RHEINISCHE FACHHOCHSCHULE KÖLN
Fachbereich für Ingenieurwesen

# Analyse und Optimierung der Betriebsparameter eines thermochemischen Solarreaktors zur regenerativen Gewinnung von Wasserstoff

Kander Akinci

**Bachelorarbeit**

| | |
|---|---|
| Studiengang: | Elektrotechnik (B.Eng.) |
| 1 Prüfer: | Prof. Dr. rer. nat. Lampe, Jörg |
| 2 Prüfer: | Prof. Dr. Glombitza, Ulrich |

Wintersemester 2020

# Inhaltsverzeichnis

# Tabellenverzeichnis



# Codeverzeichnis

# Abbildungsverzeichnis

# Kapitel 1

# Einleitung

Der Mensch trifft in seinem täglichen Leben permanent Entscheidungen. Diese Entscheidungen sind ein Prozess der Problemlösung, die zum Erreichen eines definierten Ziels führen sollen. Viele dieser Entscheidungsprobleme sind derartig geringer Komplexität, dass das Verhalten in einer automatisierten Weise an die neuen Anforderungen angepasst werden. Die Fähigkeit, Anpassungen vorzunehmen wird als Intelligenz definiert. Dabei ist die Problemlösung keine eindeutige Kausalität. Lösungen können unterschiedliche Probleme lösen und Probleme können unterschiedliche Lösungen haben.

Um einem Problem eine Lösung zuzuordnen, handelt der Mensch nach der pragmatistischen Handlungstheorie von George Herbert Meat folgendermaßen. Liegt eine Fehlanpassung vor, werden verschiedene Handlungspläne und deren Ausgang im Gehirn nachempfunden. Nachfolgend wird der Handlungsplan mit dem Ergebnis ausgewählt, dass den Zielvorstellungen am ehesten entspricht [Mead, 1925]. Diese deterministische Vorgehensweise wird in der Regel bei Handlungen in der realen Welt gewählt, da dort das Risiko im Verhältnis zum Nutzen gering bleibt.

Komplexe technische Probleme werden meist auf eine vergleichbare Weise gelöst. Zunächst wird einem klar definierten Ziel eine Fehlanpassung unterstellt. Anschließend wird der Zusammenhang eines realen Systems zwischen Ein- und Ausgangsgrößen durch mathematische Modelle nachgebildet. Bildet das Modell das reale System hinreichend ab, kann simuliert werden, was geschieht, wenn Eingangsgrößen sogenannte Parameter, verändert werden. Welche Eingangsgrößen am ehesten zum Erreichen des Ziels führen, ist Gegenstand der Optimierung.

Die Schwierigkeit liegt darin, dass die bisher entwickelten Verfahren für Optimierungsprobleme keine Konvergenz in ein globales Optimum zusichern. Ein Verfahren, welches in einem endlichen Parameterraum nach Optima sucht, ist die Response Surface Methode (RSM). Diese in dieser Arbeit, im Gegensatz zur Anwendung in [Akinci,2020], über ein neuronales Netz modelliert wird. Diese Methode wird genutzt, um irrelevante Bereiche im Parameterraum zu lokalisieren, um so das Metamodell in erfolgversprechenderen Bereichen zu konkretisieren.

## 1.1 Problemstellung

Simulationsprogramme jüngster Zeit ermöglichen eine umfassende Analyse komplexer Systeme. Aufgrund der Komplexität kann es in einer Optimierungsumgebung jedoch zu Problemen kommen. Abhängig von dem Umfang der Probleme, erfordern einzelne Systemanalysen einen hohen Berechnungsaufwand. Bei multidisziplinären Optimierungen haben Ziel- und Restriktionsfunktionen oftmals einen nichtlinearen Verlauf, was ein vielfaches Lösen der Systemgleichung erforderlich macht. Hinzukommend kennen Optimierungsalgorithmen nur das Systemverhalten in einem Designpunkt. Dies bedeutet, dass sie nicht sagen können, wie sich das System global verhält. Ohne diese globale Sicht, ist eine Bewertung der Optimierungsqualität nicht möglich. Aufgrund der Nichtlinearität sind Optimierungsaufgaben häufig nicht konvex. Dementsprechend können mehrere lokale Lösungen existieren, wobei nur eine das globale Optimum repräsentiert. Um das das globale Optimum zu finden, muss ein enormer Aufwand betrieben werden, welcher in der Regel nicht mehr vertretbar ist.

Der Anspruch nach immer effizienteren Erzeugnissen in kürzeren Entwicklungszyklen, steht diesem Tatsache entgegen. Das Ziel der Anwendung von Verfahren zur multidisziplinären Optimierung muss also sein, die Zahl der Systemanalysen auf ein Minimum zu reduzieren.

Eine Option sind globale Approximationen, wie die Response Surface Methode (RSM). Diese generiert schnell auswertbare Approximationen, sogenannte Response Surface Approximationen (RSA), die an Stelle der umfangreichen Systemanalysen genutzt werden. Absicht ist dabei nicht, dass exakte Ergebnis zu konstatieren, sondern mit möglichst geringem Berechnungsaufwand, sich so nah wie möglich an das genaue Ergebnis anzunähern.

Die RSA müssen alle für eine Optimierung maßgeblichen systeminhärenten Eigenschaften widerspiegeln und dürfen dabei nur so wenig Systemanalysen wie möglich benötigen. Die RSA wird grundsätzlich mit polynominalen Regressionen aufgebaut [Gleichmar, 2004]. Diese Arbeit verfolgt jedoch den Ansatz, die RSA anhand neuronaler Netze zu konstruieren.

## 1.2 Zielsetzung und Aufbau der Arbeit

Ziel der Arbeit ist es daher, mit Hilfe der neu ausgelegten Response Surface Methode die Betriebsparameter eines Simulationsmodells zu Optimieren und anschließend diese Optima auf ihr Parameterverhalten zu analysieren. Das Simulationsmodell ist ein thermochemischer Solarreaktor, der in einem Kreisprozess aus Solarenergie Wasserstoff produziert. Die Reaktormodellierung und Validierung ist Teil des ASTOR-Projektes (Automatisierung Solar-Thermochemischer Krei-

sprozesse zur Reduzierung von Wasserstoffgestehungskosten) an der Rheinischen Fachhochschule. Das Modell liegt als “MATLAB- Simulink“-Datei vor, was die Wahl der Realisierungssoftware auf “MATLAB“ begründet. “MATLAB“ bringt mit seinen “Toolboxen“ genügend der benötigten Funktionen mit, um eine Optimierung dieser Art zu realisieren. Die folgenden Schritte werden unternommen, um das Ergebnis auszuarbeiten:

- Grundlagen der lokalen und globalen Optimierung (Einfluss, Restriktionen, Lösungsansätze)
- Grundlagen der Metamodellierung sowie der neuronalen Netze
- Erstellung eines differenzierten, globalen Optimierungskonzept für das Problem
- Beschreibung des Simulationsmodells und dessen Parameter
- Darlegung der Optimierungsergebnisse und die Analyse der Parameter
- Kritische Beleuchtung der Optimierungsmethode
- Fazit und Ausblick

# Kapitel 2

# Grundlagen der Optimierung

Der Begriff Optimierung definiert die Suche nach einer verbesserten Lösung einer Aufgabe. Grundlegend hierbei ist, dass im Vorfeld ein Ziel formuliert wird, damit verschiedene Lösungen als gut oder optimal eingestuft werden können. Optimierungsverfahren werden allgemein zwischen experimenteller und mathematischer Optimierung unterschieden.

Eine experimentelle Optimierung ist realisierbar, wenn die physikalischen Größen gemessen werden können. Dahingegen hängt die Möglichkeit einer mathematischen Optimierung davon ab, inwieweit das System hinreichend als mathematisches Modell beschrieben werden kann.

## 2.1 Lokale Optimierung

Als Optimierungsmodell wird das Simulationsmodell gesehen, das den mathematischen Zusammenhang zwischen den Variablen des Systems und einem Funktionswert beschreibt. Wird dieser Funktion eine Bewertungsvorschrift zugeordnet, wird sie Gütefunktion oder Zielfunktion $f$ genannt. Der Funktionswert wird von einem n-dimensionalen reellwertigen Vektor bestimmt. Als Parameter gelten alle Eingangsgrößen, die Einfluss auf die Zielfunktion haben.

Um die Optimierungsaufgabe zu lösen, muss der n-dimensionale Ortsvektor $\vec{x} \in \mathbb{R}^n$ des Parameterraums gesucht werden, bei welchem der Wert der Zielfunktion $f$ ein Extremum annimmt. Die Ergebnisse der Zielfunktion bilden eine reellwertige Hyperfläche $(\vec{x}, f(\vec{x})) \in \mathbb{R}^n \times \mathbb{R}$, dessen Extremum auf die beste oder einer der besten Lösungen hindeutet.

Der Ortsvektor der Variablen ist der Zustandspunkt eines Systems im n-dimensionalen Vektorraum $\mathbb{R}^n$ und wird mit reellen Zahlen $\mathbb{R}$ gegeben.

$$\vec{x} = (x_1, \dots, x_n)^T$$

Die Optimierungsaufgabe ist als reellwertige Zielfunktion für $n$ unabhängige Variablen im Teilbereich $\mathbb{M}$ des Definitionsbereiches $\mathbb{D}$ definiert.

$$f: \begin{cases} \mathbb{M} \subseteq \mathbb{D} \subseteq \mathbb{R}^n & \to \mathbb{R} \\ \vec{x} & \mapsto q \end{cases} \quad mit\ \mathbb{M} \neq 0$$

Die Zielfunktion kann dazu jedem Zustandspunkt $\vec{x}$ einen eindeutigen Qualitätswert $q \in \mathbb{R}$ zuweisen. Der Definitionsbereich $\mathbb{D}$, ist die Teilmenge von $\mathbb{R}^n$, bei der die Zielfunktion $f$ definiert ist. Die im nächsten Abschnitt eingeführten Restriktionen schränken den Definitionsbereich weiter ein, womit der zulässige Bereich $\mathbb{M}$ übrig bleibt.

Die Minimierung einer Zielfunktion über einen zulässigen Bereich $\mathbb{M} \neq 0$ ist zulässig, wenn die Zielfunktion $f(\vec{x})$ ihr Minimum über den beschränkten Bereich $\mathbb{M}$ annimmt.

Der Wert $f(\vec{x}_{lM})$ ist das lokale Minimum der Funktion $f$, wenn eine ε-Umgebung $U_\varepsilon(\vec{x}_{lM})$ von $\vec{x}_{lM}$ existiert. Damit die Bedingung

$$f(\vec{x}_{lM}) \leq f(\vec{x})\ f\ddot{u}r\ alle\ \vec{x} \in\ \mathbb{M} \cap\ U_\varepsilon(\vec{x}_{lM}): \tag{2.1}$$

erfüllt ist. Existiert eine ε-Umgebung, muss jedes globale Minimum auch ein lokales Minimum sein. Wohingegen ein lokales Minimum keinem globalen Minimum entsprechen muss.

### 2.1.1 Restriktionen

Restriktionen sind Zulässigkeitsbereiche, die der Funktion ‚künstlich' hinzugefügt werden, um die Beschränkung der Parameterauswahl des realen Systems wiederzugeben. Restriktionsbedingungen enthalten somit auch Korrelationszusammenhänge von Parametern, die durch technische oder physikalische Gründe entstehen können.

Der Umfang der Restriktion unterteilt diese in drei verschiedene Problemgruppen:

- Restriktionsfreie Optimierungsprobleme
- Restriktionsbehaftete Optimierungsprobleme
- Quasirestriktionsfreie Optimierungsprobleme

In dieser Arbeit ist die Optimierung quasirestriktionsfrei, was bedeutet, dass die Parameter nur durch obere und untere Grenzen beschränkt sind. Dennoch wird auf restriktionsbehaftete Optimierungsprobleme eingegangen, da diese bei Erweiterung der Komplexität zwangsläufig einhergehen. Es wird jedem Parameter ein eigener Wertebereich $\mathbb{P}_k$ zugeordnet. Dieser Bereich wird durch die hinzugefügten Zulässigkeitsbereiche definiert. Der Wertebereich des Gesamtsystems $\mathbb{P} \subseteq \mathbb{R}^n$ ergibt sich aus dem Produktraum der einzelnen Wertebereiche zu:

$$\mathbb{P} = \mathbb{P}_1 \times \mathbb{P}_2 \times \ldots \times \mathbb{P}_n$$

Der Definitionsbereich der Zielfunktion unterliegt selbstverständlich derselben Eigenschaft:

$$\mathbb{D}_f = \mathbb{D}_{f,1} \times \mathbb{D}_{f,2} \times \ldots \times \mathbb{D}_{f,n.}$$

Zusätzlich muss die Bedingung $\mathbb{P}_k \subseteq \mathbb{D}_{f,k}$ erfüllt sein.

Beeinflussen sich die Parameter untereinander, kann der gesamte Parameterraum $\mathbb{P}$ weiter auf einen zulässigen Bereich $\mathbb{M}$ reduziert werden. Diese Zusammenhänge werden durch Gleichheitsrestriktionen:

$$g_j(\vec{x}) = 0 \quad \text{für alle } \vec{x} \in \mathbb{M} \text{ mit } j = 1, \dots, e$$

$$\text{mit } g_j: \begin{cases} \mathbb{D}_{gj} \subseteq \mathbb{R}^n \to \mathbb{R} \\ \vec{x} \mapsto g_j(\vec{x}) \end{cases}$$

sowie Ungleichheitsrestriktionen:

$$g_i(\vec{x}) \leq 0 \quad \text{für alle } \vec{x} \in \mathbb{M} \text{ mit } i = (e+1), \dots, m$$

$$\text{mit } g_i: \begin{cases} \mathbb{D}_{gi} \subseteq \mathbb{R}^n \to \mathbb{R} \\ \vec{x} \mapsto g_i(\vec{x}) \end{cases}$$

berücksichtigt. Allgemein reduziert eine Gleichheitsrestriktion die Dimension des Parameterraums um eins. Die zulässige Menge $\mathbb{M}$, ergibt sich als die Teilmenge von $\mathbb{P}$, wenn alle Nebenbedingungen erfüllt sind. Die Nebenbedingung $g_i \leq 0$ in einem Punkt $\vec{c} \in \mathbb{R}^n$ ist:

verletzt, falls $g_i(\vec{c}) > 0$

erfüllt, falls $g_i(\vec{c}) \leq 0$

aktiv, falls $g_i(\vec{c}) = 0$

inaktiv, falls $g_i(\vec{c}) < 0$

Die Formulierung der Nebenbedingungen ist notwendig, wenn das Optimum am Rand des zulässigen Bereichs $\mathbb{M}$ liegt.

## 2.2 Globale Optimierungsverfahren

Im Ingenieurswesen ist häufig die Frage nach einer ‚besseren' Lösung fundamental, was sich aus der Wesensart der Natur selbst interpretieren lassen kann. Ein Funktionswert mit dem Zustandspunkt $\vec{x}_{gM}$, wird als globales Minimum bezeichnet, wenn folgende Bedingung erfüllt ist:

$$f\left(\vec{x}_{gM}\right) \leq f(\vec{x}) \quad \text{für alle} \quad \vec{x} \in \mathbb{M} \tag{2.2}$$

Die Beschränktheit einer Funktion bedingt die Existenz einer Zahl $q_{min}$ bzw. $q_{max}$ so dass,

$$q_{min} \leq f(\vec{x}) \text{ bzw. } q_{max} \geq f(\vec{x}) \text{ für alle } \vec{x} \in \mathbb{M}$$

gilt. Die kleineste Zahl $q_{min}$ wird als Infimum $inf\{f(\vec{x}) | \vec{x} \in \mathbb{M}\}$ bezeichnet und entspricht dem globalen Minimum $min\{f(\vec{x}) \mid \vec{x} \in \mathbb{M} \subseteq \mathbb{R}^n\}$ mit $q_{min} \in \{f(\vec{x}) : \vec{x} \in \mathbb{M}\}$. In der vorliegenden Arbeit ist die Optimierungsaufgabe zwar ein Maximierungsproblem, doch wird sie als Minimierungsaufgabe formuliert, was aufgrund

$$-\sup_{\vec{x}\in\mathbb{M}}(f(\vec{x})) = \inf_{\vec{x}\in\mathbb{M}}(-f(\vec{x})) \tag{2.3}$$

keine Einschränkung der Allgemeinheit bedeutet.

Die bisher entwickelten Verfahren sind in der Regel nicht in der Lage ein globales Optimierungsproblem zu lösen. Die Schwierigkeit ist darin begründet, dass die Information der Extrema eines Punktes über deren erste oder zweite Ableitung gewonnen wird. Ist die Zielfunktion stetig differenzierbar, kann dadurch zugesichert werden, ob es ein lokales Minimum ist, oder falls nicht, wo sich ein niedrigerer Funktionswert befindet. Eine somit konstruierte Folge von Punkten konvergiert zwar in einem lokalen Minimum, ist aber nicht hinreichend, um eine globale Optimalität zu gewährleisten. Ein allgemeines, konstruktives Kriterium für ein globales Minimum konnte bisher nicht gefunden werden.

Die in dieser Arbeit vorliegende digitale Optimierungsaufgabe birgt darüber hinaus zusätzliche Schwierigkeiten, die aus der begrenzten Genauigkeit von Digitalrechnern resultiert. Man kann ein globales Optimierungsproblem als gelöst betrachten, wenn für $\varepsilon > 0$ entweder ein Element einer ε-Umgebung von $\vec{x}_{gM}$

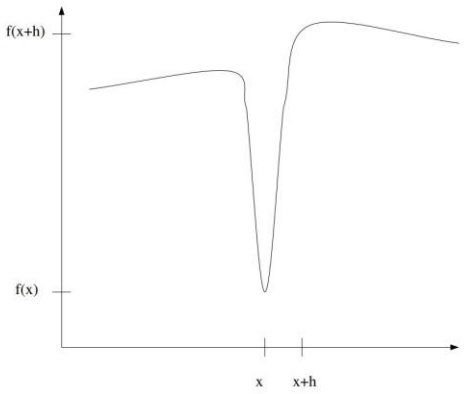
$$A_x(\varepsilon) := U_\varepsilon(\vec{x}_{gM}) = \{\vec{x} \in \mathbb{M} \mid \|\vec{x} - \vec{x}_{gM}\| \leq \varepsilon\} \tag{2.4}$$

oder ein Element der Niveaumenge

$$A_f(\varepsilon) := L_{f_{gM}+\varepsilon} = \{\vec{x} \in \mathbb{M} \mid f(\vec{x}) \leq f(\vec{x}_{gM}) + \varepsilon\} \tag{2.5}$$

berechnet werden kann. Der Nachteil der Methode von (2.4) ist, dass der Abstand zweier Punkte sehr stark verkleinert werden kann, doch die Funktionswerte sich dennoch beliebig unterscheiden können. Verdeutlicht wird das in Abb.2.1.

Bei der Möglichkeit (2.5) existiert keine Methode, ein Element der Menge mit einer endlichen Anzahl von Schritten zu finden. Sei die Zielfunktion $f(\vec{x})$ stetig und in einem zulässigen Bereich beschränkt, dann ist es dennoch möglich, dass der globale minimale Wert der Funktion $f$ vom bisher bestgefundenen Wert beliebig abweicht [Törn & Žilinskas, 1989]



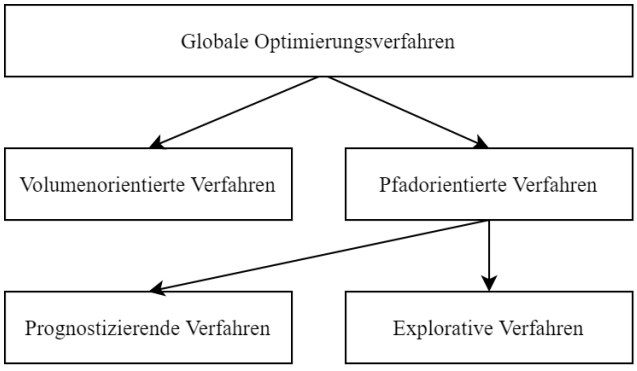

Quelle: Entnommen aus *Rudolph* et al. (1990: 30).

Abbildung 0.1: Darstellung einer Umgebung.

Damit das Problem angegangen werden kann, werden für verschiedene Lösungsideen bestimmte Annahmen getroffen, was dazu führt, dass diese Lösungsverfahren für Spezialfälle des allgemeinen Problems (2.2) gültig sind. Einige dieser Lösungsverfahren werden im folgenden Abschnitt erläutert.

### 2.2.1 Lösungsverfahren

Globale Optimierungsverfahren können in vier verschiedene Klassen eingeteilt werden, welche je nach verwendeten Unterteilungscharakteristika sehr unterschiedlich ausfallen können. Wichtig hierbei ist, dass die Klassen alle möglichen Methoden abdecken, aber dennoch die resultierenden Klassen sich nicht überlappen. Zunächst sollen die Klassifizierungen in Abbildung 2.2 kurz erläutert werden.




Quelle: In Anlehnung an *Rudolph* et al. (1990: 31).

Abbildung 0.2: Klassifikation der Lösungsverfahren.

Eine der Unterscheidungscharakteristika ist die vorliegende Suchphilosophie. Beim *volumenorientierten* Verfahren geht man davon aus, dass der gesamte zulässige Bereich $\mathbb{M}$ abgesucht werden muss, um das globale Minimum zu finden. Das Vorgehen bedingt unvermeidbarerweise einen endlichen Suchraum, was die Namensgebung dieser Klasse motiviert, denn einem begrenzen Raum kann ein Volumen zugeordnet werden.

Die *pfadorientierten* Verfahren versuchen den Aufwand von $\sim c^n$ zu vermeiden, indem sie davon ausgehen, dass der Weg im $\mathbb{R}^n$ im günstigsten Fall $\sim \sqrt{n}$ lang ist. Um einen Weg zum globalen Optimum zu finden, existieren zwei verschiedene Ansätze.

Eine dieser Ansätze ist das *prognostizierende* Verfahren, wobei der nächste Optimierungsschritt anhand eines expliziten Modells der Zielfunktion vorhergesagt wird. Treffen diese Prognosen dem weiteren Verlauf des Verfahrens nicht zu, wird das Verfahren meist abgebrochen.

Der andere Ansatz ist das *explorative* Verfahren, dass kein explizites Modell besitzt, sondern über das Probieren verschiedener Wege, versucht ans Ziel zu gelangen. Führt ein Weg im weiteren Verlauf nicht zum Ziel, wird ein alternativer Weg eingeschlagen.

Jede dieser Verfahren bietet unterschiedliche Methoden zur Lösung eines globalen Optimierungsproblems. Eine Übersicht der gängigen Methoden zeigt Tabelle 2.1 zu denen diverse Modulationen und Hybridkonfigurationen existieren.

| Volumenorientiertes Verfahren | Prognostizierende Verfahren | Explorative Verfahren |
|---|---|---|
| • Rastersuche<br>• Branch and Bound- Verfahren<br>• Überdeckungsverfahren<br>• Einfache Zufallssuche<br>• Adaptiv stochastische Automaten<br>• Cluster- Verfahren<br>• Bayes'sche Methode | • Trajektorien-Verfahren<br>• Tunnelungs- Verfahren | • Rotierende Koordinaten<br>• Muster- Suche<br>• Complex-Strategie<br>• Simulated Annealing<br>• Evolutionsstrategien |

Tabelle 0.1: Übersicht einiger Lösungsverfahren für globale Optimierungsprobleme.

Die Auswahl einer passenden Methode ist in vielen Fällen eine nichttriviale Vorleistung, zumal die vom Anwender zu berücksichtigenden Faktoren nicht immer eindeutig sind. Eine Kombination und oder Modifikation der Methoden zwecks Anpassung an Einflussfaktoren sollte wenn möglich vorgenommen werden.

# Kapitel 3

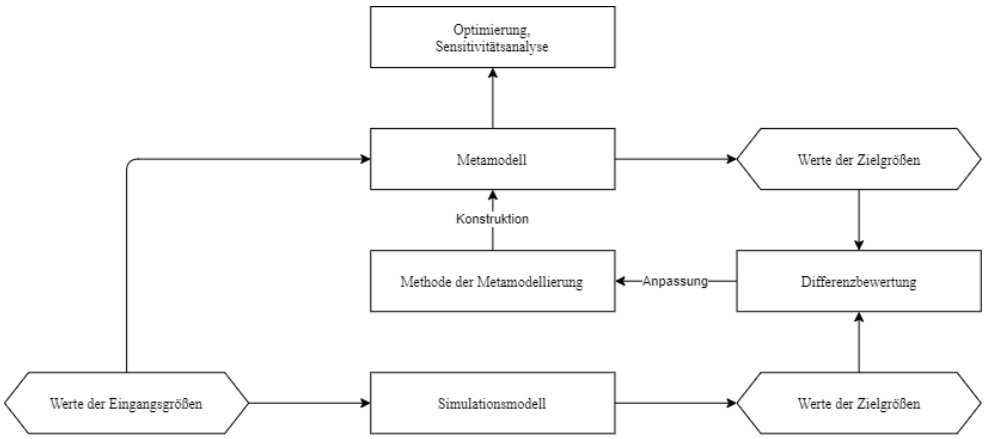
# Neuronale Netze

## 3.1 Metamodellierung

Die Metamodellierung befasst sich mit dem Aufbau eines vereinfachten Modells des Simulationsmodells, welches aus deren Simulationsergebnissen in einer moderaten Anzahl an Experimenten gewonnen wurde. Die Zusammenhänge der Eingangs- und Ausgangsgrößen der Simulation werden in mathematischen Modellen approximiert, wodurch Ausgangsgrößen nicht simulierter Zustandspunkte prognostiziert werden können. Die Verhaltensprognose komplexer Systeme bringt den Vorteil, dass rechenaufwendige Simulationen auf das Nötigste beschränkt werden können.

Ein Metamodell bildet die wesentlichsten Eigenschaften eines Modells ab, was sich als praktisches und robustes Werkzeug zur Interpretation und Analyse hochkomplexer Modelle erwiesen hat [Friedman, 1996].

Existieren keinerlei Informationen über das Modellverhalten, wird die Metamodellierung rein über die vorhandenen Daten von Eingangs- und Ausgangsgrößen des Modells entwickelt. Die Abbildung 3.1 zeigt ein allgemeines Schema zur Konstruktion eines Metamodells. Die Eingangswerte sind für Metamodell und Simulationsmodell dieselben, wobei die Ausgänge voneinander abweichen. Das Metamodell ist so zu konstruieren, dass dessen Zielgrößen, die Zielgrößen der Simulation approximieren. Dies kann Beispielsweise über ein iterativinkrementelles Vorgehen erreicht werden, bei dem die Differenzbewertung in ein ausreichend positives Maß getrieben wird.




Quelle: In Anlehnung an Nissen (1994).

Abbildung 0.1: Allgemeines Schema zur Metamodellierung.

Ein Ziel bei der Simulationsstudie ist es, Eingangswerte zu gesuchten Zielgrößen ausfindig zu machen. Eine globale Betrachtung dieses Ziels kann erfolgen, indem das Metamodell im gesamten zulässigen Parameterraum erzeugt wird. Entspricht ein globales Metamodell mit einer gewissen Adäquatheit dem Modell, können klassische Optimierungsverfahren angewendet werden. Die Schwierigkeit besteht darin, das nichtlineare Systeme mit einem geringen kontingent an Experimenten nur ausreichend approximiert werden können, wenn das Systemverhalten schon bekannt ist. Die Zahl der benötigten Systempunkte steigt zudem exponentiell mit der Anzahl der Eingangsparameter, was die Approximation rechenintensiver Simulationen begrenzt.

Eine Möglichkeit globale Metamodelle zu verwenden, besteht darin, sie als eine Art Indiz Geber zu behandeln, deren Aufgabe es nicht ist, eine möglichst exakte Prognose zu erstellen, sondern den ungefähren Verlauf der Zielfunktion anzudeuten. Die in dieser Arbeit ausgewählte Methode ist eine Regression mit künstlichen neuronalen Netzen (KNN). Diese Entscheidung geht aus der Vermutung hervor, dass der Anpassungsfehler bei kleinen Datensätzen zwar hoch ist, jedoch die Generalisierungsfähigkeit erhalten bleibt. Der Verlauf eines solchen ‚Urnetzes' kann auf erfolgsversprechende Bereiche hinweisen.

Ein anderes Ziel der Simulationsstudie ist das bessere Verständnis des Simulationsmodells. Hierfür können durch den geringen Rechenaufwand Einflüsse von Eingangsparametern auf die Zielfunktion bestimmt werden. Die durch den Approximationsvorgang verlorene Informationsmenge erzeugt die Komplexitätsdifferenz zum Simulationsmodell, was die Interpretierbarkeit verbessert [Friedman, 1996]. Kleijnen und Sargent [Kleinjnen & Sargent, 2000] unterscheiden hierbei drei Verständnisstufen, die erreicht werden können:

- *Richtung* der Zielgröße: zeigt, ob die Zielgröße bei steigendem Eingangswert steigt oder fällt. Hierfür können einfache Metamodelle verwendet werden.

- *Screening:* zeigt die wichtigsten Faktoren auf. Einfache Metamodelle reichen in der Regel aus (Polynom ersten Grades)
- *Haupteffekte, Korrelation, quadratische und höhergradige Effekte:* Effekte eines Spezifischen Parameters ggf. in Kombination mit anderen Faktoren, werden sichtbar. Polynome ersten oder zweiten Grades reichen in der Regel für solche Betrachtungen aus.

Die Wahl des Richtigen Metamodells ist vom Ziel abhängig.

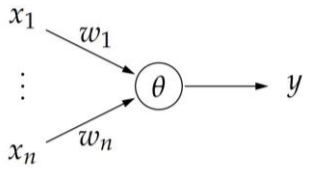
## 3.2 Neuronale Netze

Künstliche neuronale Netze sind Informationsverarbeitende Systeme, deren Struktur und Funktionsweise dem Gehirn von Menschen oder Tieren nachempfunden ist. Sie bestehen aus einer großen Anzahl parallel arbeitender Einheiten, sogenannten *Neuronen*. Diese senden Informationen der Aktivierungszustände über gerichtete Verbindungen.

Der Grund für den Erfolg vielschichtiger neuronaler Netze ist die Fähigkeit, komplizierte Konzepte aus einfachen herzuleiten. Man kann sich vorstellen, dass bei der Erkennung von handgeschriebenen Zahlen, jede Schicht eine andere geometrische Eigenschaft erkennt. Somit lassen sich Konturen und Muster zu einem Gesamtkonstrukt verbinden. Von einem solchen inneren Entscheidungsmuster kann aber nicht allgemein ausgegangen werden. Es ist in der Regel nicht genau klar, wie ein neuronales Netzt zu seinen Ergebnissen kommt.

### 3.2.1 Schwellenwertelement

Eine schematische Darstellung eines künstlichen Neurons mit den Eingängen (*Inputs*) $x_1, \dots, x_n$ und Ausgang (*Output*) ist in Abbildung 3.2 zu sehen.




Quelle: Entnommen aus *Kruse* et al. (2011: 14).

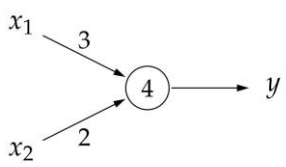
Abbildung 0.2: Schwellwertelement.

Durch den Aktivierungscharakter von Neuronen liegt es nahe, sie als Schwellenwertelemente zu modellieren. Erhält ein Neuron genügend Impulse, die nicht durch entsprechend starke Gegenimpulse ausgeglichen werden, wird es aktiviert und sendet ein Signal an andere Neuronen.

Ein Schwellenwertelement ist eine Verarbeitungseinheit für reelle Zahlen mit $n$ Eingängen $x_1, \ldots, x_n$ und einem Ausgang $y$. Der Einheit ist ein Schwellenwert $\theta$ und jedem Eingang $x_k$ ein Gewicht $w_k$ zugeordnet. So berechnet ein Schwellenwertelement die Funktion

$$y = \begin{cases} 1, & falls \sum_{k=1}^{n} w_k x_k \geq \theta, \\ 0, & sonst. \end{cases} \tag{3.1}$$

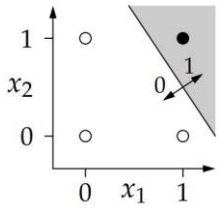
Die Eingänge $x_1, \ldots, x_n$ werden zu einem Eingangsvektor $\vec{x} = (x_1, \ldots, x_n)^T$ und die Gewichte $w_1, \ldots, w_n$ zu einem Gewichtsvektor $\vec{w} = (w_1, \ldots, w_n)^T$ zusammengefasst. Unter Verwendung des Skalarproduktes kann die Bedingung auch $\vec{w}\vec{x} \geq \theta$ geschrieben werden. Um die Funktionsweise zu verstehen, betrachten wir das Beispiel in Abbildung 3.2 a), dass eine Schwellwerteinheit mit den Gewichten $w_1 = 3$ bzw. $w_2 = 2$ und den Eingängen $x_1$ und $x_2$ zeigt. Der Schwellenwert ist $\theta = 4$. Können die Eingabevariablen nur einen Wert von 0 oder 1 annehmen, wird der Ausgang y nur aktiviert, wenn beide Eingangsvariablen 1 sind. Die somit erzeugte Konjunktion kann wie in Abbildung 3.2 c) auch grafisch dargestellt werden.




a)

| $x_1$ | $x_2$ | $3x_1 + 2x_2$ | $y$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 3 | 0 |
| 0 | 1 | 2 | 0 |
| 1 | 1 | 5 | 1 |

b)




c)

Quelle: Entnommen aus *Kruse* et al. (2011: 15-16).

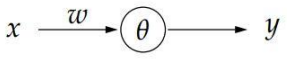
Abbildung 0.3: Schwellwertelement für die Konjunktion $x_1 \wedge x_2$.

Visuell beschränkt sich die Darstellbarkeit solcher Grafiken auf drei Eingangsvariablen, was sich durch die begrenzte Vorstellungskraft ergibt. Dennoch ist die Zahl der Eingangsvariablen nicht begrenzt.

Die zuvor besprochene geometrische Interpretation der Berechnungen hilft dabei die Vorgänge des Trainings besser zu verstehen. Um ein Verfahren zur Anpassung von Gewichten und Schwellenwerten zu finden, wird von der Überlegung ausgegangen, dass die Werte von Gewichten und Schwellenwert ein mehr oder weniger richtiges Ergebnis erzeugen. Aufgrund dessen wird eine Fehlerfunktion $e(w_1, \ldots, w_n, \theta)$ definiert, die zeigt, wie weit mit bestimmten Gewichten und

Schwellenwerten eine Funktion mit gewünschten Werten übereinstimmt. Ziel ist es, die Gewichte und den Schwellenwert so zu wählen, dass die Fehlerfunktion 0 wird.

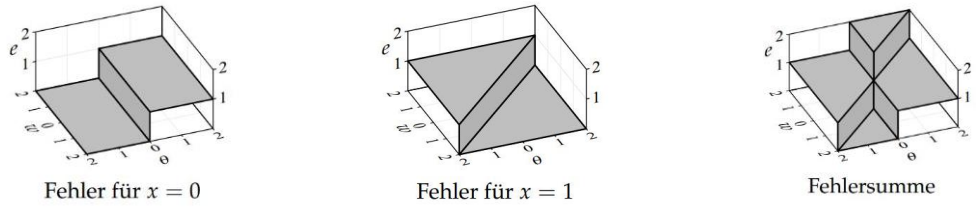
Dieses Vorgehen soll an einem Schwellenwertelement mit nur einem Eingang veranschaulicht werden, dessen Aufgabe die Negation des Eingabewertes ist. Wie Abbildung 3.4 zeigt, soll eine 1 ausgegeben werden, wenn der Eingang 0 ist und eine 0, wenn der Eingang 1 ist.




| $x$ | $y$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

Quelle: Entnommen aus *Kruse* et al. (2011: 22).

Abbildung 0.4: Ein Schwellenwertelement mit einem Eingang und ein Trainingsbeispiele für die Negation.




Quelle: Entnommen aus *Kruse* et al. (2011: 22).

Abbildung 0.5: Fehler für die Berechnung der Negation mit Schwellenwert.

Als Fehlerfunktion wird die absolute Differenz zwischen Ist- und Sollwert definiert. Das Diagramm in Abbildung 3.5 a) zeigt, dass bei einer Eingabe von $x = 0$ die Ausgabe 1 wird, denn mit $wx \geq \theta$ ist der Fehler bei positivem Gewicht 1. Das Gewicht $w$ spiel durch die Multiplikation mit $x = 0$ keine Rolle. Ist die Eingabe $x = 1$, kann man in Abbildung 3.5 b) erkennen, dass Gewicht und Schwellenwert eine Rolle spielen. Ist $wx$ kleiner als der Schwellenwert $\theta$, erfolgt eine Ausgabe und der Fehler wird 0. Die Fehlersumme in Abbildung 3.5 c) verdeutlicht, wie Gewicht und Schwellenwert gewählt werden müssen, damit das Schwellenwertelement eine Negation berechnet.

Die Suche nach den Bereichen die einen möglichst kleinen Fehler erzeugen, gestaltet sich in der Praxis schwieriger da eine solche Anschaulichkeit nicht verfügbar ist. Vielmehr muss aus dem

zufälligen Startpunkt heraus, eine Wegrichtung gesucht werden, welche den Fehler verringert. Liegt der Punkt wie in diesem Fall auf einem Plateau, kann keine Wegrichtung berechnet werden. Um dieses Problem zu umgehen, wird die Fehlerfunktion so verändert, dass sie umso ‚falscher' wird, umso weiter man sich vom Fehlerwert 0 entfernt (Beschrieben wird dieser Vorgang in Abschnitt 3.2.2). Mit der gewünschten Ausgabe $o$ und dem tatsächlichen Ausgabewert $y$ kann die Lernmethode für $y \neq o$ formuliert werden:

$$\theta^{(neu)} = \theta^{(neu)} + \Delta\theta \quad mit \quad \Delta\theta = \eta(o - y),$$
$$\forall i \in \{1, \dots, n\}: w_i^{(neu)} = w_i^{(alt)} + \Delta w_i \quad mit \quad \Delta w_i = \eta(o - y)x_i,$$

Mit der Lernrate $\eta$ wird die Stärke der Änderung skaliert.

Diese getrennten Formulierungen sind zwingend, da Gewichtung und Schwellwert in entgegengesetzte Richtungen verändert werden müssen. Um eine einheitlichere Änderungsregel zu formulieren, wird dem Schwellenwertelement ein zusätzlicher Eingang $x_0$ hinzugefügt, dessen Wert dauerhaft 1 ist. Darauffolgend kann mit einer Gewichtung von $w_0 = -\theta$ der Einfluss des Schwellenwertes als Gewichtung formuliert werden. Dies ermöglicht, dass die vorangegangene Formulierung mit dem Eingabevektor $\vec{x} = (x_0 = 1, x_1, \dots, x_n)^T$ und Gewichtsvektor $\vec{w} = (w_0 = -\theta, w_1, \dots, w_n)^T$ folgendermaßen zu verändern ist:

$$\forall i \in \{0,1, \dots, n\}: \vec{w}_i^{(neu)} = \vec{w}_i^{(alt)} + \Delta\vec{w}_i \quad mit \quad \Delta\vec{w}_i = \eta(o_i - y_i)\vec{x}_i.$$

So kann ein Datensatz der dem Eingang $\vec{x}_i$ einen Ausgangswert $o_i$ zuweist, dafür verwendet werden, die Gewichte anzupassen. Zu einer Datenmenge $i = 1, \dots, m: \; L = \{(\vec{x}_1, o_1), \dots, (\vec{x}_m, o_m)\}$ mit $m$ Datenpunkten, können somit $m$ Gewichts Änderungen berechnet werden. Passt man die Gewichte nach jeder berechneten Gewichtsänderung $\Delta w_i$ an, nennt man das *Online-Training* (engl. *online*: mitlaufen, schritthalten). Beim *Batch-Training* (engl. *batch*: Stapeln) werden zunächst alle Gewichtsänderungen mit demselben Startpunkt berechnet. Diese Änderungen werden zu einer Gesamtänderung $\vec{w}_c$ aufsummiert. Zum Schluss wird die Gesamtänderung auf den Startpunkt aufaddiert.

### 3.2.2 Netzstruktur

Auch wenn das Schwellenwertelement als Grundlage für das Prinzip lernender Systeme gilt, können Schwellenwerteinheiten nicht in Netzwerkstrukturen angeordnet werden. Für das Training eines künstlichen neuronalen Netzes ist es wichtig, dass kleine Änderungen der Parameter, eine kleine Änderung des Outputs bewirken. Wie die Abbildung 3.5 in Abschnitt 3.2.1 verdeutlicht, bewirkt die Veränderung der Parameter bei Schwellenwerteinheiten entweder keine Änderung oder eine extreme Änderung des Outputs. Dieser Umstand bewirkt, dass das Verhalten des gesamten Netzes als lineare Funktion beschreibbar ist. Die Anzahl der Netzschichten hätte somit

keinen Einfluss und das gesamte Netz wäre durch eine Einheit darstellbar [Schmitz, 2017]. Zudem verliert das Netz die Fähigkeit, nichtlineare Funktionen ausreichend anzunähern, was eine der Hauptaufgaben künstlicher neuronaler Netze ist.

Um diesen Umstand zu umgehen, wird die Ausgabe *y* der Schwellenwerteinheiten durch Einsetzen in die Gleichung (3.1) der Tangens-hyperbolicus Funktion, auch Sigmoid Funktion genannt, verändert. Wie in Abbildung 3.6 zu sehen ist, kann die unipolare Sigmoid Funktion

$$z = \sigma(y) = \frac{1}{1 + e^{-y}} \tag{3.2}$$

jedem Wert der gewichteten Summe, einen Wert zwischen eins und null zuweisen. Somit sind die Eingangswerte nicht auf eins oder null begrenzt und der Output des Netzes kann jeden Wert annehmen. Die veränderte gewichtete Summe *y*, ist somit die neue Ausgabe *z* des Neurons. Die erzeugte Differenzierbarkeit, ist die Voraussetzung einen gewünschten Parameterbereich zu finden, ohne Informationen über den Verlauf der Fehlerfunktion zu verfügen.

Es gibt neben der unipolaren noch die bipolare Sigmoid Funktion, welche der gewichteten Summe einen Wert zwischen minus eins und eins zuweisen kann. Die Einflüsse der zahlreichen Aktivierungsfunktionen sind vielfältig und können in [Krause, et al., 2011] nachgelesen werden.

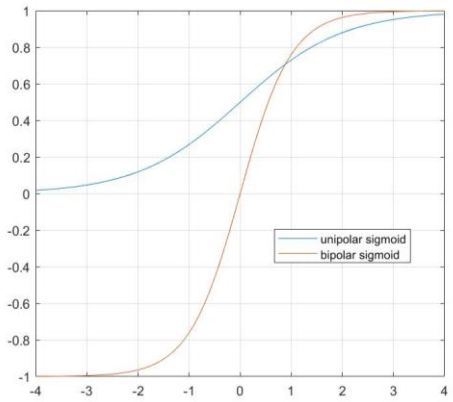



Abbildung 0.6: Unipolar und bipolare Sigmoid Funktionen.

Ein künstliches neuronales Netz besteht aus mehreren Schichten solcher Neuronen. Der Output der Neuronen aus der ersten Schicht, kann als Input der Neuronen der nächsten Schicht gesehen werden. Die Eingabe in das Netz realisiert die erste Schicht, deren Eingabeparameter *Features* genannt werden. Jene bilden den Teil des Datensatzes, dessen Vorgabe ein Ergebnis erzeugen soll. Die *Features* der letzten Schicht bilden den Output des Netzes. Das gesamte Netz und die einzelnen Neuronen verfügen über Ein- und Ausgänge. Schichten, die zwischen Inputlayer und Outputlayer liegen, werden ‚verdeckte' Schichten genannt.

Die Vernetzungstopologie der Neuronen, sowie ihre Struktur, kann individuell zum untersuchten Problem angepasst werden. Im Folgenden werden nur sogenannte *Feed Forward Neural Networks* betrachtet. Bei dieser Struktur erhalten die Neuronen keinen Input aus darauffolgenden Schichten. Der Einfachheit halber wird davon ausgegangen, dass jede Schicht aus Knoten besteht, bei dem jeder Knoten einer Schicht, mit jedem Knoten der darauffolgenden verbunden ist, sogenannte (*fully connected layer)*.

### 3.2.3 Training von künstlichen neuronalen Netzen

Das Training erfolgt ähnlich des Deltaverfahrens durch die Anpassung nach dem Kriterium des niedrigsten Fehlers. Die Abweichung vom Sollwert und somit der Fehler kann bei *Deep Neural Networks* (mehr als ein Hidden Layer) nur am Outputlayer festgestellt werden. Dort wird der Fehler meist durch die mittlere quadratische Abweichung (*Mean squared error*) MSE beschrieben. Beim Batch-Training wird das Maß der erfolgreichen Anpassung nach den Abweichungen aller Datenpunkten bestimmt. Dieses Maß wird Kostenfunktion $C$ genannt und ist als eine Abweichung der Entsprechung des KNN an einem Datensatz zu verstehen. Der Wert der Netzausgabe $z_i$, ist von den Gewichtungen $\vec{w}_j$, der einzelnen Neuronen abhängig. Mit der Anzahl $m$ der Datenpunkte und den Gewichtungen als Matrix $W$, kann die Kostenfunktion formuliert werden:

$$i = 1, \dots, m: \quad C(W) = \frac{1}{2m} \sum_{i=1}^{m} \|o_i - z_i(W)\|^2 \tag{3.3}$$

Die Kostenfunktion $C$ ist somit durch den berechneten Output von allen Gewichten der einzelnen Neuronen abhängig. Die Abweichung zu minimieren wird als Optimierungsproblem definiert. Um dieses zu lösen, betrachten wir zunächst an einem Neuron mit dem Index $j$, wie sich die Veränderung der Kostenfunktion im Verhältnis zu der Veränderung der Gewichtung zusammensetzt.

Die Änderung der gewichteten Summe $y_j$, hängt direkt von den Gewichten $w_{ij}$ ab. Die Ausgabe $z_j$ wiederum, direkt von der gewichteten Summe $y_j$ und schließlich resultiert die Kostenfunktion $C$ aus dem Ausgang $z_j$. Die partielle Ableitung der Kostenfunktion ergibt sich somit durch die Verwendung der Kettenregel:

$$\frac{\partial C}{\partial w_{ij}} = \frac{\partial C}{\partial z_j} \frac{\partial z_j}{\partial y_j} \frac{\partial y_j}{\partial w_{ij}} \tag{3.4}$$

Aus den Termen wird unter Berücksichtigung von zwei Unterscheidungen die Formel für die Veränderung der Gewichte hergeleitet. Es wird unterschieden, ob das Neuron in einer verdeckten oder einer Ausgabeschicht liegt. Dies ist wichtig, da die Änderung der Gewichte von nachfolgenden Neuronen mit dem Index $k$ beeinflusst werden.

$$\delta_j = \begin{cases} \sigma'(y_j)(z_j - o_j) & falls\ j\ Ausgabeneuron\ ist, \\ \sigma'(y_j)\sum_k \delta_k w_{jk} & falls\ j\ verdecktes\ Neuron\ ist. \end{cases}$$

$$\Delta w_{ij} = -\eta \frac{\partial C}{\partial w_{ij}} = -\eta \delta_j z_j \tag{3.5}$$

Da die Gewichtsänderungen von nachfolgenden Neuronen beeinflusst werden, dass Netz vom Ausgang-Richtung-Eingang berechnet wird, nennt man dieses Verfahren *Backpropagation* (Rückpropagierung).

Unterschiedliche Initialisierungen der Gewichte des Netzes können dazu führen, dass die Optimierung der Kostenfunktion in unterschiedliche lokale Minima konvergiert. Die Wahrscheinlichkeit in das globale Minimum zu konvergieren, steigt mit der Anzahl unterschiedlicher Initialisierungspunkte. Das Training mit einem Datensatz zu wiederholen, kann folglich bessere Ergebnisse liefern. Diese Durchläufe nennt man *epochs* (Epochen).

# Kapitel 4

# Optimierungskonzept

Bei der Erstellung des Optimierungskonzeptes wurde versucht die Gegebenheiten des Optimierungsproblems in seinen einzelnen Stadien immer wieder neu zu bewerten, um somit für jede Stufe eine passende Methode zu verwenden.

## 4.1 Betrachtung der Optimierungsumgebung

Zu Beginn fällt bei Betrachtung der Optimierungsaufgabe ins Auge, dass die zu optimierende Funktion nicht als stetige Funktion vorliegt. Stattdessen muss für jeden Zustandspunkt $\vec{x}_p \in \mathbb{M}$, der Zustandspunkte $p = 1, 2, \ldots, N$ mit $N \in \mathbb{N}$, dass zugehörige $f(\vec{x}_p)$ berechnet werden. Die abzählbare oder höchstens abzählbar unendliche Menge an Zustandspunkten unterscheidet sich zu dem Gebiet der Analysis, die sich mit kontinuierlichen Funktionen auf nicht abzählbaren unendlichen Mengen beschäftigt. Um die logischen Fundamente der Analysis auf die diskrete Mathematik anzuwenden, werden Approximationen gebildet, welche eine kontinuierliche Funktion abschätzen.

Eine Möglichkeit das Minimum einer Funktion zu bestimmen ist, für eine abzählbar unendliche Menge an Punkten, den Wert der Zielfunktion zu berechnen und den Punkt mit dem kleinsten Wert zum Minimum zu erklären. Bei dieser Art der Optimierung können theoretisch alle Versuche gleichzeitig berechnet werden, weshalb man diese Vorgehensweise als simultan bezeichnet. Wie nah das ausgewählte Minimum dabei vom tatsächlichen Minimum entfernt ist, hängt von den Punkten - ihrer Anzahl und Platzierung - ab. Die Genauigkeit des Ergebnisses steigt dabei mit der Anzahl der Versuche, die unternommen werden. Um den Ressourcenaufwand (Zeit, Material) in Grenzen zu halten, muss ein Kompromiss geschlossen werden.

Die Qualität der Suchmethode wird über das größte verbleibende Unsicherheitsintervall und der Anzahl der Versuche bestimmt. So ist die Länge des Unsicherheitsintervalls z.B. der Abstand der zwei Punkte mit dem kleinsten Wert um ein Minimum.

## 4.2 Rastersuche

Gibt es wie in diesem Fall keine Anhaltspunkte über den Ort der Minima im zulässigen Bereich $\mathbb{M}$ , ist theoretisch beweisbar, dass bei der simultanen Berechnung eine gleichmäßige Verteilung der Versuchspunkte in $\mathbb{M}$ optimal ist [Schwefel, 1977]. Werden $N$ äquidistante Punkte verwendet, in einem Intervall von $[a\ b]$, dann hat das Unsicherheitsintervall eine Länge von

$$l = \frac{1}{N+1}(b-a). \tag{2.6}$$

Was so viel bedeutet wie: Um eine Genauigkeit ε zu erzielen, benötigt die Rastersuche

$$\frac{2(b-a)}{\varepsilon} - 1 \ < N \ \leq \frac{2(b-a)}{\varepsilon} \qquad , \quad N \in \mathbb{N} \tag{2.7}$$

Versuche.

Um die Suche nach $N$ für einen n-dimensionalen quasirestriktionsfreien Parameterraum zu verallgemeinern, werden alle Intervalle $[a\ b]_p$ zu einem dimensionslosen Intervall $[c\ d]$ umgewandelt. Die Legitimität dieser Vorgehensweise wird im Themengebiet der Dimensionsanalyse in [Görtler, 1975] und [Bridgman, 1932] detailliert beschrieben.

Der zulässige Bereich, folgend auch als die zulässige Menge der Zustandspunkte genannt, wird durch ein Hyperwürfel mit einer Länge $L = \mathrm{d} - \mathrm{c}$ beschrieben. Diese Menge $M_L$, könnte auch durch eine Anzahl $N$, von kleineren Mengen $M_l$ mit einer unbekannten Länge $l < L$ angegeben werden.

$$M_L \coloneqq \bigcup_{i=1}^{N} M_l^{(i)} \ , \qquad M_l^{(i)} \cap M_l^{(j)} = 0 \ \ \forall i \neq j. \tag{2.8}$$

Das Volumen des Hyperwürfels berechnet sich mit der Länge $L$

$$v(M_L) = L^n \tag{2.9}$$

welches sich nach (2.8) auch wie folgt formulieren lässt:

$$v(M_L) = \sum_{i=1}^{N} v\left(M_l^{(i)}\right) = N\ v(M_l) \tag{2.10}$$

Durch die Umformung von (2.10) und das Einsetzen von (2.9) ergibt sich die Anzahl der benötigten Rasterpunkte $N$ [Törn & Žilinskas, 1989] durch

$$N = \frac{v(M_L)}{v(M_l)} = \frac{L^n}{l^n} = \left(\frac{L}{l}\right)^n. \tag{2.11}$$

Sollen die Punkte auf den Grenzen des Parameterraums liegen, kann (2.11) erweitert werden.

$$N = \left(\frac{L+l}{l}\right)^n \tag{2.12}$$

Wird der Abstand $l$ festgelegt, so sind $N = \left(\frac{L+l}{l}\right)^n$ Auswertungen der Zielfunktion nötig, um das Unsicherheitsintervall $l$ zu erreichen. Klar ist hierbei, dass bei einer qualitativen Suche, $l$ sehr klein sein sollte, was bedeutet, dass der Suchaufwand exponentiell mit den Dimensionen des Suchraums ansteigt.

Auf diese Weise in den Optimierungsprozess einzusteigen hat den Vorteil, dass die voranschreitenden Ressourcen der parallelen Berechnung der Zielfunktion eine erhöhte Genauigkeit in kürzerer Zeit ermöglichen. Wie viele äquidistante Punkte berechnet werden, ist von der verfügbaren Hardware- und Zeitressourcen abhängig.

Um diese Ressourcen effizient einzusetzen, ist es von Vorteil, zulässige Parameterbereiche die augenscheinlich kein ausreichend gutes Ergebnis liefern, von der weiteren Suche auszuschließen.

## 4.3 Hybridkonfiguration

Eine Möglichkeit den Suchraum einzugrenzen besteht darin, den Suchraum in Bereiche aufzuteilen. Dabei werden im normierten Suchbereich mehrere Hyperwürfel mit dem Volumen $v(M_l)$ so angeordnet, dass jeder Versuchspunkt $p$ den ‚Center- Point' von der Menge $M_p$ bildet. Die Länge dieser Hyperwürfel mit der des Unsicherheitsintervalls $l$ gleichzusetzen, bring den Vorteil, dass sich die Grenzen benachbarter Mengen nicht überlappen [Rudolph, 1990].

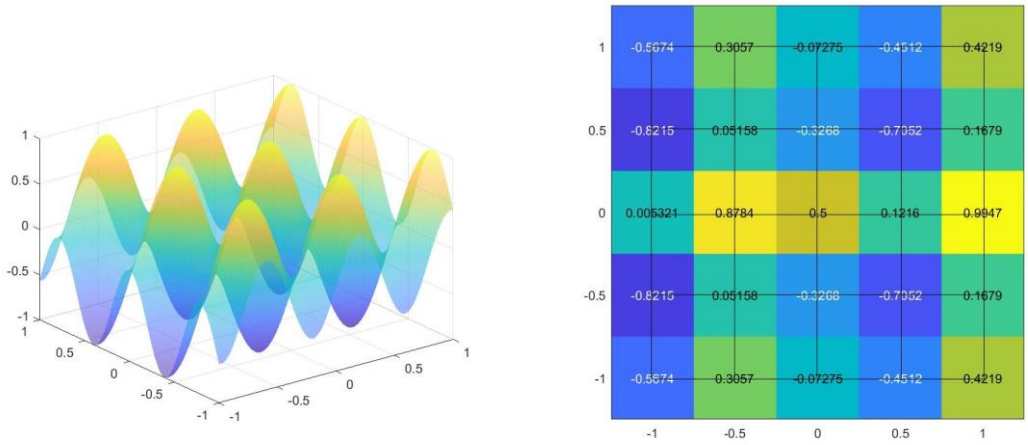
Um dieses Verfahren zu veranschaulichen, werden für die Funktion,

$$f(\vec{x}) = 0{,}5 * (\sin(8x_1) + \cos(8x_2)) \,, \vec{x} \in M_L \subseteq \mathbb{R}^2$$

$$[-1{,}1] = \{x_1 \in \mathbb{R} | -1 \leq x_1 \leq 1\},$$

$$[-1{,}1] = \{x_2 \in \mathbb{R} | -1 \leq x_2 \leq 1\}$$

Teilbereiche des Suchbereichs mit einem Unsicherheitsintervall $l = \frac{L}{4}$, gebildet. Wobei $L = 1 - (-1)$ mit einem Intervall $[-1\ 1]$ ist. Über (2.12) ergeben sich $N = 25$ Versuchspunkte. Diese Teilbereiche sind als Quadrate mit der Länge $l$ und den Versuchspunkten als ‚Center- Ponits' in Abbildung 4.1 b) dargestellt.




a) b)

Abbildung 0.1: Die Testfunktion und das berechnete Raster.

Die Schwierigkeit besteht darin zu entscheiden, in welchen Mengen $\boldsymbol{M}_p$ eine weitere Konkretisierung lohnend ist.

Nach den Methoden des Branch an Bound- Verfahren, wird eine untere Schranke $s_u(\boldsymbol{M}_p) = \min\{f(\vec{x})|\vec{x} \in \boldsymbol{M}_p\}$ für jede Untermenge $\boldsymbol{M}_p$ gebildet. Anschließend wird ein Schwellwert anhand der oberen Schranke $s_o = f(\vec{x})$ bestimmt, wobei $f(\vec{x})$ den besten bisher gefundenen Zielfunktionswert repräsentiert. Zuletzt werden diejenigen Untermengen $\boldsymbol{M}_p$ entfernt, für die $s_u(\boldsymbol{M}_p) \geq s_o$ gilt.

Problematisch ist bei dieser Vorgehensweise, dass für jede Untermenge $\boldsymbol{M}_p \subseteq \mathbb{M}$ eine untere Schranke festgelegt werden muss. Was bedeutet, dass auch für das globale Optimum eine Begrenzung angegeben werden muss. Diese Aufgabe ist jedoch derselben Schwierigkeit, wie die Bestimmung des Minimums selbst [Rudolph, 1990].

Eine Möglichkeit mehr Informationen über die Zielfunktion zu erlangen, ist es die angenommene Flachheit als Ausgangspunkt für eine Approximation zu verwenden, so können die Verläufe zwischen den berechneten Versuchspunkten abgeschätzt werden. Das in dieser Arbeit verwendete ‚Neural Net Fitting' konstruiert bei $N = 25$ Versuchspunkten eine Metamodellfunktion $f(\vec{x})$ der Simulation. Zu sehen ist dieser Funktionsverlauf in Abbildung 4.2 b).

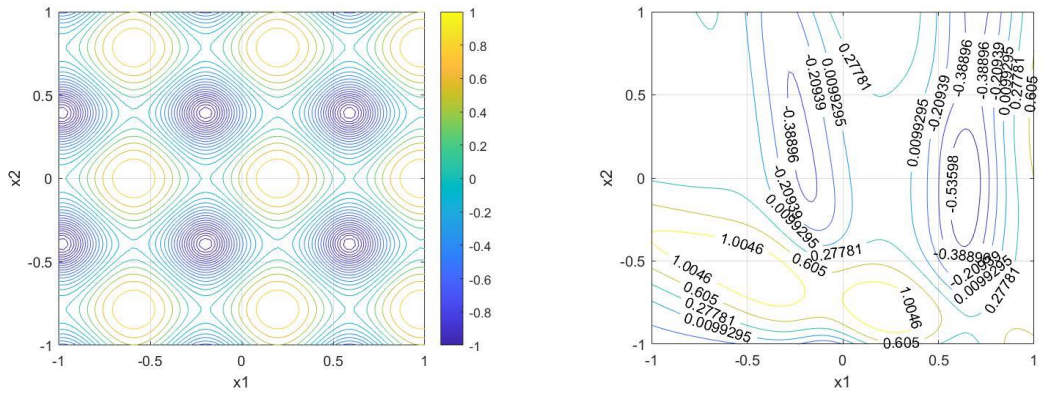



Abbildung 0.2: Konturdiagramm der Testfunktion und der Metamodellfunktion.

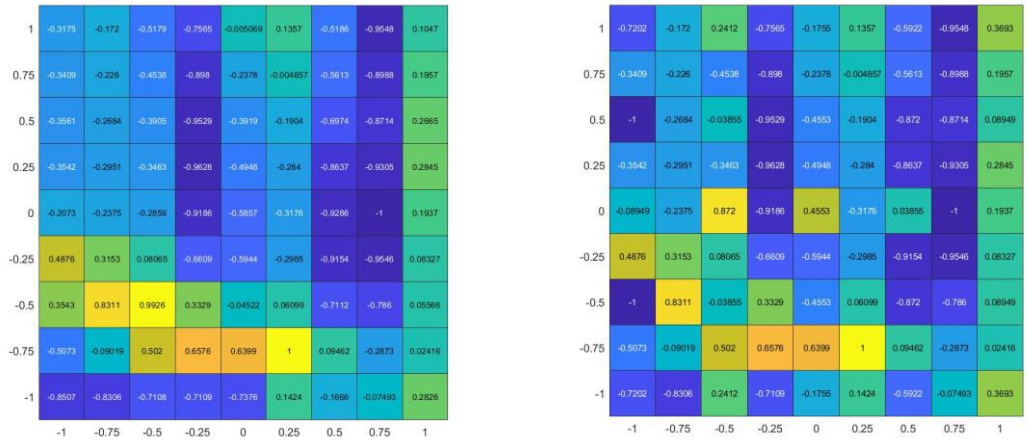
Erkennbar ist, dass die Funktionswerte der Approximation mit einem Infimum von etwa -0,63 und mit Supremum von etwa 1,37 zwar deutlich von den tatsächlichen Werten $(-1{,}1)$ abweichen, dennoch kann die Lage solcher relativen lokalen Minima auf ein tatsächliches lokales Minimum hinweisen. Vergleicht man das Rasterfeld von Abbildung 4.1 b) und die Approximation von Abbildung 4.2 b) mit den Konturen des tatsächlichen Verlaufs in Abbildung 4.2 a), fällt auf, dass die Approximation die Lage lokaler Minima in Höhe von $x_1 \approx -0{,}2$ erkennen konnte. Diese waren in der reinen Rastersuche nicht eindeutig bestimmbar.

Diese ‚neuen' Bereiche werden für die Suche im nächsten Durchlauf berücksichtigt, indem beide Raster zusammengeführt werden. Dafür können die $N$ Punkte der ersten Rastersuch im Approximationsmodell erneut berechnet werden. Normiert man die Werte der approximierten Zielfunktion und die der Versuchspunkte, kann man diese vergleichen.

Durch die geringe Auflösung des Rasters können die Informationen unleserlich werden. Aus diesem Grund wird der Parameterraum in kleinere Teilmengen $M_p$ aufgeteilt, deren Anzahl durch die neue Anzahl der Approximationsversuche bestimmt wird. Das neue Unsicherheitsintervall $l$ sollte halb so groß wie $l$ gewählt werden, damit die Parameter $\vec{x} \in \mathbb{R}^2$ der Systemantworten auf dem Raster der Approximation liegen. Dies hat den Vorteil, dass das Approximationsverfahren mit dem Rasterverfahren zusammengeführt werden kann. Wie in Abbildung 4.6 b) zu sehen ist, wurden alle normierten Zielfunktionswerte in das neue normierte Raster übertragen. Dabei werden die Werte der Approximation der genauen Systemanalysen unter Einhaltung der Bedingung:

$$f(\vec{x}_p) = f(\vec{x}_p) \; wenn \; \vec{x}_p = \vec{x}_p,$$

kompromisslos überschrieben. Diese Punkte werden als vertrauenswürdig bewertet.




a) b)

Abbildung 0.3 Raster mit $\hat{N}$=81 Punkten und $l = L/8$. a): Raster der Approximierten Lösung. b): Raster der Approximierten Lösung mit Werten der Systemantwort.

Anschließend werden einige dieser Mengen $M_p$ als eine Teilmenge eines neuen zulässig geltenden Parameterraums definiert. Die Auswahl dieser Teilmengen erfolgt ähnlich wie beim Brach and Bound Verfahren nach Schwellwerten. Es ist strategisch sinnvoll diesen niedriger anzusetzen, dafür jedoch eine Umgebung, um die ausgewählten Mengen zu definieren. So können durch die Approximation erzeugte, günstig wirkende Übergangsbereiche, zwischen lokalen Minima ausgeschlossen werden.

Der in dieser Arbeit gewählte Schwellwert ist $-0{,}5$, dass geht aus dem Vergleich der Lage und der Werte der Teilmengen hervor. Eine allgemeine Regel kann hier nicht formuliert werden, da der Erfolg des Wertes von den Suchkriterien und der Zielfunktion abhängig ist.

Die Teilmengen in ein feineres Raster aufzuteilen bringt den Vorteil, dass die Grenzen einen homogeneren Verlauf bilden. Was bei der Berechnung neuer Punkte den Vorteil bringt, dass weniger irrelevante Punkte berechnet werden. Die auftretende Problematik besteht darin, dass die Teilmengen der Systemanalysen in einem feineren Raster verhältnismäßig klein werden. Um dieser Tatsache entgegen zu wirken, wurden die Teilbereiche der genauen Punkte so skaliert, dass sie im neuen Raster einen Kreis mit dem Durchmesser $2l$ um ihren Punkt bilden. Wie in Abbildung 4.4 zu sehen ist, können mit dieser Vorgehensweise die Vorteile mehrerer Verfahren kombiniert werden, um einen sehr guten ersten Anhaltspunkt für die Berechnung weiterer Punkte zu gewinnen.

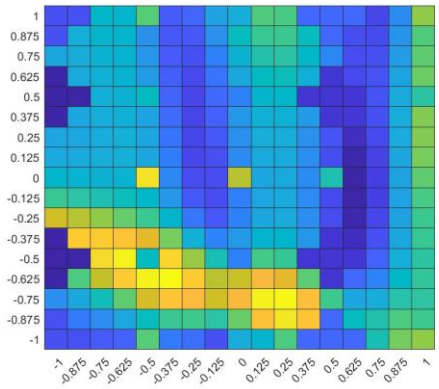



Abbildung 0.4: Verfeinertes Approximationsraster mit eingefügten Punkten und deren Umgebung.

Als neue potenzielle Parameterwerte von $x_1$ und $x_2$ werden die Koordinaten des neuen Rasters ausgewählt, deren Wert $s(M_p) \leq -0{,}5$ ist, und deren Parameter nicht bereits berechnet wurden. Die Anzahl der Experimente $E$ kann frei gewählt werden, wird jedoch durch die Menge an Elementen des neuen und alten Rasters begrenzt.

$$0 < E < \left\{ \left( \frac{L+l}{l} \right)^n - N \right\} \qquad , \quad E \in \mathbb{N} \tag{2.10}$$

Es kann sich als vorteilhaft erweisen, die Anzahl der Experimente gering zu halten, da die Effizienz der Platzierung mit der Anzahl der bereits berechneten Punkte steigt. Es ist zu beachten, dass bei einer sehr kleinen Anzahl von Experimenten die Berechnungszeiten zwischen den Simulationen stärker ins Gewicht fallen, womit der Zeitgewinn bei einem Grenzwert verschwindet.

Existieren mehr zulässige Punkte als die gewünschte Anzahl an Experimenten, wird die Auswahl nach dem Zufallsprinzip entschieden. Das verhindert, dass bei steigender Anzahl an tatsächlich berechneten Punkten unter -0,5, nur noch Bereiche berücksichtigt werden, die bereits ausreichend beleuchtet wurden. Es ist essenziell für den Erfolg, dass alle als erfolgsversprechend eingestuften Bereiche, homogen überprüft werden. Ansonsten würde der erzeugte Datensatz das Metamodell nur in den scheinbar besten Punkten konkretisieren.

### 4.3.1 Netzaufbau

Der Aufbau eines künstlichen neuronalen Netzes hat zahlreiche Konfigurationsmöglichkeiten. Wie gut die Prognosefähigkeit des Netzes ist, wird nicht nur von dem Datensatz, sondern auch von der sogenannten Architektur des Netzes beeinflusst.

Um die Prognosefähigkeit unterschiedlicher Netzwerkarchitekturen auf denselben Datensatz zu vergleichen, wird der Datensatz in Trainingsdaten und Testdaten unterteilt. Mit den Trainingsdaten wird das Netz mit einer Anzahl von Epochen trainiert. Anschließend werden die Werte aus dem Testdatensatz eingesetzt, um neue Ergebnisse zu berechnen. Deren mittlere quadratische Fehler (3.3) ist ein Maß dafür, wie gut ein Netz neue Punkte approximieren kann (Generalisierungsfähigkeit).

Der Minimierungsprozess dieses Fehlers, wird als Ansatz einer weiteren Optimierung verwendet. Die Parameter, die gesucht werden, sind die des Netzaufbaus. Hierbei wird weiterhin von einer Netzstruktur eines Feed Forward Neural Network mit fully connected layern ausgegangen. Die Optimierungsparameter sind:

- Anzahl der verdeckten Schichten (1-3)
- Anzahl der Neuronen je Schicht (10-17)
- Aktivierungsfunktion Layer 1 (Unipolarer Sigmoid, bipolarer Sigmoid, Linear)
- Aktivierungsfunktion Layer 2 (Unipolarer Sigmoid, bipolarer Sigmoid, Linear)
- Aktivierungsfunktion Layer 3 (Unipolarer Sigmoid, bipolarer Sigmoid, Linear)

Die Optimierung nach jedem Konkretisierungsschritt ermöglicht, dass die Prognosefähigkeit auf einem wünschenswerten Niveau bleibt.

Die Optimierung solcher Netzstrukturen stellt ein ebenso schwieriges Unterfangen dar, wie die Optimierung, in der sie eingesetzt wird. Aus diesem Grund, wird in dieser Arbeit nicht weiter darauf eingegangen.

```
vars = [optimizableVariable('act1',{'purelin','tansig','logsig'},'Type','categori-
cal'),
        optimizableVariable('act2',{'purelin','tansig','logsig'},'Type','categori-
cal'),
        optimizableVariable('act3',{'purelin','tansig','logsig'},'Type','categori-
cal'),
        optimizableVariable('hiddenLayernum', [1,3], 'Type', 'integer')
        optimizableVariable('hiddenLayersize', [10,17], 'Type', 'integer')];


minfn = @(T)netbuild(xtrain,xtest,ytrain,ytest,T.act1,T.act2,T.act3,T.hiddenLayer-
num,T.hiddenLayersize);


results = bayesopt(minfn, vars,'IsObjectiveDeterministic', true,...
    'AcquisitionFunctionName', 'expected-improvement-plus','PlotFcn',[],'UsePara-
llel',true);
```

Code 0.1: Bayesian Optimierung der Netzstrukturparameter.

Umgesetzt wird das ‚optimale Netz' wie in Code 4.1 zu sehen ist, indem Aufbau, Training und Validierung des Netzes als Funktion definiert wird. Mit dem MSE des Testdatensatzes als Ausgabe der Funktion, findet der Optimierer die Netzstrukturparameter, welche diesen Fehler minimieren.

### 4.3.2 Optimierungsschleife

Der Vorteil dieses Verfahrens wird deutlich, wenn das Metamodell mit jedem Durchlauf eine bessere Approximation bildet. Das ermöglicht, die neuen Experimente $E$ bei jedem Durchlauf effizient zu platzieren. Die Abbildung 4.5 zeigt die Struktur der Optimierungsschleife. Ist die maximale Verfeinerung des Rasters für das Approximationsmodell erreicht und bringt diese keine neuen Experimente hervor, ist die Abbruchbedingung der Schleife erreicht. Eine beliebige Erweiterung des Rasters ist wegen der exponentiell ansteigenden Anzahl von Experimenten nicht möglich. So erzeugen neun Parameter mit der Unterteilung $l = L/4$ ein Raster mit etwa 1,9 Millionen Punkten. Das Grundraster als Basis zur Erstellung des neuronalen Netzes wird zunächst vollständig berechnet. Dabei werden die Log-Daten abgefragt, um eine vielfache Berechnung derselben Datenpunkte zu verhindern. Die Normierung der Raster/Netz- Punkte erzeugen zwangsläufig eine Menge an Punkten, die als vielversprechend bewertet werden. Wurden diese Punkte bereits berechnet, werden sie aus der Liste entfernt. Dies führt dazu, dass bei gleichbleibender Rasterfeinheit die Menge an neu zu berechnenden Punkten kleiner wird. Werden keine neuen Experimente gefunden, wird das Approximationsraster verfeinert, um neue Punkte zugänglich zu machen.

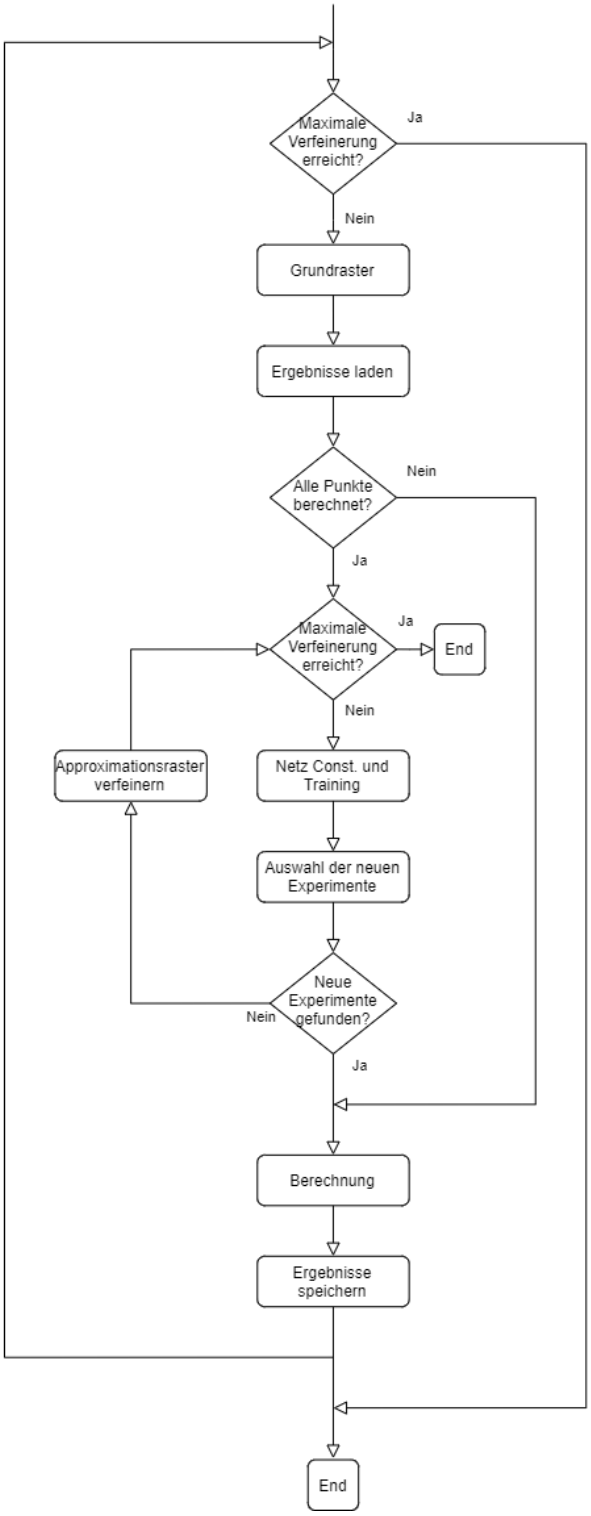



Abbildung 0.5 Konkretisierungsschleife.

Wurden die Stützstellen berechnet, ist es anschließend möglich, dass finale Netz mit einem explorativen Verfahren (Innerer Punkt Verfahren) nach lokalen Minima abzusuchen. Werden Minima gefunden, deren Umgebung nicht konkretisiert wurden, kann die Parameteranalyse durch starke Abweichungen gestört werden.

Daher wurden in dieser Arbeit die Startwerte der explorativen Suche über die besten Punkte des zuletzt trainierten Netzes bestimmt. So werden lokale Minima vermieden, die nach der globalen Betrachtung den Schwellenwert von 0.5 nicht überschreiten.

Es fällt auf, dass dieses Verfahren dennoch einen großen Nachteil hat. Der Optimierungsverlauf ist hauptsächlich von der Feinheit des anfänglich erzeugten Rasters abhängig. So können Ausreißer wie in Abbildung 2.1 dazu führen, dass nur noch in einem schlechten Bereich konkretisiert wird. Ob das Netz trotz ungleichverteilten Datensatz seine Prognosefähigkeit beibehält, ist nicht garantiert. Um die Konvergenz in ein solches lokales Minimum zu überwinden, wird ein Kontingent an gleichverteilten Experimenten für jede Iteration bereitgestellt. Dies bewirkt, dass der Datensatz seine globale Repräsentanz behält.

# Kapitel 5

# Optimierung und Analyse eines Solarreaktors durch neuronale Netze

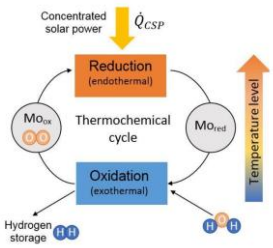
## 5.1 Beschreibung des Simulationsmodells

Zwei der großen Hürden beim Umschwung auf erneuerbare Energien ist die Ressourcenverfügbarkeit und den Einsatz im Verkehr zu verbessern. Wasserstoff wird als einer der Hoffnungsträger für diese Aufgabe gesehen. Es kann durch Brennstoffzellen effektiv in Strom umgewandelt werden, wie auch indirekt als Ausgangsprodukt zur Herstellung von synthetischen Brennstoffen wie Methan, Methanol und Diesel. Diese können in Verbrennungsmotoren und Strömungsmaschinen aller Art eingesetzt werden. Die emissionsfreie Mobilität und der Ausgleich des wetterabhängigen Angebots von erneuerbaren Energien in Europa, bilden einen wertvollen Beitrag zur Umsetzung nachhaltiger Konzepte [Deutsche Zentrum für Luft- und Raumfahrt, 2020].

Ein Weg emissionsfreien Wasserstoff aus Sonnenenergie zu gewinnen, ist der thermochemische Kreisprozess. Hierbei wird die Prozesswärme zur Spaltung von Wasser durch konzentrierte Solarstrahlung zur Verfügung gestellt. Großes Potenzial wird dem zweistufigen Kreisprozess zugeschrieben, welches ein Eisenmischoxid als Redox-System zur Wasserspaltung einsetzt. Dieser wird in zwei Schritten abwechselnd durchgeführt [Säck, 2012].

Im ersten Schritt wird ein Metalloxid in Stickstoff umspülter Umgebung bei einer Temperatur von etwa 1673K reduziert, was einen Teil des gebundenen Sauerstoffes freisetzt. Dieser Sauerstoff wird mit Hilfe des strömenden Stickstoff abgeführt.

Im zweiten Schritt reagiert das reduzierte Metalloxid bei einer Temperatur von etwa 1073 bis 1373K mit Wasserdampf. Der Sauerstoff wird im Metalloxid gebunden (oxidiert) und Wasserstoff wird freigesetzt.




Quelle: Entnommen aus *Menz* et al.(2020: 3)

Abbildung 0.1: Thermochemischer Kreisprozess mit thermisch separierten Reduktion und Oxidation.

Die schematische Darstellung dieses Kreisprozesses in Abbildung 5.1 ist eine Hilfestellung zur Verdeutlichung des Grundprinzips. Es ist zu erahnen, dass die wechselnde molekulare Zusammensetzung des Metalloxids ein Indiz auf die produzierte Wasserstoffmenge ist. Solche isomorphe Moleküle werden als nicht stöchiometrisch Verbindungen bezeichnet. So kann bei einer Reduktion und Oxidation eines Metalloxids, mit der stöchiometrische Abweichung $\delta$, die Reaktion folgendermaßen beschrieben werden:

$$RED: MO_x \rightarrow MO_{x-\delta} + \frac{\delta}{2} O_2$$

$$OX: MO_{x-\delta} + \delta H_2O \rightarrow MO_x + \delta H_2$$

Diese Wasserstoffproduktion ist also maßgeblich von dieser Abweichung abhängig. Nach [Krause, 2020], ist die Reduktion bei höheren Temperaturen und eine Oxidation bei niedrigeren Temperaturen günstiger.

Das Reaktionsmodell dieses Kreisprozesses bildet ein Teil des Gesamtmodells. Zu den drei Untermodellen gehört zusätzlich, dass Heliostatenfeld und das Temperaturmodell, welche miteinander gekoppelt sind. Das Heliostatenfeld simuliert die Flussdichte und die daraus resultierende Solarleistung. Das Temperaturmodell berechnet die Temperaturen in bedeutsamen Punkten. Im Besonderen, in den beschichteten Absorberstrukturen. Mit den Temperaturen als zentrale Parameter, kann das Reaktionsmodell die Wasserstofferzeugungsrate im Reaktionsraum berechnen [Säck, 2012]. In dieser Arbeit wurde das Heliostatenfeld durch einen Temperaturregler mit festem Solarleistungsprofil ersetzt [Anhang A].

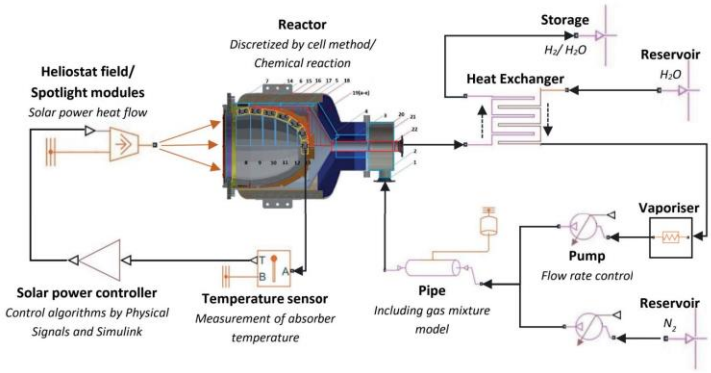



Abbildung 0.2: Vereinfachtes Prinzip des Simulationsmodells.

Wie in Abbildung 5.2 zu sehen ist, wird die Absorbertemperatur für den Reduktions- und Oxidationsprozess über die zur Verfügung stehende Solarleistung geregelt. Die jeweiligen Gase können simultan in den Reaktor eingelassen werden.

Für den Optimierungsprozess ist in erster Linie die Menge an produziertem Wasserstoff entscheidend. Dabei wird zwischen Wasserstoffmenge je Zeit und Wasserstoffmenge je eingesetzter Solarleistung differenziert. Angegeben werden die Werte in Gramm je Stunde und Gramm je Kilowattstunde.

$$H_s \left[\frac{g}{h}\right]$$

$$H_e \left[\frac{g}{kWh}\right]$$

Wie viel Solarleistung verwendet wird, um eine bestimmte Menge an Wasserstoff zu produzieren, gibt Auskunft über die benötigte Heliostatenfläche. In beiden Fällen ist ein hoher Wert wünschenswert. Aus diesem Grund wird das Optimierungsproblem als Maximierungsproblem definiert. Grundsätzlich ist eine Optimierung beider Werte lohnend. In dieser Arbeit wird jedoch nach der Produktionsmenge je Zeit optimiert.

## 5.2 Optimierungsparameter

Als variable Optimierungsparameter können alle unabhängigen Betriebsparameter gewählt werden. Um die Parameter ausfindig zu machen, werden die kontinuierlich veränderbaren Größen:

- Temperatur
- Massefluss von Stickstoff
- Massenfluss von Wasserdampf

näher betrachtet. Diese drei Größen können innerhalb ihrer Grenzen zeitlich beliebig variiert werden. Um dem Kreisprozess gerecht zu werden, wird eine Reduktiontemperatur $T_{red}$ und eine Oxidationstemperatur $T_{ox}$ definiert, welche durch jeweilige Zeitbereiche $t_{red}$ und $t_{ox}$ zeitlich definiert sind. Der thermochemische Kreisprozess definiert im Reduktionsprozess eine Stickstoffumgebung. Aus diesem Grund kann der Massenfluss von Wasserdampf $M_{ox}$, für die Reduktionsphase abgeschaltet werden. Dasselbe gilt für den Massenfluss von Stickstoff $M_{red}$ in der Oxidationsphase. Ein Produktionszyklus beginnt mit der Reduktionsphase, auf welche eine Oxidationsphase folgt.

Bevor die Berechnung eines Zyklus brauchbare Erfolge liefert, müssen die trägen Systeme des Modells berücksichtigt werden. Es wird eine Aufheizzeit von 4000 Sekunden definiert. Innerhalb dieser Dauer kann der Temperaturwert auf die maximale Reduktionstemperatur von 1673K aufheizen. Das gewährleistet, dass die Reduktionstemperatur zu Beginn des ersten Produktionszyklus eingehalten wird. Bei kleinen Werten von $t_{red}$ und oder $t_{ox}$ besteht die Möglichkeit, dass die gewünschte Temperatur der jeweiligen Phase nicht erreicht wird. Die in Abbildung 5.3 zu sehenden Temperaturverläufe des blauen Soll-Wertes und die des roten Ist-Wertes verdeutlichen, dass die gewünschten Temperaturen ($T_{red} = 1673K, T_{ox} = 1223K$) bei $t_{red} = 50s$ und $t_{ox} = 50s$ nicht erreicht werden können. Zusätzlich führt die Differenz der Aufheizrate- und Abkühlrate zu einem Einschwingvorgang. Um transiente Informationen zu relativieren, werden fünf Durchläufe in Folge simuliert und anschließend werden die Ergebnisse der letzten beiden Zyklen gemittelt.

Der Temperaturverlauf innerhalb der Zyklen in Abbildung 5.3 lässt darauf schließen, dass die als optimal bewerteten Parameter nicht die tatsächlichen Systeminhärenten Daten widerspiegeln. Vielmehr gilt die Aussage, dass bei diesen optimalen Parametern die systeminhärenten Vorgänge ein optimales Ergebnis erzeugen.

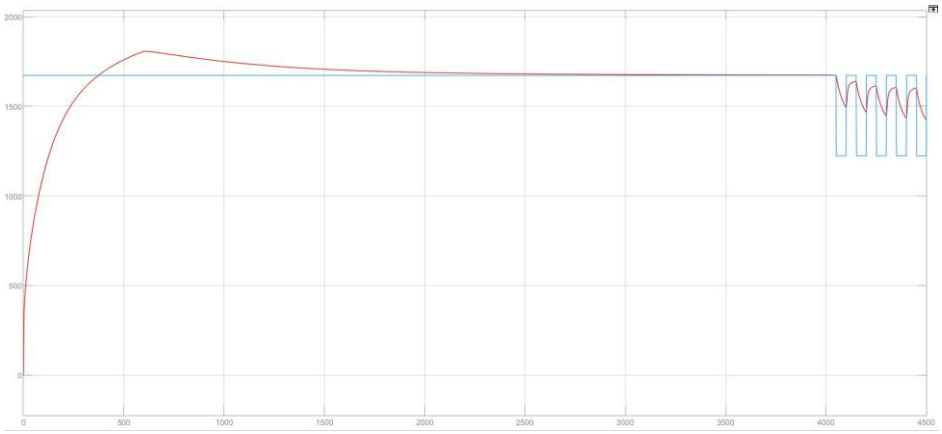



Abbildung 0.3: Temperatur Soll/Ist-Wert mit $t_{red} = 50s$ und $t_{ox} = 50s$.

Die Absorbertemperaturen werden von den Eigenschaften der Gase in den Halbzyklen beeinflusst. So ist die Temperatur der Absorber von der Vorheiztemperatur des Gases, sowie dessen Massenfluss abhängig. Die Gastemperatur wird in dieser Optimierung für beide Gase mit 400K festgelegt, um die Komplexität zu begrenzen. Das optimale Verhältnis dieser Parameter zur Erreichung des formulierten Ziels zu bestimmen, kann auch bei fundierten Kenntnissen über das System, eine große Herausforderung darstellen.

Die Suchraumbegrenzung wird von der möglichen betrieblichen Fahrweise, Erfahrungswerten, mathematischen Vorhersagen sowie Schätzungen beeinflusst. So ist der obere Grenzwert der Reduktionstemperatur durch die thermische Belastbarkeit der Absorber bestimmt, wohingegen der untere Grenzwert aus mathematischen Vorhersagen wie [Krause, 2020] stammen. Dennoch sind Grenzwerte optimierungsspezifisch kritisch zu hinterfragen. So ist beispielsweise für die Oxidationstemperatur ein günstiger Bereich von 800-1100°C bekannt. Wohingegen nicht bekannt ist, wie eine bessere Reduktionphase durch geringere Temperaturdifferenzen im Bezug zu einer schlechteren Oxidationsphase steht. Diese Unsicherheiten, die eine Optimierung motivieren, implizieren, dass der Suchraum so groß wie möglich gestaltet werden sollte, um alle Eventualitäten zu berücksichtigen. Das steht im Widerspruch mit der Knappheit der Rechenressourcen und dessen Forderung, den Suchraum so groß wie nötig zu halten. Eine nachträgliche Erweiterung des Suchraums für spezifische Parameter ist zwar möglich, ist jedoch nur eindeutig, wenn sich eine Konvergenz erahnen lässt.

Um einen Eindruck über das Grenzverhalten zu gewinnen, können extreme Werte für die jeweiligen Parameter simuliert und der Wasserstoffwert analysiert werden. Hierfür wurden die übrigen Parameter festgelegt. Zunächst werden die Grenzwerte der jeweiligen halbzyklen-Zeiten untersucht, wobei die restlichen Parameter wie folgt definiert sind:

- $T_{red} = 1400°C$
- $T_{ox} = 950°C$
- $M_{red} = 100kg/h$
- $M_{ox} = 20kg/h$

Werden die Phasendauern für beide Halbzyklen auf 2000 Sekunden eingestellt, so ergibt sich der in Abbildung 5.4 zu sehende Wasserstoffproduktionsverlauf. Erkennbar wird, dass in jeder Oxidationsphase nach etwa 700 Sekunden kein Wasserstoff mehr produziert wird. Zudem erreicht die Absorbertemperatur den Wert von $T_{ox}$ nach etwa 700 Sekunden nach Beginn der Oxidationsphase. Da stellt sich die Frage, ob die Einhaltung des günstigen Temperaturbereiches für die Oxidation notwendig ist.

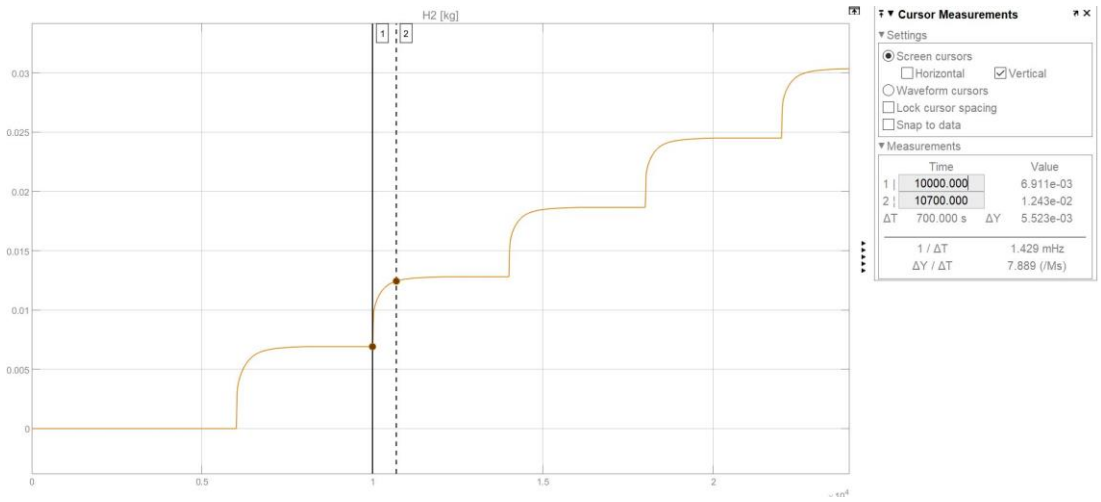
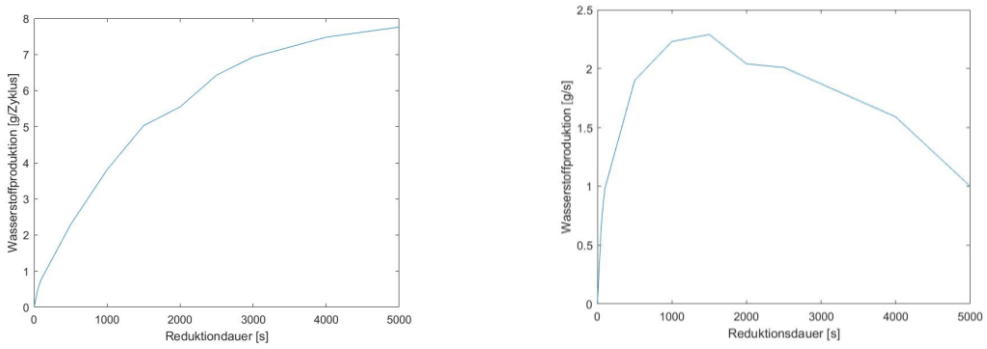



Abbildung 0.4: Wasserstoffmengenverlauf.

Es kann davon ausgegangen werden, dass ein vollständig reduzierter Reaktor in etwa 700 Sekunden oxidiert werden kann. Aus diesem Grund bildet dieser Wert den oberen Grenzwert für die Oxidationsdauer. Es kann aus der Abbildung 5.4 zusätzlich entnommen werden, dass die Reduktion von 6000 Sekunden eine Wasserstoffmenge von $6{,}911g$ in der ersten Oxidationsphase erzeugt.

Um herauszufinden, ab wann eine Erhöhung der Reduktionszeit keinen signifikanten Anstieg der Wasserstoffproduktion verursacht, wurden verschiedene Punkte berechnet und in Abbildung 5.5 a) als Verlaufsgrafik dargestellt. Ersichtlich wird, dass die Wasserstoffmenge je Zyklus mit erhöhter Reduktiondauer steigt. Die in der Optimierung formulierte Zielfunktion fordert jedoch eine maximale Produktionsmenge je Zeit. Aus diesem Grund wurden die Punkte mit ihrer jeweiligen Zykluszeit geteilt, um die Produktionsmenge je Sekunde zu erhalten. Der somit erzeugte Verlauf in Abbildung 5.4 b) verdeutlicht, dass mit den festgelegten Parametern und einer Oxidationsdauer von 700 Sekunden die maximale Produktionsrate bei einer Regenerationsdauer zwischen 500-2000 Sekunden liegt. Aus der Vermutung heraus, dass die zeitliche Effizienz mit immer weitersteigenden Reduktionszeiten sinkt, wird der obere Grenzwert auf 3000 Sekunden festgelegt. Zu beachten ist, dass die maximale Oxidationsdauer bei kleinen Reduktionszeiten stärker ins Gewicht fällt. Das der Wasserstoff hauptsächlich in den ersten Sekunden der Oxidationsphase produziert wird, lässt darauf schließen, dass bei kleineren Oxidationszeiten nicht erheblich weniger Wasserstoff produziert wird. Das sich somit bei kleinen Phasenzeiten einstellende Produktionszeitverhältnis, kann einen effektiven Ansatz darstellen. Aus diesem Grund, wurden die unteren zeitlichen Grenzen der Phasen auf 50 Sekunden gelegt.




a) b)

Abbildung 0.5: Wasserstoffproduktion je Zyklus und Wasserstoffproduktion je Sekunde.

Die folgende Tabelle zeigt die sechs Parameter des ersten Optimierungsdurchlaufs und ihre Grenzwerte. Die Grenzwerte für die Phasendauer wurden auf einen Minimalwert von 50 Sekunden festgelegt.

| **Parameter** | **Symbol** | **Min** | **Max** | **BP** |
|---|---|---|---|---|
| Reduktionstemperatur | $T_{red}$ | 1200°C | 1400°C | 1400°C |
| Oxidationstemperatur | $T_{ox}$ | 800°C | 1400°C | 800-1100°C |
| Reduktionsdauer | $t_{red}$ | 50s | 3000s | - |
| Oxidationsdauer | $t_{prod}$ | 50s | 700s | - |
| Massefluss Reduktion | $M_{red}$ | 10kg/h | 220kg/h | 100kg/h |
| Massefluss Oxidation | $M_{ox}$ | 5kg/h | 50kg/h | 20kg/h |

Tabelle 0.1: Parameter der Optimierung mit Grenzwerten und Betriebspunkten bei sechs Parametern.

## 5.3 Sechs Parameter Optimierung

### 5.3.1 Beurteilung des Optimierungskonzeptes

Um die Wirksamkeit des Optimierungskonzeptes zu untersuchen, wurden die Ergebnisse der sechs Parameteroptimierung auf ihren planmäßigen Verlauf untersucht. Es wird zu Beginn die Hypothese der Generalisierungsfähigkeit bei kleinen Datensätzen untersucht. Um diese Generalisierung veranschaulichen zu können, wurden in Abbildung 5.5 die normalisierten Daten des ersten Rasters mit dem Verlauf des damit erzeugten KNN gegenübergestellt. In Abbildung 5.6 a)

c) ist die Reduktionsdauer der Oxidationsdauer gegenübergestellt. Die restlichen Optimierungsparameter werden für die Betrachtung festgesetzt.

Erkennbar wird bei dem Vergleichen mit dem finalen KNN in Abbildung 5.7, dass vier Datenpunkte nur Informationen über eine sehr grobe Richtung geben können. So sind bei einer Grundrastergröße von $l = L$ auch größere Abweichungen des generellen Verlaufes möglich. In Abbildung 5.6 d) ist zu sehen, dass bei maximaler Reduktionstemperatur ein stärkerer Abfall in Richtung minimaler Oxidationstemperatur prognostiziert wurde. Dies kann bei einem berechnetem MSE des Testdatensatzes von 0.757 durchaus vorkommen.

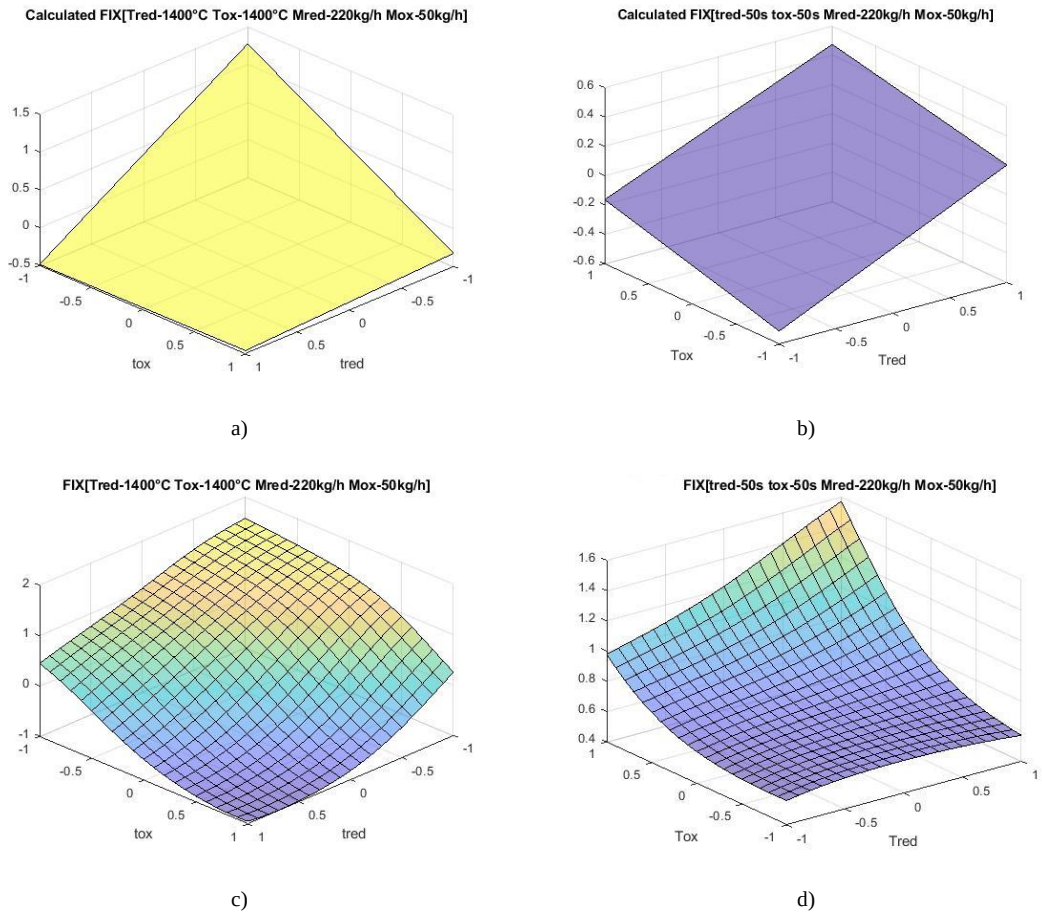



Abbildung 0.6: Normalisierte Punkte des ersten Rasters und der Verlauf der damit erzeugten KNN.

Es wird deutlich, dass die Wahl des Grenzwertes eine erweiterte Rolle bei der Nutzung der Ressourcen spielt. Denn bei einem niedrigeren Grenzwert wird die Wahrscheinlichkeit vielversprechende Bereiche zu umgehen eher vermindert, was bedeutet, dass mehr Experimente möglich sind, ehe die Schleife abgebrochen wird.

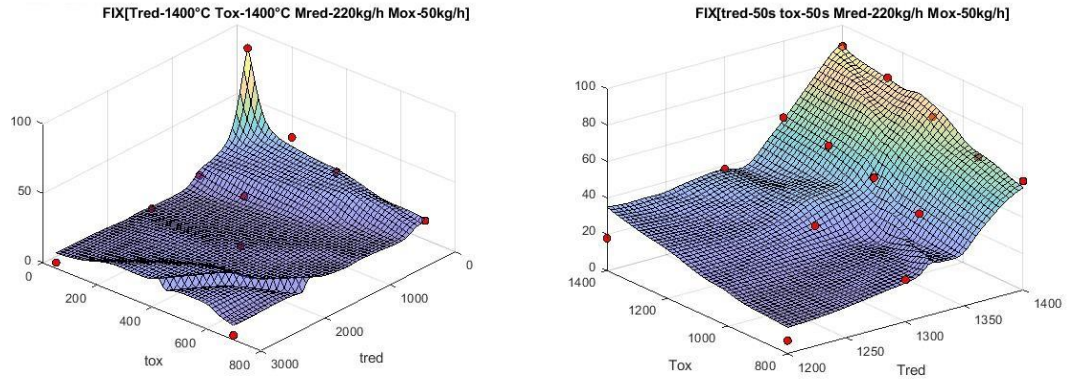
Um zu bestimmen, ob die Kombination der Approximation mit den Umgebungspunkten vielversprechende Datenpunkte in Bereiche mit hohen Zielfunktionswerten platziert, wurde der Mittelwert der neuen Experimente bei jeder Iteration berechnet.

| Iteration | *L/l* | Mittelwert Exp. | Mittelwert Ges. | Anz. Experimente |
|---|---|---|---|---|
| - | 1 | 11.24 | 11.24 | 64 |
| 1 | 2 | 40.23 | 18.41 | 64 |
| 2 | 2 | 9.93 | 14.76 | 40 |
| 3 | 2 | 20.08 | 15.58 | 27 |
| 4 | 2 | 44.80 | 16.23 | 4 |
| 5 | 2 | 17.78 | 16.29 | 7 |
| 6 | 2 | 12.05 | 15.95 | 16 |
| 7 | 4 | 26.59 | 18.50 | 64 |
| 8 | 4 | 22.42 | 19.26 | 64 |
| 9 | 4 | 25.49 | 20.27 | 64 |
| 10 | 4 | 23.06 | 20.66 | 64 |
| 11 | 4 | 20.24 | 20.61 | 64 |
| 12 | 4 | 25.76 | 21.17 | 64 |
| 13 | 4 | 21.12 | 21.16 | 64 |
| 14 | 4 | 22.04 | 21.24 | 64 |
| 15 | 4 | 24.62 | 21.52 | 64 |
| 16 | 4 | 23.33 | 21.66 | 64 |
| 17 | 4 | 19.20 | 21.48 | 64 |
| 18 | 4 | 20.29 | 21.41 | 64 |
| 19 | 4 | 14.59 | 21.27 | 19 |
| 20 | 4 | 15.15 | 21.27 | 1 |
| 21 | 8 | 39.41 | 22.37 | 64 |

Tabelle 0.2: Mittelwerte des gesamten Datensatzes und der Experimente in [g/h] (sechs Parameter Optimierung).

So sieht man in Tabelle 5.2, dass die ersten 64 Punkte des Grundrasters einen Mittelwert von etwa $11{,}25 g/h$ erzeugen. Es ist zu erkennen, dass der Mittelwert der Experimente im Iterationsschritt nach einer Verfeinerung des Rasters besonders hoch ausfallen. Das ist daher zu erklären, dass die neuen verfügbaren Punkte noch nicht berechnet wurden, und die Umgebungen der höheren Punkte, die der niedrigeren überschreiben. Werden anschließend die 64 Experimente berechnet, sind diese in den folgenden Iteration nicht mehr verfügbar und die Umgebungen der niedrigeren Werte über 0.5 werden als Experiment gewählt. Betrachtet man die ersten vier Iterationen, sind Fluktuationen der Experimentmittelwerte zu erkennen. Diese können auftreten, wenn neue Datenpunkte das KNN so verändern, dass neue Bereiche mit hohen Zielfunktionswerten besser approximiert werden. Um die Stabilität des Datensatzes zu verbessern, sind 10 Prozent aller neuen Punkte zufällig ausgewählte Punkte aller Rasterpunkte, die nicht als vielversprechend bewertet wurden.

Der steigende Gesamtmittelwert zeigt, dass die ausgewählten Punkte in Bereiche gesetzt wurden, die einen höheren Zielfunktionswert erzeugen. Wie die Punkte nach Beendigung der Optimierungsschleife in den Temperatur- und Phasenzeitdiagrammen platziert wurden, veranschaulicht Abbildung 5.7, bei dieser die roten Punkte (Experimente im Betriebspunkt) eine höhere Dichteverteilung in zweckmäßigeren Gebieten aufweisen.




a) b)

Abbildung 0.7: Temperatur- und Phasenzeitverlauf in einem festen Betriebspunkt.

Beim Vergleich mit den Abbildungen der Approximation in 5.6 wird der Vorteil der stufenweise Verfeinerung deutlich. So wird durch die Begrenzung der verfügbaren Punkte verhindert, dass zu viele Punkte in den anfänglich als gut bewerteten Bereichen berechnet werden.

Die 1055 Datenpunkte erzeugen ein viel differenzierteres neuronales Netz, welches ein MSE im normierten Netz von 0,078 und im nicht normierten Netz von 2,93$[\frac{g}{h}]$ aufweist. Der Vorteil dieser Vorgehensweise ist, dass die Fehler stärker in Bereichen vorkommen, die als nicht relevant bewertet worden sind, was bedeutet, dass einer Parameteranalyse mit diesem Netz eine gewisse Vertrauenswürdigkeit mitbringt.

### 5.3.2 Parameteranalyse

Die finale explorative Optimierung des Netzes wird genutzt, um einen genauen Betriebspunkt im erwarteten Bereich zu bestimmen und um Überraschungen im Funktionsverlauf auszuschließen. Das mit den normierten Daten trainierte KNN wurde als Optimierungsfunktion für eine Explorative Optimierung genutzt. Dabei wurde, wie bei der Platzierung der neuen Experimente die Wasserstoffmenge je Zeit als *Target* gewählt. Die Experimente deren normierte Produktionsrate über 0,5 waren, wurden als Startpunkte genutzt, um den zulässigen Bereich abzusuchen.

Die Betriebspunkte in Tabelle 5.3 sind die gefundenen lokalen Maxima, mit ihrem Zielfunktionswerte $F$, welcher ein Maß für die Wasserstoffmenge je Stunde $H_{2t}$ ist. Zusätzlich sind die zugehörigen Wert für die Wasserstoffmenge je Zyklus $H_{2c}$ und die benötigte Energie je Wasserstoff $E_{H2}$ aufgetragen.

| $T_{red}[C°]$ | $T_{ox}[C°]$ | $t_{red}[s]$ | $t_{ox}[s]$ | $M_{red}\left[\frac{kg}{h}\right]$ | $M_{ox}\left[\frac{kg}{h}\right]$ | $F$ | $H_{2c}\left[\frac{g}{Z}\right]$ | $H_{2t}\left[\frac{g}{h}\right]$ | $E_{H2}\left[\frac{kWh}{g}\right]$ |
|---|---|---|---|---|---|---|---|---|---|
| 1400 | 1400 | 50 | 50 | 188.50 | 26.82 | 3.12 | 2,46 | 88,56 | 2,08 |
| 1400 | 1400 | 50 | 50 | 68.80 | 26.15 | 3.35 | 2,65 | 95,49 | 1,69 |

Tabelle 0.3: Im Metamodell explorativ bestimmte lokale Minima.

Um einen visuellen Überblick über den Einfluss der Parameter zu erlangen, wurden 4000 Experimente raumfüllend mit den Ergebnissen $F$ des KNN in Abbildung 5.8 gegenübergestellt.

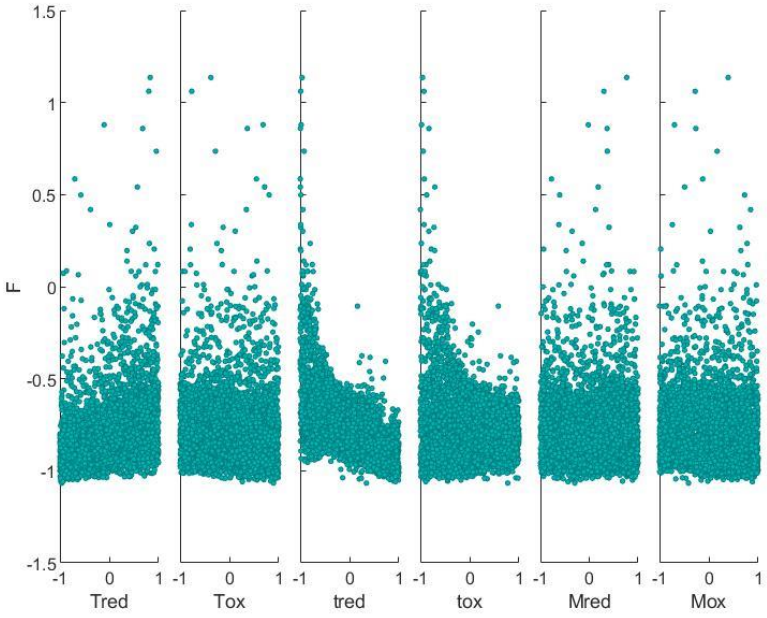



Abbildung 0.8: Wasserstoffmengenverlauf.

Es wird deutlich, dass die Zielfunktion nur große Werte erzeugt, wenn die Werte für $t_{red}$ und $t_{ox}$ in der Nähe ihrer unteren Grenzen gesetzt werden. Wie im Kapitel Optimierungsparameter erwartet wurde, wird durch die ungleichverteilte Produktionsmenge in der Oxidationsphase, so wie die ungleichverteilte Reduktion in der Reduktionsphase, der Wert für die Wasserstoffmenge je Zeit, mit kleinen Parameterwerten in die Höhe getrieben. Das längere Phasenzeiten durch günstigere Temperaturbereiche einen positiven Effekt haben, wird durch den Umstand gemildert, dass bei Beginn des Halbzyklus direkt mit der Reduktion bzw. Oxidation begonnen wird. Sind die günstigen Temperaturbereiche erreicht, so sind die freien Stellen der Moleküle soweit gefüllt, dass der Produktionsgewinn im Zeitverhältnis einen ungünstigen Verlauf nimmt.

Die lineare Korrelationen der Parameter zum Zielfunktionswert in Tabelle 5.3 geben einen Eindruck über den linearen Zusammenhang. So ist zu sehen, dass der positive Korrelationswert von

$T_{red}$ bei steigendem Parameter einen steigenden Zielfunktionswert bedeutet. Interpretierbar ist dieser Umstand dadurch, dass eine höhere Temperatur einen besser ablaufenden Reduktionszyklus bedeutet.

| | F |
|---|---|
| Tred | 0.31 |
| Tox | -0.02 |
| tred | -0.40 |
| tox | -0.35 |
| Mred | -0.01 |
| Mox | 0.01 |

Tabelle 0.4: Lineare Korrelation der Parameter mit dem Zielfunktionswert.

Auffällig ist, dass die Korrelation von $T_{ox}$ mit fast 0 keinen signifikanten Einfluss auf die Zielgröße bedeutet. Möchte man meinen, dass die bessere Oxidation bei niedrigeren Temperaturen Einfluss auf den Zielfunktionswert aufweist. Wie bereits erläutert wurde, wird dies einerseits dadurch begründet, dass der Einfluss des niedrigeren Temperaturniveaus durch teilweise Oxidation bei $T_{red}$ verringert wird und andererseits durch spezifische Absorbertemperaturen kein definierter Temperaturzustand erreicht werden kann.

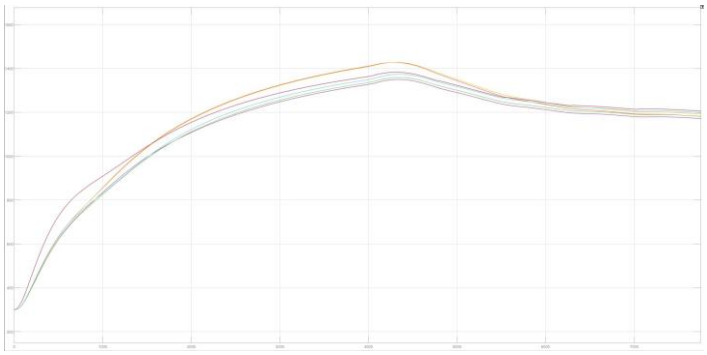


Abbildung 0.9: Temperaturverlauf der Absorberrückseiten.

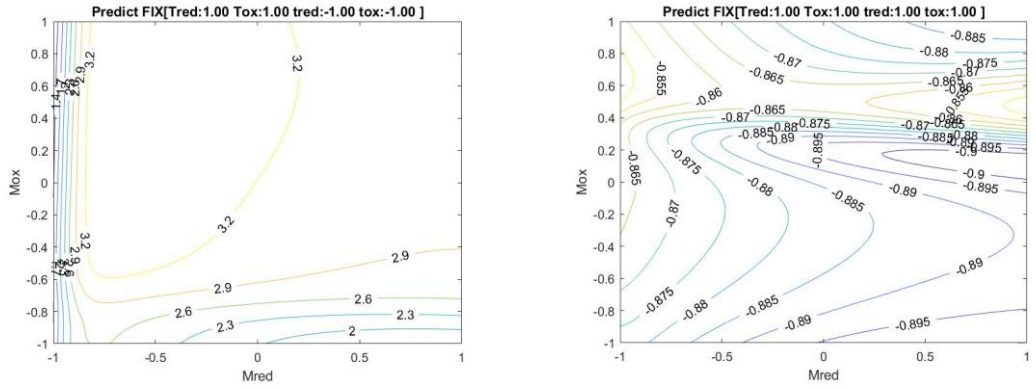
Die in Abbildung 5.9 zu sehenden Temperaturen der Absorberrückseiten zeigt, dass der Einfluss von $T_{ox}$ durch die lange Aufheizzeit mit $T_{red}$ gemildert werden. Durch die lange Aufheizzeit von 4000 Sekunden werden die Temperaturen der Absorberrückseiten maßgeblich von der Reduktionstemperatur beeinflusst. Auch wenn sich die einstellende durchschnittliche Temperatur niedriger liegt, reichen fünf Zyklen nicht aus, um den Einfluss der Oxidationstemperatur ausreichend einzubeziehen.

Die niedrige lineare Korrelation der Massenflüsse $M_{red}$ und $M_{ox}$ zum Zielfunktionswert deuten an, dass es eine geringe Rolle spielt wie diese Parameter gewählt werden. So ergibt sich aus den

Experimenten, dass sie mit $0{,}01^2 = 0{,}01\%$ der Varianz des Zielfunktionswertes ausmachen. Die niedrigen linearen Korrelationen bedeuten allerdings nicht, dass kein nichtlinearer Zusammenhang zwischen den Parametern und des Zielfunktionswert besteht. Um den Einfluss der Masseflüsse genauer zu untersuchen, wird im Betriebspunkt:

- $T_{red} = 1400°C$
- $T_{ox} = 1400°C$
- $t_{red} = 50s$
- $t_{ox} = 50s$

der Verlauf des Zielfunktionswertes untersucht. Wie in Abbildung 5.8 a) zu sehen ist, haben die Masseflüsse in diesem Betriebspunkt, ab einen Wert von $22{,}6 kg/h$ für Stickstoff und $14 kg/h$ für Wasserdampf, keinen nennenswerten Einfluss auf den Zielfunktionswert.




a) b)

Abbildung 0.10: Masseflüsse im Betriebspunkt [1 1 -1 -1] und [1 1 1 1] bei einem Test MSE von 0,078.

Betrachtet man den Zielfunktionsverlauf der Abbildung 5.10 b) dessen Betriebspunkt dahingehend verändert würde, dass die Phasenzeiten in ihren maximalen Grenzwerten liegen, wird deutlich, dass der Einfluss bei langen Phasenzeiten im gesamten Parameterraum sehr gering ist. Der negative Einfluss bei einem minimalen Grenzwerten ist nicht zu sehen, denn solche steilen Funktionsverläufe sind in unzureichend approximierten Bereichen nicht berücksichtigt.

## 5.4 Sieben Parameter Optimierung

In der Abbildung 5.3 ist zu sehen, dass die Soll-Temperaturen der jeweiligen Halbzyklen nicht mit den Ist-Temperaturen übereinstimmen. Das bewirkt, dass die Informationen über die günstigen Temperaturbereiche keinen Optimierungsvorteil erzeugen, denn die Phasen laufen in undefinierten Temperaturen ab. Zusätzlich fällt auf, dass die Halbzyklen bündig mit den jeweiligen

Aufheiz- und Abkühlvorgängen beendet werden. Das hat die Folge, dass die durchschnittlichen Phasentemperaturen stark von der gewünschten Temperatur abweichen. Ein Verbesserungsansatz ist, dass die Reduktionsphase um eine gewisse Aufheizzeit $t_h$ so verschoben wird, dass sie eine günstigere Lage hat. Zusätzlich wird nach der Reduktionphase eine Abkühlphase $t_k$ definiert, um eine beschleunigte Abkühlung durch einen Massenfluss $M_k$ von Stickstoff zu erreichen. So ergibt sich die Zykluszeit aus der Summe der einzelnen Zeiten. Die Reduktions- und Oxidationstemperatur werden jeweils für die Hälfte der Gesamtzykluszeit eingeschaltet. Auf einen zusätzlichen Parameter als Schaltzeitverhältnis wird verzichtet, da die einzelnen Zeiten so verändert werden, dass sie zu einem 1:1 Verhältnis optimal sind.

Damit der Erkenntnisgewinn in einem guten Verhältnis zum Berechnungsaufwand steht, wurden die Reduktionsmasseflusses und des Oxidationsmasseflusses als Optimierungsparameter entfernt. Diese wurden als Mittelwert der 22 besten Experimente festgelegt, welche bei Stickstoff zwischen 66kg/h bis 220kg/h und bei Wasser 24kg/h bis 50kg/h liegen.

- $M_{red} = 125 kg/h$
- $M_{ox} = 37 kg/h$

Die zusätzlichen Parameter ermöglichen, dass im Optimierungsprozess die Phasentemperaturen eingehalten werden können. Die Abbildung 5.11 zeigt, dass es somit möglich wird, dass erst oxidiert wird, wenn die Oxidationstemperatur erreicht wurde.

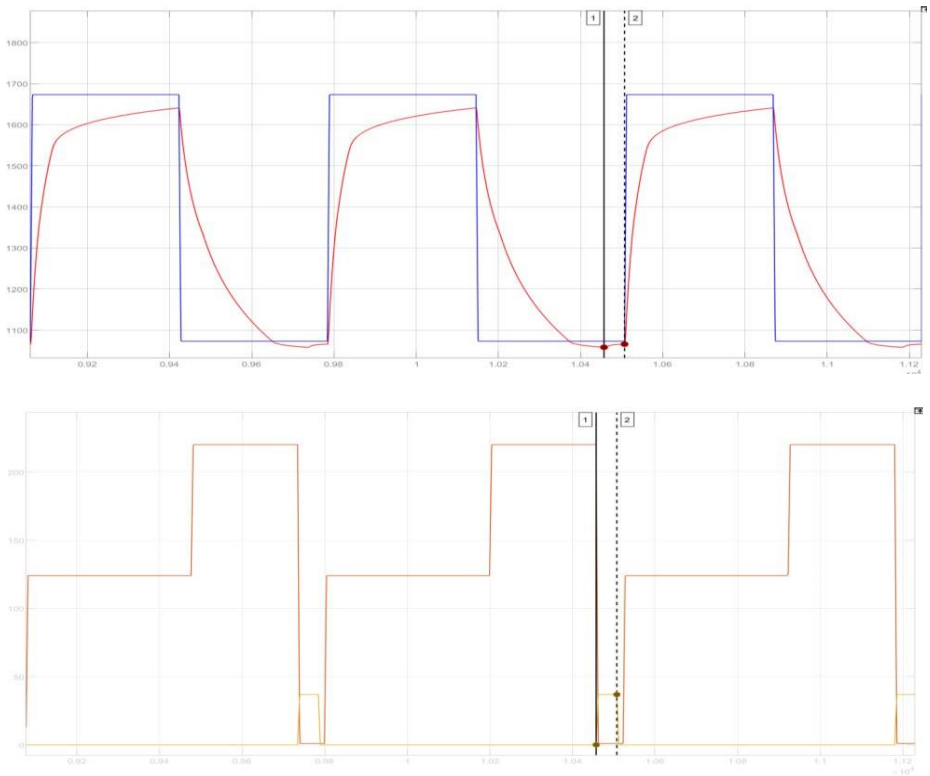


Abbildung 0.11: Temperaturverlauf der Absorberfrontseite und der Masseflussverlauf.

Die Einhaltung der Phasentemperatur ist jedoch kein Zwang, denn die unteren Grenzwerte der Aufheiz- und Abkühlzeiten sind nicht der Aufheiz- oder Abkühlrate angepasst. So besteht die Möglichkeit, definierte Zyklen mit einzubeziehen, ohne die Optimierung zu stark zu restriktiveren.

Zusätzlich wurde die Anzahl der berechneten Zyklen auf 10 erhöht und die Ergebnisse der letzten beiden Zyklen gemittelt. So wird versucht, dem eingeschwungenen Zustand näher zu kommen, um die Übertragbarkeit auf das reale System zu verbessern.

Es wurden die Parametergrenzen bei dieser Modellierung zunächst nahe des vermuteten optimalen Bereiches gesetzt. Der somit erzeugte Datensatz, soll die Genauigkeit des Netzes und der Parameteranalyse verbessern. Der Vermutung nach, dass das Optimum bei möglichst kleinen Zeitparametern liegt, wurden die oberen Grenzen der Aufheiz- und Abkühlzeiten auf 100 Sekunden festgelegt.

| **Parameter** | **Symbol** | **Min** | **Max** | **BP** |
|---|---|---|---|---|
| Reduktionstemperatur | $T_{red}$ | 1200°C | 1400°C | 1400°C |
| Oxidationstemperatur | $T_{ox}$ | 800°C | 1400°C | 800-1100°C |
| Reduktionsdauer | $t_{red}$ | 50s | 3000s | - |
| Oxidationsdauer | $t_{prod}$ | 50s | 700s | - |
| Aufheizdauer | $t_h$ | 15s | 100s | - |
| Abkühldauer | $t_k$ | 15s | 100s | - |
| Massefluss Abkühldauer | $M_k$ | 20kg/h | 100kg/h | - |

Tabelle 0.5: Parameter der Optimierung mit Grenzwerten und Betriebspunkten bei sieben Parametern (1.Durchlauf).

### 5.4.1 Parameteranalyse

Wird die Konkretisierungsschleife nach einer Feinheit von $l = L/4$ abgebrochen, so werden 1424 Experimente berechnet. Diese Experimente erzeugen ein KNN mit einem Test-MSE von 0,016. Dieser Wert und die Höhe des durchschnittlichen nicht normierten Zielfunktionswertes im Iterationsprotokoll in [Anhang B] lassen darauf schließen, dass die Parameteranalyse und eine explorative Suche mit dem Netz zielführend sind.

Wieder wurden die als vielversprechend geltenden Bereiche als Startpunkte für die Optimierung ausgewählt. In Tabelle 5.6 sind die Ergebnisse der explorativen Suche aufgelistet. Es wird deut-

lich, dass diese kein besseres Ergebnis als die Optimierung mit sechs Parametern erzeugen konnten. Sind dieselben Ursachen der sechs Parameter Optimierung für eine solche Konvergenz verantwortlich, so liegt die Vermutung nahe, dass bei minimalen Grenzwerten von 0 für $t_h$ und $t_k$, dass Ergebnis der ersten Optimierung angenähert wird.

| $T_{red}[C°]$ | $T_{ox}[C°]$ | $t_{red}[s]$ | $t_{ox}[s]$ | $t_h[s]$ | $t_k[s]$ | $M_k[s]$ | $F$ | $H_{2c}\ \left[\frac{g}{Z}\right]$ | $H_{2t}\ \left[\frac{g}{h}\right]$ | $E_{H2}\ \left[\frac{kWh}{g}\right]$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1400 | 1400 | 50 | 50 | 15 | 15 | 22 | 3.48 | 2.98 | 82.52 | 2.22 |

Tabelle 0.6: Im Metamodell explorativ bestimmte lokale Minima bei sieben Parametern.

Wie sich der Betriebspunkt von $F = 3{,}48$ bezüglich seiner Parameter verhält, zeigen die Verläufe in [Anhang C]. Die Verläufe der Zeitparameter $t_{red}$ und $t_{ox}$, sowie die Phasentemperaturen $T_{red}$ und $T_{ox}$ zeigen, dass die Systemeigenschaften eine Konvergenz in das Optimum der sechs Parameter Optimierung verursachen. Vergleicht man die Verläufe der angenäherten Zielfunktion, ist in diesem Betriebspunkt zu sehen, dass sie sehr stark den der Abbildung 5.7 ähneln. Ein vermutetes neues Produktionsoptimum, durch die Verschiebung der Phasen in günstigere durchschnittliche Reaktionstemperaturen, ist noch nicht eingetreten.

Für einen Überblick der Parameterwirkung werden wieder approximierte Zielfunktionswerte $F$ für raumfüllende Versuchspunkte berechnet. Die folgende bildliche Darstellung dieser Berechnungen zeigt die Tendenzen im sieben Parameterfall.

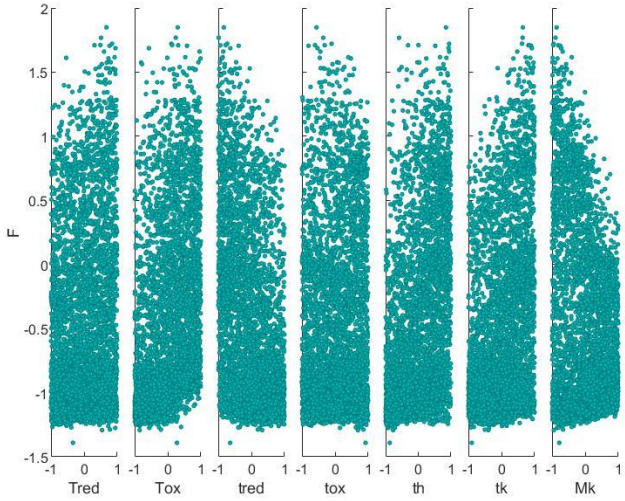



Abbildung 0.11: Ausgabe des normierten KNN der sieben Parameter Optimierung mit reduzierten Grenzen.

Es sind bei einigen Parametern widersprüchliche Tendenzen zum gefundenen Optimum zu erkennen. So führt, wie auch anhand der linearen Korrelationen in Tabelle 5.7 zu erkennen ist, ein Anstieg der Parameter $t_h$ und $t_k$, zu einem Anstieg des Zielfunktionswertes. Beachtet man jedoch den Wert für $F$, wird deutlich, dass die somit erzeugten Werte bei etwa 1,8 liegen. Dieser ist etwa halb so groß wie der explorativ gefundene Zielfunktionswert. Bildlich gesprochen bedeutet das, dass die Umgebung eines Zustandspunktes eine gewisse Flachheit ihrer Steigung bezüglich der Parameter aufweist, so dass viele Startpunkte in einen Funktionswert von $F \approx 1{,}8$ konvergieren.

| | F |
|---|---|
| Tred | 0.12 |
| Tox | 0.40 |
| tred | -0.40 |
| tox | 0.13 |
| th | 0.42 |
| tk | 0.36 |
| Mk | -0.30 |

Tabelle 0.7: Lineare Korrelation der Parameter mit dem Zielfunktionswert bei sieben Parametern mit reduzierten Grenzen.

Um die abweichenden Korrelationen zu untersuchen, wurden lokale Maxima der nicht genau untersuchten Bereiche gesucht. Diese konvergierten aus Startpunkten zufällig verteilter Zustandspunkten. Die letzte Reihe der Tabelle 5.8 repräsentiert einen Betriebspunkt, der die lineare Tendenzen widerspiegelt. Beim Vergleich der tatsächlichen Produktionsrate der Betriebspunkte, mit dem des approximierten Zielfunktionswertes wird deutlich, dass sich die schütter platzierten Versuchspunkte negativ bemerkbar machen.

| $T_{red}[C°]$ | $T_{ox}[C°]$ | $t_{red}[s]$ | $t_{ox}[s]$ | $t_h[s]$ | $t_k[s]$ | $M_k[s]$ | $F$ | $H_{2c}\ \left[\frac{g}{Z}\right]$ | $H_{2t}\ \left[\frac{g}{h}\right]$ | $E_{H2}\ \left[\frac{kWh}{g}\right]$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1400 | 1400 | 50 | 50 | 100 | 20.25 | 51.20 | 1.07 | 3.06 | 50.12 | 6.63 |
| 1400 | 1400 | 50 | 173.50 | 23.50 | 82.15 | 20 | 2.38 | 3.49 | 38.17 | 4.59 |
| 1400 | 1394 | 50 | 176.75 | 100 | 100 | 20 | 1.98 | 3.45 | 29.12 | 5.74 |

Tabelle 0.8: Im Metamodell explorativ bestimmte lokale Minima bei sieben Parametern.

Die im Anhang befindliche Abbildung [Anhang C] veranschaulicht das Verhalten der Parameter in diesem Betriebspunkt. Die flacheren Funktionsverläufe deuten auf einen stabileren Betriebspunkt hin, denn eine große Änderung eines Parameter erzeugt nur eine kleine Veränderung des Funktionswertes. Diese Wertestabilität kann nicht auf den Energieeinsatz je Wasserstoff übertragen werden. Die Validierung solcher Betriebspunkte bezüglich ihrer Wirtschaftlichkeit und Betriebstauglichkeit muss separat untersucht werden.

Die bisherige Parameterbegrenzung hatte den Vorteil, dass der vielversprechende Bereich bei kleinen Zeitparametern gut approximiert wurde. Es ist jedoch anzumerken, dass die Vorüberlegung die Phasentemperaturen einzuhalten, bei Aufheiz- und Abkühlzeiten von 100 Sekunden nicht möglich ist. Aus diesem Grund werden final die oberen Grenzwerte für $t_h$, $t_k$ und $M_k$ ausgeweitet.

Die somit entstehenden neuen Grenzen sind in Tabelle 5.9 zu sehen.

| **Parameter** | **Symbol** | **Min** | **Max** | **BP** |
|---|---|---|---|---|
| Reduktionstemperatur | $T_{red}$ | 1200°C | 1400°C | 1400°C |
| Oxidationstemperatur | $T_{ox}$ | 800°C | 1400°C | 800-1100°C |
| Reduktionsdauer | $t_{red}$ | 50s | 3000s | - |
| Oxidationsdauer | $t_{prod}$ | 50s | 700s | - |
| Aufheizdauer | $t_h$ | 15s | 1000s | - |
| Abkühldauer | $t_k$ | 15s | 1000s | - |
| Massefluss Abkühldauer | $M_k$ | 20kg/h | 220kg/h | - |

Tabelle 0.9: Parameter der Optimierung mit Grenzwerten und Betriebspunkten bei sieben Parametern.

Beim erneuten Start der Konkretisierungsschleife wird der neue zulässige Parameterraum erneut durch das Grundraster aufgeteilt. Die entstehenden Zustandspunkte weichen mehr oder weniger stark von den Punkten des ersten Grundrasters ab. So kann man dem Iterationsprotokoll des zweiten Durchlaufes in [Anhang B] entnehmen, dass 16 Punkte des zweiten Grundrasters bereits berechnet worden sind. Zusätzlich wird deutlich, dass die Mittelwerte der Experimente deutlich geringer ausfallen. Hervorgerufen wird das durch die grobe Aufteilung eines großen Parameterraumes, so fallen die günstigen Bereiche im Vergleich zum gesamten Parameterraum kleiner aus, wodurch das Raster diese nicht mehr erfassen kann.

Aus zeitlichen Gründen wurde die Konkretisierungsschleife nach 527 Experimenten abgebrochen. Die finalen 2001 Experimente erzeugen ein KNN mit einem Test-MSE von 0.034. Wie sich der etwa doppelt so große Fehler im Vergleich zum ersten Durchlauf bemerkbar macht, kann anhand der Abbildung 5.13 veranschaulicht werden.

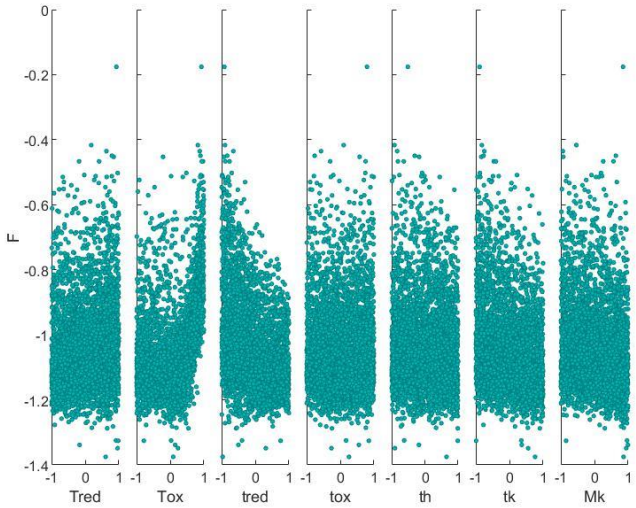



Abbildung 0.12: Ausgabe des normierten KNN der sieben Parameter Optimierung mit erweiterten Grenzen.

So ist im Vergleich mit Abbildung 5.12 zu sehen, dass der Minimalwert des Datensatzes von -1,122 öfter überschritten wird. Diese Fehlerstreuung kommt durch den ungleichverteilten Datensatz zwar nicht gleichmäßig vor, beeinträchtigt er jedoch die explorative Suche nach Betriebspunkten.

Der Abbildung ist zusätzlich zu entnehmen, dass die flächenfüllenden Experimente niedrigere approximierte Zielfunktionswerte erzeugen. Das eine große Menge an Experimenten kein zufriedenstellendes Ergebnis erzeugt, weist auf eine ungünstige Wahl der Grenzwerte hin.

So können die Grenzwerte einen Parameterraum ergeben, der positive Bereiche einschließt, jedoch dessen Gesamtheit im Verhältnis zum Ausmaß der positiven Bereichen zu groß ausfällt. Es kann bei der Wahl der Grenzwerte aber auch ein Parameterraum erzeugt werden, der zwar größentechnisch klein ist, jedoch keine positiven Bereiche umschließt.

Der ausgeweitete Parameterraum kann trotz durchschnittlich niedriger Funktionswerte dennoch wichtige Informationen erzeugen. So ist die gefundene Lösung nach dem höchsten Wert des im gesamtmöglichen Suchbereiches zu bewerten. Aus diesem Grund wurde das Netz explorativ nach Optima untersucht, um eine Gesamtbetrachtung vorzunehmen.

| $T_{red}[C°]$ | $T_{ox}[C°]$ | $t_{red}[s]$ | $t_{ox}[s]$ | $t_h[s]$ | $t_k[s]$ | $M_k[s]$ | $F$ |
|---|---|---|---|---|---|---|---|
| 1234 | 1400 | 2454.25 | 609 | 15 | 15 | 20 | -0.79 |
| 1395 | 932 | 64.75 | 378.25 | 192.30 | 29.77 | 33 | -0.22 |
| 1395 | 932 | 64.75 | 381.50 | 197.22 | 29.77 | 33 | -0.22 |
| 1395 | 935 | 64.75 | 397.75 | 212 | 29.77 | 64.00 | -0.26 |
| 1396 | 950 | 64.750 | 368.50 | 177.52 | 29.77 | 29 | -0.20 |
| 1396 | 956 | 64.75 | 378.25 | 172.60 | 24.85 | 30 | -0.20 |
| 1396 | 971 | 64.75 | 381.50 | 212 | 24.850 | 44 | -0.22 |
| 1397 | 959 | 64.75 | 362 | 167.67 | 24.85 | 31 | -0.19 |
| 1397 | 992 | 64.75 | 375 | 207.07 | 24.85 | 29 | -0.19 |
| 1397 | 995 | 64.75 | 378.25 | 192.30 | 24.85 | 29 | -0.18 |
| 1397 | 1016 | 64.75 | 368.50 | 192.30 | 24.85 | 25 | -0.16 |
| 1398 | 980 | 50 | 329.50 | 133.20 | 19.92 | 30 | -0.14 |
| 1398 | 1013 | 50 | 362 | 123.35 | 19.92 | 26 | -0.13 |
| 1400 | 800 | 50 | 115 | 15 | 15 | 220 | 2.66 |
| 1400 | 800 | 551.50 | 700 | 650.32 | 812.85 | 20 | -0.51 |
| 1400 | 1301 | 50 | 124.75 | 15 | 15 | 20 | 3.54 |
| 1400 | 1400 | 50 | 50 | 15 | 15 | 20 | 3.86 |

Tabelle 0.10: Im Metamodell explorativ bestimmte lokale Minima bei sieben Parametern mit erweiterten Grenzen.

Abgesehen von den vielzähligen lokalen Minima, die kleine Funktionswerte erzeugen, konnten keine neuen Hinweise auf weitere zielführende Bereiche gefunden werden. Das bedeutet nicht, dass keine weitaus höheren Zielfunktionswerte existieren können.

Vielmehr gilt die Behauptung, dass die mit den gewählten Experimenten berechneten genauen Systemanalysen ein KNN erzeugen, dessen Approximation keine besseren Ergebnisse im erweiterten Suchbereich beinhaltet.

Existieren ein hoher Wert mit einer steilen Umgebung in diesem Bereich, ist die Wahrscheinlichkeit sehr hoch, dass durch die Begrenztheit der Rasterfeinheit keine Kenntnis über diesen Punkt erlangt werden kann.

Möchte man den Zielfunktionswert weiter erhöhen, empfiehlt es sich die Phasenzeiten weiter zu reduzieren, bis sie bei einem Grenzwert größer 0 ihr Maximum erreichen. Inwieweit sich ein optimaler Nutzen in Bezug auf eine gesteigerter Produktionsrate und Materialverschleiß einstellt, kann bei schwer quantifizierbaren Verschleißparametern nur geschätzt werden.

# Fazit

Betriebsparameter eines unbekannten Systems zu optimieren stellt eine besondere Herausforderung dar, denn um einen deterministischen Ansatz zu verfolgen, bedarf es einem Anhaltspunkt. Die angenäherte optimale Erstellung eines solchen, bildet die bestmögliche Vorgehensweise. Erreicht wir dies, durch die angenähert simultane Berechnung der als Raster angeordneten Zustandspunkte.

Die Forderung nach stabilen Betriebspunkten wurde genutzt, um dem unbekannten System eine gewisse Flachheit zu unterstellen. Eine so erzeugte Begrenzung der Potenzialität ermöglicht es, die eingefügten Informationen durch die Formulierung einer Umgebung der genauen Zustandspunkte direkt zu verwenden.

Die Zusammenführung der Informationen der genauen Analysen, mit denen der geschätzten tiefschichtigen Informationen des künstlichen neuronalen Netzes, erzeugt darüber hinaus eine Ressourceneffizienz. Auch vermeidlich ungünstig platzierte Punkte können in mehrschichtigen neuronalen Netzen zur Komplettierung der Mustererkennung beitragen. Das Grundproblem der Ressourcenintensivität wird dahingehend gelöst, dass durch die Verbesserung der Suche, durch die Suche selbst, in einer Art Rückkopplungskreislauf den Ressourcenaufwand mildert.

Es hat sich herausgestellt, dass neuronale Netze auch mit geringen Datenmengen Systeme grundlegend beschreiben können. Was bedeutet, dass kein Wechsel zu anderen Regressionsmethoden vorgenommen werden muss. Für die sieben Parameter Optimierung hat das Netz mit 1424 Datenpunkten global gesehen ein sehr genaues Abbild des Systemverhaltens erzeugt. Diese Genauigkeit ist bei stark unregelmäßigen Funktionsverläufen vorteilhaft. So hilft das neuronale Netz diese Ungewissheiten mit einer hohen Sicherheit auszuschließen.

Die hohe Anzahl an Simulationen, ist der Konzeption als direkt analysierbares Netz geschuldet. Die Parameteranalyse über den gesamten Parameterraum hat zwar keine konstante Fehlerabweichung, kann aber vor allem in besseren Bereichen sehr gute Auskünfte über das Systemverhalten geben.

Die beingeschränkte Auflösung des Approximationsrasters begrenzt die Anwendungsfähigkeit des Optimierungskonzeptes. Die Platzierung der neuen Punkte folgt einem zielführenden Prinzip, jedoch ist die Ausführung sehr rustikal aufgebaut. Eine bessere Anwendbarkeit und Effizienz

könnte erzielt werden, wenn begrenzte Teilbereiche mit höherer Auflösung approximiert werden und der Radius der Umgebungen von Werten der Nachbarpunkte abhängig sind.

Der Einsatz der neuronalen Netze in der Response Surface Methode ist eine Erweiterung, welche ihre Vorteile nicht in der Reduzierung der benötigten Analysen aufweist. Vielmehr sollte diese Erweiterung verwendet werden, wenn das Systemverhalten unbekannt und der Parameterbereich weitläufig ist. Die gewonnen ersten Erkenntnisse können eine Regression mit anderen Funktionstypen motivieren.

Die Optimierung der Betriebsparameter des thermochemischen Solarreaktors nach der Wasserstoffproduktionsrate kann in den einzelnen Auslegungen als erfolgreich angesehen werden. Die globale Betrachtung des Suchbereichs in den einzelnen Optimierungsauslegungen schließt ein besseres Ergebnis mit einer geringen Restwahrscheinlichkeit aus.

Versuchen zufolge, strebt die Produktionsrate einen Wert von etwa 200[g/h] an, bei welchem die Phasenzeiten zwischen 10 und 50 Sekunden liegen. Es konnten keine anderen Betriebspunkte mit vergleichbarer Produktionsrate bestimmt werden. So liegen die Produktionsraten in flacheren Bereichen bei 30-50[g/h]. Auch wenn der optimale Betriebspunkt eine sehr harte Fahrweise für den im Wechselbetrieb genutzten Reaktor darstellt, muss die Nutzenausbeute im Vergleich zu anderen Punkten abgewogen werden.

Trotz der globalen Betrachtung ist dieses Optimum nicht als absolutes globales Optimum zu sehen. Durch die Erweiterung der Optimierungsparameter können neue Extrema entstehen, welche höhere Werte aufweisen. Teilt man den Zyklus beispielsweise in deutlich mehr Zeitbereiche auf und weist man diesen eine Temperatur, Massefluss und Gasverhältnis zu, kann ein nahezu kontinuierlicher optimaler Verlauf der Parameter entstehen. Diese Art der Optimierung ist qualitativ überlegen. Die Durchführbarkeit solcher Optimierungen wird durch die voranschreitenden Rechenressourcen in Form von Quantencomputern und Netzwerktechnologien, wie die des Clustering verbessert. Verschachtelt man diese verbesserten Optimierungen, so entstehen selbstoptimierende problemlösende Algorithmen die autonom Lösungen finden können.

# Literaturverzeichnis

Akinci, K. (2020). *Optimierung der Auslegungsparameter eines permanentmagneterregten Synchronmotors am beispiel eines FE-Entwicklungs-Tools.* Köln: Rheinische Fachhochschule Köln.

Bridgman, P. W. (1932). *Theorie der physikalischen Dimensionen.* Leibzig und Berlin: Teubner.

Deutsche Zentrum für Luft- und Raumfahrt. (15. 11 2020). *https://www.dlr.de*. Von https://www.dlr.de/sf/de/desktopdefault.aspx/tabid-9315/16078_read-39605/22259_read-51115/ abgerufen

Friedman, L. W. (1996). *The Simulation Metamodel.* Springer US.

Gleichmar, R. (2004). *Approximationen und paralleles Rechnen bei der multidisziplin¨aren Strukturoptimierung.* München: Fakultät für Maschinenwesen der Technischen Universität München.

Görtler, H. (1975). *Dimensionsanalyse: Eine Theorie der physikalischen Dimensionen mit Anwendungen.* Berlin: Springer Verlag.

Kleijnen, J., & Sargent, R. (2000). *A methodology for ®tting and validating metamodels in simulation.* European Journal of Operational Research.

Krause, J. (2020). *Validierung und Erweiterung eines Reaktionskinetikmodells für einen Solarreaktor zur thermochemischen Spaltung von Wasser.* Köln: Rheinische Fachhochschule Köln.

Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Ruß, G., & Steinbrecher, M. (2011). *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze,Evolutionäre Algorithmen,Fuzzy-Systeme und Bayes-Netze*. Magdeburg: Vieweg+Teubner Verlag.

Mead, G. H. (1925). *Die Genesis der Identität und die soziale Kontrolle. In: Ders.: Gesammelte Aufsätze. Bd. 1.* Frankfurt a.M: Suhrkamp.

Menz, S., Lampe, J., Tröltzsch, U., Weiler, P., Pahl, A., Fend, T., & Seeger, T. (2020). *Real time executable model for dynamic heat flow analysis of solar hydrogen reactor.*

Nissen, V. (1994). *Evolutionäre Algorithmen.* Deutscher Universitätsverlag.

Rudolph, G. (1990). *Globale optimierung mit parallelen Evolutionsstrategien.* Dortmund: Universität Dortmund.

Säck, J. P. (2012). *Entwicklung und Simulation einer Wasserstofferzeugungsanlage auf einem Solarturm.* Köln: Universität Duisburg-Essen.

Schmitz, J. (2017). *Grundlagen Neuronaler Netze.* Technische Universität München.

Schwefel, H. P. (1977). *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie.* Basel, Stuttgart: Birkhäuser.

Törn, A., & Žilinskas, A. (1989). *Global Optimization, Lecture Notes in Computer Science Vol. 350.* Berlin: Springer.

# Eigenständigkeitserklärung

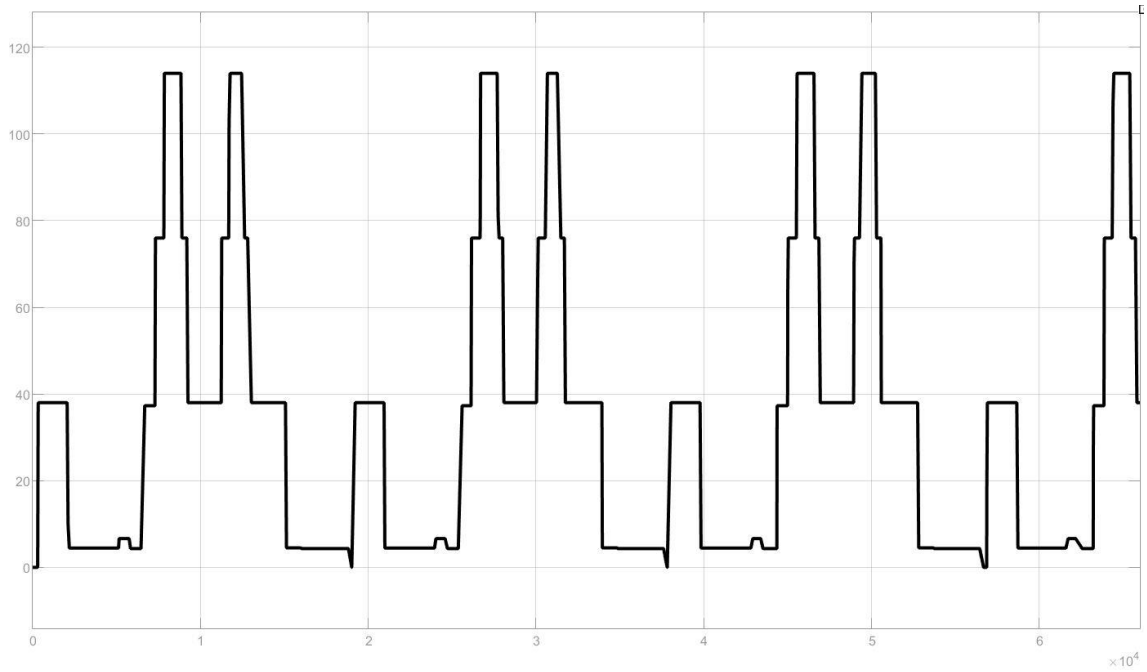
Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen Publikationen, Vorlagen und Hilfsmitteln als die angegebenen benutzt habe. Alle Teile meiner Arbeit, die wortwörtlich oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht. Gleiches gilt für von mir verwendete Internetquellen. Ich versichere, dass ich diese Arbeit oder nicht zitierte Teile daraus vorher nicht in einem anderen Prüfungsverfahren eingereicht habe. Mir ist bekannt, dass meine Arbeit zum Zwecke eines Plagiatsabgleichs mittels einer Plagiatserkennungssoftware auf eine ungekennzeichnete Übernahme von fremdem geistigen Eigentum überprüft werden kann. Ich versichere, dass die elektronische Form meiner Arbeit mit der gedruckten Version identisch ist.

Köln, 03.12.2020 

Ort, Datum Unterschrift

# Anhang A

Festes Solarprofil für die Verwendung im Temperaturregler



# Anhang B

Teil eins des Iterationsprotokolls der sieben Parameteroptimierung. Angegeben ist die Feinheit des Approximationsraster, Mittelwerte des gesamten Datensatzes sowie der Experimente in [g/h] und die Anzahl der Experimente der jeweiligen Iteration.

| Iteration | *L*/*l* | Mittelwert Exp. | Mittelwert Ges. | Anz. Experimente |
|---|---|---|---|---|
| - | 1 | 9.924 | 9.924 | 128 |
| 1 | 2 | 26.77 | 14.65 | 50 |
| 2 | 2 | 38.31 | 18.72 | 50 |
| 3 | 2 | 23.39 | 19.01 | 20 |
| 4 | 2 | 20.82 | 19.04 | 26 |
| 5 | 2 | 39.26 | 19.30 | 12 |
| 6 | 4 | 22.63 | 19.88 | 50 |
| 7 | 4 | 23.40 | 20.40 | 50 |
| 8 | 4 | 27.03 | 21.26 | 50 |
| 9 | 4 | 24.08 | 21.58 | 50 |
| 10 | 4 | 21.42 | 21.57 | 50 |
| 11 | 4 | 21.87 | 21.60 | 50 |
| 12 | 4 | 19.32 | 21.40 | 50 |
| 13 | 4 | 22.05 | 21.45 | 50 |
| 14 | 4 | 19.37 | 21.30 | 50 |
| 15 | 4 | 24.38 | 21.51 | 50 |
| 16 | 4 | 17.97 | 21.28 | 50 |
| 17 | 4 | 24.00 | 21.45 | 50 |
| 18 | 4 | 24.55 | 21.62 | 50 |
| 19 | 4 | 20.56 | 21.57 | 50 |
| 20 | 4 | 19.26 | 21.45 | 50 |
| 21 | 4 | 21.14 | 21.43 | 50 |
| 22 | 4 | 22.85 | 21.50 | 50 |
| 23 | 4 | 24.56 | 21.63 | 50 |
| 24 | 4 | 19.46 | 21.54 | 50 |
| 25 | 4 | 21.16 | 21.53 | 50 |
| 26 | 4 | 20.16 | 21.47 | 50 |
| 27 | 4 | 19.68 | 21.41 | 32 |
| 28 | 4 | 15.00 | 21.26 | 3 |
| 29 | 4 | 13.69 | 21.24 | 50 |
| 30 | 4 | 9.04 | 20.81 | 3 |

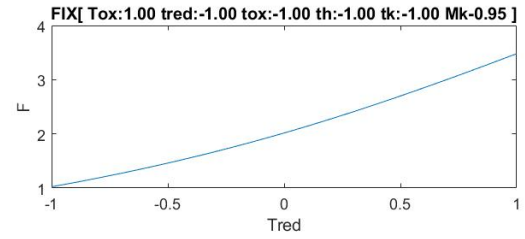
Teil zwei des Iterationsprotokolls der sieben Parameteroptimierung. Angegeben ist die Feinheit des Approximationsraster, Mittelwerte des gesamten Datensatzes sowie der Experimente in [g/h] und die Anzahl der Experimente der jeweiligen Iteration.

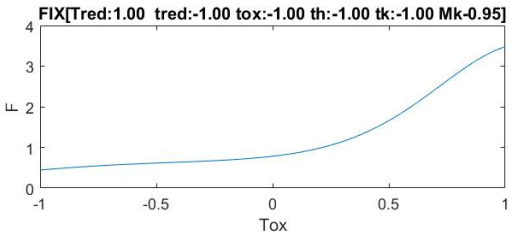
| Iteration | *L*/*l* | Mittelwert Exp. | Mittelwert Ges. | Anz. Experimente |
|---|---|---|---|---|
| - | 1 | 3.89 | 19.70 | 112 |
| 1 | 2 | 31.57 | 19.77 | 9 |
| 2 | 2 | 27.71 | 19.78 | 6 |
| 3 | 4 | 14.61 | 19.56 | 50 |

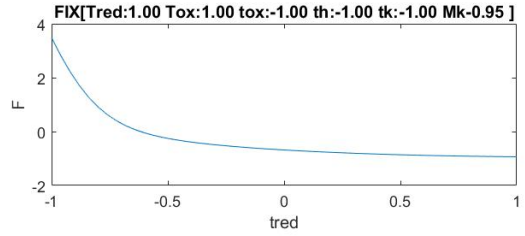
| | | | | |
|---|---|---|---|---|
| 4 | 4 | 14.48 | 19.28 | 50 |
| 5 | 4 | 14.98 | 19.09 | 50 |
| 6 | 4 | 12.82 | 18.91 | 50 |
| 7 | 4 | 15.54 | 18.83 | 50 |
| 8 | | 13.87 | 18.68 | 50 |
| 9 | 4 | 10.39 | 18.61 | 50 |
| 10 | 4 | 11.82 | 18.44 | 50 |

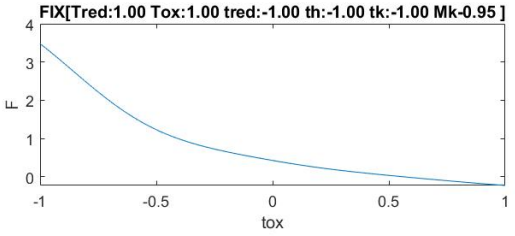
# Anhang C

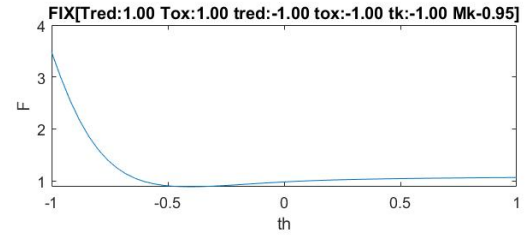
Funktionsverläufe der approximierten Zielfunktion im normierten Betriebspunkt [1 1 -1 -1 -1 -1 1]. welche dem Betriebspunkt der realen Werte [1400°C 1400°C 50s 50s 15s 15s 22kg/h] entspricht.

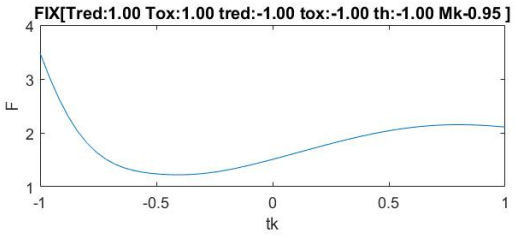


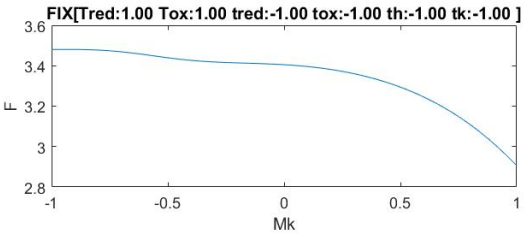



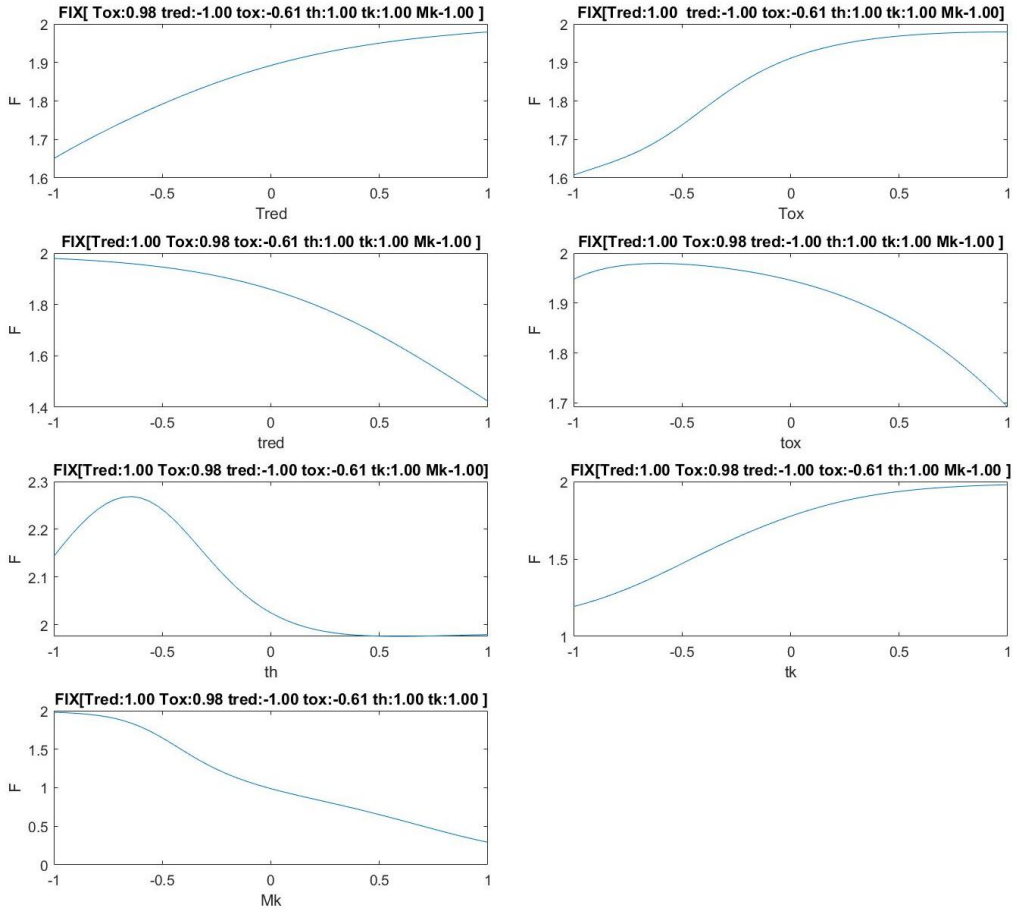



Funktionsverläufe der approximierten Zielfunktion im normierten Betriebspunkt [1 0.98 -1 -0.61 1 1 -1]. welche dem Betriebspunkt der realen Werte [1400°C 1394°C 50s 176.75s 100s 100s 20kg/h] entspricht



# Anhang D

# main.m

```
clear


%Löschen aller Cluster/Jobs
delete(gcp('nocreate'));
myCluster = parcluster('local');
delete(myCluster.Jobs);


%Oberen Grenzwerte/Falls eingegrenzt werden kann, tuen!!!
opti.Tred_ub     =   1400;
opti.Tprod_ub    =   1400;
opti.tred_ub     =   3000;
opti.tox_ub      =   700;

opti.M_red_ub    =   220;
opti.M_ox_ub     =   50;

opti.th_ub       =   100;
opti.tk_ub       =   100;

opti.Mk_ub       =   100;



%Unteren Grenzwerte
opti.Tred_lb     =   1200;
opti.Tprod_lb    =   800;

%25K/min zum aufheizen
opti.tred_lb     =   50;
opti.tox_lb      =   50;

opti.M_red_lb    =   10;
opti.M_ox_lb     =   5;

opti.th_lb       =   15;
opti.tk_lb       =   15;

opti.Mk_lb       =   20;







%////////////////////////////Optionen/////////////////////////////////////

%Welche simset-Funktion wird genutzt
opti.simset=7;

%Model
opti.model = 'reactor_5_5_MA_model';

%Raster Start a=L/Teiler n=4 Dim.  (2+a/a)^n
opti.Teiler=1;
%Raster Approx.
opti.Teileraprox=opti.Teiler*2;

%Anzahl der Experimente in der konkretisierung
opti.N2=50;
%opti.N2=((2+(2/opti.Teiler))/(2/opti.Teiler))^opti.simset;
[~,nfactors]=size(opti.bounds);

%////////////////////////////////////////////////////////////////////////
```

```
if opti.simset==4
    opti.bounds =   [opti.Tred_ub opti.Tprod_ub opti.tred_ub opti.tox_ub ; opti.Tred_lb
opti.Tprod_lb opti.tred_lb opti.tox_lb ];
end
if opti.simset==6
    opti.bounds =   [opti.Tred_ub opti.Tprod_ub opti.tred_ub opti.tox_ub opti.M_red_ub
opti.M_ox_ub; opti.Tred_lb opti.Tprod_lb opti.tred_lb opti.tox_lb opti.M_red_lb
opti.M_ox_lb];
    opti.Teileraproxmax=4;
end
if opti.simset==7
    opti.bounds =   [opti.Tred_ub opti.Tprod_ub opti.tred_ub opti.tox_ub opti.th_ub
opti.tk_ub opti.Mk_ub; opti.Tred_lb opti.Tprod_lb opti.tred_lb opti.tox_lb opti.th_lb
opti.tk_lb opti.Mk_lb];
    opti.Teileraproxmax=8;
end




opti.iter=1;
opti.protokoll=[];

%------------------Konkretisierungsschleife--------------------------------
while opti.Teileraprox < opti.Teileraproxmax

%-------------------------------------------------------------------------
    %Laden der gespeicherten Ergebnisse
    if opti.simset==4
        opti.Mat=load('4P_res');
    end
    if opti.simset==6
        opti.Mat=load('6P_res');
    end
    if opti.simset==7
        opti.Mat=load('7P_res');
    end

%-------------------------------------------------------------------------

    %Erstellen des Grundrasters
    [opti.C,opti.Cnorm,opti.meshalt]=raster(opti.Teiler,opti.simset,opti.bounds);


    %Falls Ergebnisse vorhanden, mit Raster Abgleichen
    if isempty(opti.Mat.Eges)==0
    opti.A=round(opti.Mat.Eges(:,1:opti.simset),0);
    opti.B=round(opti.C,0);
    [~,aid,bid] = intersect(opti.A,opti.B,'rows');
    opti.C( bid, : ) = [];
    end



    %Wurden Alle Versuche aus dem Stützraster berechnet, Startet die
    %Schätzung
    while isempty(opti.C)==1

    %Konstruktion des KNN
    [opti.Cnorm,opti.ZappYh2snormnorm,opti.tablenewnorm,opti.Yh2snorm]=NNet-
const(opti.Teileraprox,opti.Mat.Eges,opti.simset,opti.bounds);



     %Neue Punkte erstellen
    [opti.Znewschlecht,opti.Znewnorm,opti.tablenewnorm]  = posnice(opti.Zap-
pYh2snormnorm,opti.tablenewnorm,opti.Yh2snorm,opti.Cnorm,opti.simset,opti.Teileraprox);


    %Auswahl aus den neuen Punkten
    [opti.C]=newexp(opti.Znewschlecht,opti.Znewnorm,opti.tab-
lenewnorm,opti.Cnorm,opti.bounds,opti.N2,opti.simset);
```

```
    if isempty(opti.C)==1
    opti.Teileraprox=opti.Teileraprox*2;
    end

    end



    delete(gcp('nocreate'));
    myCluster = parcluster('local');
    delete(myCluster.Jobs);

    opti.Cnotdone=opti.C;

    backup=[];
    save('backupparcom','backup');

        %Solange nicht alle Berechnungen fertig sind
        while isempty(opti.Cnotdone)==0

            %Umwandling der Parameter und parallele Berechnung
            [par]=parcomfunc(opti.Cnotdone,opti.model,opti.simset);

            %Extraktion der Ergebnisse aus dem Logfile
            [par.Yh2z,par.Yh2s,par.Yh2e,opti.error]=resultstovector(opti.Cnot-
done,par.out,opti.simset);

            %Ungültige oder ERROR Versuche wurden zuvor 0 gesetzt und jetzt
            %rausgesucht.
            opti.index0=find(par.Yh2s==0);
            opti.indexnot0=find(par.Yh2s~=0);

            %Alle Versuche mit Ergebnisse != 0 werden opti.Cdone
            %zugeordnet
            opti.Cdone=removerows(opti.Cnotdone,'ind',opti.index0);
            par.Yh2z=removerows(par.Yh2z,'ind',opti.index0);
            par.Yh2s=removerows(par.Yh2s,'ind',opti.index0);
            par.Yh2e=removerows(par.Yh2e,'ind',opti.index0);

            %Backup verhindert bei abstürzen komplett neu Berechnet werden
            %muss
            opti.backupload=load('backupparcom');
            if isempty(opti.backupload.backup)==0
            backup=[opti.backupload.backup;[opti.Cdone par.Yh2z,par.Yh2s,par.Yh2e]];
            else
                backup=[opti.Cdone par.Yh2z,par.Yh2s,par.Yh2e];
            end

            save('backupparcom','backup');

            %Alle Versuche mit Ergebnisse = 0 werden opti.Cnotdone
            %zugeordnet
            opti.Cnotdone=removerows(opti.Cnotdone,'ind',opti.indexnot0);


        end


    opti.backupload=load('backupparcom');
    opti.C=opti.backupload.backup(:,1:opti.simset);
    opti.Yh2z=opti.backupload.backup(:,end-2:end-2);
    opti.Yh2s=opti.backupload.backup(:,end-1:end-1);
    opti.Yh2e=opti.backupload.backup(:,end:end);


    savedeldouble(opti.C,opti.Yh2z,opti.Yh2s,opti.Yh2e,opti.simset);



    opti.L=load('7P_res');
    Eges=opti.L.Eges;
    opti.protokoll=[opti.protokoll; [opti.iter opti.Teileraprox (sum(par.Yh2s)/nu-
mel(par.Yh2s)) sum(Eges(:,end-1))/numel(Eges(:,end-1)) numel(Eges(:,end-1))]];

    opti.iter = opti.iter + 1;
end
```

# raster.m

```
function [C,grid,mesh] = raster(Teiler,simset,bounds)
        l=2/Teiler;
        %Raster Erstellen
        if simset==4
            x1=-1:l:1;
            x2=-1:l:1;
            x3=-1:l:1;
            x4=-1:l:1;
        [X1,X2,X3,X4] =  ndgrid(x1,x2,x3,x4);
        grid = [X1(:), X2(:), X3(:), X4(:)];
        mesh=  ndgrid(x1,x2,x3,x4);
        end

        if simset==6
        x1=-1:l:1;
        x2=-1:l:1;
        x3=-1:l:1;
        x4=-1:l:1;
        x5=-1:l:1;
        x6=-1:l:1;
        [X1,X2,X3,X4,X5,X6] =  ndgrid(x1,x2,x3,x4,x5,x6);
        grid = [X1(:), X2(:),X3(:), X4(:),X5(:), X6(:)];
        mesh =  ndgrid(x1,x2,x3,x4,x5,x6);
        end
        if simset==7
        x1=-1:l:1;
        x2=-1:l:1;
        x3=-1:l:1;
        x4=-1:l:1;
        x5=-1:l:1;
        x6=-1:l:1;
        x7=-1:l:1;

        [X1,X2,X3,X4,X5,X6,X7] =  ndgrid(x1,x2,x3,x4,x5,x6,x7);
        grid = [X1(:), X2(:),X3(:), X4(:),X5(:), X6(:), X7(:)];
        mesh=ndgrid(x1,x2,x3,x4,x5,x6,x7);
        end
        %Normiertes Raster umwandeln
        C=[];
        if isempty(bounds)==0
            C=grid;
            for q=1:numel(grid(:,1))
                for i=1:simset
                    Pm=(C(q,i)*(bounds(1,i)-bounds(2,i))+bounds(1,i)+bounds(2,i))/2;
                    C(q,i) = Pm;
                end
            end
        end
end
```

# NNetconst.m

```
function [Cnorm,ZappYh2snormnorm,grid,Yh2snormnorm] = NNetconst(Teileraprox,Eges,sim-
set,bounds)
%Ergebnis aufteilen
C=Eges(:,1:simset);
Yh2s=Eges(:,end-1:end-1);
Yh2z=Eges(:,end-2:end-2);
Yh2e=Eges(:,end:end);

%Target normalisieren
Yh2snorm=normalize( Yh2s );
Yh2znorm=normalize( Yh2z );
Yh2enorm=normalize( Yh2e );

Yh2snormnorm=normalised_diff(Yh2snorm);

Cnorm=normwbounds(C,bounds);

%Optimales Netz
[make.netz,~]=optimalnet([Cnorm Yh2znorm],0.8);
```

```
[make.nets,~]=optimalnet([Cnorm Yh2snorm],0.8);
[make.nete,~]=optimalnet([Cnorm Yh2enorm],0.8);


function [Yh2zpred]=predYh2z(x)
     x=x.';
     Yh2zpred=make.netz(x);
end

function [Yh2spred]=predYh2s(x)
     x=x.';
     Yh2spred=make.nets(x);
end

function [Yh2epred]=predYh2e(x)
     x=x.';
     Yh2epred=make.nete(x);
end

        l=2/Teileraprox;
        if simset==4
            x1=-1:l:1;
            x2=-1:l:1;
            x3=-1:l:1;
            x4=-1:l:1;
        [X1,X2,X3,X4] =  ndgrid(x1,x2,x3,x4);
        grid = [X1(:), X2(:), X3(:), X4(:)];
        mesh=  ndgrid(x1,x2,x3,x4);
        end

        if simset==6
        x1=-1:l:1;
        x2=-1:l:1;
        x3=-1:l:1;
        x4=-1:l:1;
        x5=-1:l:1;
        x6=-1:l:1;
        [X1,X2,X3,X4,X5,X6] =  ndgrid(x1,x2,x3,x4,x5,x6);
        grid = [X1(:), X2(:),X3(:), X4(:),X5(:), X6(:)];
        mesh=ndgrid(x1,x2,x3,x4,x5,x6);
        end
        if simset==7
        x1=-1:l:1;
        x2=-1:l:1;
        x3=-1:l:1;
        x4=-1:l:1;
        x5=-1:l:1;
        x6=-1:l:1;
        x7=-1:l:1;

        [X1,X2,X3,X4,X5,X6,X7] =  ndgrid(x1,x2,x3,x4,x5,x6,x7);
        grid = [X1(:), X2(:),X3(:), X4(:),X5(:), X6(:), X7(:)];
        mesh=ndgrid(x1,x2,x3,x4,x5,x6,x7);
        end


ZappYh2snorm=predYh2s(grid);
ZappYh2enorm=predYh2e(grid);
ZappYh2znorm=predYh2z(grid);




ZappYh2snorm=transpose(ZappYh2snorm);
ZappYh2enorm=transpose(ZappYh2enorm);
ZappYh2znorm=transpose(ZappYh2znorm);


ZappYh2snormnorm=normalised_diff(ZappYh2snorm);
ZappYh2enormnorm=normalised_diff(ZappYh2enorm);
ZappYh2znormnorm=normalised_diff(ZappYh2znorm);


end

function X = normalised_diff( data )
```

```
ma=max(max(data));
mi=min(min(data));
X=[];
    for i=1:numel(data(1,:))
        Xcol=[];
        for q=1:numel(data(:,1))
            zaeler=data(q,i)-(ma+mi)/2;
            nenner=(ma-mi)/2;
            Xcol=[Xcol;zaeler/nenner];


        end
        X=[X Xcol];
    end
end
```

# posnice.m

```
function [Znewschlecht,Znewnormfin,tablenewnormfin]=posnice(Znewnorm,tablenewnorm,Zold-
norm,tableoldnorm,simset,Teileraprox)


%Finde Index gleicher tablerow
[~,aid,~] = intersect(tablenewnorm,tableoldnorm,'rows');

%Werte von Znewnorm löschen für die Werte von Zoldnorm existieren (aid)
Znewnorm( aid, : ) = [];
tablenewnorm( aid, : ) = [];

%Anzahl von N
Nnew=numel(Znewnorm);
Nold=numel(Zoldnorm);
%Alten werte werden absteigend sortiert damit die Umgebund der besten Werte
%die der schlechteren überschreibt.

[Zoldnorm,I]=sort(Zoldnorm);
tableoldnorm=tableoldnorm(I,:);

%Anschließend die Alten Werte anhängen/wieder alte größe erreicht
Znewnorm=[Znewnorm;Zoldnorm];
tablenewnorm=[tablenewnorm;tableoldnorm];

step=2/Teileraprox ;
%Umgebung

%Dafür müssen nun nur die letzten Zeilen abgefragt werden
Znewnormcopy=zeros(Nnew,1);
for i=Nnew:Nnew+Nold
    %Die Umgebung wird nur für Werte >0.5 erzeugt
    if Znewnorm(i)>0.5


        %Ein Hyperwürfel mit der Seitenlänge l' *2 wird um den Punkt
        %konstruiert


        if simset==4
        x1=tablenewnorm(i,1)-step:step:tablenewnorm(i,1)+step;
        x2=tablenewnorm(i,2)-step:step:tablenewnorm(i,2)+step;
        x3=tablenewnorm(i,3)-step:step:tablenewnorm(i,3)+step;
        x4=tablenewnorm(i,4)-step:step:tablenewnorm(i,4)+step;
        [X1,X2,X3,X4] =  ndgrid(x1,x2,x3,x4);
        Hw = [X1(:), X2(:),X3(:), X4(:)];
        end

        if simset==6
        x1=tablenewnorm(i,1)-step:step:tablenewnorm(i,1)+step;
        x2=tablenewnorm(i,2)-step:step:tablenewnorm(i,2)+step;
        x3=tablenewnorm(i,3)-step:step:tablenewnorm(i,3)+step;
        x4=tablenewnorm(i,4)-step:step:tablenewnorm(i,4)+step;
        x5=tablenewnorm(i,5)-step:step:tablenewnorm(i,5)+step;
        x6=tablenewnorm(i,6)-step:step:tablenewnorm(i,6)+step;
        [X1,X2,X3,X4,X5,X6] =  ndgrid(x1,x2,x3,x4,x5,x6);
        Hw = [X1(:), X2(:),X3(:), X4(:),X5(:), X6(:)];
        end
```

```
    if simset==7
    x1=tablenewnorm(i,1)-step:step:tablenewnorm(i,1)+step;
    x2=tablenewnorm(i,2)-step:step:tablenewnorm(i,2)+step;
    x3=tablenewnorm(i,3)-step:step:tablenewnorm(i,3)+step;
    x4=tablenewnorm(i,4)-step:step:tablenewnorm(i,4)+step;
    x5=tablenewnorm(i,5)-step:step:tablenewnorm(i,5)+step;
    x6=tablenewnorm(i,6)-step:step:tablenewnorm(i,6)+step;
    x7=tablenewnorm(i,7)-step:step:tablenewnorm(i,7)+step;

    [X1,X2,X3,X4,X5,X6,X7] =  ndgrid(x1,x2,x3,x4,x5,x6,x7);
    Hw = [X1(:), X2(:),X3(:), X4(:),X5(:), X6(:), X7(:)];
    end


    %Punkte des Hyperwürfels werden gelöscht wenn sie die Grenzen
    %überschreiten

    Hw2=[];
    for f=1:numel(Hw(:,1))

        ma=max(Hw(f,:));
        mi=min(Hw(f,:));

        if mi>=-1    &&  ma<=1
            Hw2=[Hw2;Hw(f,:)];
        end

    end


    %oder die Hyperkugel Bedingung verletzen
    %Abstand Hw(f,:) Punkt und tablenewnorm(i) wird berechnet
    Hw3=[];
    for f=1:numel(Hw2(:,1))

        Ab1=Hw2(f,1)-tablenewnorm(i,1);
        Ab2=Hw2(f,2)-tablenewnorm(i,2);
        Ab3=Hw2(f,3)-tablenewnorm(i,3);
        Ab4=Hw2(f,4)-tablenewnorm(i,4);

        dist=sqrt((Ab1^2)+(Ab2^2)+(Ab3^2)+(Ab4^2));

        if simset==6
        Ab5=Hw2(f,5)-tablenewnorm(i,5);
        Ab6=Hw2(f,6)-tablenewnorm(i,6);

        dist=sqrt((Ab1^2)+(Ab2^2)+(Ab3^2)+(Ab4^2)+(Ab5^2)+(Ab6^2));
        end

        if simset==7
        Ab5=Hw2(f,5)-tablenewnorm(i,5);
        Ab6=Hw2(f,6)-tablenewnorm(i,6);
        Ab7=Hw2(f,7)-tablenewnorm(i,7);


        dist=sqrt((Ab1^2)+(Ab2^2)+(Ab3^2)+(Ab4^2)+(Ab5^2)+(Ab6^2)+(Ab7^2));
        end

        if dist<= 1
           Hw3=[Hw3;Hw2(f,:)];
        end



    end
%Die Punkte Hw3 werden in tablenenorm gesucht und ersetzt durch
%Znewnorm(i) so wird die Umgebung mit dem Abstand 1 für alle Werte über
%0 gleichgesetzt

[~,aid,~] = intersect(tablenewnorm,Hw3,'rows');

for f=1:numel(aid)
    Znewnormcopy(aid(f),:)=Znewnorm(i);
end

end%EndeZnewnorm(i)>0
```

```
end%Ende von i=(Nnew-Nold):Nnew

%Die Werte von Znewnormcopy werden in Znewnorm geschrieben

indexnot0=find(Znewnormcopy~=0);

for f=1:numel(indexnot0)

    Znewnorm(indexnot0(f,1),1)=Znewnormcopy(indexnot0(f,1));
end



%Am Ende werden alle Werte < 0.5 entfernt
Znewnormfin=[];
Znewschlecht=[];
tablenewnormfin=[];
for f=1:numel(Znewnorm)
    if Znewnorm(f)>0.5
        Znewnormfin=[Znewnormfin;Znewnorm(f)];
        tablenewnormfin=[tablenewnormfin;tablenewnorm(f,:)];
    else
        Znewschlecht=[Znewschlecht;tablenewnorm(f,:)];
    end

end

end
```

# newexp.m

```
function [C] = newexp(Znewschlecht,Znewnorm,tablenewnorm,CnormEges,bounds,N,simset)

%Mit Hilfe der günstigen Punkte in 'Znewnorm', werden die neuen Experimente bestimmt.


tablenewnormsort=tablenewnorm;

% Schon Berechnete Punkte werden Entfernt
[~,aid,~] = intersect(tablenewnormsort,CnormEges,'rows');
tablenewnormsort(aid,:)=[];
[~,aid,~] = intersect(Znewschlecht,CnormEges,'rows');
Znewschlecht(aid,:)=[];

%Sortieren
tablenewnormsort=unique(tablenewnormsort,'rows');

%Test auf doppelte
Znewschlecht=unique(Znewschlecht,'rows');

%Wenn noch Punkte bleiben
if isempty(tablenewnormsort)==0

    a = randn(numel(tablenewnormsort(:,1)),1);
   indicesa = randperm(length(a));

   b = randn(numel(Znewschlecht(:,1)),1);
   indicesb = randperm(length(b));

    %Ggf. anpassung von N
    if numel(tablenewnormsort(:,1))<N
    N=numel(tablenewnormsort(:,1));
    end


 %Anzahl der Experimente N muss auf ein Kontingent für Zufallspunkte Z und
 %gute Punkte G aufgeteilt werden /////Kontingen 10%

 Z=floor(N/10);
 G=N-Z;

    indicesvona = indicesa(1:G);
    indicesvonb = indicesb(1:Z);
    dCE = [tablenewnormsort(indicesvona,:);Znewschlecht(indicesvonb,:)];
```

```
    C=dCE;
    for q=1:numel(dCE(:,1))
        for i=1:simset
            Pm=(C(q,i)*(bounds(1,i)-bounds(2,i))+bounds(1,i)+bounds(2,i))/2;
            C(q,i) = Pm;
        end
    end
else
C=tablenewnormsort;
end
end
```

# parcomfunc.m

```
function [par] = parcomfunc(C,model,simset)

par.C=C;

isModelOpen       = bdIsLoaded(model);
load_system(model);
if isModelOpen==0
    load_system(model);
end

%Auswahl der Umwandlung
if simset==4
    par.S=simset_4P(par.C);
end

if simset==6
    par.S=simset_6P(par.C);
end

if simset==7
    par.S=simset_7P(par.C);
end


[par.anzexp,par.anzpara]=size(par.S);

%Hinterlegen der Variablen
for i=1:par.anzexp

    in(i)=Simulink.SimulationInput(model);

    in(i)=in(i).setVariable('t_ges',par.S{i,1});
    in(i)=in(i).setVariable('t_sim_T',par.S{i,2});
    in(i)=in(i).setVariable('t_sim_M',par.S{i,3});
    in(i)=in(i).setVariable('T_sim',par.S{i,4});
    in(i)=in(i).setVariable('M_sim_N2',par.S{i,5});
    in(i)=in(i).setVariable('M_sim_H2O',par.S{i,6});

end


delete(gcp('nocreate'));
assignin('base','par', par)

par.in=in;

%Paralelle Berechnung
par.out=parsim(par.in,'ShowProgress',"on","TransferBaseWorkspaceVariables","off","Use-
FastRestart","off");

if(~isModelOpen)
    close_system(model, 0);
end
delete(gcp('nocreate'));

end
```

# simset_4P.m

```
function [S]    =   simset_4P(C)
```

```
    S=[];

    cy=5;

    for c=C.'
        T_red=c(1);
        T_ox=c(2);
        t_red=c(3);
        t_ox=c(4);
        M_N2=100;
        M_H2O=20;

        t_ges=t_red+t_ox;

        t_sim       =   [];
        T_sim       =   [];
        M_sim_N2    =   [];
        M_sim_H2O   =   [];
        Mredmin     =   1;


        for i=0:cy-1

            t_sim       = [t_sim i*(t_red+t_ox)+4001 i*(t_red+t_ox)+t_red+4000
i*(t_red+t_ox)+t_red+4001 i*(t_red+t_ox)+t_red+t_ox+4000];
            T_sim       = [T_sim T_red T_red T_ox T_ox];
            M_sim_N2    = [M_sim_N2 M_N2 M_N2 Mredmin Mredmin];
            M_sim_H2O   = [M_sim_H2O 0 0 M_H2O M_H2O];

        end

        t_sim=[0 4000 t_sim];
        t_sim_T=t_sim;
        t_sim_M=t_sim;

        T_sim=[T_red T_red T_sim];

        M_sim_N2 = [Mredmin Mredmin M_sim_N2];
        M_sim_H2O = [0 0 M_sim_H2O] ;

        t_ges=cy*(t_red+t_ox)+4000;



        simvec={t_ges,[t_sim_T],[t_sim_M], [T_sim], [M_sim_N2], M_sim_H2O};

        if size(S)==0
            S=[simvec];
        else
            S=[S;simvec];
        end
    end

end
```

# simset_6P.m

```
function [S]    =   simset_6P(C)

    S=[];

    cy=5;

    for c=C.'
        T_red = c(1);
        T_ox  = c(2);
        t_red = c(3);
        t_ox  = c(4);
        M_N2  = c(5);
        M_H2O = c(6);

        t_ges = t_red+t_ox;

        t_sim       =   [];
        T_sim       =   [];
        M_sim_N2    =   [];
```

```
        M_sim_H2O   =   [];
        Mredmin     =   1;


        for i=0:cy-1

            t_sim       = [t_sim i*(t_red+t_ox)+4005 i*(t_red+t_ox)+t_red+4000
i*(t_red+t_ox)+t_red+4005 i*(t_red+t_ox)+t_red+t_ox+4000];
            T_sim       = [T_sim T_red T_red T_ox T_ox];
            M_sim_N2    = [M_sim_N2 M_N2 M_N2 Mredmin Mredmin];
            M_sim_H2O   = [M_sim_H2O 0 0 M_H2O M_H2O];

        end

        t_sim=[0 4000 t_sim];

        t_sim_T = t_sim;
        t_sim_M = t_sim;

        T_sim=[T_red T_red T_sim];


        M_sim_N2 = [Mredmin Mredmin M_sim_N2];
        M_sim_H2O = [0 0 M_sim_H2O] ;

        t_ges   = cy*(t_red+t_ox)+4000;



        simvec = {t_ges,[t_sim_T],[t_sim_M], [T_sim], [M_sim_N2], M_sim_H2O};

        if size(S)==0
            S=[simvec];
        else
            S=[S;simvec];
        end
    end


end
```

# simset_7P.m

```
function [S] = simset_7P(C)
S=[];
cy=10;

AZ=4000;

    for c=C.'
        T_red   = c(1);
        T_ox    = c(2);
        t_red   = c(3);
        t_ox    = c(4);


        M_red   = 124;
        M_ox    = 37;

        th      = c(5);
        tk      = c(6);


        Mk      = c(7);


        t_ges   = t_red+t_ox+th+tk;
        ton     = t_ges/2;
        toff    = t_ges-ton;

        t_sim_T     =   [];
        t_sim_M     =   [];
        T_sim       =   [];
        M_sim_N2    =   [];
        M_sim_H2O   =   [];
        Mredmin     =   1;
```

```
        step2       =  th+t_red;
        step3       =  step2+tk;
        step4       =  step3+t_ox;

        for i=0:cy-1

            t_sim_T     = [t_sim_T i*(ton+toff)+AZ+5 i*(ton+toff)+ton+AZ
i*(ton+toff)+ton+AZ+5 i*(ton+toff)+ton+toff+AZ];
            t_sim_M     = [t_sim_M i*(step4)+AZ+5 i*(step4)+th+AZ i*(step4)+th+AZ+5
i*(step4)+step2+AZ i*(step4)+step2+AZ+5 i*(step4)+step3+AZ i*(step4)+step3+AZ+5
i*(step4)+step4+AZ];
            T_sim       = [T_sim T_red T_red T_ox T_ox];
            M_sim_N2    = [M_sim_N2 Mredmin Mredmin M_red M_red Mk Mk Mredmin Mredmin];
            M_sim_H2O   = [M_sim_H2O 0 0 0 0 0 0 M_ox M_ox];

        end

        t_sim_T=[0 AZ t_sim_T];
        t_sim_M=[0 AZ t_sim_M];
        T_sim=[T_red T_red T_sim];
        M_sim_N2 = [Mredmin Mredmin M_sim_N2];
        M_sim_H2O = [0 0 M_sim_H2O] ;

        t_ges=cy*t_ges+AZ;



        simvec={t_ges,[t_sim_T],[t_sim_M], [T_sim], [M_sim_N2], M_sim_H2O};



        S=[S;simvec];

    end
end
```

# resultvector.m

```
function [Yh2z,Yh2s,Yh2e,Errorall] = resultstovector(C,out,simset)

Yh2z=[];
Yh2s=[];
Yh2e=[];

[~,anz]=size(out);
format longG
Errorall=0;
for i =1:anz
    %Zeitwerte zu t-zyklus
    if simset==4
       n=(C(i,3)+C(i,4))*5;
       ztime=C(i,3)+C(i,4);
    end
    if simset==6
       n=(C(i,3)+C(i,4))*5;
       ztime=C(i,3)+C(i,4);
    end
    if simset==7
       n=(C(i,3)+C(i,4)+C(i,5)+C(i,6))*8;
       ztime=(C(i,3)+C(i,4)+C(i,5)+C(i,6));
    end
    %Zeitpunkt andem der erste Zyklus startet
    n=n+4000;
    Enames=getElementNames(out(1,i));

    %Falls kein log geschrieben wurde (lo==0)
    if ~any(strcmp(Enames,'logsout'))
    lo=0;
    else
    lo=1;
    end

    %Falls ERROR oder lo==0 Ergebnis = 0
    if isempty(out(1,i).ErrorMessage)==0 || lo==0
        y=0;
        y2=0;
```

```
        Errorall=1;
    else
    a=out(1,i).logsout{1}.Values.Time;
    [ ~, ix ] = min( abs( a-n ) );
    y=out(1,i).logsout{1}.Values.Data;%H2 [kg]
    y=y.*1000;%h2 [g]
    l=y(end,1);
    m=y(ix+1,1);
    y=(l-m)/2;%H2/Zyklus[g/Z]


    y2=out(1,i).logsout{2}.Values.Data;%Energieverbrauch [Wh]
    y2=(y2(end,1)-y2(ix+1,1))/2000;%Energieverbrauch/Zyklus[kWh/Z]
    y2=y/y2;%H2/Energieverbrauch[g/kWh]
    end

    Yh2z=[Yh2z;y];

    Yh2e=[Yh2e;y2];

    y=(y/ztime);%H2/Zeit [g/s]
    y=y.*3600;%H2/Zeit [g/h]
    Yh2s=[Yh2s;y];%H2/Zeit [g/h]


end
format short
end
```

# savedeldouble.m

```
function []=savedeldouble(C,Yh2z,Yh2s,Yh2e,simset)
%Beim vergleich werden die Sets gerundet
 rou=0;

 %Die neuen Daten werden in die jeweiligen Dateien geschrieben,
 %dabei werden dublikate entfern.
    if simset==4
        Mat=load('4P_res');
        Resplus=Mat.Eges;


            if isempty(Resplus)==0


            Cplus=Resplus(:,1:end-3);
            Yh2zplus=Resplus(:,end-2);
            Yh2splus=Resplus(:,end-1);
            Yh2eplus=Resplus(:,end);


            A=round(Cplus,rou);
            B=round(C,rou);

            [~,~,bid] = intersect(A,B,'rows');


            C( bid, : ) = [];
            Yh2z(bid,:)=[];
            Yh2s(bid,:)=[];
            Yh2e(bid,:)=[];



            C=[C;Cplus];
            Yh2z=[Yh2z;Yh2zplus];
            Yh2s=[Yh2s;Yh2splus];
            Yh2e=[Yh2e;Yh2eplus];

            index0=find(Yh2z==0);
            C=removerows(C,'ind',index0);
            Yh2z=removerows(Yh2z,'ind',index0);
            Yh2s=removerows(Yh2s,'ind',index0);
            Yh2e=removerows(Yh2e,'ind',index0);
```

```
        end

Etablezyklus=table(C(:,1),C(:,2),C(:,3),C(:,4),Yh2z);
Etablezeit=table(C(:,1),C(:,2),C(:,3),C(:,4),Yh2s);
Etableenergie=table(C(:,1),C(:,2),C(:,3),C(:,4),Yh2e);

format longG
Eges = [C Yh2z Yh2s Yh2e];
format short

save('4P_res.mat','Eges');
end



if simset==6
    Mat=load('6P_res');
    Resplus=Mat.Eges;


        if isempty(Resplus)==0


        Cplus=Resplus(:,1:end-3);
        Yh2zplus=Resplus(:,end-2);
        Yh2splus=Resplus(:,end-1);
        Yh2eplus=Resplus(:,end);


        A=round(Cplus,rou);
        B=round(C,rou);

        [~,~,bid] = intersect(A,B,'rows');


        C( bid, : ) = [];
        Yh2z(bid,:)=[];
        Yh2s(bid,:)=[];
        Yh2e(bid,:)=[];


        C=[C;Cplus];
        Yh2z=[Yh2z;Yh2zplus];
        Yh2s=[Yh2s;Yh2splus];
        Yh2e=[Yh2e;Yh2eplus];


        index0=find(Yh2z==0);
        C=removerows(C,'ind',index0);
        Yh2z=removerows(Yh2z,'ind',index0);
        Yh2s=removerows(Yh2s,'ind',index0);
        Yh2e=removerows(Yh2e,'ind',index0);




        end

Etablezyklus=table(C(:,1),C(:,2),C(:,3),C(:,4),C(:,5),C(:,6),Yh2z);
Etablezeit=table(C(:,1),C(:,2),C(:,3),C(:,4),C(:,5),C(:,6),Yh2s);
Etableenergie=table(C(:,1),C(:,2),C(:,3),C(:,4),C(:,5),C(:,6),Yh2e);

format longG
Eges = [C Yh2z Yh2s Yh2e];
format short

save('6P_res.mat','Eges');
end



if simset==7
    Mat=load('7P_res');
    Resplus=Mat.Eges;
```

```
            if isempty(Resplus)==0


            Cplus=Resplus(:,1:end-3);
            Yh2zplus=Resplus(:,end-2);
            Yh2splus=Resplus(:,end-1);
            Yh2eplus=Resplus(:,end);


            Cplusanf=round(Cplus(:,1:end-1),rou);
            Cplusende=round(Cplus(:,end),1);
            A=[Cplusanf Cplusende];


            Canf=round(C(:,1:end-1),rou);
            Cende=round(C(:,end),1);
            B=[Canf Cende];




            [~,~,bid] = intersect(A,B,'rows');


            C( bid, : ) = [];
            Yh2z(bid,:)=[];
            Yh2s(bid,:)=[];
            Yh2e(bid,:)=[];



            C=[C;Cplus];
            Yh2z=[Yh2z;Yh2zplus];
            Yh2s=[Yh2s;Yh2splus];
            Yh2e=[Yh2e;Yh2eplus];


            index0=find(Yh2z==0);
            C=removerows(C,'ind',index0);
            Yh2z=removerows(Yh2z,'ind',index0);
            Yh2s=removerows(Yh2s,'ind',index0);
            Yh2e=removerows(Yh2e,'ind',index0);




            end

    format longG
    Eges = [C Yh2z Yh2s Yh2e];
    format short

    save('7P_res.mat','Eges');
    end
```

# optimalnet.m

```
function [net,rmse] = optimalnet(Daten,P)

%Daten mischen
rindx = randperm(size(Daten,1));
Daten = Daten(rindx,:);

[m,n] = size(Daten) ;

%Trainings und testdaten aufteilen
Training = Daten(1:round(P*m),:) ;
Testing = Daten(round(P*m)+1:end,:);

XTrain = Training(:,1:n-1);
YTrain = Training(:,n);
XTest = Testing(:,1:n-1);
YTest = Testing(:,n);
```

```
xtrain = XTrain';
ytrain = YTrain';

[n,m] = size(xtrain);

%Validierungsdaten separieren
xval = xtrain(:,round(P*m)+1:end);
yval = ytrain(:,round(P*m)+1:end);

xtrain = xtrain(:,1:round(P*m)) ;
ytrain = ytrain(:,1:round(P*m)) ;

xtest  = XTest';
ytest  = YTest';

%Hyperparameter definieren
vars = [optimizableVariable('act1',{'purelin','tansig','logsig'},'Type','categorical'),
        optimizableVariable('act2',{'purelin','tansig','logsig'},'Type','categorical'),
        optimizableVariable('act3',{'purelin','tansig','logsig'},'Type','categorical'),
        optimizableVariable('hiddenLayernum', [1,3], 'Type', 'integer')
        optimizableVariable('hiddenLayersize', [10,17], 'Type', 'integer')];

%Optimierungsfunktion
minfn = @(T)netbuild(xtrain,xtest,ytrain,ytest,T.act1,T.act2,T.act3,T.hiddenLayer-
num,T.hiddenLayersize);


results = bayesopt(minfn, vars,'IsObjectiveDeterministic', true,...
    'AcquisitionFunctionName', 'expected-improvement-plus','PlotFcn',[],'UseParal-
lel',true);
T = bestPoint(results);


%Netz aus T bauen
layer=[];
for i =1:T.hiddenLayernum
        layer=[layer T.hiddenLayersize];

end

net = fitnet(layer,'trainlm');


if T.hiddenLayernum==1
net.layers{1}.transferFcn = string(T.act1);
end

if T.hiddenLayernum==2
net.layers{1}.transferFcn = string(T.act1);
net.layers{2}.transferFcn = string(T.act2);
end

if T.hiddenLayernum==3
net.layers{1}.transferFcn = string(T.act1);
net.layers{2}.transferFcn = string(T.act2);
net.layers{3}.transferFcn = string(T.act3);
end


%Netz final trainieren
net = train(net, xtrain,ytrain,'useParallel','no');
% Final test MSE
ypredictcl = net(xtest);
rmse = sqrt(mean((ypredictcl- ytest).^2));


function [rmse]=netbuild(xtrain,xtest,ytrain,ytest,act1,act2,act3,numLay,hidSize)

trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Erstellt ein Netz

layer=[];
for i =1:numLay
        layer=[layer hidSize];
```

```
end


net = fitnet(layer,trainFcn);

if numel(layer)==1
net.layers{1}.transferFcn = string(act1);
end

if numel(layer)==2
net.layers{1}.transferFcn = string(act1);
net.layers{2}.transferFcn = string(act2);
end

if numel(layer)==3
net.layers{1}.transferFcn = string(act1);
net.layers{2}.transferFcn = string(act2);
net.layers{3}.transferFcn = string(act3);
end

% Data Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train
[net,tr] = train(net,xtrain,ytrain);

% Test
ypredictcl = net(xtest);

rmse = sqrt(mean((ypredictcl- ytest).^2));


end
end
```

# Anhang E

Die erzeugte Datenmenge befindet sich auf dem beiliegenden Datenträger.