muss der Nenner $Q^{[\lambda/\mu+m']}(z)$ einen Grad (echt) kleiner $\mu+m'$ aufweisen.

Für $m' = 1$ (Induktionsanfang) folgt also, dass $[\lambda/\mu + 1](z)$ vom selben Grad (λ/μ) wie $[\lambda/\mu](z)$ ist oder sogar kleiner. Aufgrund der Eindeutigkeit von $[\lambda/\mu](z)$ müssen die beiden aber übereinstimmen (siehe Definition 4.15 und Bemerkung über Eindeutigkeit darunter).
Für den Induktionsschritt von $[\lambda/\mu+m'] = [\lambda/\mu+m'-1]$ kann ganz analog argumentiert werden.

(Die Induktionsannahme besagt ja, dass $[\lambda/\mu + m' - 1]$ existiert und mit $[\lambda/\mu + m' - 2]$ übereinstimmt.)

Der Beweis für die erste Spalte ist also erbracht. Sei nun $\ell' \in \{1, 2, \ldots, r-1\}$ für ein beliebiges Element $[\lambda + \ell'/\mu]$ aus der ersten Zeile gewählt. Aus Gleichung (6) folgt für dessen Zähler

$$P^{[\lambda+\ell'/\mu]}(z) = \begin{vmatrix} c_{(\lambda+\ell')+1-\mu} & c_{(\lambda+\ell')+2-\mu} & \cdots & c_{(\lambda+\ell')+1} \\ c_{(\lambda+\ell')+2-\mu} & c_{(\lambda+\ell')+3-\mu} & \cdots & c_{(\lambda+\ell')+2} \\ \vdots & \vdots & & \vdots \\ c_{(\lambda+\ell')} & c_{(\lambda+\ell')+1} & \cdots & c_{(\lambda+\ell')+\mu} \\ \sum_{i=0}^{(\lambda+\ell')-\mu} c_i z^{\mu+i} & \sum_{i=0}^{(\lambda+\ell')-\mu+1} c_i z^{\mu+i-1} & \cdots & \sum_{i=0}^{(\lambda+\ell')} c_i z^i \end{vmatrix}$$

und erneutem Entwickeln nach der letzten Zeile für den $(\lambda + \ell')$-ten Koeffizienten

$$[z^{\lambda+\ell'}]P^{[\lambda+\ell'/\mu]}(z) = \pm \left(c_{(\lambda+\ell')-\mu} \, |P_1| - c_{(\lambda+\ell')-\mu+1} \, |P_2| + \ldots \pm c_{(\lambda+\ell')} \, |P_{\mu+1}| \right)$$

und einer Rückentwicklung ähnlich wie im Beweis von Satz 4.17 wieder

$$[z^{\lambda+\ell'}]P^{[\lambda+\ell'/\mu]}(z) = \pm \begin{vmatrix} c_{(\lambda+\ell')+1-\mu} & c_{(\lambda+\ell')+2-\mu} & \cdots & c_{(\lambda+\ell')+1} \\ c_{(\lambda+\ell')+2-\mu} & c_{(\lambda+\ell')+3-\mu} & \cdots & c_{(\lambda+\ell')+2} \\ \vdots & \vdots & & \vdots \\ c_{(\lambda+\ell')} & c_{(\lambda+\ell')+1} & \cdots & c_{(\lambda+\ell')+\mu} \\ c_{(\lambda+\ell')-\mu} & c_{(\lambda+\ell')-\mu+1} & \cdots & c_{\lambda+\ell'} \end{vmatrix} \tag{9}$$

und bringt die letzte Zeile an die erste Stelle (vertauschen der Zeilen ändert die Determinante ja nicht), so erhält man die Übereinstimmung aus Definition 4.16 mit

$$C(\lambda + \ell'/\mu + 1) = \begin{vmatrix} c_{(\lambda+\ell')+1-(\mu+1)} & c_{(\lambda+\ell')+2-(\mu+1)} & \cdots & c_{(\lambda+\ell')} \\ c_{(\lambda+\ell')+2-(\mu+1)} & c_{(\lambda+\ell')+3-(\mu+1)} & \cdots & c_{(\lambda+\ell')+1} \\ \vdots & \vdots & \vdots & \vdots \\ c_{(\lambda+\ell')} & c_{(\lambda+\ell')+1} & \cdots & c_{(\lambda+\ell')+(\mu+1)-1} \end{vmatrix} \tag{10}$$

was per Definition des inneren Blocks ja wieder gleich 0 ist. Das heißt es kann nun analog wie bei der vollständigen Induktion nach dem Spaltenindex m vorgegangen werden.

2.Teil:
Für den letzten Padé-Approximanten in der ersten Zeile $[\lambda + r/\mu](z)$ folgt aus den Gleichungen (9) und (10) für $\ell' = r$ und $C(\lambda + r/\mu + 1) \neq 0$, dass der Führungskoeffizient $[z^{\lambda+r}]P[\lambda + r/\mu](z) \neq 0$ ist. Die Erlaubnis zur Verwendung von $C(\lambda + r/\mu + 1) \neq 0$ folgt aus Satz 4.20.

Da nun $C(\lambda + r/\mu) \neq 0$ ist und somit ein eindeutiger Padé-Approximant der Ordnung $(\lambda + r/\mu)$ existiert, der aufgrund $[z^{\lambda+r}]P[\lambda/\mu](z) = 0$ nicht mit $[\lambda/\mu](z)$ übereinstimmen kann, ist also die Gleichung (1) durch die Koeffizienten

$$[z^{\lambda+r+\mu}]f(z) \neq [z^{\lambda+r+\mu}][\lambda/\mu](z) \tag{11}$$

nicht mehr erfüllt. (Bis zur Ordnung $\lambda + (r-1) + \mu$ musste sie ja noch übereinstimmen, da laut 1. Teil ja $[\lambda/\mu] = [\lambda + 1/\mu] = \ldots = [\lambda + (r-1)/\mu]$ gilt).

Sollte nun für ein L mit $\lambda + 1 \leq L \leq \lambda + (r-1)$ und ein M mit $\mu + 1 \leq M \leq \mu + (r-1)$ der Padé-Approximant $[L/M]$ dennoch existieren und die explizite Lösung von (2) und (3) sein, so muss sich dieser auch kürzen lassen. Dies folgt daraus, dass $C(L/M) = 0 = Q^{[L/M]}(0)$ (also $b_0 = 0$ von $[L/M](z)$) und $[L/M]$ laut Definition (1) anders kein Padé-Approximant sein könnte. Somit gilt also durch Kürzen (nur eines Linearfaktors) des Nenners und Zählers, dass $[L/M] = [(L-1)/(M-1)]$ ist. Analog kann für $[L - 1/M - 1]$ vorgegangen werden, bis man schließlich in der 1. Zeile oder 1. Spalte ankommt und aus dem 1. Teil folgt dass $[L/M] = [\lambda/\mu]$ ist.
Da nun $[\lambda/\mu](z)$ als Potenzreihe bis zur Ordnung $\lambda + (r-1) + \mu$ mit der von $f(z)$ übereinstimmt,

ist das für alle $[L/M](z)$ mit $L+M \leq \lambda+\mu+(r-1)$ der Fall. Für $L+M > \lambda+\mu+(r-1)$ existieren also keine Padé-Approximanten, da ja (11) gilt. ■

Korollar 4.22. Die C-Tabelle einer rationalen Funktion der Ordnung (L, M) besitzt einen unbeschränkten Block an Nullen der bei $C[L+1/M+1]$ beginnt.
Die betroffenen Padé-Approximanten stimmen alle mit $[L/M]$ überein.

Beweis. Für jedes $\lambda, \mu \geq 1$ folgt, dass die Hankel-Determinante $C(L+\lambda/M+\mu) = 0$ ist.
Es wurde auch bereits festgestellt, sofern ein Padé-Approximant gewisser Ordnung existiert, dass dieser dann auch eindeutig ist. Da nun für sämtliche $[L+\lambda/M+\mu]$ mit $\lambda, \mu \geq 0$ die rationale Funktion $[L/M]$ selbst auch die Definition 4.15 eines Padé-Approximanten der Ordnung $(L+\lambda/M+\mu)$ erfüllt, ist der Beweis auch schon gebracht. ■

Die Theoreme haben gezeigt, wie mit Hilfe der C-Tabelle Rückschlüsse auf die Existenz und Form von Padé-Approximanten gezogen werden können. Zu guter Letzt folgt noch ein Satz, der später noch eine konvergente Folge von Padé-Approximanten sicherstellt:

Satz 4.23. Gegeben sei eine MacLaurin-Reihe $\Sigma_{n=0}^{\infty} c_n z^n$. In dessen Padé-Tabelle existieren unendlich viele Approximanten

(1) in jeder Zeile $[L/M]$ mit L fest gewählt und $M \to \infty$

(2) in jeder Spalte $[L/M]$ mit M fest gewählt und $L \to \infty$

(3) in jeder Nebendiagonalen $[M+J/M]$ mit J fest gewählt und $M \to \infty$

Beweis.

(1) Dies ist über einen Widerspruch leicht zu zeigen. Angenommen es gäbe ein M' so, dass für alle $[L/M]$ mit $M > M'$ kein Padé-Approximant mehr existiert. Satz 4.21 besagt nun, dass diese Kandidaten in einem Block liegen müssen. Endlich kann dieser nicht sein, da sonst ein $M'' > M'$ am Rand des Blocks gewählt werden kann für das $[L/M'']$ existiert. Unbeschränkt kann der Block auch nicht sein, da aus Korollar 4.22 folgt, dass alle $[L/M]$ mit $M > M'$ existieren. Also existieren in jeder Zeile unendlich viele Approximanten.

(2) Der Beweis für Spalten ist analog zu dem mit Zeilen. Es muss nur die Indizierung nach L erfolgen.

(3) Genauso wie im Beweis für Zeilen, denn jeder endliche Block kann auch diagonal durchlaufen werden und würde einen Approximanten liefern.

■

4.3 Kettenbruchdarstellung

4.3.1 Eine allgemeine Betrachtung

In diesem Kapitel wird der Zusammenhang von rationalen Funktionen und Kettenbrüchen hergestellt. Die Theorie der Kettenbrüche ist schon sehr alt und vor allem ist die Padé-Approximation aus ihr gereift und nicht wie die Gliederung dieser Arbeit vermuten lässt etwa umgekehrt. Nur als kleines Beispiel, wie lange das schon zurückreicht, sei nur Rafael Bombelli [Bombelli, 1579] und seine 1572 erschienene "Algebra" erwähnt. Er befasste sich unter anderem mit der Kettenbruchdarstellung von Quadratwurzeln wie

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \ddots}}}$$

Bombelli selbst soll laut [Strick, 2016] auf ein Exemplar der Arithmetica des Diophant, aus der Bibliothek des Vatikans zurückgegriffen haben. Diophantos von Alexandria gilt als der bedeutendste Algebraiker der Antike und lebte um 250 nach Christus.
Aber was macht nun Kettenbrüche so besonders? Laut [Sauer, 2005] sollen nicht nur die alten Griechen reelle Zahlen (im speziellen Irrationale) über Kettenbrüche definiert haben, eine reelle Zahl lässt sich auch "genauer/leichter" über einen Kettenbruch darstellen als über rationale Zahlen oder später noch anstatt rationaler Funktionen für meromorphe Funktionen. Bevor dies vonstattengeht, erstmals folgende Definitionen.

Definition 4.24. Als **Integritätsbereich/Integritätsring** R wird ein nullteilerfreier kommutativer Ring mit Einselement bezeichnet. Nullteilerfrei bedeutet für alle $a, b \in R$ mit $ab = 0$ folgt stets, dass entweder $a = 0$ oder $b = 0$ ist.
Erweitert man den Integritätsbereich R nun auch noch um eine Bewertungsfunktion $g : R \setminus \{0\} \to \mathbf{N}$ mit der Eigenschaft

- Für alle $a, b \in R$, $b \neq 0$ existieren $q, r \in R$, so dass $a = qb + r$ mit $r = 0$ oder $g(r) < g(a)$,

so sprechen wir von einem **euklidischen Ring**. Es wird hier g auch als euklidische Bewertungsfunktion bezeichnet und deren Eigenschaft ist genau das, was man als "Division mit Rest" auffasst.

Die Menge der ganzen Zahlen $\mathbb{Z}$ ist zusammen mit der Addition und der Multiplikation ein Integritätsbereich. Nimmt man auch noch die Betragsfunktion $|.| : \mathbf{Z} \to \mathbf{N}$ zur Bewertung hinzu, so ergibt sich auch schon ein euklidischer Ring.
Ein euklidischer Ring ist nun der natürliche Rahmen, um die Existenz von Kettenbrüchen sicherzustellen. Betrachtet man als Beispiel $a = 1071$ und $b = 1029$. Es soll nun der Quotient a/b als Kettenbruch dargestellt werden. Man sieht sofort dass b in a genau einmal enthalten ist und somit

$$1071 = 1 \cdot 1029 + 42 \Rightarrow \frac{1071}{1029} = 1 + \frac{42}{1029} = 1 + \frac{1}{\frac{1029}{42}}.$$

Wird das wiederum analog für den Nenner auf der rechten Seite durchgeführt

$$1029 = 24 \cdot 42 + 21 \Rightarrow \frac{1029}{42} = 24 + \frac{21}{42} = 24 + \frac{1}{\frac{42}{21}}$$

und schließlich noch ein letztes Mal

$$42 = 2 \cdot 21 + 0 \Rightarrow \frac{42}{21} = 2,$$

so erhält man durch rekursives Einsetzen

$$\frac{1071}{1029} = 1 + \frac{1}{24 + \frac{1}{2}}.$$

Man sieht sofort, dass dieses Vorgehen nichts anderes als der moderne euklidische Algorithmus ist. Da Euklid von Alexandria bereits im 3. Jahrhundert vor Christus gelebt hatte, geht der Beginn der Kettenbruchtheorie also noch weiter zurück als zuvor erwähnt.

Definition 4.25. Gegeben sei ein euklidischer Ring R mit Bewertungsfunktion g und den Elementen $a, b \in R$. Der **moderne euklidische Algorithmus** liefert $ggT(a, b)$.
Zur Erinnerung, man sagt ein $u \in R$ teilt ein $w \in R$, wenn ein $v \in R$ existiert mit $uv = w$. Auch kurz nur $u|w$. Ein größter gemeinsamer Teiler von a und b (kurz $ggT(a, b)$) ist ein $c \in R$ das $c|a$ und $c|b$ erfüllt und für jedes weitere Element $d \in R$ mit dieser Eigenschaft $d|c$ gilt.
Die Vorgehensweise ist nun wie folgt:
Man definiert einfachheitshalber $r_0 := b$ und findet

$$a = q_1 r_0 + r_1 \qquad \text{mit} \qquad 0 \leq g(r_1) < g(r_0).$$

Solche $q_1, r_1 \in R$ müssen ja existieren, da R ein euklidischer Ring ist. Analog wird für die r_i's so lange vorgegangen, bis man ein r_n mit $g(r_n) = 0$ erhält:

$$\begin{aligned}
r_0 &= q_2 r_1 + r_2 && \text{mit} && 0 \leq g(r_2) < g(r_1) \\
r_1 &= q_3 r_2 + r_3 && \text{mit} && 0 \leq g(r_3) < g(r_2) \\
&\vdots \\
r_{n-2} &= q_n r_{n-1} + r_n && \text{mit} && 0 \leq g(r_n) < g(r_{n-1}) \\
r_{n-1} &= q_{n+1} r_n + 0
\end{aligned}$$

So ein $n \in \mathbb{N}$ muss ja existieren, da die Bewertungen der r_i's iterativ immer strikt kleiner werden. Das r_n ist nun der $ggT(a, b)$.

Beweis. Betrachtet man die letzte Gleichung in der Gleichungskette, so ist ersichtlich, dass r_n ein Teiler von r_{n-1} ist. Eingesetzt in die vorletzte Gleichung ergibt sich $r_{n-2} = q_{n+1} q_n r_n + r_n = (q_{n+1} q_n + 1) r_n$ und somit, dass $r_n | r_{n-2}$. Durch analoges Fortsetzen bis zur zweiten und zur ersten Zeile, haben wir mit r_n also tatsächlich einen Teiler von a und b gefunden.
Angenommen es gäbe einen weiteren Teiler $d \in R$ und $d_a, d_b \in R$ mit $d_a d = a$ und $d_b d = b$ jedoch $d \nmid r_n$, dann folgt aus der ersten Gleichung und dem Distributivgesetz $r_1 = d_a d - q_1 d_b d = d(d_a - q_1 d_b)$, dass d ein Teiler von r_1 ist und analog $d|r_2, d|r_3, \ldots, d|r_n$ und somit dass r_n tatsächlich $ggT(a, b)$ ist. ∎

In obigem Beispiel wurde nicht nur 1071/1029 in einen Kettenbruch entwickelt, in dem die Zähler alle 1 sind, es wurde auch direkt der größte gemeinsamen Teiler von 1071 und 1029 mit 21 bestimmt. Die Kehrwertbildung bereitet allerdings noch Schwierigkeiten, denn sie ist um einen Schritt weiter als was bisher aufgebaut wurde. Man benötigt nämlich noch den Begriff des Quotientenkörpers.

Definition 4.26. Es wird wieder von einem Integritätsbereich R ausgegangen. Aus diesem kann über folgende Definitionen ein **Quotientenkörper** K gebildet werden. Als Ausgangsmenge hierfür nimmt man das kartesische Produkt $R \times R \setminus \{0\}$. Darauf wird die Äquivalenz-

relation $(r_1, s_1) \equiv (r_2, s_2)$, wenn $r_1 s_2 = r_2 s_1$ festgelegt, die bezüglich den Operatoren

$$\begin{aligned}(r_1, s_1) +_K (r_2, s_2) &:= (r_1 s_2 + r_2 s_1, s_1 s_2)\\ -_K(r, s) &:= (-r, s)\\ (r_1, s_1) \cdot_K (r_2, s_2) &:= (r_1 r_2, s_1 s_2)\end{aligned}$$

sogar eine Kongruenzrelation ist. Die Faktoralgebra $K := (Q, +, 0_K, -_K, \cdot_K, 1_K)$ mit der Grundmenge $Q := \big(R \times R \setminus \{0\}\big)/_{\equiv}$, den neutralen Elementen $0_K := [(0,1)]_{\equiv}$, $1_K := [(1,1)]_{\equiv}$ und der multiplikativ Inversen $[(r,s)]_{\equiv}^{-1} := [(s,r)]_{\equiv}$ ist schließlich der gewünschte Körper.

Nun ist man endlich in der Lage Kettenbrüche sinnvoll zu bilden und zu definieren.

Definition 4.27. Sei R ein euklidischer Ring und K dessen Quotientenkörper. Als einen **regulären Kettenbruch** bezeichnet man ein $f \in K$ mit

$$f = g_0 + \cfrac{1}{g_1 + \cfrac{1}{\ddots + \cfrac{1}{g_{n-1} + \cfrac{1}{g_n}}}}$$

wobei $g_0, g_1, \ldots, g_n \in R$ und f wohldefiniert sein muss.
Es wird hierfür auch kurz $[g_0; g_1, g_2, \ldots, g_n]$ geschrieben.

Vergleicht man den im obigen Beispiel aus 1071/1029 gewonnenen Kettenbruch mit dieser neuen Definition, so ist ersichtlich, dass dieser regulär ist. Es wird also eine Modifizierung des modernen euklidischen Algorithmus vorgenommen und erhält:

Definition 4.28. Seien ein Euklidischer Ring R und dessen Quotientenkörper K gegeben. Der **modifizierte euklidische Algorithmus** liefert aus jedem $\frac{a}{b} \in K$ mit $a, b \in R$ einen regulären Kettenbruch f.
Hierfür wird wie im modernen euklidischen Algorithmus die Division mit Rest durchgeführt und zwar so lange, bis der $ggT(a, b)$ mit $r_n \in R$ gefunden ist. Aus den q_i's ergibt sich dann

$$f = [q_1; q_2, q_3, \ldots q_{n+1}]$$

die gewünschte Darstellung.

Beweis. Der Beweis wird über vollständige Induktion nach n (Anzahl der Divisionen mit Rest bis zum ggT) geführt. Begonnen wird mit dem Induktionsanfang $n = 1$. Sei also $q_1 \in R$ gegeben mit $a = q_1 b + 0$. Da $\frac{a}{b} \in K$ gilt $b \neq 0$ und da K ein Körper ist, darf man b invertieren und erhält $\frac{a}{b} = q_1$. Es wurde also ein gewünschter regulärer Kettenbruch $f := q_1$ gefunden.
Für jedes $\tilde{a}, \tilde{b} \in R$, wobei $\frac{\tilde{a}}{\tilde{b}} \in K$ und mit n Schritten des euklidischen Algorithmus bis zum $ggT(\tilde{a}, \tilde{b})$ gibt es nun einen regulären Kettenbruch $\tilde{f}$ mit $\tilde{f} = \frac{\tilde{a}}{\tilde{b}}$. (Induktionsvoraussetzung)

Seien für $\frac{a}{b} \in K$ nun $n+1$ Schritte nötig, um den $ggT(a,b)$ zu finden ($r_0 := b$)

$$\begin{aligned} a &= q_1 r_0 + r_1 \qquad \text{mit} \qquad 0 \leq g(r_1) < g(r_0) \\ r_0 &= q_2 r_1 + r_2 \qquad \text{mit} \qquad 0 \leq g(r_2) < g(r_1) \\ &\vdots \\ r_{n-1} &= q_{n+1} r_n + 0 \end{aligned}$$

so ist sofort ersichtlich, dass für $\frac{r_0}{r_1}$ nur noch n Schritte für den $ggT(r_0, r_1)$ nötig sind. Durch den Einfluss der Induktionsvoraussetzung, erhält man den regulären Kettenbruch $f := [q_2; q_3, q_4, \ldots, q_{n+1}]$ mit $f = \frac{r_0}{r_1}$.
Da $f \neq 0$ ist auch f invertierbar und es gilt $\frac{a}{b} = q_1 + \frac{r_1}{r_0} = q_1 + f^{-1}$. Unter Anwendung dass $f^{-1} = [0; f]$ und $q_1 + [0; q_2, q_3, \ldots, q_{n+1}] = [q_1; q_2, q_3, \ldots, q_{n+1}]$, folgt der Induktionsschritt $n \to n+1$.

■

Umgekehrt soll nun von (wohldefinierten) regulären Kettenbrüchen ausgegangen und diese in rationale Elemente geformt werden. Hierfür wird erst einmal eine weitere Definition eingeführt.

Definition 4.29. Gegeben sei eine Folge $(g_n)_{n\in\mathbb{N}}$ im euklidischen Ring R und für alle $n \in \mathbb{N}$ sei $[g_0; g_1, \ldots, g_n]$ im Quotientenkörper K von R wohldefiniert. Man schreibt für diese Folge auch

$$[g_0; g_1, g_2, g_3, \ldots] = g_0 + \cfrac{1}{g_1 + \cfrac{1}{g_2 + \cfrac{1}{g_3 + \ddots}}}$$

und spricht von einem **unendlichen regulären Kettenbruch**. Die aus den endlichen Teilfolgen gebildeten Kettenbrüche $[g_0; g_1, \ldots, g_n]$ werden auch als **n-te Konvergenten** des unendlichen Kettenbruchs bzeichnet.

Man sieht schon, dass bereits der Ring der formalen Potenzreihen angepeilt wird. Von Konvergenz soll hier allerdings noch nicht gesprochen werden. Zunächst einmal die Betrachtung der ersten vier Konvergenten eines unendlichen Kettenbruchs und deren Darstellung als rationale Elemente:

$$\frac{p_0}{q_0} = \frac{g_0}{1} \qquad \frac{p_2}{q_2} = \frac{g_2(g_1 g_0 + 1) + g_0}{g_2 g_1 + 1}$$
$$\frac{p_1}{q_1} = \frac{g_1 g_0 + 1}{g_1} \qquad \frac{p_3}{q_3} = \frac{g_3(g_2(g_1 g_0 + 1) + g_0) + (g_1 g_0 + 1)}{g_3(g_2 g_1 + 1) + g_1}$$

Wie bereits Euler [Euler, 1744] feststellte und sich bei näherem Betrachten auch ableiten lässt, gilt für die Konvergenten eine rekursive Beziehung:

Satz 4.30. Sei R ein euklidischer Ring und f=$[g_0; g_1, g_2, g_3, \ldots]$ ein unendlicher regulärer Kettenbruch aus dessen Quotientenkörper K. Für die n-ten Konvergenten gilt

$$[g_0; g_1, g_2, \ldots, g_n] = \frac{p_n}{q_n} := \frac{g_n p_{n-1} + p_{n-2}}{g_n q_{n-1} + q_{n-2}} \qquad \text{für alle } n \geq 1$$

wobei hier die ersten vier Glieder definiert sind als

$$p_{-1} := 1 \qquad p_0 := g_0$$
$$q_{-1} := 0 \qquad q_0 := 1.$$

Es soll auch nochmal angemerkt werden, dass $p_n, q_n \in R$ für alle $n \geq 0$.

Beweis. Der Beweis wird wie erwartet über vollständige Induktion nach der Anzahl der Konvergenten n durchgeführt.
Für $n = 0$ folgt aus der Definition bereits $\frac{p_0}{q_0} = \frac{g_0}{1}$ und für n=1 dass

$$\frac{p_1}{q_1} := \frac{g_1 p_0 + p_{-1}}{g_1 q_0 + q_{-1}} = \frac{g_1 g_0 + 1}{g_1 \cdot 1 + 0}$$

mit der oben gezeigten Herleitung für $\frac{p_1}{q_1}$ übereinstimmt.
Der Induktionsanfang ist also erfüllt und im Weiteren folgt der Induktionsschritt $(n-1) \to n$. Seien also alle Konvergenten eines (beliebigen) unendlichen regulären Kettenbruchs bis $n-1$ rekursiv darstellbar (Induktionsvoraussetzung). Somit gilt für den $(n-1)$-ten Konvergenten von f die Gleichung

$$\frac{p_{n-1}}{q_{n-1}} = \frac{g_{n-1} p_{n-2} + p_{n-3}}{g_{n-1} q_{n-2} + q_{n-3}}$$

Der n-te Konvergent von f darf auch dargestellt werden als

$$[g_0; g_1, \ldots, g_{n-2}, g_{n-1}, g_n] = [g_0; g_1, \ldots, g_{n-2}, g_{n-1} + 1/g_n].$$

Dies ist ebenfalls einfach über vollständige Induktion zu zeigen. Zu bemerken ist hier allerdings, dass die Rekursion nicht nur für $g_0, g_1, \ldots \in R$ gilt, sondern ebenfalls auch für Folgen in K. Diese Tatsache wird nur kurz benötigt, um hier die Induktionsvoraussetzung geschickt einbinden zu können. Einen Schritt später wird wieder der Fall in R aufgegriffen.
Unter Anwendung der Induktionsvoraussetzung auf die rechte Seite folgt

$$f = \frac{(g_{n-1} + \frac{1}{g_n}) p_{n-2} + p_{n-3}}{(g_{n-1} + \frac{1}{g_n}) q_{n-2} + q_{n-3}} = \frac{\frac{1}{g_n} p_{n-2} + g_{n-1} p_{n-2} + p_{n-3}}{\frac{1}{g_n} q_{n-2} + g_{n-1} q_{n-2} + q_{n-3}}.$$

Hier bleiben p_{n-2}, p_{n-3}, q_{n-2} und q_{n-3} unverändert, das liegt daran, dass diese nicht auf das Element $g_{n-1} + 1/g_n$ zurückgreifen. Durch Erweitern der rechten Seite mit g_n/g_n und dem Einfluss der Gleichung für den $(n-1)$-ten Konvergenten, erhält man schließlich

$$f = \frac{p_{n-2} + g_n(g_{n-1} p_{n-2} + p_{n-3})}{q_{n-2} + g_n(g_{n-1} q_{n-2} + q_{n-3})} = \frac{g_n p_{n-1} + p_{n-2}}{g_n q_{n-1} + q_{n-2}}$$

und somit wurde auch der Induktionsschritt gezeigt. ■

Fasst man die oben gezeigten Beziehungen zusammen, so ergibt sich folgendes Resultat:

Korollar 4.31. Bezeichne R einen euklidischen Ring und K den daraus gebildeten Quotientenkörper. Jedes Element $\frac{p}{q} \in K$ lässt sich als regulärer Kettenbruch $f = [g_0; g_1, \ldots, g_n]$ mit Elementen $g_0, g_1, \ldots, g_n \in R$ darstellen.
Umgekehrt lässt sich auch jeder (wohldefinierte) reguläre Kettenbruch f als rationales Element $\frac{p}{q} \in K$ mit $p, q \in R$ darstellen.

Beweis. Den ersten Teil hat der modifizierte euklidische Algorithmus geliefert und der zweite Teil folgt aus dem Satz von Euler, denn die Rekursion gilt natürlich auch für (endliche) reguläre Kettenbrüche. ■

Zu bemerken ist, dass sich ein rationales Element nicht eindeutig als regulärer Kettenbruch darstellen lässt. Man betrachte hierfür als Beispiel $[g_0; g_1, \ldots, g_n]$ und den daraus gebildeten Kettenbruch $[g_0; g_1, \ldots, g_n - 1, 1]$. Beide beschreiben ein und dasselbe rationale Element.
Es wird nun eine weitere Form eines Kettenbruchs vorgestellt. Diese ist besser geeignet, um eine Verbindung zu den Padé-Approximanten zu schaffen.

Definition 4.32. Sei R ein euklidischer Ring und K dessen Quotientenkörper. Als einen **(verallgemeinerten) Kettenbruch** bezeichnet man ein $f \in K$ mit

$$f = g_0 + \cfrac{h_1}{g_1 + \cfrac{h_2}{\ddots + \cfrac{h_n}{g_n}}}$$

wobei $g_0, g_1, \ldots, g_n, h_1, h_2, \ldots, h_n \in R$ sind und f wohldefiniert sein muss.
Es wird auch oft eine eigene Notation für solche Kettenbrüche verwendet wie

$$g_0 + \frac{h_1}{g_1} + \frac{h_2}{g_2} + \cdots + \frac{h_n}{g_n} \qquad \text{oder auch} \qquad g_0 + \operatorname*{\mathcal{K}}_{i=1}^{n}\left(\frac{h_n}{g_n}\right).$$

Es wird auch hier der Begriff eines **unendlichen (verallgemeinerten) Kettenbruchs** eingeführt. Dies sei wieder eine Folge $(g_n)_{n\in\mathbb{N}}$ in R bei der jede n-te Konvergente

$$g_0 + \frac{h_1}{g_1} + \frac{h_2}{g_2} + \cdots + \frac{h_n}{g_n}$$

wohldefiniert ist. Geschrieben wird dafür auch

$$g_0 + \frac{h_1}{g_1} + \frac{h_2}{g_2} + \cdots = g_0 + \cfrac{h_1}{g_1 + \cfrac{h_2}{g_2 + \ddots}}.$$

Zu Beginn wird gleich einmal der Zusammenhang zwischen der allgemeinen- und der regulären Darstellung eines Kettenbruchs gezeigt.

Lemma 4.33. Es existiert eine Äquivalanztransformation zwischen einem Kettenbruch der allgemeinen- und der regulären Darstellung. Man spricht von **äquivalenten Kettenbrüchen** wenn deren Konvergenten übereinstimmen.

Beweis. Es ist klar, dass jeder (unendliche) reguläre Kettenbruch $f_r = [g_0; g_1, g_2, g_3, \ldots]$ als (unendlicher) verallgemeinerter Kettenbruch $f_v := g_0 + \mathcal{K}_{i=1}^{\infty}\left(\frac{1}{g_i}\right)$ dargestellt werden kann und die Konvergenten beider Kettenbrüche übereinstimmen.

Die Umkehrung gilt ebenfalls, jedoch ist darauf zu achten, dass dann von einem regulären Kettenbruch gesprochen wird, der eine Folge in K und nicht mehr in R ist. Durch Umformen eines verallgemeinerten Kettenbruches erhält man

$$g_0 + \cfrac{h_1}{g_1 + \cfrac{h_2}{g_2 + \cfrac{h_3}{g_3 + \ddots}}} = g_0 + \cfrac{h_1/g_1}{1 + \cfrac{h_2/g_1}{g_2 + \cfrac{h_3}{g_3 + \ddots}}} = g_0 + \cfrac{h_1/g_1}{1 + \cfrac{h_2/(g_1g_2)}{1 + \cfrac{h_3/g_2}{g_3 + \ddots}}} = g_0 + \cfrac{h_1/g_1}{1 + \cfrac{h_2/(g_1g_2)}{1 + \cfrac{h_3/(g_2g_3)}{1 + \ddots}}}$$

und somit wird durch $f_r := [g_0, h_1/g_1, h_2/(g_1g_2), h_3/(g_2g_3), \ldots]$ ein regulärer Kettenbruch gebildet. ■

Ein Satz von Bernoulli besagt laut [Sauer, 2005] nun, dass für eine Folge $(q_n)_{n\in\mathbb{N}}$ mit $q_n \neq q_{n+1}$ für alle $n \in \mathbb{N}$ in einem Quotientenkörper Q eines komutativen Ring mit Einselement R ein verallgemeinerter unendlicher Kettenbruch f_v mit Elementen aus Q existiert, dessen Konvergentenfolge $(f_{v_n})_{n\in\mathbb{N}}$ mit $(q_n)_{n\in\mathbb{N}}$ übereinstimmt.
Nimmt man wieder $R = \mathbb{Z}$ und $Q = \mathbb{Q}$ und ein irrationales $x \in \mathbb{R}$ (der Vervollständigung von $\mathbb{Q}$), so existiert eine streng monoton wachsende Folge $(q_n)_{n\in\mathbb{N}}$ in $\mathbb{Q}$, die gegen x konvergiert ($\mathbb{Q}$ liegt ja dicht in $\mathbb{R}$ und z.B. durch geschickte Wahl in Dezimalbruchentwicklung). Bernoulli sagt also nun, es gibt einen verallgemeinerten Kettenbruch f_v in $\mathbb{Q}$ mit obiger Eigenschaft, dessen Konvergenten streng monoton gegen x konvergieren.
Analog kann dies auf Polynome ($= R$), rationale Funktionen ($= Q$) und meromorphe Funktionen (Vervollständigung von Q) angewendet werden.
Eine Anwendung auf die formalen Potenzreihen $R = \mathbb{C}[[X]]$ und die formalen Laurentreihen $Q = \mathbb{C}((X))$ (siehe Definition 4.36) ist ebenfalls möglich.

Es treten also zwei Probleme auf. Erstens muss eine Folge von Padé-Approximanten (rationale Funktionen) ungleiche Nachbarn haben. Zweitens sind laut Bernoulli dann die Glieder eines Kettenbruchs rationale Funktionen und keine Polynome (und selbst bei diesen möchte man, dass sie Monome sind). Diese Schwierigkeiten müssen im Folgenden noch ausgebadet werden.

Bevor man näher darauf eingehen kann noch ein Korollar für die Konvergenten eines unendlichen verallgemeinerten Kettenbruchs. Um eine Übersicht zu erhalten, sind hier schon mal die ersten drei Konvergenten aufgezeigt:

$$\frac{p_0}{q_0} = \frac{g_0}{1} \qquad \frac{p_1}{q_1} = \frac{g_0g_1 + h_1}{g_1}$$
$$\frac{p_2}{q_2} = g_0 + \frac{g_2h_1}{g_1g_2 + h_2} = \frac{g_0g_1g_2 + g_0h_2 + g_2h_1}{g_1g_2 + h_2}$$

Wie auch bei den n-ten Konvergenten eines regulären unendlichen Kettenbruchs, lässt sich auch bei den verallgemeinerten Kettenbrüchen eine rekursive Darstellung finden.

Korollar 4.34. Sei R ein Euklidischer Ring und

$$f = g_0 + \frac{h_1}{g_1} + \frac{h_2}{g_2} + \frac{h_3}{g_3} + \cdots$$

ein unendlicher verallgemeinerter Kettenbruch aus dessen Quotientenkörper. Für die n-ten Konvergenten gilt

$$f_n = \frac{p_n}{q_n} := \frac{g_np_{n-1} + h_np_{n-2}}{g_nq_{n-1} + h_nq_{n-2}} \qquad \text{für alle } n \geq 1$$

wobei hier die ersten vier Glieder definiert sind als

$$p_{-1} := 1 \qquad p_0 := g_0$$
$$q_{-1} := 0 \qquad q_0 := 1.$$

Es gilt auch hier wieder, dass $p_n, q_n \in R$ für alle $n \geq 0$.

Beweis. Der Beweis ist analog zu dem von Satz 4.30 zu führen. Der entscheidende Unterschied liegt nur in der Darstellung des n-ten Konvergenten, wo man $\tilde{g}_{n-1} := g_{n-1} + h_n/g_n$ definiert. ■

4.3.2 Kettenbrüche und Padé-Approximation

Im vorigen Kapitel wurde versucht alles sehr allgemein aufzubauen. Dies liegt vor allem daran, dass sich nun die bekannten Vorgehensweisen auf die formalen Potenzreihen $\mathbb{C}[[X]]$ (oder $\mathbb{R}[[X]]$) umsetzen lassen, denn der Ring der formalen Potenzreihen ist sogar ein euklidischer Ring. Es gibt also wiederum die Möglichkeit Division mit Rest durchzuführen. Desweiteren ist der Quotientenkörper davon die Menge der formalen Laurentreihen. Man könnte also jede Laurentreihe wieder als einen Kettenbruch darstellen und da sich ja meromorphe Funktionen lokal als Laurentreihe (mit endlichem Hauptteil) darstellen lassen auch alles auf diese gewünschte Funktionenmenge überführen.
Das war also im Groben, was hier gezeigt werden soll. Selbstverständlich wird auch der Zusammenhang von Padé-Approximanten (dass sind ja die endlichen Laurentreihen) und dessen Darstellung als Kettenbrüche behandelt. Im Weiteren werden auch Konvergenzsätze angesprochen und wie alles auch für Potenzreihen in mehreren Variablen umgesetzt werden kann.

Das waren erst einmal genug Zusammenhänge. Begonnen wird wieder ganz vorne und damit zu zeigen, wie aus dem Ring der formalen Potenzreihen $\mathbb{C}[[X]]$ ein euklidischer Ring gewonnen wird. Von der Tatsache, dass $\mathbb{C}[[X]]$ ein Integritätsbereich ist, wird ausgegangen. Also fehlt nur noch die Bewertungsfunktion mit den in Definition 4.24 beschriebenen Eigenschaften. Diese ist mit

$$\begin{aligned} ord : \mathbb{C}[[X]] \setminus \{0\} &\to \mathbb{N} \\ \sum_{n=0}^{\infty} c_n X^n &\mapsto \min\{n \in \mathbb{N} \text{ mit } c_n \neq 0\} \end{aligned}$$

gegeben. Um zu zeigen, dass die Eigenschaft tatsächlich erfüllt ist, wird hier erst einmal der klassische Divisionsalgorithmus zur Bewertung formaler Potenzreihen angeführt. Seien also $A(X), B(X) \in \mathbb{C}[[X]] \setminus \{0\}$. Man bestimmt die ersten Koeffizienten von $C(X) = A(X)/B(X)$ wie folgt:

Divisionsalgorithmus bei formalen Potenzreihen							
$(a_0$	$+a_1X$	$+a_2X^2+\ldots)$	/	$(b_0+b_1X+b_2X^2+\ldots)$			
$-b_0\frac{a_0}{b_0}$	$-b_1c_0X$	$-b_2c_0X^2-\ldots$			$\frac{a_0}{b_0}$	$=$	c_0
	$(a_1-b_1c_0)X$	$+(a_2-b_2c_0)X^2+\ldots$					
	$-b_0\frac{a_1-b_1c_0}{b_0}X$	$-b_1c_1X^2-\ldots$			$\frac{a_1-b_1c_0}{b_0}X$	$=$	$+c_1X$
		$(a_2-b_2c_0-b_1c_1)X^2+\ldots$					
		$-b_0\frac{a_2-b_2c_0-b_1c_1}{b_0}X^2-\ldots$			$+\frac{a_2-b_2c_0-b_1c_1}{b_0}X^2$	$=$	$+c_2X^2$
		$+\ldots$					
		$-\ldots$				$=$	$+\ldots$

Hier fallen sofort ein paar Merkmale auf. Erstens wird (im Gegensatz zu dem Beispiel in $\mathbb{Z}$) die Ordnung des Restes von Zeile zu Zeile immer strikt größer (also Vorsicht bei der Wahl des $<$ Zeichens in der Bewertung). Zweitens spielt hier $b_0 \neq 0$ eine bedeutende Rolle, um die Invertierbarkeit zu garantieren. Dieses Problem wurde ja auch schon im Kapitel der Padé-Approximanten angemerkt und selbst Korollar 4.13 nimmt solche Fälle (vorerst) aus. Vergleicht man die Aussage des Lemmas 4.12 mit der Struktur des Quotienten, so kann die rekursive Beziehung

$$c_n = b_0^{-1}(a_n - \sum_{i=1}^{n} b_i c_{n-i}) \qquad \text{für alle } n \geq 0$$

folgern. Durch das Setzen von $A(X) := (1, 0, 0, \ldots)$ als neutrales Element, bestimmt der Algorithmus nichts anderes als die Inverse von $B(X)$ und die c_n stimmen mit denen aus dem Lemma überein.

Hier soll nun eine Möglichkeit angegeben werden, um eine MacLaurin Reihe in einen Kettenbruch zu entwickeln [Salzer, 1962]. Dazu geht man wieder von einer formalen Potenzreihe

$$A(X) = a_0 + a_1X + a_2X^2 + a_3X^3 + \ldots$$

aus und heben den linearen Term heraus

$$A(X) = a_0 + a_1X(1 + \frac{a_2}{a_1}X + \frac{a_3}{a_1}X^2 + \ldots).$$

Hier wurde vorausgesetzt, dass a_1 invertierbar ist ($a_1 \neq 0$) und weil der Führungskoeffizient ungleich 0 ist, lässt sich die nun gebildete Potenzreihe invertieren

$$(1 + \frac{a_2}{a_1}X + \frac{a_3}{a_1}X^2 + \ldots)^{-1} =: (1 + a_1^{(1)}X + a_2^{(1)}X^2 + \ldots)$$

und da die Inverse der Inversen wieder die Ursprüngliche ist, kann man schreiben

$$A(X) = a_0 + \frac{a_1X}{(1 + a_1^{(1)}X + a_2^{(1)}X^2 + \ldots)}.$$

Analog wird aus dieser durch Herausheben des linearen Terms

$$A(X) = a_0 + \frac{a_1X}{(1 + a_1^{(1)}X(1 + \frac{a_2^{(1)}}{a_1^{(1)}}X + \frac{a_3^{(1)}}{a_1^{(1)}}X^2 + \ldots)}.$$

Vorausgesetzt $a_1^{(1)}$ ist hier wieder invertierbar und iterativ auch im Weiteren alle $a_1^{(n)}$, so erhält man

$$A(X) = a_0 + \frac{a_1X}{1} + \frac{a_1^{(1)}X}{1} + \frac{a_1^{(2)}X}{1} + \frac{a_1^{(3)}X}{1} + \cdots. \tag{12}$$

Es sind hier zwar noch jede Menge an Voraussetzungen nötig, um eine formale Potenzreihe in einen Kettenbruch entwickeln zu können, jedoch gibt es schon jetzt eine interessante Beobachtung [Baker et al., 1996]. Davor benötigt man aber wieder die rekursive Darstellung der Konvergenten von (12). Desweiteren wird die Darstellung von (12) verallgemeinert und betrachtet im Weiteren Kettenbrüche der Form

$$F(z) = b_0 + \frac{a_1z}{b_1} + \frac{a_2z}{b_2} + \frac{a_3z}{b_3} + \frac{a_4z}{b_4} + \cdots.$$

Durch Bestimmung der ersten Konvergenten

$$\frac{A_0}{B_0} = \frac{b_0}{1} \qquad \frac{A_1}{B_1} = \frac{b_0b_1 + a_1z}{b_1}$$
$$\frac{A_2}{B_2} = b_0 + \frac{b_2a_1z}{b_1b_2 + a_2z} = \frac{b_0b_1b_2 + b_0a_2z + b_2a_1z}{b_1b_2 + a_2z}$$

sieht man, dass wie in Korollar 4.34 eine rekursive Darstellung möglich ist. Es müssen nur alle h_i mit $a_i z$ und g_i mit b_i ersetzt werden und man erhält

$$F_n(z) = \frac{A_n(z)}{B_n(z)} := \frac{b_n A_{n-1} + a_n z A_{n-2}}{b_n B_{n-1} + a_n z B_{n-2}} \qquad \text{für alle } n \geq 1 \tag{13}$$

mit den ersten 4 Gliedern definiert als

$$A_{-1}(z) := 1 \qquad A_0(z) := b_0$$
$$B_{-1}(z) := 0 \qquad B_0(z) := 1.$$

Es folgt nun endlich der Satz aus [Baker et al., 1996], auf den schon lange abgezielt wird.

Satz 4.35. Sei $f(z)$ in eine komplexe MacLaurin-Reihe mit Konvergenzradius $R > 0$ entwickelbar und $F(z) \in \mathbb{C}[[Z]]$ deren Vertreter als formale Potenzreihe. Falls alle $a_1, a_1^{(1)}, a_1^{(2)}, \ldots$ ungleich 0 sind ($F(z)$ ist also in einen unendlichen Kettenbruch entwickelbar), dann gilt für die Konvergenten von $F(z)$ aufgefasst als rationale Funktionen

$$[M, M](z) = \frac{A_{2M}(z)}{B_{2M}(z)} \qquad \text{und} \qquad [M+1, M](z) = \frac{A_{2M+1}(z)}{B_{2M+1}(z)} \qquad \text{für alle } M \geq 0$$

dass sie eine Stufenfolge in der Padé-Tabelle von $f(z)$ bilden.

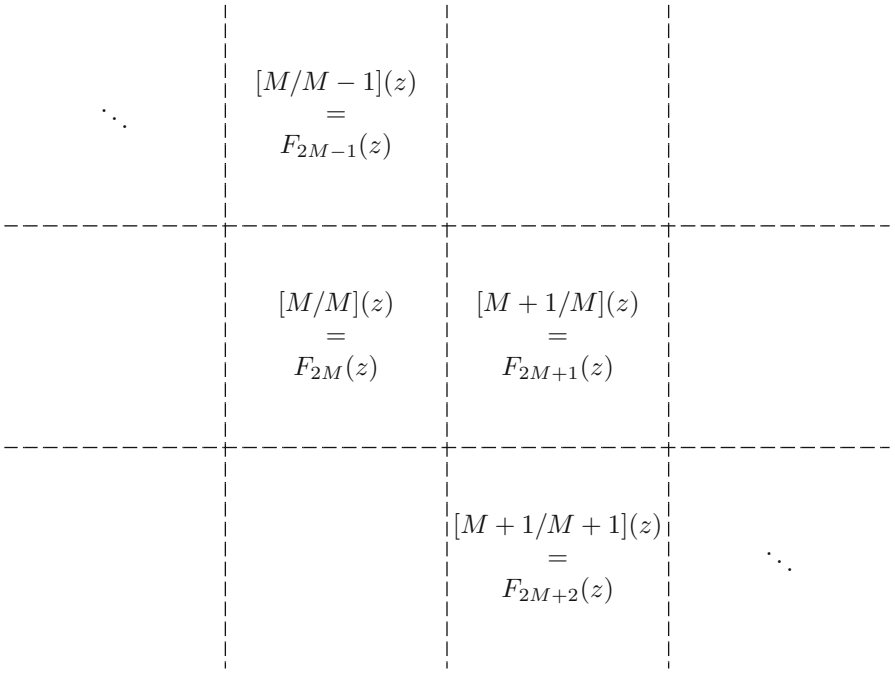
Beweis. Die Voraussetzung $a_1, a_1^{(1)}, a_1^{(2)}, \ldots \neq 0$ erlaubt $F(z)$ in einen unendlichen Kettenbruch zu entwickeln, was somit die Existenz aller Konvergenten von $F(z)$ sicherstellt. Da man nun einen Konvergenten $F_n(z)$ mit $n \in \mathbb{N}$ über eine Äquivalenzumformung aus $F(z)$ erhält, stimmen die Koeffizienten der Reihendarstellung von $F_n(z)$ mit denen von $F(z)$ bis zum Grad n überein. (Kann über vollständige Induktion gezeigt werden). Das bedeutet also, dass die Koeffizienten von $F_{2M} = \frac{A_{2M}(z)}{B_{2M}(z)}$ bzw. $F_{2M+1} = \frac{A_{2M+1}(z)}{B_{2M+1}(z)}$ mit denen von $F(z)$ bis zum Grad $2M$ bzw. $2M+1$ übereinstimmen.
Da nun $f(z)$ analytisch in 0 ist muss man für die Definition 4.15 eines Padé-Approximanten der Ordnung (M/M) bzw. $(M+1/M)$ noch zeigen, dass $\deg(A_{2M}(z)) = M$ und $\deg(B_{2M}(z)) = M$ bzw. $\deg(A_{2M+1}(z)) = M+1$ und $\deg(B_{2M+1}(z)) = M$.

Der Beweis hierfür ist aufwendig und beruht auf zwei Teilschritten:
Im ersten Schritt verwendet man die Rekursion (13) um über vollständige Induktion zu zeigen, dass die Ungleichungen

$$\deg(A_{2M}(z)) \leq M \qquad \text{und} \qquad \deg(B_{2M}) \leq M$$
$$\text{sowie}$$
$$\deg(A_{2M+1}(z)) \leq M+1 \qquad \text{und} \qquad \deg(B_{2M+1}) \leq M$$

gelten. Um den zweiten Schritt für Gleichheit zu erhalten, kann man wie in [Baker et al., 1996] vorgehen und einen Beweisschritt aus dem Verallgemeinerten Viskovatov Algorithmus verwenden. ■

Wenn also die Folge an Konvergenten betrachtet und diese in der Padé-Tabelle eingetragen wird



sieht man sofort, dass sich diese treppenartig fortsetzen. Einfach ausgedrückt: durch konstruktives Ausbauen eines Kettenbruchs nach unten, kann eine diagonale Folge von Padé-Approximanten gebildet werden.

Dieses Konzept soll im Folgenden noch verallgemeinert werden:
Da man den Ring $\mathbb{C}[[X]]$ der formalen Potenzreihen über $\mathbb{C}$ in einer Unbestimmten X betrachtet, und $\mathbb{C}$ ein Körper ist folgt, dass auch der Quotientenkörper (siehe Definition 4.26) davon wieder ein Körper ist. Weiters gilt auch, dass die Lokalisierung an X von $\mathbb{C}[[X]]$ mit diesem Quotientenkörper übereinstimmt. Diese wird als Ring der formalen Laurent-Reihen bezeichnet. Ohne weiter auf dessen Konstruktion einzugehen deren Definition:

Definition 4.36. Sei $K((X))$ der Körper der **formalen Laurentreihen** mit $K = \mathbb{C}$ (oder $K = \mathbb{R}$) und $A \in K((X))$ mit

$$A(X) = \sum_{n=m}^{\infty} a_n X^n \qquad \text{mit } m \in \mathbb{Z} \text{ und } a_n \in K \text{ für alle } n \geq m.$$

Als Ordnungsfunktion auf der Menge der formalen Laurentreihen bezeichnet man die Funktion

$$ord : K((X)) \to \mathbb{Z} \cup \{+\infty\}$$

$$B(X) = \sum_{n \in \mathbb{Z}} b_n X^n \mapsto \begin{cases} +\infty & \text{wenn } B(x) = 0 \\ \min\{n \in \mathbb{Z} \text{ mit } b_n \neq 0\} & \text{sonst} \end{cases}$$

Diese ist wohldefiniert, da ja jede formale Laurentreihe einen endlichen Hauptteil besitzt.
Es wird auch $ord(A(X))$ als die **Ordnung** von A bezeichnet.

Das Schöne an den formalen Laurentreihen $K((X))$ ist, dass man nun in der Lage ist zu jedem $A(X) \in K((X)) \setminus \{0\}$ ein inverses Element zu bilden. Hierfür muss nur

$$B(X) := X^{-ord(A(X))} A(X)$$

definiert werden und da nun weiters $b_0 \neq 0$ gilt, lässt sich der auf den formalen Potenzreihen definierte Divisionsalgorithmus anwenden. Daraus erhält man schließlich

$$A(X)^{-1} = X^{-ord(A(X))} B(X)^{-1}.$$

Es ist auch wieder ersichtlich, warum von Beginn an von meromorphen Funktionen gesprochen wird. Denn für genau diese ist man in der Lage eine Polstelle in 0 zu heben und dann eine Kettenbruchentwicklung durchzuführen.
Als Beispiel dient nun die Kettenbruchentwicklung einer geraden Funktion $f_g(z)$. Da die Potenzreihe nur aus Koeffizienten mit geradem Index (ungleich 0) besteht

$$f_g(z) = c_0 + c_2 z^2 + c_4 z^4 + c_6 z^6 + \dots$$

erhält man durch umformen und bilden der reziproken Reihe

$$f_g(z) = c_0 + c_2 z^2 (1 + \frac{c_4}{c_2} z^2 + \frac{c_6}{c_2} z^4 + \dots) = c_0 + \frac{c_2 z^2}{1 + c_2^{(1)} z^2 + c_4^{(1)} z^4 + \dots}$$
$$= c_0 + \frac{c_2 z^2}{1 + c_2^{(1)} z^2 (1 + \frac{c_4^{(1)}}{c_2^{(1)}} z^2 + \dots)} = c_0 + \frac{c_2 z^2}{1 + \frac{c_2^{(1)} z^2}{1 + c_2^{(2)} z^2 + \dots}}.$$

Somit hat also eine gerade Funktion eine Form

$$f_g(z) = b_0 + \frac{a_1 z^2}{b_1} + \frac{a_2 z^2}{b_2} + \frac{a_3 z^2}{b_3} + \dots. \tag{14}$$

Analog ist auch für eine ungerade Funktion

$$f_u(z) = c_1 z + c_3 z^3 + c_5 z^5 + c_7 z^7 + \dots$$

eine spezielle Kettenbruchentwicklung erkennbar

$$f_u(z) = \frac{a_1 z}{b_1} + \frac{a_2 z^2}{b_2} + \frac{a_3 z^2}{b_3} + \dots.$$

Es gibt auch unterschiedliche Algorithmen zur Bestimmung der Kettenbrüche zu gegebener MacLaurin-Reihe, wie zum Beispiel den Algorithmus von Viskovatov. Dieser liefert eine Form

$$f(z) = b_0 + \frac{a_1 z^{\alpha_1}}{1} + \frac{a_2 z^{\alpha_2}}{1} + \frac{a_3 z^{\alpha_3}}{1} + \cdots$$

mit $\alpha_n \geq 1$ für alle $n \in \mathbb{N}$. Die Konvergenten müssen dann auch nicht mehr nur stufenförmig in der Padé-Tabelle angeordnet sein.

Es gibt auch mehrere Erweiterungen dieses Algorithmus. In [Bultheel, 1980] sowie auch wieder [Baker et al., 1996] finden sich Ideen und Vorgehensweisen zur Kettenbruchentwicklung des Quotienten $C(Z) = A(Z)/B(Z)$ zweier Potenzreihen $A(Z)$ und $B(Z) \neq 0$ und erhalten

$$C(Z) = \Pi_0(Z) + \frac{Z^{\alpha_1}}{\Pi_1(Z)} + \frac{Z^{\alpha_2}}{\Pi_2(Z)} + \frac{Z^{\alpha_3}}{\Pi_3(Z)} + \dots.$$

Vorteile dieser Darstellungsform werden auch ausführlich in [Magnus, 1962] besprochen. Diese Kettenbrüche werden von Magnus als P-Brüche bezeichnet.
Interessant ist auch, dass durch eine weitere Modifikation alle Padé-Approximanten (vom Quotienten) vom Typ $[M+J/M]$ mit $J \geq -1$ bestimmt werden können. Solche Folgen in Nebendiagonalen der Padé-Tabelle bezeichnet man auch als paradiagonale Folgen. Deren Existenz wurde ja schon mit Satz 4.23 gezeigt.

Angemerkt soll hier noch werden, dass sich sogenannte Stieltjes-Reihen in reguläre Kettenbrüche der Form (12) entwickeln lassen. Für weitere Betrachtungen wie die Kettenbrüche spezieller Funktionen aussehen, siehe auch [Backeljauw und Cuyt, 2009].

4.4 Werte- und Koeffizienten Problem

Hier ein kurzer Ausflug in die numerische Mathematik. In der rationalen Inter- oder Extrapolation und somit auch in der Padé-Approximation, wird zwischen **zwei Problematiken** unterschieden. Eine vorzeitige Klassifikation kann hier sehr sinnvoll sein, da hier eigens effiziente Algorithmen entwickelt wurden.

Die erste Problematik ist das sogenannte **Werte Problem**. Dabei kommt es nur auf einzelne Auswertungen von Padé-Approximanten (einer analytischen Funktion) an. Es müssen also hier die Padé-Approximanten nicht explizit bestimmt werden. Einer der ersten hierfür entwickelten Algorithmen ist der sogenannte Epsilon-Algorithmus. So wurden ursprünglich von Shanks [Shanks, 1955] und später von Wynn [Wynn, 1956] Methoden zur Transformation von langsam konvergierenden (oder gar divergenten – man denke an Konvergenz von Reihen außerhalb des Konvergenzradius) in zügig konvergierende Folgen vorgestellt. Wie effizient und vielseitig einsetzbar dann der finale Epsilon-Algorithmus ist, kann auch im vielzitierten Review von Wynn [Wynn, 1966] nachgelesen werden. Speziell für Anwendung auf Padé-Approximanten und Kettenbrüche siehe auch Graves-Morris [Graves-Morris et al., 2000]. Eine weiters aus dem Epsilon-Algorithmus hervorgehende und numerisch stabilere Variante ist der Eta-Algorithmus von Bauer. Für Vergleich siehe Gragg [Gragg, 1972].

Die zweite Problematik ist das **Koeffizienten Problem**. Das Interesse liegt hier in der expliziten Darstellung der Padé-Approximanten und somit der Bestimmung der Koeffizienten von Nenner- und Zählerpolynom. Damit lassen sich Untersuchungen auf Konvergenz oder auch anschließend eine Vielzahl von Auswertungen (wie im CFR-Algorithmus nötig) durchführen. Im letzten Kapitel wurde gezeigt, wie sich eine formale Potenzreihe in einen Kettenbruch überführen lässt und wann dessen Konvergenten eine Stufenfolge in der Padé-Tabelle bilden. Es lassen sich die Konvergenten sogar über eine rekursive Vorschrift aus der C-Tabelle bilden. Der Quotienten-Differenz-Algorithmus (QD-Algorithmus) ist hierfür ein numerisch stabiles Verfahren von Rutishauser [Rutishauser, 1954]. Im Gegensatz dazu gibt es noch den Kronecker Algorithmus [Kronecker, 1881] und den darauf aufbauenden Baker Algorithmus [Baker, 1972], die allerdings auf dem Prinzip des euklidischen Algorithmus basieren. Ein weiteres Verfahren zur Bestimmung einer Stufenfolge in der Padé-Tabelle, welches die Bildung von reziproken Reihen vermeidet, ist das Viskovatov-Verfahren [Baker et al., 1996]. Geht es einem hierbei nur um die Koeffizienten des Nennerpolynoms, so kann dafür auch ein modifizierter Berlekamp-Massey-Algorithmus verwendet werden [Gashkov und Gashkov, 2006].

Eine gute Übersicht zu den oben genannten (und auch weitere und modifizierte) Methoden zur Padé-Approximation findet sich in [Claessens, 1975].

4.5 Konvergenztheorie

In diesem Unterkapitel sollen Überlegungen und Sätze zur Konvergenz von Padé-Approximanten angesprochen werden. Diese sollen Bezug auf den CFR-Algorithmus haben und ein vollständiger Aufbau wäre hier also nicht zielführend. Aber vorerst noch eine kurze Wiederholung der nötigen Konvergenzbegriffe:

Definition 4.37. Eine Funktionenfolge $(f_n(z))_{n\in\mathbb{N}}$ mit $f_n : D \to \mathbb{C}$ heißt **gleichmäßig konvergent** gegen eine Funktion $f : D \to \mathbb{C}$ wenn gilt

$$\lim_{n\to\infty} \sup_{z\in D} |f_n(z) - f(z)| = 0.$$

Eine leichte Abschwächung ist der Begriff der **lokal gleichmäßigen Konvergenz**. Hier reicht es aus, wenn es zu jedem $z_0 \in D$ eine offene Umgebung $D_{z_0} \subset D$ gibt, so dass

$$\lim_{n\to\infty} \sup_{z\in D_{z_0}} |f_n(z) - f(z)| = 0.$$

Die Funktionenfolge heißt auf D **gleichmäßig beschränkt**, falls eine Konstante $c \in \mathbb{R}$ existiert, so dass $|f_n(z)| \leq c$ für alle $z \in D$ und $n \in \mathbb{N}$.
Zu guter Letzt noch der etwas schwächere Begriff **Konvergenz in Kapazität** [Stahl, 1997]. Diese sei erfüllt, wenn für jedes $\epsilon > 0$ und jede kompakte Teilmenge $V \subset D$ auch

$$\lim_{n\to\infty} \operatorname{cap}\big(\{z \in D : |f_n(z) - f(z)| > \epsilon\}\big) = 0.$$

Hier ist die (logarithmische) **Kapazität** einer kompakten Teilmenge $U \subset \mathbb{C}$ definiert als

$$\operatorname{cap}(U) := \lim_{k\to\infty} \inf_{h\in\mathbb{P}_k} \max_{z\in U} |h(z)|^{1/k}$$

und hier $\mathbb{P}_k$ die Menge aller Polynome $h(z) = z^k + \ldots$ vom Grad $k \in \mathbb{N}$ bezeichnet [Pommerenke, 1973]. Für eine beliebige Teilmenge $W \subset \mathbb{C}$ sei die (innere) Kapazität

$$\operatorname{cap}(W) := \sup\{\operatorname{cap}(K) : K \subset W \text{ und kompakt}\}.$$

Es folgt nun ein Theorem von Baker, welches besagt wann eine Padé-Folge auch außerhalb des Konvergenzradius einer Mac-Laurin-Reihe konvergiert:

Satz 4.38. Sei $f(z)$ eine in 0 analytische Funktion und desweiteren $\big([L_n/M_n](z)\big)_{n\in\mathbb{N}}$ eine Folge von Padé-Approximanten von $f(z)$ und für $n \to \infty$ strebt auch $L_n + M_n \to \infty$.
Sei nun D eine 0 enthaltende, einfach zusammenhängende und beschränkte Menge in der $\big(|[L_n/M_n](z)|\big)_{n\in\mathbb{N}}$ gleichmäßig beschränkt ist.
Dann folgt daraus, dass $\big([L_n/M_n](z)\big)_{n\in\mathbb{N}}$ auf jedem kompakten Gebiet $R \subset D$ gleichmäßig gegen $f(z)$ konvergiert. Damit wird der Definitionsbereich von $f(z)$ erweitert und $f(z)$ ist analytisch in R.

Beweis. Für einen ausführlichen Beweis siehe [Baker, 1964]. ■

Ersichtlich ist also, obwohl der Konvergenzradius der Reihenentwicklung einer analytischen Funktion endlich sein kann, können über die Padé-Approximanten auch darüber hinaus interessante Schlüsse gezogen werden.

Die im CFR-Algorithmus definierten Kettenbrüche bilden eine Stufenfolge in der Padé-Tabelle. Diese enthält die Hauptdiagonale als eine Teilfolge. Deshalb wird nun eine Vermutung betrachtet, die sich speziell auf die Approximation meromorpher Funktionen über die Hauptdiagonale der Padé-Tabelle bezieht:

Satz 4.39. Baker-Gammel-Wills Vermutung [Baker et al., 1961] (hat sich allerdings als falsch erwiesen)
Gegeben sei eine auf der Einheitsscheibe $\mathbb{D}$ meromorphe Funktion $f(z)$ und deren Polstellenmenge P. Es wird mit $([N,N](z))_{N\in\mathbb{N}}$ die diagonale Folge von Padé-Approximanten von $f(z)$ bezeichnet. Achtung: Die einzelnen Approximanten müssen hier nicht existieren.
So soll es eine unendliche Teilfolge $([N',N'](z))_{N'\in\mathbb{N}}$ von $([N,N](z))_{N\in\mathbb{N}}$ geben, die lokal für alle $z \in \mathbb{D}\setminus P$ gleichmäßig gegen $f(z)$ konvergiert.

Beweis. Diese Vermutung wurde von Lubinsky [Lubinsky, 2003] durch ein Gegenbeispiel widerlegt. ■

Auf dem Gegenbeispiel dieser Vermutung aufbauend, lieferte Baker eine Verschärfung. So könnte durch die Kombination endlich vieler unendlicher Teilfolgen doch eine gleichmäßige Konvergenz sichergestellt werden [Lubinsky, 2021].
Ein fundamentaler Satz, der allerdings Konvergenz in Kapazität verwendet, ist der Satz von Nuttal-Pommerenke [Pommerenke, 1973]. Eine brauchbare Folgerung lautet [Lubinsky, 2014]:

Satz 4.40. Gegeben sei f eine auf $\mathbb{C}\setminus P$ und in 0 analytische Funktion, wobei $P \subset \mathbb{C}$ und $\operatorname{cap}(P) = 0$ gilt. Dann folgt für $R, \epsilon > 0$ beliebig, dass

$$\lim_{N\to\infty} \operatorname{cap}\big(\{z : |z| \le R \text{ und } |f(z) - [N,N](z)| \ge \epsilon^N\}\big) = 0.$$

(Somit stellt ϵ^N auch die Konvergenz in Kapazität sicher.)

Ursprünglich wurde dieser Satz von Nuttal [Nuttall, 1970] für Konvergenz im Maß (von paradiagonalen Approximanten $[N, N+J]$ und $J \ge 0$) gezeigt und erst durch Pommerenke [Pommerenke, 1973] für Konvergenz in Kapazität verallgemeinert. Der Vorteil liegt darin, dass sich nun auch die Pole der Approximanten im Konvergenzbereich von f häufen dürfen [Stahl, 1997]. Laut Stahl soll nun auch die Baker-Gammel-Wills Vermutung mit Konvergenz in Kapazität anstatt der lokal gleichmäßigen Konvergenz funktionieren. Eine Auflistung und Diskussion ähnlich nötiger Verschärfungen findet sich in [Lubinsky, 2014]. Aufgrund dieser Problematik gibt es auch Sätze, die auf schwächeren Konvergenzbegriffen beruhen, wie der bereits angesprochenen Konverzgenz im Maß [Lubinsky, 1995] (analog zu Konvergenz in Kapazität [Meyer, 1976]) oder der punktweisen Konvergenz (siehe [Baker et al., 1996]).

4.6 Zusammenhang CFR-Algorithmus

Zur Beurteilung der Kettenbruch-Regression wurden in den letzten Publikationen verschiedene physikalische Datenmengen verwendet. Diese haben sich über reelle Funktionen in mehreren Variablen beschreiben lassen. Selbst die in dieser Arbeit vorgestellte Implementierung bezieht sich auf solche. Deshalb soll angemerkt werden, dass sich die auf analytischen (komplexen) Funktionen hergeleiteten Resultate auch auf beliebig oft stetig differenzierbare (reelle) Funktionen anwenden lassen. Dies liegt vor allem daran, dass sich ja beide als MacLaurin Reihe entwickeln lassen.

In der Veröffentlichung [Sun und Moscato, 2019] des CFR-Algorithmus wird angesprochen, dass sich meromorphe Funktionen mit mehreren Polstellen gut über Kettenbrüche approximieren lassen und dies auf der Padé-Theorie beruht. In **Kapitel 4.1** wurde ersichtlich, wie meromorphe Funktionen definiert sind und auch wie sich diese charakterisieren lassen. Ebenfalls konnte man gebrochenrationale Funktionen als solche identifizieren.

In **Kapitel 4.2** wurde gezeigt, wie sich eine analytische Funktion über spezielle gebrochenrationale Funktionen approximieren lässt (sofern existent). Solche sogenannten Padé-Approximanten sollen dabei bis zu einer möglichst hohen Ordnung dieselbe Reihenentwicklung aufweisen. Speziell wurde auch auf ein Konzept für die Feststellung deren Existenz und deren anschließenden Bestimmung eingegangen. Zum Schluss des Unterkapitels wurde noch festgestellt, dass eine analytische Funktion beliebig genau durch eine Folge von Padé-Approximanten angenähert werden kann, die sich auf Nebendiagonalen in der Padé-Tabelle bewegen. Diese dürfen (später) als analytische Kettenbrüche aufgefasst werden.

Im Weiteren (**Kapitel 4.3**) wurde gezeigt, wie sich formale Potenzreihen (jede analytische Funktion kann ja als solche aufgefasst werden) in (reguläre) Kettenbrüche entwickeln lassen. Man hat auch gesehen, dass dies nur möglich ist, sofern der konstante Term der Reihe ungleich 0 ist. Schließlich lässt sich laut Bernoulli eine Folge von rationalen Funktionen (speziell hier wieder Padé-Approximanten) in der benachbarte Glieder voneinander verschieden sind, mit der Konvergentenfolge eines verallgemeinerten Kettenbruchs darstellen. Kombiniert mit dem vorigen Kapitel gibt es nun für reellsymmetrische analytische Funktionen sogar einen regulären Kettenbruch mit dieser Eigenschaft. Das bedeutet für GP, man kann solche Funktionen zuerst durch Kettenbrüche mit niedriger Tiefe approximieren und bei anschließender Erweiterung muss nur noch über die neu hinzugekommenen Koeffizienten optimiert werden. Interessant ist auch, dass sich jeder Kontinuant aus einem verallgemeinerten Kettenbruch über Rekursion der beiden vorhergehenden Kontinuanten darstellen lässt. Dies würde Anlass für die Definition eines Rekombinationsoperators liefern. Meromorphe Funktionen lassen sich lokal als formale Laurentreihen darstellen. Da sich nun solche Reihen invertieren lassen, kann immer eine Kettenbruchentwicklung durchgeführt werden. Selbst bei analytischen Funktionen ist dadurch eine vollständige Entwicklung möglich. Jedoch muss hier die Struktur der Kettenbrüche im CFR Algorithmus gezielt verändert werden, um eine Approximation zu ermöglichen. Als ein gutes Beispiel ist hier die ungerade Sinusfunktion behandelt worden.

In **Kapitel 4.4** wurden numerische Verfahren zur Berechnung eines Padé-Approximanten vorgestellt. Bei der Kettenbruch-Regression spricht man von einem sogenannten Koeffizientenproblem, da hier nicht nur einzelne Auswertungen reichen, sondern eine vollständige Bestimmung der Approximanten erfolgen muss. Es wurden gebräuchliche Algorithmen vorgestellt, die auch für eine Wahl der Operatoren und der lokalen Suche in CFR von Interesse sein können.

Es gibt eine Vielzahl an Konvergenzbegriffen, die zur Untersuchung einer Folge von Padé-Approximanten eingesetzt werden. **Kapitel 4.5** behandelt wichtige Definitionen und Sätze, die eine Erweiterung des Definitionsbereichs einer analytischen Funktion sicherstellt. Ein Satz von Baker und eine Folgerung des Theorems von Nuttal-Pommerenke liefern Anhaltspunkte, dass sich paradiagonale Folgen der Padé-Tabelle (CFR bildet solche) besonders eignen. Diese sind mit Vorwissen der Zielfunktion und der Folge an Padé-Approximanten behaftet und liefern einen theoretischen Wert für die Kettenbruch-Regression.

5 Analyse der Nelder-Mead Methode

In diesem Kapitel soll die Nelder-Mead-Methode etwas genauer untersucht werden. Verwendet werden hierfür größere Datenmengen mit ein und zwei Eingangsvariablen. Die von ihnen repräsentierten Funktionen sollen durch Kettenbrüche, unter alleiniger Verwendung des Downhill-Simplex-Verfahrens (also ohne genetische Operatoren), approximiert werden. Von Interesse ist der Nutzen der lokalen Suche im Zusammenhang mit der Anzahl an Eingangsvariablen und der verwendeten Kettenbruchtiefe.
Zu Beginn wird hierfür eine festgelegte Anzahl an Kettenbrüche mit ebenfalls festgewählter maximaler Kettenbruchtiefe generiert. Dabei werden die Konstanten und Koeffizienten mit zufälligen Werten aus einem vordefinierten Intervall belegt. Anschließend wird über die gesamte Population optimiert, wobei das Nelder-Mead-Verfahren jeweils eine genaue Anzahl an Iterationen ausführt.
Ebenfalls soll auch verglichen werden, wie sich die optimierten Lösungen zu den dazugehörenden Padé-Approximanten verhalten. Hierfür werden die Herleitungen aus dem vorigen Kapitel angewendet. Dies wird aus notationellen Gründen nur für die eindimensionale Funktion durchgeführt.

5.1 Sinusfunktion (univariat)

Es soll hier die Funktion $f_1(x) = 0,3\sin(2\pi x)x$ im Intervall $[-1,1]$ angenähert werden (Abb. 13). Die Versuchsdaten für das Downhill-Simplex Verfahren ergeben sich durch eine Unterteilung der Definitionsmenge in äquidistante Stützstellen mit einem Abstand von $0,001$ und der exakten Auswertung daran. Zur Bestimmung der Padé-Approximanten muss $f_1(x)$ in eine Reihe entwickelt werden. Mit Hilfe der Potenzreihendarstellung von $\sin(x) = \sum_{n=0}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!}$ erhält man

$$
\begin{aligned}
f_1(x) &= \sum_{n=0}^{\infty} \underbrace{\frac{0,3(-1)^n(2\pi)^{2n+1}}{(2n+1)!}}_{:=a_n} x^{2n+2} \\
&= 0,3(2\pi)x^2 - \frac{0,3(2\pi)^3}{3!}x^4 + \frac{0,3(2\pi)^5}{5!}x^6 - \frac{0,3(2\pi)^7}{7!}x^8 + \frac{0,3(2\pi)^9}{9!}x^{10} - \dots
\end{aligned}
$$

und hat bereits die nötigen Koeffizienten zur Bestimmung der Padé-Approximanten hergeleitet.

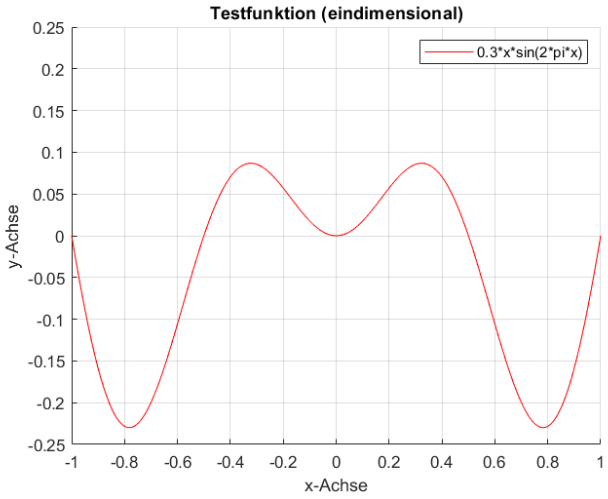
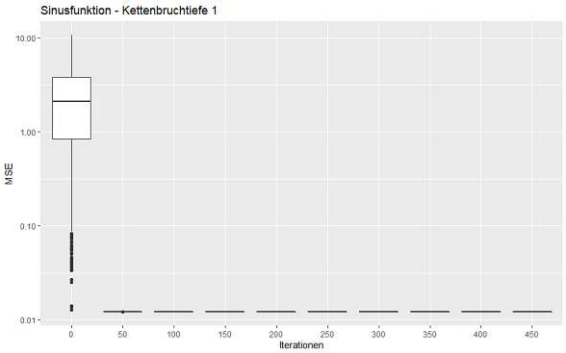


Abbildung 13: Darstellung der Testfunktion

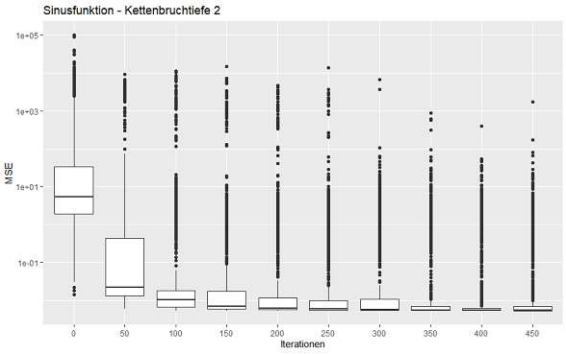
Test über die Iterationszahl

Im Folgenden wird die Auswirkung der Iterationszahl für die Nelder-Mead-Methode untersucht. Um hier keinen frühzeitigen Abbruch zu erhalten, wurde die Abweichungsgenauigkeit als $0,0$ definiert. Jeder Testversuch wird mit einer Anzahl von 1000 Kettenbrüchen geführt. Variiert wird über die Kettenbruchtiefe von 1 bis 5 und die Koeffizienten werden zufällig aus dem Intervall (-3,3) gewählt. (Wie auch im CFR-Algorithmus in [Moscato et al., 2021] empfohlen)

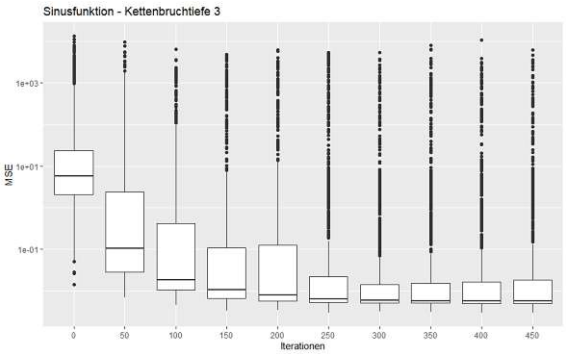
Durch die einfache Struktur von Kettenbrüchen in nur einer Variablen der Tiefe 1, liefert eine Optimierung durch das Nelder-Mead-Verfahren nahezu immer dieselbe Lösung. Der MSE-Wert der besten Lösungskandidaten liegt bei $0,012142$ gerundet. Man sieht, dass bereits weniger als 50 Iterationen ausreichen, bis keine nennenswerte Verbesserung durch lokale Suche mehr erfolgt.



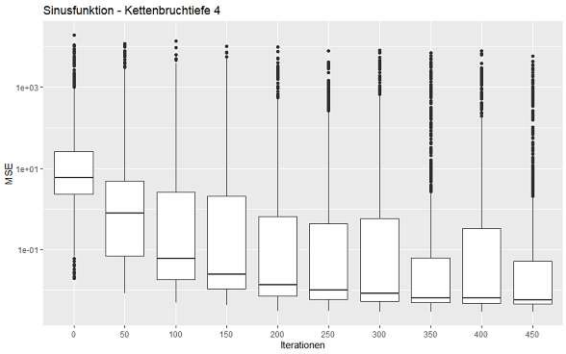
Betrachtet man nun Kettenbrüche der Tiefe 2 so ist sofort erkennbar, dass alleinige Anwendung der Downhill-Simplex-Methode nicht mehr ausreicht, um den besten Lösungskandidaten zu finden. Mit $0,005414$ wurde der MSE-Wert der besten Lösung bestimmt und dies erfolgte im letzten Testlauf mit 450 Iterationen. Im dritten Testlauf mit 100 Iterationen sind unteres, mittleres und oberes Quartil bereits nahe zusammen und so sollte diese Anzahl für CFR ausreichen.



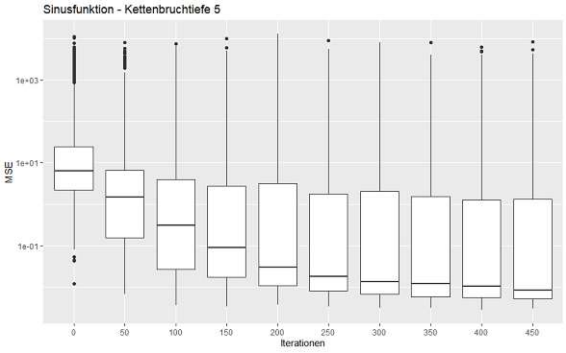
Für eine Tiefe von 3 liegt der MSE-Wert der besten Lösung bei $0,002979$ und wurde mit 450 Iterationen gefunden. Das 0,25-Quantil und der Median der Kettenbrüche liegen nach 150 Iterationen bereits sehr nahe am MSE-Wert der besten Lösung und sollte somit für CFR genügen.



Der niedrigste MSE-Wert eines in der Population mit Kettenbruchtiefe 4 gefundenen Individuums liegt bei $0,002975$ und wurde nach 450 Iterationen erreicht. Man kann erkennen, dass die Verteilung der MSE-Werte der Kandidaten nach 200 Iterationen sehr ähnlich zu denen mit 400 sind. Um Effizienz beizubehalten, reichen wohl 200 Iterationen mit Nelder-Mead aus.



Zu guter Letzt noch eine Untersuchung der Kettenbrüche mit Tiefe 5. Hierbei liegt der MSE-Wert der besten Lösung bei $0,002813$ und wurde im vorletzten Lauf mit 400 Iterationen bestimmt. Das 0,75-Quantil bleibt über höhere Iterationszahlen nahezu gleich. Der Median macht bei 200-250 doch noch einen deutlichen Sprung, weshalb 250 Iterationen bei Nelder-Mead genügen sollten.



Zusammenfassend kann also gesagt werden, dass mit zunehmender Tiefe auch ein langsameres Konvergenzverhalten zu beobachten ist. Dies liegt wie erwartet an der höheren Anzahl an Koeffizienten. Der MSE Wert der besten gefundenen Lösung wird mit zunehmender Kettenbruchtiefe nur geringfügig besser. Betrachtet man den Verlauf des Median mit zunehmender Kettenbruchtiefe, so ist ersichtlich, dass die meisten Versuche gegen denselben MSE Wert konvergieren. Eine Iterationszahl von 250 des Nelder-Mead-Verfahrens bietet wohl ausreichend Optimierungspotential. Dies ist allerdings nur eine erste Beurteilung. Deshalb wird im Folgenden noch eine Funktion in zwei Eingangsvariablen und schließlich auch noch unsere Datenmenge (in drei Eingangsvariablen) diesem Test unterzogen.

Vergleich zu Padé-Approximanten

Im Folgenden wird die C-Tabelle von $f_1(x)$ bestimmt. Dies dient dazu, die Padé-Theorie aus dem vorigen Kapitel zu veranschaulichen und des Weiteren auch die Existenz der hier benötigten Approximanten sicherzustellen. Um ein Mitschleifen der Konstanten zu ersparen, wird bereits zu Beginn $c_1 := 0.3$, $c_2 := 2\pi$ definiert und man erhält:

M\L	0	1	2	3	4	5
0	1	1	1	1	1	1
1	0	0	c_1c_2	0	$-\frac{c_1c_2^3}{6}$	0
2	0	0	$-c_1^2c_2^2$	$-\frac{c_1^2c_2^4}{6}$	$-\frac{c_1^2c_2^6}{36}$	$-\frac{c_1^2c_2^8}{720}$
3	0	0	$-c_1^3c_2^3$	0	$\frac{7c_1^3c_2^9}{2160}$	0
4	0	0	$c_1^4c_2^4$	$\frac{7c_1^4c_2^8}{360}$	$\frac{49c_1^4c_2^{12}}{129600}$	$\frac{11c_1^4c_2^{16}}{15552000}$
5	0	0	$c_1^5c_2^5$	0	$-\frac{31c_1^5c_2^{15}}{777600}$	0

Tabelle 3: C-Tabelle der Sinusfunktion

In Kapitel 4.2 wurde bereits hergeleitet, wie sich die Einträge bestimmen lassen. So besagt Lemma 4.19, dass die erste Zeile immer konstant 1 ist und für die 2. Zeile gilt $C[L/1] = a_L$ für alle $L \geq 0$. Die erste Spalte ergibt sich aus dem Koeffizienten a_0 und deren Spaltenindex als Potenz, womit alle bis auf den ersten Eintrag gleich 0 sind. Der Rest lässt sich anschließend über die Frobenius Formel (Gleichung (8)) bestimmen.
Da nun alle Einträge in der oberen Nebendiagonale ungleich 0 sind, existieren laut Satz 4.21 alle für den CFR-Algorithmus relevanten Padé-Approximanten.
Es soll auch hier nochmal die Darstellung über die Determinante aus Satz 4.17 veranschaulicht werden. So ergibt sich $C[5/4]$ als

$$Q^{[5/4]}(0) = \left| \begin{pmatrix} a_2 & a_3(=0) & a_4 & a_5(=0) \\ a_3(=0) & a_4 & a_5(=0) & a_6 \\ a_4 & a_5(=0) & a_6 & a_7(=0) \\ a_5(=0) & a_6 & a_7(=0) & a_8 \end{pmatrix} \right| = \frac{11c_1^4c_2^{16}}{15552000},$$

wobei hier bereits nach der letzten Zeile entwickelt wurde. Eine weitere Anwendung des Laplace'schen Entwicklungssatzes, desweiteren der Regel von Sarrus und man erhält den Koeffizienten auf der rechten Seite.

Nach Satz 4.17 lässt sich daraus nun auch das Nennerpolynom $B[5/4](x)$ des Padé-Approximanten $[5/4](x)$ bestimmen als:

$$\begin{aligned}\frac{Q^{[5/4]}(x)}{Q^{[5/4]}(0)} &= \left|\begin{pmatrix} a_2 & a_3(=0) & a_4 & a_5(=0) & a_6 \\ a_3(=0) & a_4 & a_5(=0) & a_6 & a_7(=0) \\ a_4 & a_5(=0) & a_6 & a_7(=0) & a_8 \\ a_5(=0) & a_6 & a_7(=0) & a_8 & a_9(=0) \\ x^4 & x^3 & x^2 & x & 1 \end{pmatrix}\right| \frac{1}{C[5/4]} \\ &= \frac{11c_2^4}{5880}x^4 + \frac{360c_2^2}{5880}x^2 + 1\end{aligned}$$

Man sieht hier, dass durch die Division von $C[5/4]$ auch die gewünschte Normierung erfolgt. Analog kann auch das Zählerpolynom $A[5/4](x)$ bestimmt werden:

$$\begin{aligned}\frac{P^{[5/4]}(x)}{Q^{[5/4]}(0)} &= \left|\begin{pmatrix} a_2 & a_3(=0) & a_4 & a_5(=0) & a_6 \\ a_3(=0) & a_4 & a_5(=0) & a_6 & a_7(=0) \\ a_4 & a_5(=0) & a_6 & a_7(=0) & a_8 \\ a_5(=0) & a_6 & a_7(=0) & a_8 & a_9(=0) \\ 0 & a_2x^5 & a_2x^4 & a_2x^3 + a_4x^5 & a_2x^2 + a_4x^4 \end{pmatrix}\right| \frac{1}{C[5/4]} \\ &= -\frac{31c_1c_2^3}{294}x^4 + c_1c_2x^2\end{aligned}$$

Da nun solche Determinantenbestimmungen immer aufwändiger werden, lässt sich hier effizienter über die in Kapitel 4.4 angesprochenen Methoden vorgehen. In der folgenden Tabelle 4 sind nun die interessanten Approximanten angegeben:

M\L	0	1	2	3	4	5
0	-	0	-	-	-	-
1	-	-	$c_1c_2x^2$	-	-	-
2	-	-	-	$\frac{c_1c_2x^2}{\frac{c_2^2}{6}x^2+1}$	-	-
3	-	-	-	-	$\frac{-\frac{7c_1c_2^3}{60}x^4+c_1c_2x^2}{\frac{c_2^2}{20}x^2+1}$	-
4	-	-	-	-	-	$\frac{-\frac{31c_1c_2^3}{294}x^4+c_1c_2x^2}{\frac{11c_2^4}{5880}x^4+\frac{360c_2^2}{5880}x^2+1}$
5	-	-	-	-	-	-

Tabelle 4: Padé-Approximanten der Sinusfunktion

Es ist hier sofort ersichtlich, dass sämtliche Zähler- und Nennerpolynome von gerader Ordnung sind. Dies folgt daraus, dass $f_1(x)$ eine gerade Funktion ist und aus der Entwicklung in einen regulären Kettenbruch (Gleichung (14)). Nun können die Padé-Approximanten mit den von HeuristicLab vorhandenen Daten ausgewertet und die MSE Werte bestimmt werden:

M\L	0	1	2	3	4	5
0	-	0,01442	-	-	-	-
1	-	-	0,88039	-	-	-
2	-	-	-	0,07172	-	-
3	-	-	-	-	0,52342	-
4	-	-	-	-	-	0,08489
5	-	-	-	-	-	-

Tabelle 5: MSE der Padé-Approximanten

In Tabelle 5 weist der Padé-Approximant $[1/0](x)$, welcher der konstanen Nullfunktion entspricht, den geringsten MSE-Wert auf. Durch die Zunahme der Ordnung der Approximanten muss nicht zwingend eine Abnahme des Fehlers erfolgen, wie $[4/3](x)$ zeigt. Um den Zusammenhang zu veranschaulichen, sind die Funktionen in Abbildung 14 dargestellt:

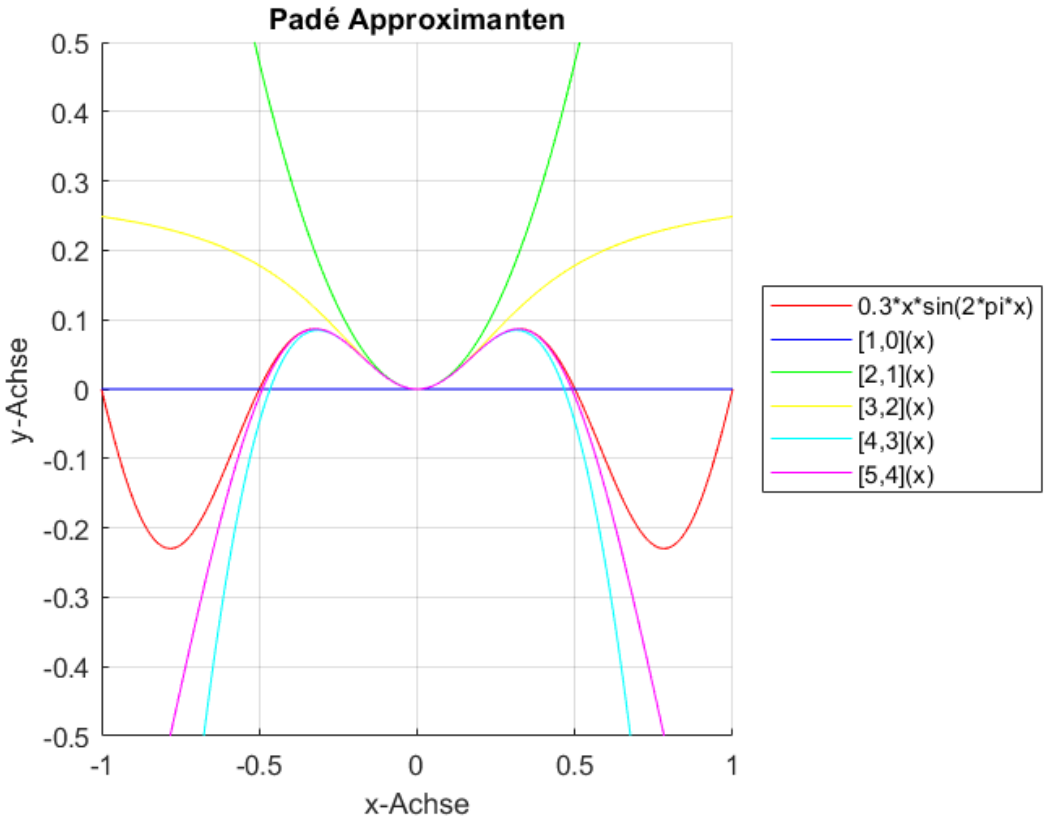


Abbildung 14: Vergleich der Padé-Approximanten

Die Padé-Approximanten liefern lokal um 0 eine gute Annäherung an $f_1(x)$. Ansonsten scheinen mit höherer Ordnung die Funktionswerte potentiell zuzunehmen und somit den MSE-Wert nach oben zu treiben. Ein ähnliches Verhalten kennt man bereits von der Reihenentwicklung der Sinusfunktion, weshalb diese zum Vergleich ebenfalls dargestellt sind:

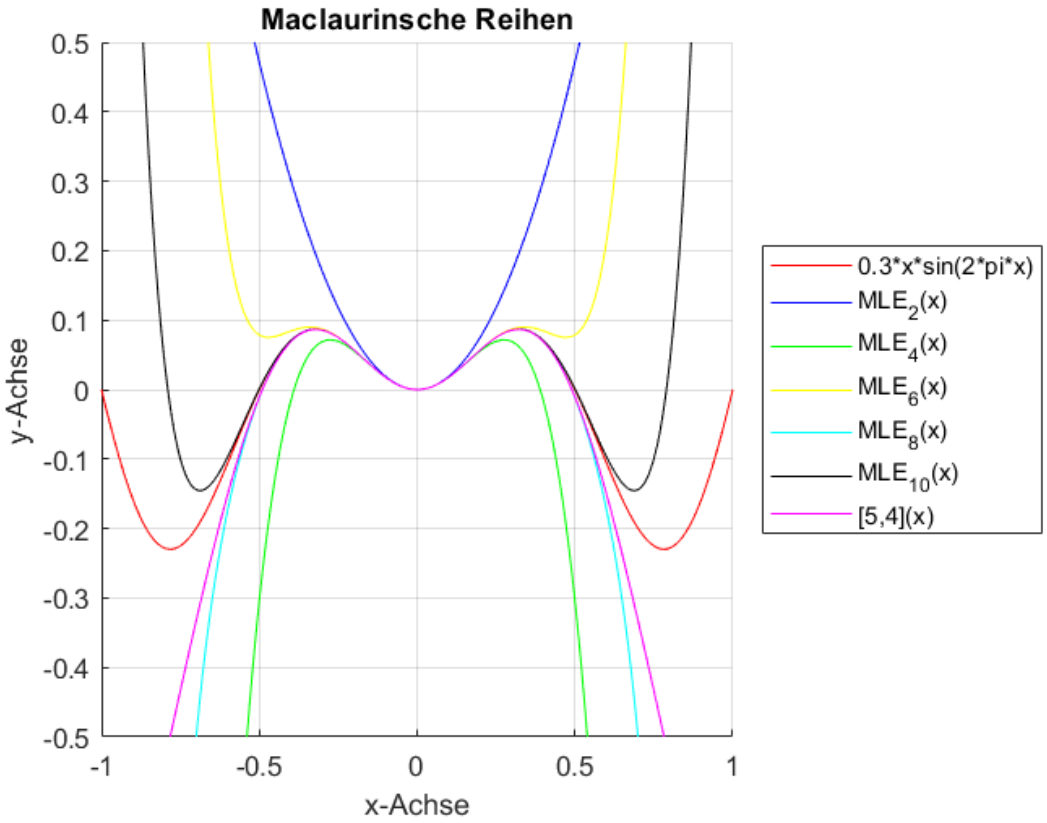


Abbildung 15: Vergleich der Maclaurinschen Reihen

In Abbildung 15 sind die geraden MacLaurin Reihen Entwicklungen (MLE) von 2 bis 10 abgebildet. Der Padé-Approximant $[5/4](x)$, der als Reihenentwicklung bis zum Grad von 8 mit $\text{MLE}_8(x)$ übereinstimmt, wurde ebenfalls aufgetragen. Durch den Vergleich beider Kurven miteinander, erkennt man das etwas bessere Extrapolationsverhalten von $[5/4](x)$.

Die Tatsache, dass sich die Reihenentwicklung von der durch den CFR-Algorithmus gewonnenen Kettenbrüchen stark unterscheidet, ist in Abbildung 16 ersichtlich:

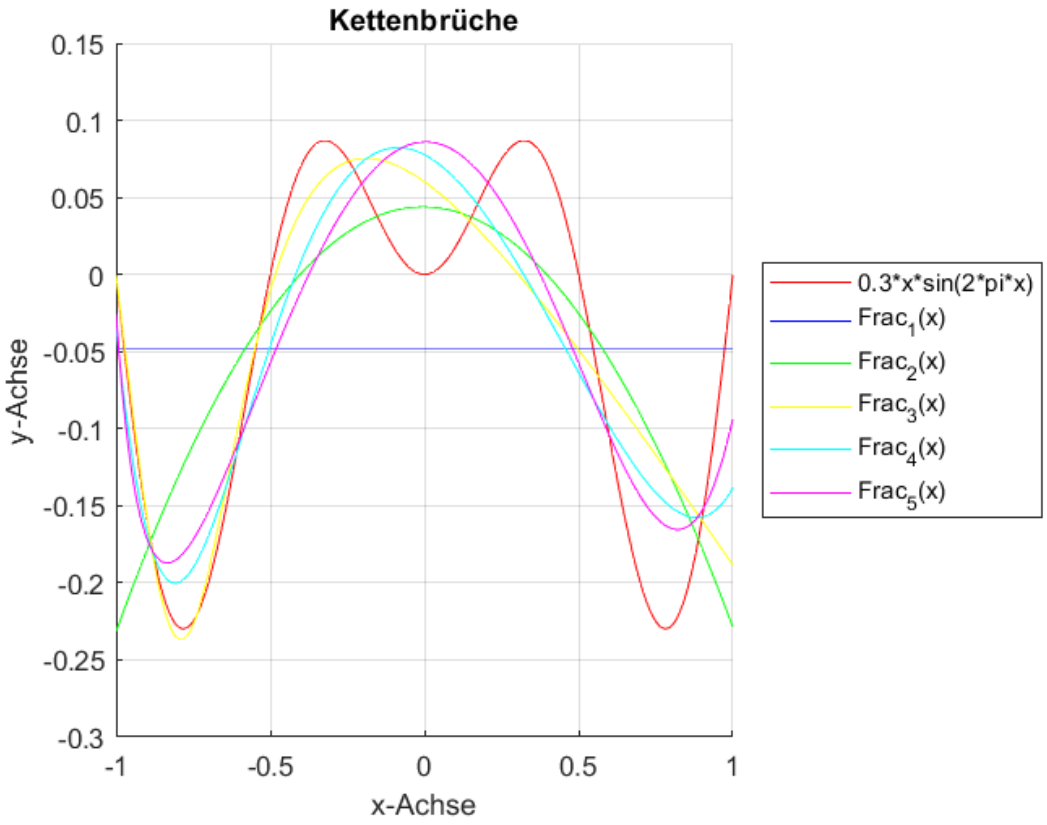


Abbildung 16: Vergleich der Kettenbrüche

Die Kettenbrüche 1-5 (Frac) sind die jeweils besten Kandidaten aus dem im vorigen Kapitel durchgeführten Tests über die Iterationsanzahl des Nelder-Mead-Verfahrens. Man sieht einen wesentlichen Unterschied zwischen Funktionen, die als Annäherung an Reihenentwicklungen gedacht sind und jener, die lediglich den MSE minimieren sollen.

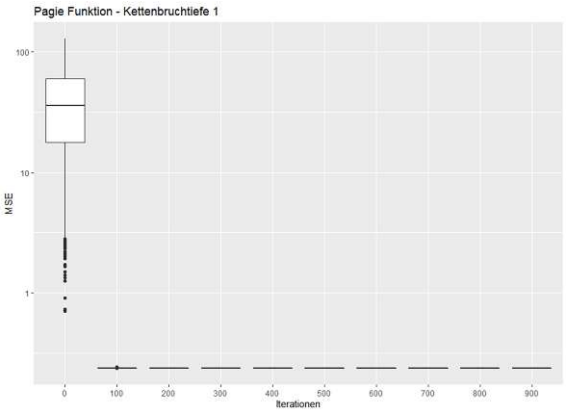
5.2 Pagie Funktion (bivariat)

Für ein univariates Beispiel wurde bereits gezeigt, dass ein Downhill-Simplex-Verfahren mit 250 Iterationen zur Optimierung ausreicht. Im Vergleich dazu nun eine bivariate Funktion (Pagie Funktion) aus [Pagie und Hogeweg, 1997], definiert als

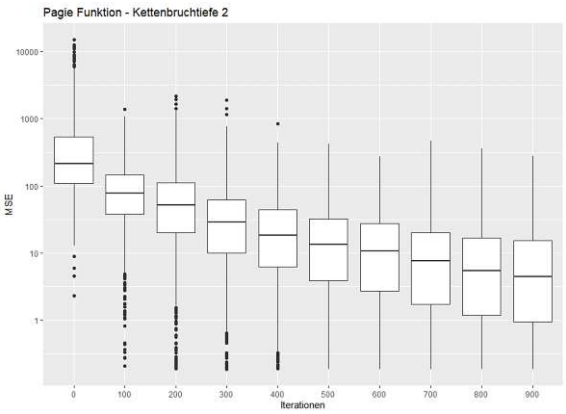
$$f_2(x, y) := \frac{1}{1 + x^{-4}} + \frac{1}{1 + y^{-4}}.$$

Die Versuchsdaten ergeben sich aus der Definitionsmenge $[-5, 5] \times [-5, 5]$ durch zufällige Wahl von 1000 Elementen und den $26 \times 26 = 676$ Elementen mit einem äquidistanten Abstand von $0,4$ und deren exakten Auswertung daran. Um einen Vergleich anstellen zu können, wird ebenfalls wieder mit denselben Voraussetzungen wie aus Kapitel 5.1 vorgegangen. Im Folgenden sind die Boxplots zu dem mit Nelder-Mead optimierten Kettenbrüchen der Tiefe 1 bis 4 erläutert:

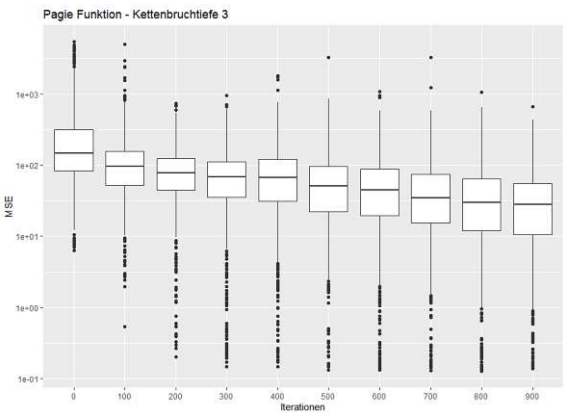
Aufgrund der in Kettenbrüchen der Tiefe 1 wenig zu variierenden Koeffizienten, ist bereits eine lokale Suche mit 100 Iterationen ausreichend. Bis auf wenige Nachkommastellen genau, erhält man Kandidaten mit einem MSE-Wert von $0,237090$.



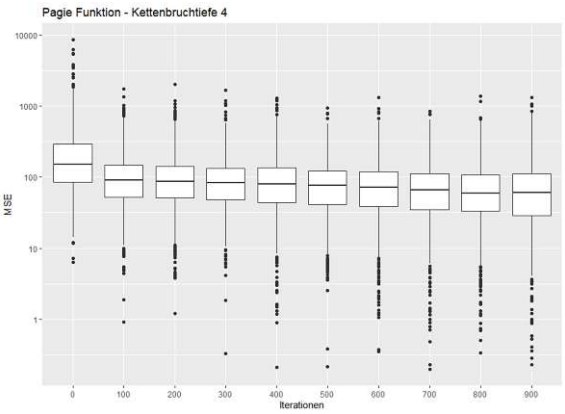
Im Gegensatz zu vorigem Boxplot sind Kettenbrüche der Tiefe 2 nicht mehr ausreichend über Nelder-Mead optimierbar und bleiben in lokalen Optima stecken. Betrachtet man alle drei Quartile, so scheint hier keine Konvergenz des MSE über die Iterationszahl einzutreten. Das untere Quartil sinkt von einem MSE Wert von 100 (bei 100 Iterationen) bis auf einem mit 1 (bei 900 Iterationen) ab. Der beste Lösungskandidat hat einen MSE von $0,185113$ nach 700 Iterationen.



Im Gegensatz zu einer Tiefe von 2 ist bei der von 3 nur noch ein langsames Abklingen ersichtlich. So liegt das untere Quartil im Versuch mit 0 Iterationen bei etwa einem MSE von 100. Dies war auch bei einer Kettenbruchtiefe von 2 der Fall. Das untere Quartil bei 900 Iterationen liegt hingegen nur bei etwa 10. Der beste Kandidat wurde im Versuch von 800 Iterationen gefunden und weist einen MSE-Wert von $0,127159$ auf.



Bei einer Kettenbruchtiefe von 4 ist ein Unterschied der Quartile zwischen 100 und 900 Iterationen kaum noch merkbar. Nur wenige Läufe finden ein Optimum und viele bleiben bei schlechten Optima stecken. Erwähnt sei auch, dass die beste Lösung im Versuch über 700 Iterationen mit einem MSE-Wert von $0,198423$ Iterationen gefunden wurde.



Aus den Boxplots und den darin eingezeichneten Quartilen ist gut ersichtlich, dass mit Zunahme der Kettenbruchtiefe die Verbesserungsspanne der MSE-Werte abnimmt. Es kann also gesagt werden: Je höher die Kettenbruchtiefe ist, desto geringer ist der Einfluss einer einzelnen Nelder-Mead-Iteration. Da jedoch so ein Optimierungsschritt eine relativ große Testdatenmenge zur Fehlerberechnung verwendet, ist dieser auch mit einem hohen Rechenaufwand verbunden. Deshalb soll auch für diese bivariate Funktion $f_2(x)$, die Anzahl der Iterationsschritte bei 250 belassen werden. Für grobe Sprünge des MSE-Wertes müssen die genetischen Operatoren sorgen.

6 Anwendung auf die Daten

In diesem Kapitel soll der CFR Algorithmus auf die Heißkompressionsdaten einer AA6082 Legierung angewendet werden. Die Versuche wurden vom Leichtmetallkompetenzzentrum in Ranshofen durchgeführt und aufgezeichnet. Abgezielt wird dabei auf eine möglichst genaue Beschreibung der Daten und auf die Erkenntnis der Inter- und Extrapolationseigenschaften. Um auch feststellen zu können wie gut CFR im Vergleich zu bekannten Regressionsalgorithmen abschneidet, wird die in [Schützeneder, 2020] beschriebene Datenaufbereitung verwendet. Dort wurden bereits, mit Hilfe derselben Datenmenge, Experimente in genetischer Programmierung und symbolischer Regression durchgeführt und daraus gewonnene Modelle vorgestellt. Das dort verwendete Feature-Engineering soll im Folgenden auch noch genauer behandelt werden. Um ein besseres Verständnis der Heißkompression zu erhalten, wird gleich zu Beginn auf den Versuchsaufbau und dem allgemeinen Kurvenverlauf eingegangen.

6.1 Heißkompressionstests

Um einen Werkstoff industriell nutzen zu können, müssen dessen Eigenschaften bekannt sein. So sind die Festigkeit, die Elastizität sowie die Plastizität für die Umformbarkeit von großer Bedeutung.

Abbildung 17: Prüfvorbereitung im LKR in Ranshofen

Zu diesem Zweck werden Materialproben in einem Zugversuch oder auch bei einem Stauchversuch getestet. Lässt sich die Probe vor Beginn des Versuches auf eine vordefinierte Temperatur erwärmen, spricht man auch von einer Heißkompression. Durchgeführt wurden die Heißkompressionstests mit einem Abschreck- und Umformdilatometer DIL 805A/D von TA Instruments (siehe Abbildung 17).
Dieses liefert eine, wie in Abbildung 18, dargestellte Spannungs-Dehnungskurve. Um dann eine Weiterverarbeitung, wie zum Beispiel in der FEM-Simulation zu ermöglichen, wird dabei auf eine einfache mathematische Darstellung Wert gelegt. So wäre eine explizite Funktion

$$k_f = f(T, \phi, \dot{\phi})$$

in den Variablen der Temperatur T, der Dehnung ϕ und der Umformgeschwindigkeit $\dot{\phi}$ gewünscht, die für jeden Materialzustand dessen Festigkeit zurückliefert. Da bei einer Variation der Temperatur und der Umformgeschwindigkeit eine große Menge an Daten generiert wird, kommt hier auch das maschinelle Lernen ins Spiel.

6.2 Spannungsdehnungsdiagramm und Fließkurven

Bei einem Zug- bzw. Druckversuch wird ein Prüfling in eine Prüfvorrichtung eingespannt. Dieser hat dabei oft eine zylindrische Form. Anschließend wird dieser mit konstanter Geschwindigkeit $\dot{\phi}$ so lange gezogen, bis er schließlich reißt bzw. auf eine vorab definierte Länge gestaucht wird. Kontinuierlich wird dabei die Zugfestigkeit σ (in Newton / mm^2) über dessen Anteil der Längenänderung ϕ (bzw. ϵ in Prozent) aufgezeichnet:

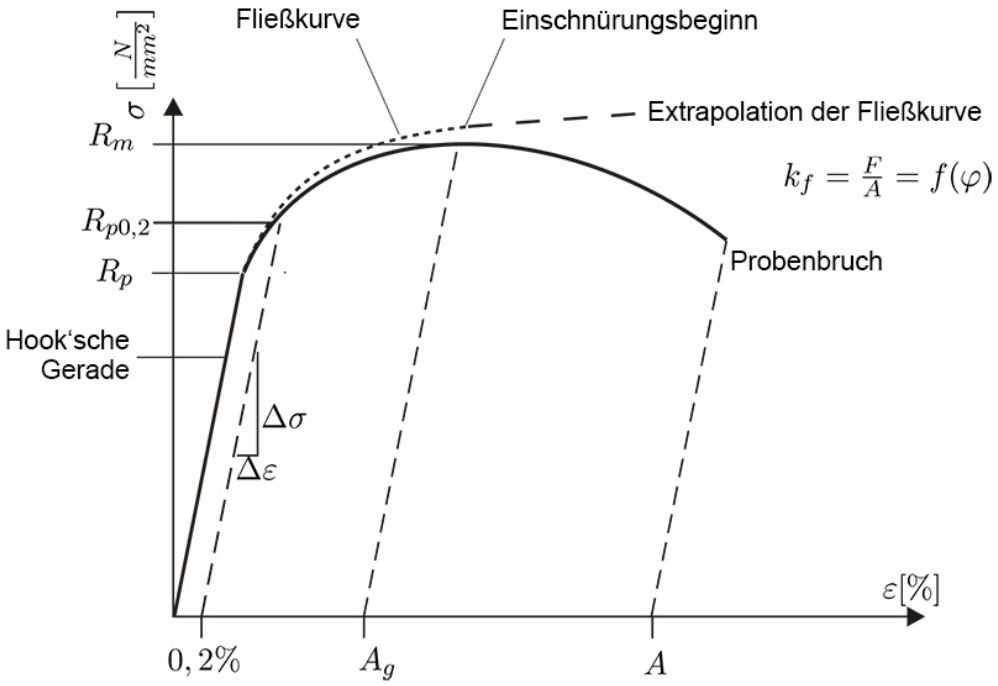


Abbildung 18: Spannungs-Dehnungs-Diagramm nach Doege

Wie in Abbildung 18 ersichtlich, wird eine Spannungs-Dehnungs-Kurve durch mehrere Bereiche und Punkte charakterisiert. So bezeichnet R_p die Streckgrenze, bis zu deren Belastung die Probe noch in ihren ursprünglichen Zustand zurückkehren kann. Der Anstieg bis dahin ist durch eine Gerade (nach Hook) gekennzeichnet. Da die Streckgrenze R_p im Zugversuch nicht unmittelbar bestimmt werden kann, liefern die meisten Testverfahren die $0,2\%$-Dehngrenze $R_{p0,2}$ zurück, bei der bereits eine bleibende plastische Verformung von $0,2\%$ stattfindet.
Bei der Zugfestigkeit R_m beginnt sich die Probe an einer Stelle einzuschnüren und die dortige Querschnittsfläche nimmt bei weiterer Belastung rapide ab. Die Querschnittsflächenänderung kann während des Versuchs nicht ermittelt werden. Deshalb auch das Abklingen der Kurve, obwohl die tatsächliche Spannung (auch als Fließkurve bezeichnet) noch weiter zunimmt. Die Gleichmaßdehnung A_g und die Bruchdehnung A sind die bleibenden Längenänderungen bei vollständiger Entlastung kurz vor Einschnürungsbeginn bzw. Probenbruch.
Im Gegensatz zu der im Bild dargestellten Spannungs-Dehnungs-Kurve aus einem Zugversuch, kann bei einem Zylinderstauchversuch die Fließkurve direkt bestimmt werden.

6.3 Versuchsdurchführung

Noch kurz zu den mit dem Abschreck- und Umformdilatometer gewonnen Daten. So wurden für die Warmumformversuche aus einer AA6082 Legierung Zylinder gefertigt:

Die Zylinder weisen einen Durchmesser von 5mm auf und haben eine Länge von 10mm. Die Probe wird in der Vorrichtung eingespannt und bei 15°C/s auf gewünschte Temperatur erwärmt. Anschließend wird der Zylinder mit konstanter Geschwindigkeit auf 30% seiner ursprünglichen Länge gestaucht. Dabei wird kontinuierlich die Stauchung Epsilon und die Festigkeit Sigma gemessen und aufgezeichnet. Die Festigkeit wird in Megapascal angegeben und liegt im Bereich von 0 – ca. 50 MPa. So ein Versuch wird 35-mal durchgeführt und zwar mit Temperaturen von 350 – 500°C in 25°-Schritten und Umformgeschwindigkeiten von $0,001$ bis 10 pro Sekunde.

Es ergibt sich schließlich eine große Datenmenge mit 28.559 Auswertungen:

Temperatur T	**Umformung** ϕ	**Geschwindigkeit** $\dot{\phi}$	**Festigkeit** k_f
350,28	2,6E-5	0,001	0,23
350,26	4,3E-5	0,001	0,20
350,23	5,8E-5	0,001	0,26
⋮	⋮	⋮	⋮
506,38	0,64	10	48,92
506,53	0,65	10	48,73

6.4 Feature Engineering

6.4.1 Datenbereinigung

In den Rohdaten befinden sich oft noch fehlerhafte Werte (zum Beispiel Ausreißer) und diese müssen vor der Verarbeitung gelöscht oder ergänzt werden. So kann dann auch im Weiteren eine brauchbare Analyse der Daten erfolgen. Bei Zug- oder Druckversuchen wird ein Probenstück in eine Messvorrichtung eingespannt. Dadurch entstehen leichte Abweichungen vom Nullreferenzpunkt, die zu Beginn des Versuches bereits aufgezeichnet werden. Um diese Verfälschung nicht in den Lernprozess einzubinden, löscht man alle Datenpunkte mit Spannung $\mathrm{k}_f < 0,5$ MPa.

6.4.2 Normalisierung

Ziel der Normalisierung ist es, mehrere Features (Merkmale) in einen vergleichbaren Wertebereich überzuführen/ zu skalieren. Ebenfalls sollen die Features in diesem auch ein ähnliches Verhalten aufweisen. Zwei der gängigsten Skalierungsmethoden sollen im Folgenden vorgestellt und anschließend auf den Datensatz angewendet werden:

Min-Max Skalierung

Hierbei wird der Wertebereich eines Features in ein Einheitsintervall $[0, 1]$ oder $[-1, 1]$ transformiert. Da meist Features als numerische Werte in Spalten einer Datenmatrix abgespeichert werden, spricht man im Folgenden auch schlicht von einer Datenspalte $\underline{x}$. Es bezeichnet wiederum $x_{\min}$ den kleinsten Wert in der Datenspalte und analog $x_{\max}$ den sich darin befindlichen größten Wert. Anschließend wird die Transformation

$$x' = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})} \qquad \text{bzw.} \qquad x' = 2 * \frac{(x - x_{\min})}{(x_{\max} - x_{\min})} - 1$$

auf alle Werte x aus der Datenspalte $\underline{x}$ angewendet und x durch x′ ersetzt. Welche dieser beiden Transformationen angewendet werden soll, hängt von der Natur der Daten, sowie dem oft vorgegebenen Definitionsintervall des zu verwendenden Algorithmus ab. Es ist darauf zu achten, dass Ausreißer keinen (oder kaum) Einfluss nehmen und die Daten somit im Intervall nahezu gleich verteilt sind. Vorteile bietet diese Skalierung nur, wenn mehr als ein Feature betrachtet wird und dadurch alle Features nahezu gleichgroßen Einfluss auf das Modell nehmen können. Ebenfalls ist dieses Vorgehen (somit) auch hilfreich, wenn die Algorithmen mit Gradientenabstiegsverfahren arbeiten.

Logarithmische Transformation

Die logarithmische Transformation soll dabei helfen die Dichte eines beobachteten Merkmals anzupassen. So hängt die Performance vieler Machine Learning Algorithmen von einer gleichmäßigen Verteilung der Datenpunkte ab. Bei hoher bzw. niedriger Dichte um den Ursprung stellt sich so

$$x' = \log(x)$$

oft als eine wirkungsvolle Transformation heraus. Als Beispiel denke man nur an eine exponentiell abklingende Beobachtung, wie die beim Entladen eines Kondensators über einen Widerstand. In konstanten Zeitabständen wird hierbei die Entladespannung des Kondensators über der Entladezeit gemessen. Zu Beginn ist hier der (euklidische) Abstand der Datenpunkte zueinander noch sehr hoch und nimmt dann rapide ab. Nach einer logarithmischen Transformation der Zeit hingegen liegen die Punkte nahezu auf einer Geraden mit konstantem Abstand. Zur Beschreibung des Sachverhaltes zeigen lineare Regressionsalgorithmen anschließend eine gute Performance. Das Feature Zeit hatte hier eine niedrige Dichte in 0 und das Feature Spannung hingegen eine hohe.

Anwendung

Der Datensatz besteht nun aus vier verschiedenen Features. Das Erste, welches transformiert wird, ist die Umformgeschwindigkeit $\dot{\phi}$. Denn da sich diese in Zehnerpotenzen (0,001, 0,01, 0,1, 1 und 10) steigert und in 0 sozusagen ein Häufungspunkt bildet, eignet sich hier eine logarithmische Transformation (zur Basis 10) besonders gut. Anschließend sind konstante Abstände (-3, -2, -1, -0 und 1) zu erkennen. Die nächsten beiden Features, die wir transformieren möchten, sind die Umformung ϕ und die Spannung k_f. Betrachtet man die Abbildung 19, so sind große Abstände der Datenpunkte im Anfangsbereich der Umformung erkennbar, die im weiteren Verlauf schließlich abnehmen. Wie bereits oben erwähnt wurde, eignet sich hier eine logarithmische Transformation (Abbildung 20):

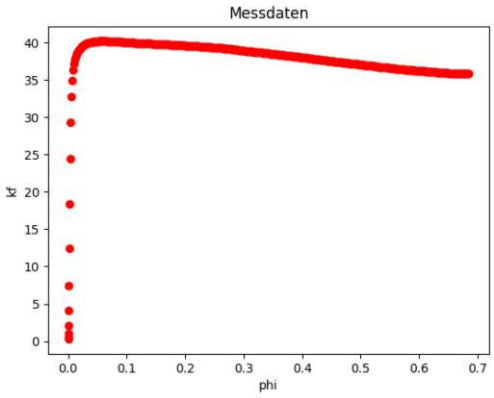


Abbildung 19: T=400 und $\dot{\phi}$=0,01

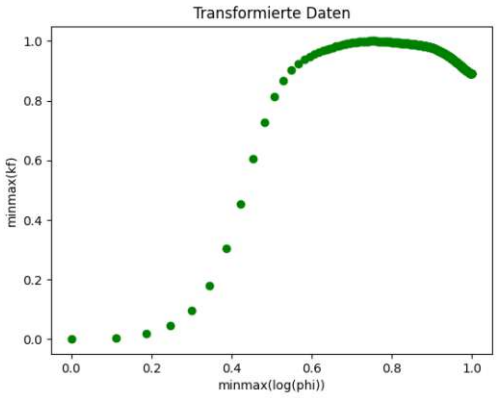


Abbildung 20: T′=0,33 und $\dot{\phi}'$=0,25

Da nun erreicht wurde, dass sämtliche Features gleichmäßiger verteilt sind, führt man auf alle vier noch eine Min-Max Skalierung in das Einheitsintervall [0,1] aus, um die Performance des CFR Algorithmus zu verbessern. Für ϕ und k_f kann dies in Abbildung 20 betrachtet werden.

6.4.3 Datenauswahl (Data Sampling)

Aus wie vielen Datenpunkten ein Heißkompressionstest besteht, ist von der Umformgeschwindigkeit $\dot{\phi}$ abhängig. Dies liegt vor allem daran, dass die Abtastrate bei allen Tests nahezu gleich ist und alle Proben auf dieselbe Länge gestaucht werden.

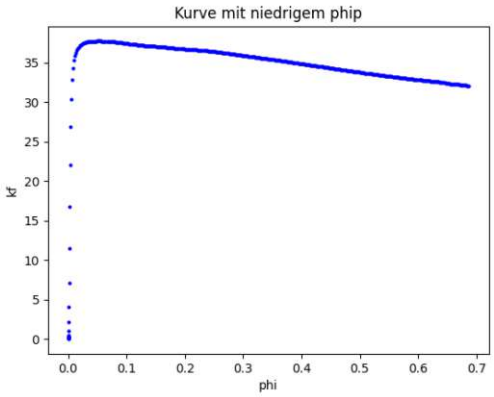


Abbildung 21: T=375 und $\dot{\phi}$=0,001

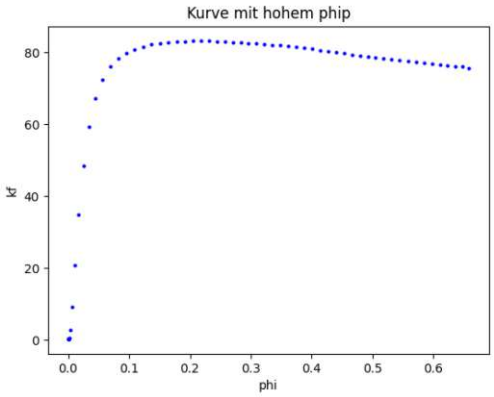


Abbildung 22: T=375 und $\dot{\phi}$=10

Gut beobachtbar ist, dass in Abbildung 22 mit einer Geschwindigkeit von 10/s weniger Punkte generiert wurden als in Abbildung 21 mit einer Umformgeschwindigkeit von 0,001/s. Würde man nun beide Datenmengen dem Lernalgorithmus übergeben, so hätte die Kurve mit niedrigem $\dot{\phi}$ einen größeren Einfluss auf das Modell als die mit hohem $\dot{\phi}$. Dies liegt daran, dass man beim CFR Algorithmus ja über den Mean-Squared-Error minimiert. Um dem entgegenzuwirken, sollen alle Kurven auf dieselbe Anzahl an Datenpunkte von 300 gebracht werden. Dies erfolgt durch Löschen oder Duplizieren von einzelnen und zufällig gewählten Datenpunkten. Für das dazugehörende Python-Skript siehe [Schützeneder, 2020].

6.4.4 Datenaufteilung

Wie bereits in Kapitel 2.2.1 und Abbildung 5 angemerkt wurde, erfolgt nun eine Aufteilung in eine Trainings- und Testmenge der Daten. Dies soll uns ein mögliches Overfitting eines Modells erkennen lassen. Das aus 35 Kurven bestehende Datenset soll wie in folgender Tabelle 6 aufgeteilt werden:

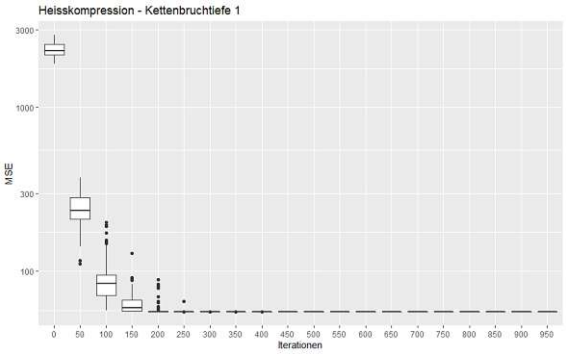
	phip				
T	**0,001/s**	**0,01/s**	**0,1/s**	**1/s**	**10/s**
350°C	E	T	T	E	T
375°C	V	I	T	T	E
400°C	T	V	I	T	T
425°C	E	T	V	I	T
450°C	T	I	T	V	E
475°C	T	T	I	T	V
500°C	T	T	T	E	T

Tabelle 6: Aufteilung der Daten

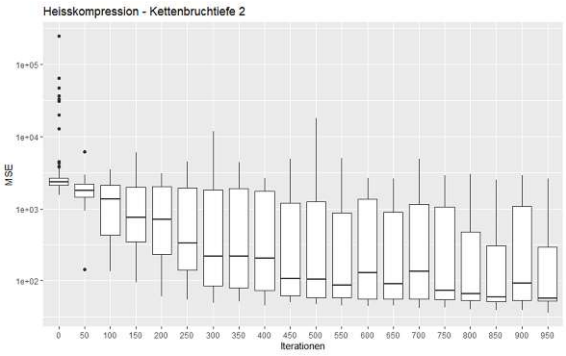
Für das Training T sollen also 19 Kurven (5700 Datenpunkte) und für die anschließende Validierung V fünf Kurven (1500 Datenpunkte) verwendet werden. Die so gewählte Aufteilung der Kurven und speziell auch das Zusammenhalten der Datenpunkte in ganzen Kurven, soll die Fähigkeit des CFR-Algorithmus zu verallgemeinern (Interpolation I und Extrapolation E) ersichtlich machen.

6.4.5 Nelder-Mead-Verfahren

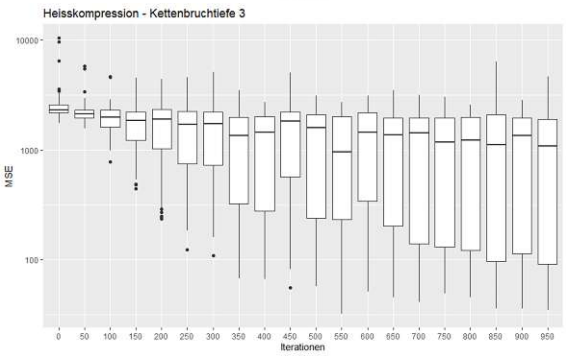
Wie in den Kapiteln 5.1 und 5.2 wird auch für die Heißkompressionsdaten die Auswirkung der Optimierung nach Nelder-Mead untersucht. Die Resultate sollen einen einfachen Vergleichswert für die Modelle anderer Algorithmen liefern. Da hier eine Datenmenge in drei Eingangsvariablen vorliegt und erste Versuche gezeigt haben, dass ein Abklingverhalten über die Iterationen kaum feststellbar ist, erhöht man deren Anzahl auf bis zu 950 und betrachtet nur Kettenbruchtiefen von 1 bis 4.



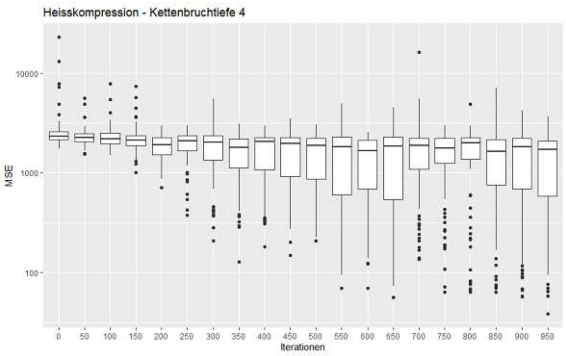
Auf Grund der simplen Struktur eines Kettenbruches der Tiefe 1, konvergieren nahezu alle Kandidaten gegen denselben Repräsentanten. Dieser kann bereits nach 300 Iterationen mit einem MSE-Wert von $56,5136$ ausgemacht werden.



Bei Kettenbrüchen der Tiefe 2 ist ein rasches Abklingen des unteren Quartils und des Medians mit Zunahme der Iterationszahl zu beobachten. Mehr als 450 Wiederholungen scheinen hier nicht nötig zu sein. Der geringste MSE liegt hier bei $36,1650$ nach einer Iterationszahl von 950.



In einer Kettenbruchtiefe von 3 ist eine kontinuierliche Abnahme des unteren Quartils der Testversuche erkennbar. Das obere- und mittlere Quartil hingegen scheinen bei 250 Iterationen, im Vergleich zu einer höheren Anzahl, kaum merkbar zu sinken. Der beste Lösungskandidat wurde nach 550 Iterationen mit einem MSE-Wert von $32,2506$ gefunden.



Bei den drei vorhergehenden Experimenten ist man zumindest mit dem unteren Quartil nahe einem MSE-Wert von 100 gekommen. Es ist erkennbar, dass durch alleiniges Optimieren nach Nelder-Mead oft ein schlechtes lokales Optimum gefunden wird. Hierbei hatte die beste Lösung einen MSE-Wert von $38,4711$ nach 950 Iterationen.

Das vermutlich wichtigste Resultat aus diesen Experimenten ist, dass man durch reine Nelder-Mead-Optimierung in viele lokale Minima gerät. Es wird im Folgenden ersichtlich, dass durch eine gute Wahl der Parameter bei Kettenbruch-Regression nahezu immer ein Lösungskandidat mit einem MSE-Wert unter 10 zu finden ist. Die Iterationszahl kann auch hier bei 250 belassen werden.

6.5 Originaler CFR-Algorithmus

In diesem Unterkapitel soll nun der originale CFR-Algorithmus getestet werden. Die Funktionsweise wurde bereits in Kapitel 3 ausführlich beschrieben und die von den Autoren verwendeten Standardparameter sind in Tabelle 1 angegeben.
Diese Parameter werden nun in der Implementierung von HeuristicLab [Kronberger, 2021] festgelegt und ein Experiment allein über die Kettenbruchtiefe durchgeführt. Verglichen werden die Tiefen 2, 3, 4 und 5 miteinander und für jede Tiefe jeweils 10 Versuche durchgeführt (insgesamt also 40 Versuche). Obwohl die Auswahl der besten Lösungskandidaten über eine variablenregulierte MSE Variante erfolgt, wird für eine objektive Beurteilung der MSE Wert herangezogen. Die 4 Boxplots mit den jeweils 10 Validierungs-MSE Werten (logarithmisch skaliert) sind in Abbildung 23 dargestellt:

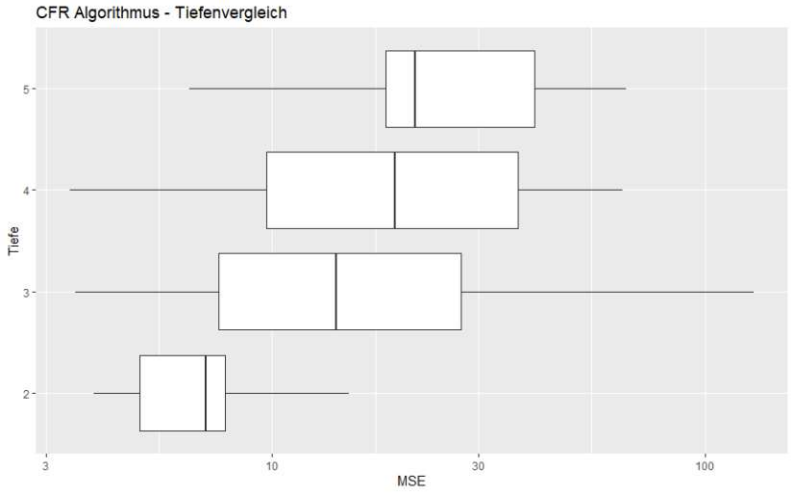


Abbildung 23: Experiment über die Tiefe der Kettenbrüche

Man stellt leider fest, dass keiner der Kettenbrüche einen MSE Wert unter 3 aufweist. Mit steigender Tiefe nimmt auch der Wert sämtlicher Quartile zu. Eine Kettenbruchtiefe von 2 scheint für die Daten (als Modellparameter) die logisch beste Wahl zu sein. In den folgenden zwei Abbildungen 24 und 25 ist das Konvergenzverhalten zweier Versuche dargestellt:

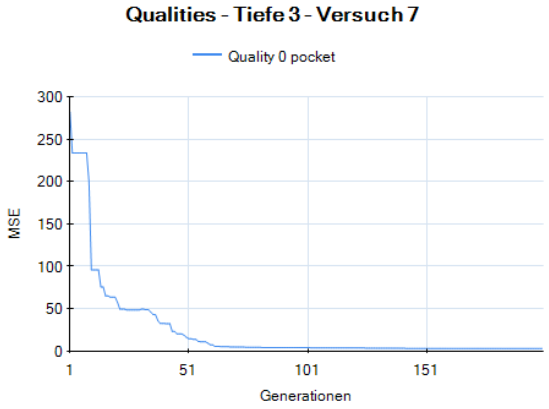


Abbildung 24: Konvergenzverhalten bei Tiefe 3

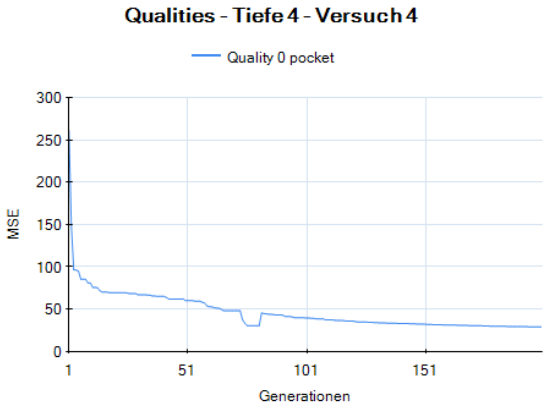


Abbildung 25: Konvergenzverhalten bei Tiefe 4

Wie bereits schon erwähnt wurde, befindet sich der beste Lösungskandidat in der Pocket Solution von Agent 0. Deshalb wurde auch hier nur dessen Entwicklung betrachtet. In den Versuchen mit Tiefe 3 ist allgemein eine rasche Konvergenz beobachtbar. Allerdings scheint für die Versuche mit Tiefe 4 (und auch höher) der Standardwert mit 200 Generationen oftmals zu wenig zu sein und ein Underfitting tritt ein. Was in Abbildung 25 ebenfalls erkennbar ist, ist eine Neuwahl der Pocket Solution, die nach 81 Generationen ohne das Erzielen einer Verbesserung stattgefunden hat.

6.6 Eigene Implementierung in HeuristicLab

HeuristicLab ist eine frei erhältliche Softwareumgebung für heuristische und evolutionäre Algorithmen, die an der FH Oberösterreich entwickelt wurde. Sie zeichnet sich durch eine benutzerfreundliche Oberfläche aus und erlaubt es dennoch die Funktionalität auf Code-Level zu modifizieren. Es soll nun eine eigene Implementierung des CFR Algorithmus in HeuristicLab vorgestellt werden. Hierfür wurde ein sogenanntes ProgrammableProblem (Skript in C#) geschrieben, welches ein Multi-Encoding des Koeffizienten- und Binärvektors ermöglicht. Darauf wird anschließend ein klassischer genetischer Algorithmus angewendet.
Zur Auswahl stehen eine Reihe an Operatoren. Für Crossover wird auf beiden Vektoren das Single-point Crossover (Beschreibung in Kap 2.3.3) verwendet. Die Auswahl erfolgt über einen Tournament Selektor (Beschreibung in Kap 2.3.3), der für eine Populationsgröße von 30, 50 oder 100 Kettenbrüche auf zwei, zwei oder drei Gruppen festgelegt wird. Der Mutationsoperator wird entweder völlig ausgeblendet oder der zur Verfügung stehende Some-Positions-Bitflip-Manipulator und der Self-Adaptive-Normal-All-Positions-Manipulator auf Binär- bzw. Koeffizientenvektor angewendet. Die Anzahl der Generationen wird wie im Standardalgorithmus bei 200 belassen.
Zur Beurteilung der Individuen kommt einmal der klassische Mean-Squared-Error und des Weiteren auch der regulierte Mean-Squared-Error zum Einsatz (beide wie in Kap 3.4).

6.6.1 Angepasster Nelder Mead Algorithmus

Wie im originalen CFR-Algorithmus beschrieben, kann auch hier wieder das Downhill-Simplex-Verfahren angewendet werden. Allerdings wird dieses etwas angepasst um die, wie in Abbildung 25 und Abbildung 23, ersichtliche langsame Konvergenz bei erhöhter Kettenbruchtiefe zu beheben. Die Idee liegt darin die Individuen nach Anwendung der Operatoren zu kürzen und über diese Kettenbrüche geringerer Tiefe zu optimieren. Bringt dies eine Verbesserung des MSE so soll das Individuum in die neue Population aufgenommen werden. Hierbei werden nur die letzten Stellen des Binärvektors auf False gesetzt und der Koeffizientenvektor wird vollständig beibehalten.

Als Beispiel sei nun die maximale Kettenbruchtiefe mit 5 definiert und es soll ein Individuum mit dem Nelder-Mead-Algorithmus optimiert werden. Das heißt man wendet (5+1)-mal (selbe Anzahl wie maximale Tiefe + 1) dieses Verfahren an. Dieses wird im ersten Lauf noch auf den originalen Kettenbruch mit 250 Iterationen und 20% der Trainingsdaten (Parameter wie im originalen Algorithmus) durchgeführt. Beim zweiten Mal entfernt man das letzte Glied (keine Reduktion um eine vollständige Tiefe) und optimiert darüber wieder. Danach über zwei Glieder und so weiter und so fort, bis schließlich nach (5+1) Vorgängen der Kettenbruch halbiert vorgefunden wird. Den besten dieser optimierten Kandidaten nimmt man in der neuen Population auf. (Es soll vermerkt werden, dass hier auch Schritte, die eine ganze Tiefe umfassen oder bis sich nur noch über ein Glied optimieren lässt, interessante Ergebnisse liefern)

6.6.2 Wahl der Model-Parameter

Es sollen nun möglichst gute Parameter für das Modell bestimmt werden. Hierfür variiert man die Populationsgröße (30, 50 und 100), die Wahrscheinlichkeit der Mutation (0% und 30%), die Fehlerbestimmung (MSE und regulierter MSE) und die Kettenbruchtiefe (4, 5 und 6) durch. Dies ergibt also ein Experiment mit 36 Variationen. Für jedes werden 20 Einzeltests durchgeführt, um ein stochastisches Mittel zu erhalten.
Für die Boxplots wurde nun aus jedem Test der Kettenbruch mit dem geringsten MSE-Wert über die Trainingsdaten verwendet. Für diesen wurde anschließend der MSE-Wert über die Validierungsdaten genommen und aufgetragen.
Zu Beginn soll gleich einmal der Einfluss des Mutationsoperators festgestellt werden. Hierfür wurden die 18 Experimente mit MSE (weiters unterteilt in mit und ohne Mutationsoperator) und auch die 18 Experimente mit reguliertem MSE (weiters unterteilt in mit und ohne Mutationsoperator) in jeweils einen Boxplot aufgetragen:

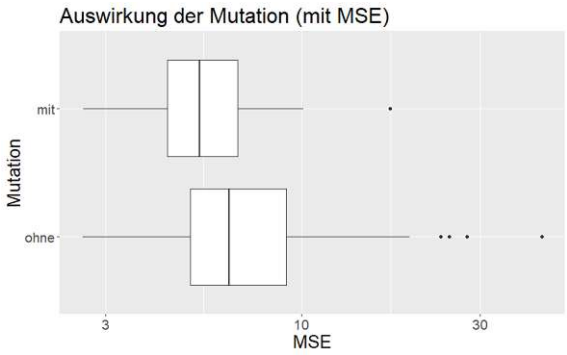


Abbildung 26: Boxplot 1 für Mutation

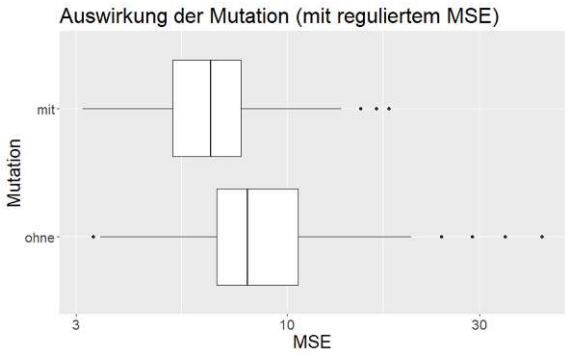


Abbildung 27: Boxplot 2 für Mutation

Es ist aus den Abbildungen 26 und 27 sofort ersichtlich, dass sich der Mutationsoperator positiv auf den Lernprozess auswirkt. So liegen Median sowie auch unteres und oberes Quartil der Versuche mit Anwendung von Mutation immer unterhalb derer ohne.
Im Weiteren verwendet man die 18 übrigbleibenden Experimente, um den Einfluss der Populationsgröße auf das Training zu ermitteln. Wie vorher werden diese in zwei Boxplots mit und ohne reguliertem MSE aufgetragen:

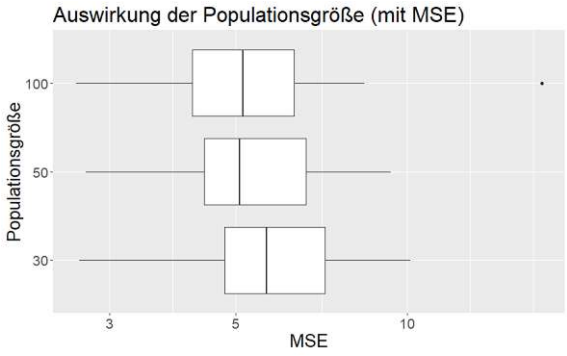


Abbildung 28: Boxplot 1 für Populationsgröße

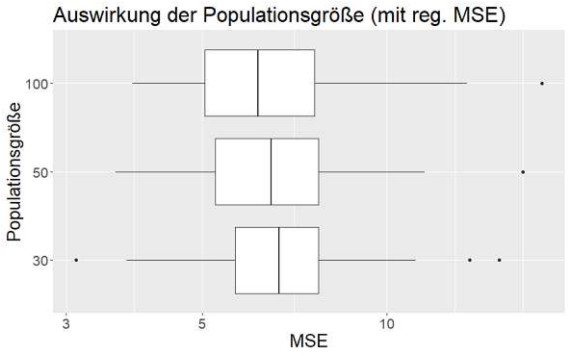


Abbildung 29: Boxplot 2 für Populationsgröße

In Abbildung 29 ist erkennbar, dass mit zunehmender Populationsgröße der Median, unteres und oberes Quartil über die dazugehörenden Versuche abnimmt. Analoges ist auch in Abbildung 28 ersichtlich, abgesehen vom Median, bei den zwei Experimenten mit höherer Populationsgröße. Da diese Abweichung aber nur gering ist, fällt die Entscheidung auf eine Populationsgröße von 100 Individuen und einem Tournament Selektor mit drei Gruppen.
Im Weiteren, ob in einem Modell mit MSE oder reguliertem MSE gearbeitet werden soll, fällt die Entscheidung auf ersteres. Denn im Vergleich der beiden Balken mit einer Populationsgröße von 100 ist sofort erkennbar, dass die entsprechenden Quartile alle darunter liegen.

Zu guter Letzt bleibt noch die Wahl der Kettenbruchtiefe über. Hierfür trägt man wieder die drei in Frage kommenden Experimente zu je 20 Versuchen auf:

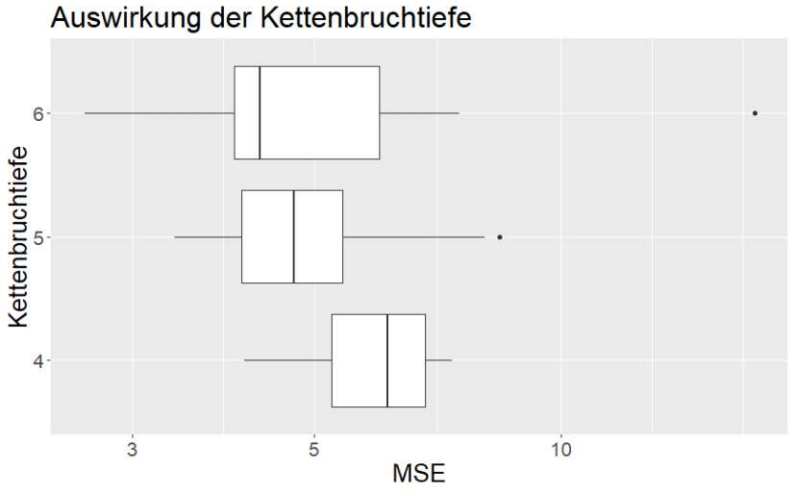


Abbildung 30: Boxplot für Kettenbruchtiefe

Was in Abbildung 30 deutlich auffällt ist, dass eine Kettenbruchtiefe von 4, weder im Bezug auf den besten Lösungskandidaten noch auf die der Quartile mit den Tiefen 5 und 6 mithalten kann. Bis auf das obere Quartil schneidet hier eine Kettenbruchtiefe von 6 am Besten ab, für welche dann auch entschieden wird. Anzumerken ist noch, der beste Lösungskandidat über alle Experimente stammt aus einem Versuch ohne Mutationsoperator, einer Populationszahl von 50, mit MSE und einer Kettenbruchtiefe von 6 und weist einen MSE von 2,61 auf.
Verglichen mit dem originalen Algorithmus (Abbildung 23) ist hier mit Zunahme der Kettenbruchtiefe eine deutliche Abnahme des Median gegeben, was auch beabsichtigt war.

6.6.3 Anwendung auf die Testdaten

Es wurden nun die am Besten geeignetsten Parameter bestimmt und damit wird nun eine finale Auswertung durchgeführt. Für das Antrainieren eines Modells dient die gesamte Trainings- und Validierungsdatenmenge. Die Abbildung 31 zeigt die Kettenbruchform der besten Lösungskandidaten in jeder Generation an:

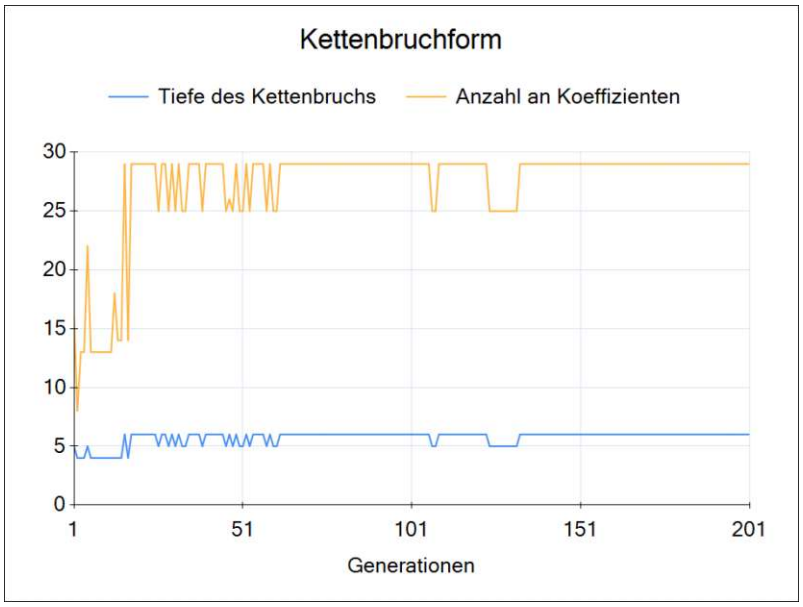


Abbildung 31: Komplexität der besten Lösungskandidaten

In den ersten 15 Generationen ist ersichtlich, dass sich Kettenbrüche mit geringerer Tiefe besser eignen. So weisen diese meist eine Tiefe von 4 auf und bestehen meist aus weniger als 15 Koeffizi-

enten. Erst bei zunehmender Anzahl an Generationen werden wieder Individuen erhöhter bzw. von maximaler Tiefe von 6 und 29 Koeffizienten erzeugt. Ausgewertet wird nun der beste gefundene Lösungskandidat an den Interpolationsdaten, an den Extrapolationsdaten, sowie der Kombination aus beidem (Test). Dies soll eine Beurteilung ermöglichen, ob der CFR Algorithmus in der Lage ist zu verallgemeinern:

Model	**Training**	**Test**	**Interpolation**	**Extrapolation**
Kettenbruch-Regression Eigene Implementierung	2,0320	29,1572	28,1092	30,3744
Grammar Enumeration Symbolic Regression	0,7387	1,3532	1,0097	1,7521
Genetische Programmierung mit Nachkommensselektion	0,1643	0,4204	0,2626	0,6037
Support Vector Regression	0,3362	1,3676	0,5692	2,2950
Mehrschichtige Perzeptren	0,0188	0,3654	0,1669	0,5959

Tabelle 7: Vergleich der MSE-Werte mit anderen Modellen

In Tabelle 7 sieht man den Vergleich der MSE-Werte der in [Schützeneder, 2020] trainierten Modelle. Bereits im Training konnten die Daten durch die Kettenbruch-Regression nicht, wie durch die anderen vier Modelle, ausreichend genau approximiert werden. Ebenfalls ist auch keine zufriedenstellende Annäherung an die Interpolations- sowie auch Extrapolationsdaten erfolgt. Hier sind noch weitere Modifizierungen des CFR-Algorithmus nötig, um deren Genauigkeit zu erreichen.
Dennoch muss auch erwähnt werden, dass es sich bei mehrschichtige Perzeptren um ein Informationsverarbeitungssystem handelt. Es ist eine spezielle Form eines künstlichen neuronalen Netzes und somit ein Blackboxmodell. Diese Uneinsichtbarkeit gilt leider auch für das Support Vector Regression Modell. Bei der Grammar Enumeration Symbolic Regression hingegen erhält man eine explizite Funktion, die aus einer Komposition von $+, -, *, \log(), \exp(), \sqrt{.}$ und $\sqrt[3]{.}$ besteht. Allerdings wird hier ein Funktionsbaum gebildet, der ein hohes Variablenvorkommen von nahezu 1000 aufweist. Ähnlich komplex ist auch das Modell aus der genetischen Programmierung mit Nachkommensselektion. Es verwendet die Funktionssymbole $+, -, *, /, \log()$ und $\exp()$. Die Ausdruckslänge beträgt hier zwar nur etwa 150, ist jedoch nicht annähernd so kompakt und interpretationsfähig wie ein verallgemeinerter Kettenbruch mit 29 Koeffizienten.

7 Fazit und Diskussion

Der (originale) CFR-Algorithmus hat mit zunehmender Komplexität der Modelle (Erhöhung der Kettenbruchtiefe) ein langsameres Abklingen des Mean-Squared-Errors zur Folge. Da die maximale Anzahl an Generationen vorab festgelegt wurde, ist eine bessere Approximation der Trainingsdaten (und somit auch der Testdaten) über die Zunahme der Komplexität nicht sichergestellt.
Eine Abhilfe konnte über die Bildung und einer anschließenden Optimierung von Kettenbrüchen mit geringerer Tiefe in den ersten Generationen geschaffen werden. Hier hatte auch der Einsatz des Nelder-Mead-Verfahrens eine große Auswirkung gezeigt. Erst im weiteren Verlauf wurden daraus durch Mutation und Rekombination Modelle mit erhöhter und schließlich maximaler Komplexität erzeugt. Der springende Punkt hierbei ist, dass sich im Allgemeinen der Einfluss, der in Kettenbrüchen neu hinzukommenden Glieder verringert. Dies wurde auch mit einer stochastischen Analyse an den Spannungs-Dehnungs-Daten gezeigt. Die exakte Vorgehensweise wurde in einer eigenen Variante des CFR Algorithmus beschrieben.
Die Idee Rekombination von Kettenbrüchen mit geringer Tiefe, zur Bildung welcher mit hoher zu verwenden, wurde in Korollar 4.34 hergeleitet. Den abnehmenden Einfluss der unteren Glieder eines Kettenbruchs bezüglich MSE, ist eine Schlussfolgerung aus Satz 4.35 und der Tatsache, dass sich eine konvergente Stufenfolge von Padé-Approximanten als Konvergentenfolge eines Kettenbruchs darstellen lässt. Eine Zusammenfassung dieser Resultate und weiters auch, weshalb Padé-Approximanten bezüglich Konvergenz (Inter- und Extrapolation) besser geeignet sind als eine klassische Reihendarstellung der Zielfunktion, findet sich in Kapitel 4.6.
Angemerkt sei auch, ein Vorwissen über die zu minimierende Zielfunktion und deren Reihenentwicklung ermöglicht eine effizientere Wahl der Modelle. So müssen nicht alle Glieder eines Kettenbruchs linear in den Eingangsvariablen gewählt werden. Da andere Modelle eine bessere Anpassung an die Stauchversuchsdaten gezeigt haben, wäre eine solche Umsetzung oder auch eine weitere Recherche zur multivariaten Padé-Approximation überlegenswert. Ebenfalls könnte ein Rekombinationsoperator, der speziell die Kontinuanten verwendet, sinnvoll sein.

Literatur

[Aunkofer, 2017] Aunkofer, Benjamin. *Überwachtes vs unüberwachtes maschinelles Lernen.* 2. Juli 2017 data-science-blog. https://data-science-blog.com/blog/2017/07/02/uberwachtes-vs-unuberwachtes-maschinelles-lernen/

[Backeljauw und Cuyt, 2009] Backeljauw, Franky, and Annie Cuyt. *Algorithm 895: A continued fractions package for special functions.* ACM Transactions on Mathematical Software (TOMS) 36.3 (2009): 1-20.

[Baker et al., 1961] Baker Jr, George A., J. L. Gammel, and John G. Wills. *An investigation of the applicability of the Padé approximant method.* Journal of Mathematical Analysis and Applications 2.3 (1961): 405-418.

[Baker, 1964] Baker Jr, George A. *The theory and application of the Padé approximant method.* No. LA-DC-6526. Los Alamos Scientific Lab., Univ. of California, N. Mex., 1964.

[Baker, 1972] Baker Jr, George A. *Recursive calculation of Padé approximants.* No. BNL-17111; CONF-720612-6. Brookhaven National Lab., Upton, NY, 1972.

[Baker et al., 1996] Baker, George A., et al. *Pade Approximants: Encyclopedia of Mathematics and It's Applications.* Vol. 59. Cambridge University Press, 1996.

[Bernecker, 2019] Bernecker, Michael. *Künstliche Intelligenz – Was ist das eigentlich?.* 4.Juli 2019 Deutsches Institut für Marketing. https://www.marketinginstitut.biz/blog/kuenstliche-intelligenz/

[Bombelli, 1579] Bombelli, Raffaele. *L' Algebra: Opera di Rafael Bombelli da Bologna.* ETH-Bibliothek Zürich. https://doi.org/10.3931/e-rara-3918

[Bultheel, 1980] Bultheel, Adhemar. *Division algorithms for continued fractions and the Padé table.* Journal of Computational and Applied Mathematics 6.4 (1980): 259-266.

[Claessens, 1975] Claessens, G. *A new look at the Padé table and the different methods for computing its elements.* Journal of Computational and Applied Mathematics 1.3 (1975): 141-152.

[Cuyt, 1990] Cuyt, A. *A multivariate convergence theorem of the "de Montessus de Ballore" type.* Journal of Computational and Applied Mathematics 32.1-2 (1990): 47-57.

[Cuyt, 1999] Cuyt, Annie. *How well can the concept of Padé approximant be generalized to the multivariate case?.* Journal of Computational and Applied Mathematics 105.1-2 (1999): 25-50.

[Dantzig, 1990] Dantzig, George B. *Origins of the simplex method.* June 1990 Association for Computing Machinery New York, NY, United States. https://doi.org/10.1145/87252.88081

[De Jong, 2016] De Jong, Kenneth. *Evolutionary computation: a unified approach.* The MIT Press Cambridge, Massachusetts London, England, 2006.

[Euler, 1744] Euler, Leonhard. *De fractionibus continuis dissertatio.* Commentarii academiae scientiarum Petropolitanae (1744): 98-137.

[Fajfar et al., 2017] Fajfar, Iztok, Janez Puhan, and Árpád Bűrmen. *Evolving a Nelder–Mead algorithm for optimization with genetic programming.* Evolutionary computation 25.3 (2017): 351-373.

[Ferus, 2010] Ferus, Dirk. *Komplexe Analysis.* Vorlesungsskript, TU Berlin (2010). https://page.math.tu-berlin.de/~ferus/KA/komplexeAnalysis.pdf

[Fogel et al., 1966] Fogel, Lawrence J., Alvin J. Owens, and Michael J. Walsh. *Artificial intelligence through simulated evolution.* Wiley, 1966.

[Gashkov und Gashkov, 2006] Gashkov, Sergey Borisovich, and Igor Borisovich Gashkov. *Berlekamp—Massey Algorithm, Continued Fractions, Pade Approximations, and Orthogonal Polynomials.* Mathematical Notes 79.1-2 (2006): 41-54.

[Goldstern und Winkler, 2018] Goldstern, Martin., Winkler, Reinhard. *ALGEBRA I und II.* Vorlesungsskript, 2018 TU Wien. http://dmg.tuwien.ac.at/winkler/skripten/algebra.pdf

[Gragg, 1972] Gragg, William B. *The Padé table and its relation to certain algorithms of numerical analysis.* SIAM review 14.1 (1972): 1-62.

[Graves-Morris et al., 2000] Graves-Morris, P. R., D. E. Roberts, and A. Salam. *The epsilon algorithm and related topics.* Journal of Computational and Applied Mathematics 122.1-2 (2000): 51-80.

[Guillaume und Huard, 2000] Guillaume, Philippe, and Alain Huard. *Multivariate padé approximation.* Journal of computational and applied mathematics 121.1-2 (2000): 197-219.

[Hart et al., 2004] Hart, William E., Natalio Krasnogor, and James E. Smith. *Recent advances in memetic algorithms.* Springer Berlin Heidelberg, 2004.

[Borchers, 2011] Borchers, Detlef. *((Requiescat) in (pace)): Zum Tod von John McCarthy.* heise.de, 25. Oktober 2011. https://www.heise.de/developer/meldung/Requiescat-in-pace-Zum-Tod-von-John-McCarthy-1366069.html

[Höfling, 1994] Höfling, Oskar. *Physik. Band II Teil 1, Mechanik, Wärme.* 15. Auflage. Ferd. Dümmlers Verlag, Bonn 1994.

[Holland, 1992] Holland, John Henry. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, 1992.

[Jacobi, 1846] Jacobi, Carl Gustav Jacob. *Über die Darstellung einer Reihe gegebner Werthe durch eine gebrochne rationale Function.* Journal für die reine und angewandte Mathematik 1846.30 (1846): 127-156.

[Jänich, 2004] Jänich, Klaus. *Funktionentheorie.* Springer Berlin Heidelberg, 2004.

[Kabliman et al., 2019] Kabliman, Evgeniya, et al. *Prediction of stress-strain curves for aluminium alloys using symbolic regression.* AIP Conference Proceedings. Vol. 2113. No. 1. AIP Publishing LLC, 2019.

[Koza, 1992] Koza, John R., and John R. Koza. *Genetic programming: on the programming of computers by means of natural selection.* Vol. 1. MIT press, 1992.

[Kronberger, 2021] Kronberger, Gabriel. *CFR - Continued Fraction Regression, 04.07.2022.* https://github.com/heal-research/HEAL.CFR/

[Kronecker, 1881] Kronecker, Leopold. *Zur Theorie der Elimination einer Variabeln aus zwei algebraischen Gleichungen.* Buchdruckerei der Königl. Akademie der Wissenschaften (G. Vogt), 1881.

[Lorentzen, 2010] Lorentzen, Lisa. *Padé approximation and continued fractions.* Applied numerical mathematics 60.12 (2010): 1364-1370.

[Lubinsky, 1995] Lubinsky, D. S. *Convergence of diagonal Padé approximants for functions analytic near 0.* Transactions of the American Mathematical Society 347.8 (1995): 3149-3157.

[Lubinsky, 2003] Lubinsky, Doron S. *Rogers-ramanujan and the baker-gammel-wills (padé) conjecture.* Annals of mathematics (2003): 847-889.

[Lubinsky, 2014] Lubinsky, Doron S. *Analytic number theory, approximation theory, and special functions.* Springer, New York, NY, 2014. 561-571.

[Lubinsky, 2021] Lubinsky, Doron S. *On Baker's Patchwork Conjecture for Diagonal Padé Approximants.* Constructive Approximation 53.3 (2021): 545-567.

[Magnus, 1962] Magnus, Arne. *Expansion of power series into P-fractions.* Math Z 80, 209–216 (1962).

[McCarthy et al., 1955] McCarthy, John, et al. *A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955.* AI magazine 27.4 (2006): 12-12.

[Meyer, 1976] Meyer, Burnett. *On convergence in capacity.* Bulletin of the Australian Mathematical Society 14.1 (1976): 1-5.

[Miller und Goldberg, 1995] Miller, Brad L., and David E. Goldberg. *Genetic algorithms, tournament selection, and the effects of noise.* Complex systems 9.3 (1995): 193-212.

[Moscato, 1989] Moscato, Pablo. *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms.* Caltech concurrent computation program, C3P Report 826, 1989.

[Moscato et al., 2020] Moscato, Pablo, Haoyuan Sun, and Mohammad Nazmul Haque. *Analytic Continued Fractions for Regression: Results on 352 datasets from the physical sciences.* 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2020.

[Moscato et al., 2021] Moscato, Pablo, Haoyuan Sun, and Mohammad Nazmul Haque. *Analytic Continued Fractions for Regression: A Memetic Algorithm Approach.* Expert Systems with Applications 179 (2021): 115018.

[Musch, 2004] Musch, Marc. *Genetisches Programmieren.* Proseminar - Soft Computing, 2004 Universität Ulm. http://www.informatik.uni-ulm.de/ni/Lehre/SS04/ProsemSC/ausarbeitungen/Musch.pdf

[Nilsson, 2009] Nilsson, Nils J. *The quest for artificial intelligence.* Cambridge University Press, 2009.

[Nissen, 1998] Nissen, Volker. *Einige Grundlagen Evolutionärer Algorithmen.* In: Biethahn, J., Hönerloh, A., Kuhl, J., Leisewitz, MC., Nissen, V., Tietze, M. (eds) Betriebswirtschaftliche Anwendungen des Soft Computing. Computational Intelligence. Vieweg+Teubner Verlag. https://doi.org/10.1007/978-3-322-86843-5_3

[Nuttall, 1970] Nuttall, J. *The convergence of Padé approximants of meromorphic functions.* Journal of Mathematical Analysis and Applications 31.1 (1970): 147-153.

[Padé, 1892] Padé, Henri. *Sur la représentation approchée d'une fonction par des fractions rationnelles.* Annales scientifiques de l'école Normale Supérieure. Vol. 9. 1892.

[Pagie und Hogeweg, 1997] Pagie, Ludo, and Paulien Hogeweg. *Evolutionary consequences of coevolving targets.* Evolutionary computation 5.4 (1997): 401-418.

[Poli et al., 2008] Poli, Riccardo, et al. *A field guide to genetic programming.* Lulu Enterprises, UK Ltd., 2008.

[Pommerenke, 1973] Pommerenke, Ch. *Padé approximants and convergence in capacity.* Journal of Mathematical Analysis and Applications 41.3 (1973): 775-780.

[Rechenberg, 1973] Rechenberg, Ingo. *Evolutionsstrategie — Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog-Verlag, Stuttgart 1973.

[Reitmaier, 2015] Reitmaier, Tobias. *Aktives Lernen für Klassifikationsprobleme unter der Nutzung von Strukturinformationen.* Vol. 5. kassel university press GmbH, 2015.

[Rudolph, 2015] Rudolph, Günter. *Evolutionäre Algorithmen.* 11. September 2015 TU Dortmund. https://ls11-www.cs.tu-dortmund.de/de/rudolph/ea

[Rutishauser, 1954] Rutishauser, Heinz. *Der quotienten-differenzen-algorithmus.* Zeitschrift für angewandte Mathematik und Physik ZAMP 5.3 (1954): 233-251.

[Salzer, 1962] Salzer, Herbert E. *Note on osculatory rational interpolation.* Mathematics of Computation 16.80 (1962): 486-491.

[Sauer, 2005] Sauer, Tomas. *Kettenbrüche.* Vorlesungsskript, Justus–Liebig–Universität Gießen (2005).

[Scherk et al., 2017] Scherk, Johannes, Gerlinde Pöchhacker-Tröscher, and Karina Wagner. *Künstliche Intelligenz-Artificial Intelligence.* A Report commissioned by the German Federal Ministry of Transport and Infrastructure (2017).

[Schützeneder, 2020] Schützeneder, Gabriel. *Die Modellierung von Spannungs-Dehnungs-Kurven für die Aluminiumlegierung 6082 mit Methoden des maschinellen Lernens.* Diplomarbeit, 2020 Fachhochschule Oberösterreich.

[Schwefel, 1977] Schwefel, Hans-Paul. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing-und Zufallsstrategie.* Vol. 1. Basel: Birkhäuser, 1977.

[Searle, 1992] Searle, John. *The Rediscovery of the Mind.* M.I.T. Press, Cambridge MA 1992.

[Shanks, 1955] Shanks, Daniel. *Non-linear transformations of divergent and slowly convergent sequences.* Journal of Mathematics and Physics 34.1-4 (1955): 1-42.

[Stahl, 1997] Stahl, Herbert. *Conjectures around the baker-gammel-wills conjecture.* Constructive approximation 13.2 (1997): 287-292.

[Stone et al., 2016] Stone, Peter, et al. *Artificial intelligence and life in 2030: the one hundred year study on artificial intelligence.* Report - Stanford University (2016). https://ai100.stanford.edu/sites/g/files/sbiybj18871/files/media/file/ai100report10032016fnl_singles.pdf

[Strick, 2016] Strick, Heinz Klaus. *Rafael Bombelli (1526–1572), Bombellis Werk steht in der Tradition des antiken Mathematikers Diophant.* Spektrum.de, August 2016. https://www.spektrum.de/wissen/rafael-bombelli-1526-1572/1416043

[Sun und Moscato, 2019] Sun, Haoyuan, and Pablo Moscato. *A memetic algorithm for symbolic regression.* 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2019, pp. 2167-2174.

[Turing, 1950] Turing A. M. *I.—COMPUTING MACHINERY AND INTELLIGENCE.* Mind, Volume LIX, Issue 236, October 1950, Pages 433–460. https://doi.org/10.1093/mind/LIX.236.433

[Wernick et al., 2010] Wernick, Miles N., et al. *Machine learning in medical imaging.* IEEE signal processing magazine 27.4 (2010): 25-38.

[Wright, 2010] Wright, Margaret H. *Nelder, Mead, and the other simplex method.* Documenta Mathematica 7 (2010): 271-276.

[Wuttke, 2020] Wuttke, Laurenz. *Machine Learning: Definition, Algorithmen, Methoden und Beispiele.* 2020 datasolut. https://datasolut.com/was-ist-machine-learning/#FAQ---Die-wichtigsten-Fragen-schnell-beantwortet

[Wynn, 1956] Wynn, Peter. *On a device for computing the $e_m(S_n)$ transformation.* Mathematical Tables and Other Aids to Computation (1956): 91-96.

[Wynn, 1966] Wynn, Peter. *On the convergence and stability of the epsilon algorithm.* SIAM Journal on Numerical Analysis 3.1 (1966): 91-122.