FH Vorarlberg
University of Applied Sciences

Stochastische Optimierung im Demand Side Management

Masterarbeit
zur Erlangung des akademischen Grades

Master of Science in Engineering

Fachhochschule Vorarlberg
Energietechnik und Energiewirtschaft

Betreut von
Dr. Klaus Rheinberger

Vorgelegt von
Thomas Sohm

Dornbirn, 31. Juli 2015

Kurzreferat

Stochastische Optimierung im Demand Side Management

Aufgrund des stetig steigenden weltweiten Energiebedarfs werden alternative Möglichkeiten zur Energiegewinnung und zur effizienten Nutzung der vorhandenen Energie immer wichtiger. Im Bereich Energiebereitstellung wird durch die Gewinnung regenerativer Energien schon sehr viel getan. Da diese Energie aber nicht stetig sondern meistens nur volatil zur Verfügung steht, müssen auch Verbrauchergeräte effizient betrieben werden. Die vorliegende Arbeit beschäftigt sich daher damit, verschiedene Verbraucher durch stochastische Optimierung, also mittels Wahrscheinlichkeitsberechnung behafteter Optimierung, kostenoptimal zu betreiben. Für die Optimierung wird ein volatiler Strompreis, wie er an der Strombörse gehandelt wird, vorausgesetzt. Da die Verbraucherseite gesteuert wird, nennt sich diese Art der Optimierung auch Demand Side Management (DSM).

Optimiert wurden jeweils über den Zeitraum eines ganzen Jahres die folgenden vier Verbrauchertypen: *verschiebbar – nicht unterbrechbar*, *verschiebbar – unterbrechbar*, *verschiebbar – unterbrechbar mit Wirkungsgrad behaftet* und *verschiebbar – unterbrechbar mit stetigen Verlusten behaftet*. Um die stochastische Optimierung vergleichen zu können, wurden an diesen Verbrauchertypen auch drei deterministische Optimierungen durchgeführt und in eine Ergebnismatrix eingetragen. Dies sind: die Optimierung anhand eines konstanten Strompreises, eines Mittelwertstrompreis über historische Daten und anhand des aktuellen Strompreises am untersuchten Tag.

Als Ergebnis aus den Optimierungsdurchläufen ergab sich, dass der deterministische Mittelwertstrompreis mit den resultierenden Kosten am nächsten an die Optimierungen anhand korrekten Wissens über den Tagesstrompreis heran gekommen ist. Dies kann auf die über alle Tage hinweg gesehenen sehr gleichmäßigen Strompreisverläufe zurückgeführt werden. Werden die Strompreise unregelmäßiger, liefern die stochastischen Optimierungen die besten Resultate.

Abstract

Stochastic optimization in demand side management

Due to the ever increasing global energy consumption, alternative ways of generating energy and the efficient use of the existing energy are becoming more and more important. Currently, a lot of effort is invested in the production of renewable energies. Since this kind of energy is not continuously available but mostly volatile, the efficient operation of consumer devices plays an important role. Therefore, the present thesis addresses the cost-efficient operation of various consumers using stochastic, or, to be more precise, probability-calculation-based optimization. The premise for this optimization is a volatile electricity price, as traded on the power exchange. Since the consumer side is controlled, this kind of optimization is also known as demand side management (DSM).

The following four types of consumer devices were optimized over a period of one year: *deferrable – not interruptible*, *deferrable – interruptible*, *deferrable – interruptible*, *afflicted by an efficiency factor,* and *deferrable – interruptible, afflicted by steady losses*. In order to compare the stochastic optimization, the consumer types were subjected to three deterministic optimizations and then entered into a results matrix. The price basis of the deterministic optimizations were: a constant electricity price, an average electricity price based on historical data, and an optimization based on the price of electricity on the analyzed day.

The conclusion of the optimization runs is that the resulting costs related to the optimization by deterministic average electricity price came closest to the optimizations based on correct knowledge about the daily electricity price. Reason for this is the consistency of the electricity prices throughout the considered days. With varying prices, stochastic optimization delivers the best results.

Vorwort

Im Rahmen des Studiums Energietechnik und Energiewirtschaft an der Fachhochschule Vorarlberg ist es vorgesehen eine Masterarbeit im vierten und damit letzten Semester zu schreiben. Da ich ab dem zweiten Semester die Vertiefungsrichtung Energietechnik besuchte, wollte ich auch ein technisches Thema für diese Arbeit wählen. Ausschlaggebend für die Methodik, welche in dieser Arbeit angewandt wird, war der Kurs „Angewandte Mathematik“ im ersten Semester. In dieser Vorlesung behandelten wir lineare und nichtlineare Optimierungen. Mich faszinierte, was mit Mathematik und entsprechendem Grundgerüst alles möglich ist, und so war mir klar, dass ich in meiner Masterarbeit eine Optimierung behandeln möchte. Ein weiterer Aspekt für die Themenwahl war, dass mich regenerative Energien und deren Einflüsse auf das tagtägliche Leben sehr interessieren.

Mit diesen Wünschen und Anregungen suchte ich das Gespräch mit meinem Studiengangsleiter, der mir nach kurzer Diskussion den Vorschlag der stochastischen Optimierung im Demand Side Management unterbreitete. Mein Interesse war sofort geweckt. Als aus diesem Gespräch ebenso hervorging, dass einer der Gründe für die Schwankungen des Strompreises die Einspeisung von Strom aus regenerativen Energien ist, war mir klar, dass ich dieses Thema wählen muss.

Ich möchte somit einen großen Dank an meinen Studiengangsleiter und Betreuer Dr. Klaus Rheinberger aussprechen, der mir nicht nur mit der Themenwahl ein sehr interessantes Gebiet der Mathematik eröffnete, sondern mich auch während des Verfassens der Arbeit stets tatkräftig unterstützte.

Für die stets liebevolle Unterstützung möchte ich ebenfalls ein Dankeschön an die Administration des Studiengangs richten.

Herzlich bedanken möchte ich mich auch bei Maria Lampert, die sich meiner Arbeit angenommen und Korrektur gelesen hat.

Ein ganz besonderer Dank gilt meiner Freundin Mirjam Jakober, die mich während meiner 2 Jahre Studienzeit stets unterstützt, motiviert und vor allem sehr viel Geduld bewiesen hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Aufgrund des stetig steigenden weltweiten Energiebedarfs werden alternative Möglichkeiten zur Energiegewinnung und zur effizienten Nutzung der vorhandenen Energie immer wichtiger. Im Bereich Energiebereitstellung wird durch die Gewinnung regenerativer Energien schon sehr viel getan. Da diese Energie aber nicht stetig sondern meistens nur volatil zur Verfügung steht, müssen auch Verbraucher, welche in diesem Kontext elektrische Geräte darstellen, effizient betrieben werden. Die der Arbeit zugrunde liegende Motivation ist es daher, verschiedene Verbraucher durch stochastische Optimierung, also mittels Wahrscheinlichkeitsberechnung behafteter Optimierung, kostenoptimal zu betreiben. Da die Verbraucherseite gesteuert wird, nennt sich diese Art der Optimierung auch Demand Side Management (DSM).

In Kapitel 2 werden die Ausgangslage, die Gründe und die Motivation für diese Arbeit beschrieben. Weiters wird erläutert, was DSM bedeutet, und welche finanziellen und systemtechnischen Vorteile es mit sich bringt eine intelligente, verbraucherseitige Steuerung zu verwenden. Abschließend werden die Möglichkeiten durch stochastische Optimierung beschrieben und welche Kosteneinsparungen damit möglich sind.

Material und Methoden, genauer gesagt das deterministische und stochastische Optimierungsverfahren, die vier Verbrauchertypen, die jeweiligen Preisbasen für die Optimierungen und die Strompreisdaten, werden in Kapitel 3 beschrieben. Außerdem wird erklärt, wie die Ergebnisse dargestellt und verglichen werden.

In Kapitel 4 werden Resultate und Varianten der Optimierungsverfahren behandelt, welche anhand der jeweiligen Preisbasis auf die vier Verbrauchertypen angewandt wurden. Dabei wird die Ergebnismatrix gefüllt wodurch ersichtlich wird, welches Optimierungsverfahren für welchen Verbrauchertyp am geeignetsten ist.

Das abschließende Kapitel 5 dient der Diskussion und Zusammenfassung. Die Auswahl der entsprechenden Methoden dieser Arbeit wird begründet und nicht behandelte Aspekte werden erwähnt sowie hinsichtlich ihres möglichen Einflusses auf die Ergebnisse der Optimierung diskutiert.

2. Ausgangslage

Laut einer Fachanalyse aus dem Jahre 2013 „Energy Outlook 2030“ [13] wird prognostiziert, dass der weltweite Energiebedarf von 2013 bis 2030 um 39% steigen wird. Da unsere Energieressourcen beschränkt sind, ist Handlungsbedarf geboten. In der folgenden von BP veröffentlichten Abbildung 1 wird diese Prognose bildhaft dargestellt.

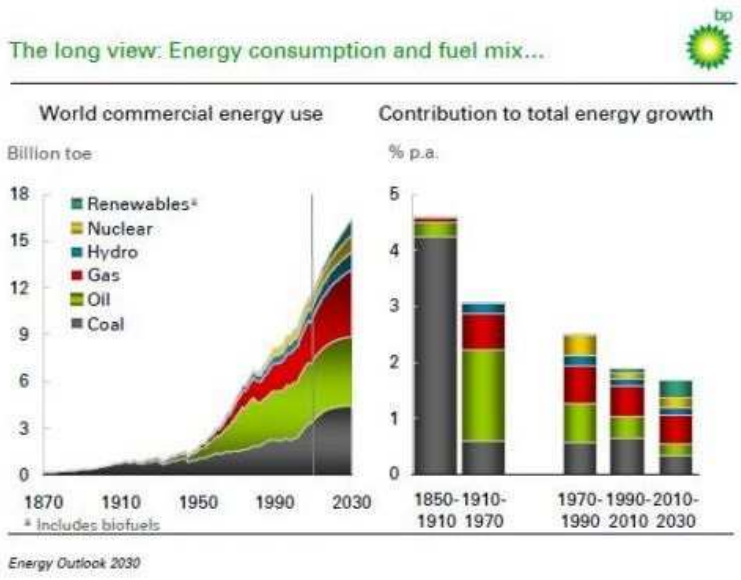

Abbildung 1: Entwicklungsprognose des weltweiten Energiebedarfs bis 2030
Quelle: BP

Energie aus erneuerbaren Quellen wie Solarenergie, Geothermie oder auch Gezeitenkraftwerken stellen heute schon eine gute Alternative zu fossilen oder nuklearen Kraftwerken dar. Daher sollten diese erneuerbaren Quellen weiter entwickelt und ausgebaut werden, um Energie umweltschonender bereitstellen zu können. Ein Nachteil dieser Quellen ist allerdings, dass ihre Energie nicht immer zu gewünschten Zeitpunkten verfügbar ist, denn die Sonne scheint beispielsweise nicht immer genau dann, wenn gerade Energie benötigt wird. Dasselbe gilt auch für den Wind, der nicht immer in dem Moment weht, wann er gebraucht wird, um Strom zu erzeugen.

Daher ist es enorm wichtig die Energie der neuen Energieträger so effizient und kontrolliert wie möglich einzusetzen. Speicherkraftwerke können überschüssige Energie zwischenspeichern und sie dann wieder abgeben, wenn Bedarf vorhanden ist. Leider sind diese Kraftwerke teuer und aufwändig im Bau, beinhalten Verluste und können aufgrund von Geländeeigenschaften nicht immer in den Gebieten errichtet werden, in denen Energie erzeugt wird. Es ist somit leider eine Tatsache, dass elektrische Energie schwer speicherbar ist.

Eine Ergänzung dazu bietet das Demand Side Management (DSM). DSM bedeutet, dass produzierte Energie durch intelligente Steuerung seitens der Verbraucher kosteneffizient eingesetzt werden kann. Es wird somit zum einen die Netzqualität sichergestellt, indem Überschüsse im Netz kompensiert werden, und zum anderen wird der Bedarf an verlustbehafteten Zwischenspeicherungen minimiert.

Das Projekt „LoadShift“ [5], welches von Klima- und Energiefonds und weiteren Projektpartnern gefördert wird, weist bereits darauf hin, welches Potential mittels DSM technisch als auch marktbezogen genutzt werden kann, und auch welche Probleme noch zu bewältigen sind. Es wurden dabei die Bereiche Haushalt, Industrie, Gewerbe und die kommunale Infrastruktur untersucht und ihre Potenziale hinsichtlich Smart Grids analysiert. So wurde aufgezeigt, dass die technische Basis für Varianten wie Peak Clipping, Valley Filling oder Load Shifting größtenteils bereits vorhanden sind. Diese muss aber in eine Steuerung implementiert werden, welche Daten von Verbrauchern benötigt, die zusätzlich ausgewertet werden müssen. Das bedeutet, dass man anhand dieser Daten Rückschlüsse auf das Nutzerverhalten des Menschen ziehen kann. Das macht diese Daten zu sensiblen Daten. Ein weiteres Problem ist verbraucherseitig Anreize setzen zu können, damit die Möglichkeit der Lastverschiebung auch kundenseitig eindeutig als positiv und gewinnbringend angenommen und verstanden werden kann.

Die Arbeit „Demand Side management: Benefits and challenges“ von Goran Strbac [4] weist ebenfalls auf die Potentiale des DSM hin. Es wird beschrieben, dass in mehreren Bereichen wie z.B. bei der Erzeugung von Energie, speziell in Bezug auf erneuerbare Energiequellen, in der Transportebene, in Verteilernetzwerken als auch direkt beim Endkunden kostensparende Vorteile geschaffen werden können.

Mittels stochastischer Optimierung können nicht eindeutig vorhersehbare Veränderungen von für ein System wichtigen Kennwerten berücksichtigt werden. Beispiele für diese Volatilitäten sind das Wetter und damit verbunden auch der Strompreis oder der natürliche Zufluss des Wassers in ein Staubecken eines Pumpspeicherkraftwerks. Ein weiteres Beispiel ist die Verbrauchercharakteristik, welche nicht eindeutig vorhersehbar ist, aber Einfluss auf die zu erzeugende Menge an Energie und auch auf die Netzqualität hat. In der Ausgabe „Energie 2.0 Kompendium 2014“ des Publish-industry Verlags [6] wird beschrieben, welchen Mehrwert die stochastische Optimierung in Bezug auf die Effizienz hat. Es wird aufgezeigt, dass die Berücksichtigung dieser Unsicherheiten in der Optimierung eine Verbesserung von bis zu 7% erreichen kann.

Deterministische Optimierungsverfahren basieren auf der Kenntnis entsprechender Kenngrößen, welche im vorliegenden Fall Strompreise sind. Stochastische

Optimierungsverfahren beruhen darauf, durch historische Kennwerte und Wahrscheinlichkeitsrechnung den zukünftigen Stromverlauf so gut wie möglich voraus zu sehen und anhand diesem in jedem neuen Zeitintervall wieder neu zu entscheiden. Diese beiden Optimierungsverfahren werden an verschiedenen Verbrauchertypen angewandt. Mit den erhobenen Daten wird anschließend eine Matrix befüllt, bei der die Zeilen die Verbrauchertypen und die Spalten die Preisbasen der Optimierungsverfahren darstellen. Mit dieser Matrix soll zudem gezeigt werden, dass die Stochastik einen messbaren Mehrwert gegenüber heuristischen Algorithmen liefert. Zusätzlich sollte sie bessere Ergebnisse als eine deterministische Optimierung über ein Erwartungswertszenario liefern. Dies wird auch „Expected value of perfect information“ genannt.

In vorliegender Arbeit soll somit aufgezeigt werden, wie groß bei verschiedenen elektrischen Verbrauchertypen und konstantem oder variablem Strompreis der Mehrwert einer stochastischen Optimierung im Vergleich zu einem deterministischen Optimierungsverfahren ist.

3. Material und Methoden

In dieser Arbeit werden drei deterministische Optimierungsverfahren mit stochastischen Optimierungsverfahren verglichen. Diese werden auf vier verschiedene Verbrauchertypen angewandt. Zum Modellieren der Optimierungsverfahren und der Verbrauchertypen wird die Programmiersprache Python in der Umgebung Canopy verwendet.

Eine prinzipielle Annahme in Bezug auf den Strompreis ist, dass dem Endnutzer ein volatiler Strompreis vom Stromlieferanten weitergegeben wird. Da dieser Preis ein schwankender, nicht vorhersehbarer Strompreis ist, wird die Stochastik auf den Strompreis bezogen. Die Menge an verbrauchter Energie wird als konstant angenommen, damit nur ein unsicherer Einfluss auf die Optimierung einwirkt. Sie wird zudem für alle Optimierungsverfahren als gleich groß angenommen, um die Ergebnisse vergleichen zu können.

In diesem Kapitel werden somit die Methode der Stochastik, den „Expected value of perfect information“, die verwendeten Strompreis-Daten, die Verbrauchertypen, die verwendete jeweilige Preisbasis der Optimierungsverfahren und die Ergebnismatrix vorgestellt.

3.1 Deterministische Optimierung allgemein

Die deterministische, lineare Optimierung basiert darauf alle Kennwerte des Optimierungsproblems zu kennen und daraufhin die Entscheidungsvariablen x so zu wählen, dass die Zielfunktion minimiert wird. Die folgenden Formeln beschreiben die grundsätzliche Form des Optimierungsverfahren, die „General Form“, die in dieser Arbeit verwendet wird.

$$\min. c^T \mathrm{x} \tag{1}$$

$$s.t. Ax \leq b \tag{2}$$

$$Gx = h \tag{3}$$

Formel (1) beschreibt die Zielfunktion, welche die Strompreise c und die Entscheidungsvariablen x jeweils als Vektoren enthält. Schematisch kann dies wie in Abbildung 2 dargestellt werden.

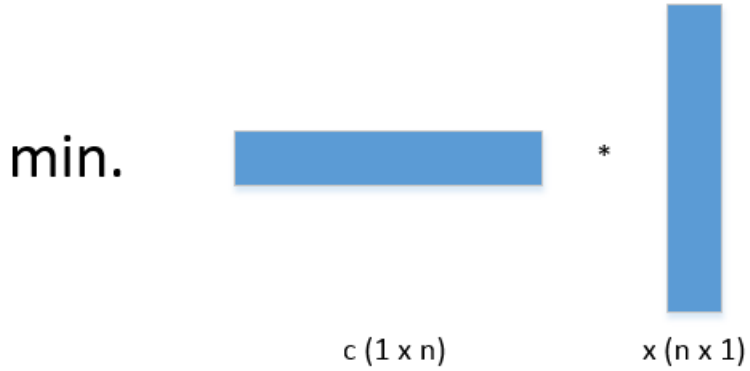

Abbildung 2: Schematische Darstellung der Zielfunktion
Quelle: Eigene Ausarbeitung

Formel (2) beschreibt die Ungleichheitsform der Nebenbedingungen. Dabei stellt A eine Matrix und x wiederum die Entscheidungsvariablen in Form eines Vektors dar. Die Matrix A beinhaltet die Faktoren, mit denen die Entscheidungsvariablen gewichtet werden. Auf der rechten Seite der Ungleichung stehen im Vektor b die Grenzwerte. Diese Ungleichung wird in Abbildung 3 schematisch dargestellt.

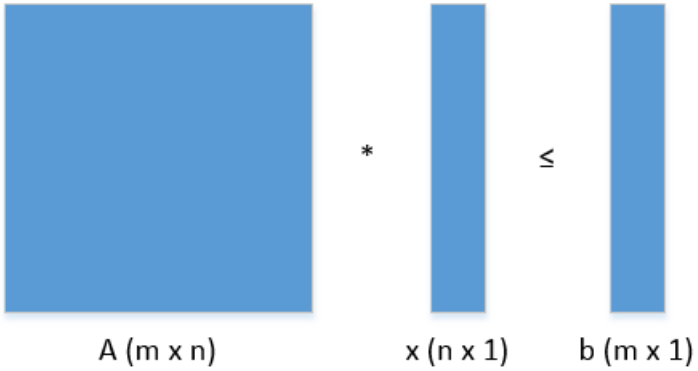

Abbildung 3: Schematische Darstellung der Ungleichheitsform der Nebenbedingungen
Quelle: Eigene Ausarbeitung

Hat eine Zeile die Form „≥“, so muss sie einfach mit -1 multipliziert werden, um auf die Form „≤“ gebracht zu werden.

Formel (3) stellt die Gleichheitsform der Nebenbedingungen dar. Dabei beschreibt G eine Matrix und x wiederum die Entscheidungsvariablen in Form eines Vektors. Die Matrix G beinhaltet die Faktoren, mit denen die Entscheidungsvariablen gewichtet werden. Auf der rechten Seite der Gleichung stehen mit h die Grenzwerte, welche ebenfalls in Form eines Vektors zu sehen sind. In Abbildung 4 wird dies schematisch dargestellt.

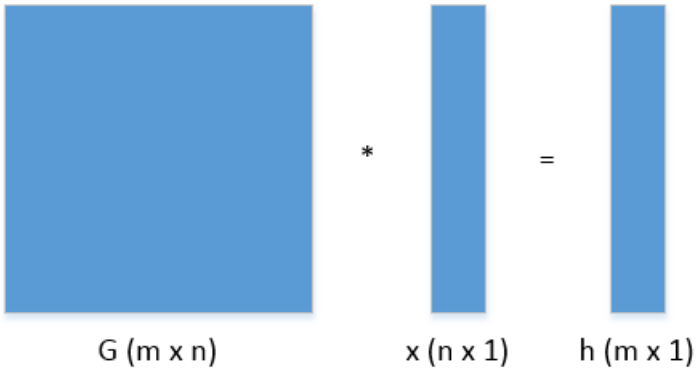

Abbildung 4: Schematische Darstellung der Gleichheitsform der Nebenbedingungen
Quelle: Eigene Ausarbeitung

3.2 Stochastische Optimierung allgemein

Grundsätzlich basiert die stochastische Optimierungsmethode auf derselben Form wie die deterministische. Es wird ebenfalls die „General Form“ aus Punkt 3.1 angewandt. Der Unterschied liegt darin, dass die Entwicklung eines bestimmten Kennwertes, in unserem Fall des Strompreises, nicht bekannt ist. Bekannt sind nur die Preise bis zum jeweiligen Zeitpunkt. Daher muss zu jedem Zeitpunkt eine Bestandsaufnahme durchgeführt und anhand bekannter, historischer Werte die zukünftigen Strompreise so gut wie möglich abgeschätzt werden. Diese Optimierungsmethode unterteilt sich daher in zwei Teile: in alles, was bekannt ist, und alles, was unbekannt ist. Dies nennt sich Two-Stage Modell und wird unter Punkt 3.2.1 genauer erläutert. Eine Möglichkeit zukünftige Strompreise abschätzen zu können liefert das „Nearest Neighbourhood Verfahren“, welches unter Punkt 3.2.2 genauer beschrieben wird. Um schlussendlich abschätzen zu können, wie gut die Stochastik gegenüber dem perfekten Wissen ist, also ob der Strompreisverlauf des Tages bereits im Vorhinein bekannt ist, wird der „Expected value of perfect information“ errechnet. Dies wird unter Punkt 3.2.4 genauer erläutert.

3.2.1 Two-Stage Modell

Die allgemeine Form des Two-Stage Modells [7] wird in der folgenden Formel(4) dargestellt.

$$\min. c^T \mathrm{x} + E_\xi Q(x, \xi) \tag{4}$$

Der Teil vor dem Additionszeichen beschreibt die „first stage decision“, also jene Entscheidungen, welche anhand bekannter Kennwerte gemacht werden können. Der Teil hinter dem Additionszeichen beschreibt die „second stage decision“, also jene

Entscheidungen, die anhand von Wahrscheinlichkeiten E_ξ getroffen werden. ξ stellt eine Zufallsvariable der Szenarien dar, welche im vorliegenden Fall vom Nearest Neighbourhood Verfahren abhängig ist, weil die Szenarien anhand der errechneten Wahrscheinlichkeiten ausgewählt werden. Q(x,ξ) definiert eine separate Minimierungsfunktion, welche die Optimierung der „second stage" übernimmt. In der Umsetzung des Codes wird die extensive Form der stochastischen Methode verwendet, was bedeutet, dass alle Szenarien gewichtet mit ihren Wahrscheinlichkeiten an die Minimierungsfunktion der „first stage" angehängt werden, und daher nur diese eine Minimierung implementiert werden muss. Die Matrizen und Vektoren werden dadurch allerdings um die Anzahl der Szenarien größer.

3.2.2 Nearest Neighbourhood Verfahren

Nearest Neigbourhood (knn) ist ein Verfahren, um die zukünftige Entwicklung eines Verlaufes, in diesem Fall des Strompreises, anhand bereits bekannter Werte so gut wie möglich voraus zu sehen. Die bereits bekannten Werte stellen im vorliegenden Kontext die vergangenen Stunden eines Tages dar, d.h. zu Beginn des Tages ist ein Wert bekannt und mit jeder vergangenen Stunde kommt ein neuer hinzu. Es werden nun aus historischen Daten die k nächsten Nachbarn gesucht. Um die nächsten Nachbarn bestimmen zu können, ist eine Methode erforderlich, um den Abstand zwischen zwei Punkten zu ermitteln. Eine Möglichkeit den geringsten Abstand zwischen zwei Punkten X und Y zu ermitteln ist die euklidische Distanz [11]. Anhand der folgenden Formel (5) ist dies dargestellt.

$$\mathrm{Dist}_j(X_i, Y_i) = \sqrt{\sum (X_i - Y_i)^2} \tag{5}$$

Um nun der geringsten Distanz die höchste Wahrscheinlichkeit zuordnen zu können, wird von jeder Distanz Dist_j der Kehrwert ($\frac{1}{Dist_j}$) gebildet. Anschließend wird jeder Kehrwert durch die Summe aller Kehrwerte dividiert um die Ergebnisse zu normieren und damit eine Wahrscheinlichkeit zu bekommen. Die Distanz wird zusätzlich mit „+1" addiert, um eine Division durch null zu verhindern. Dies wird in Formel (6) nochmals dargestellt.

$$P_j = \frac{\frac{1}{1 + Dist_j}}{\sum_i \frac{1}{1 + Dist_i}} \tag{6}$$

Somit können anhand der bereits bekannten Strompreise und des knn-Verfahrens, wenn beispielsweise k = 3 ist, die wahrscheinlichkeitsmäßig 3 nächsten Strompreisverläufe aus den historischen Verläufen herausgefiltert werden.

3.2.3 Bildung von Szenarien

Um die Szenarien für die stochastische Optimierung zu generieren, gibt es zwei Methoden. Zum einen die Bildung eines Szenarienbüschels und zum anderen den Szenarienbaum. Im Folgenden werden beide erläutert.

3.2.3.1 Szenarienbüschel

Ausgehend von einem bekannten Startwert zum Zeitpunkt t_1 werden mehrere Szenarien generiert. Dies geschieht mittels des knn-Verfahrens, welches die k-nächsten Verläufe findet. Daraus ergibt sich ein Büschel mit k-Verläufen. Abbildung 5 stellt dies bildhaft dar. Dabei stellen t_i die Zeitschritte in Stunden dar.

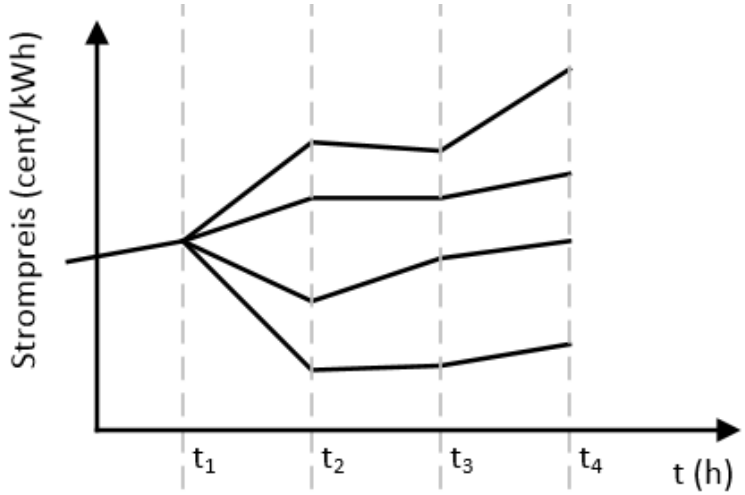

Abbildung 5: Szenarienbüschel
Quelle: Eigene Ausarbeitung

3.2.3.2 Szenarienbaum

Ausgehend von einem bekannten Startwert werden mehrere Szenarien generiert. Diese verzweigen sich nach einer bestimmten Anzahl an Zeitschritten erneut, wobei die Anzahl an jeder neuen Verzeigung vom knn-Verfahren abhängt. Dabei entstehen k neue Verzweigungen, wie in Abbildung 6 bildhaft dargestellt. t_i stellen die Zeitschritte in Stunden dar.

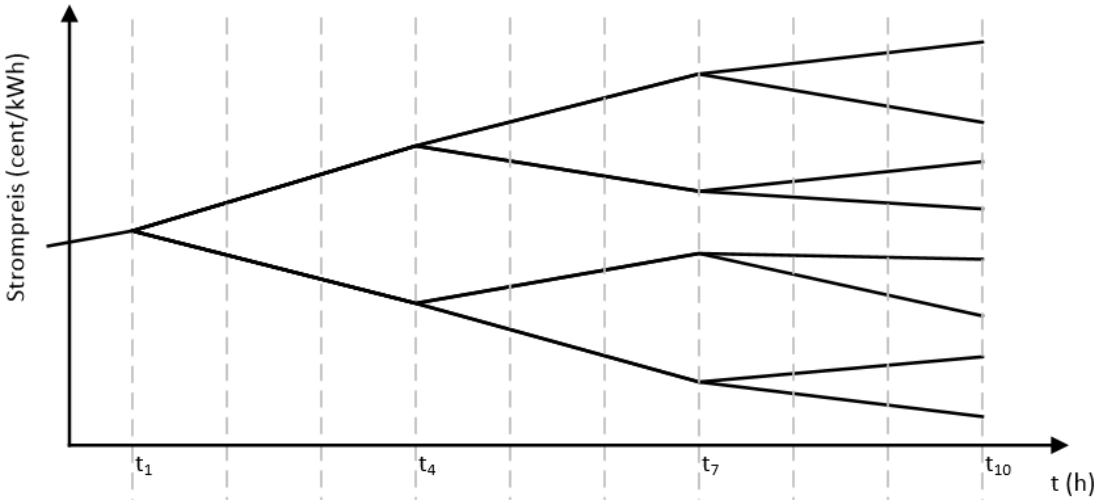

Abbildung 6: Szenarienbaum
Quelle: Eigene Ausarbeitung

3.2.4 Expected value of perfect information

Der Wert „Expected value of perfect information“ (EVPI) beschreibt wie gut die stochastische Optimierung im Vergleich zum bestmöglichen Ergebnis, also wenn die Strompreiskurve zu Beginn der Optimierung bereits bekannt gewesen wäre, abgeschnitten hat. Dabei wird die Differenz zwischen dem stochastischen Optimierungsergebnis und den Kosten aus der Optimierung mit völlig bekannten Strompreisen berechnet. Je kleiner dieser Wert ist, desto näher kommt die Stochastik an die optimalen, geringsten Kosten heran.

3.3 Strompreis Daten

Die Strompreisdaten, die für die Optimierungsverfahren verwendet wurden, stammen einerseits von European Energy Exchange AG (EEX) welche den größten Handelsplatz für in Deutschland produzierten Strom darstellt, und andererseits von European Power Exchange (EPEX) welcher der zentraleuropäische Spotmarkt für Energie ist. Dabei werden zwei Arten von Preistypen an der Strombörse gehandelt: One-Day-Ahead Preise und Intraday Preise [12].

3.3.1 One-Day-Ahead Preis

Der One-Day-Ahead Preishandel, welcher auch Auktionshandel genannt wird, beschreibt den Handel von Strom an der EPEX in Paris für den folgenden Tag. Im Markt Deutschland/Österreich können Gebote bis 12:00 Uhr abgegeben werden. Die Zuschläge werden um 12:40 verkündet. Nachfolgend beginnt ab 15:00 Uhr der Intraday Handel für den nächsten Tag.

Mit dem One-Day-Ahead Preis ist somit die vollständige Preiskurve für den nächsten Tag bekannt. Die in dieser Arbeit verwendeten One-Day-Ahead Preisdaten stammen aus der EEX Datenbank.

3.3.2 Intraday Preis

Ist der Auktionshandel abgeschlossen, beginnt der Intradayhandel, bei dem Preise bis 45 min vor Lieferung gehandelt werden können. Dieser Handel ist notwendig, um kurzfristige Lastschwankungen und damit kurzfristige Lieferengpässe oder Energieüberschüsse von Stromlieferanten ausgleichen zu können. Dieser Preis kann in 15-Minuten- oder in Stundenblöcken gehandelt werden.

Beim Intradaypreis wird zu jedem neuen Zeitschritt der aktuelle Preis bekannt gegeben. Die in dieser Arbeit verwendeten Intraday Preisdaten stammen aus der EPEX Datenbank.

3.4 Verbrauchertypen

3.4.1 Verschiebbare, nicht unterbrechbare Verbraucher – Typ A

Dieser Verbrauchertyp wird innerhalb von 24 Stunden zu einem bestimmten Zeitpunkt gestartet und muss die benötigte Zeit zur Fertigstellung seiner Aufgabe durchgehend betrieben werden. Beispiele hierfür sind die Waschmaschine oder der Geschirrspüler. Abbildung 7 stellt diesen Verbrauchertyp schematisch dar.

Abbildung 7: Verschiebbarer, nicht unterbrechbarer Verbraucher
Quelle: [3]

3.4.1.1 Deterministische Modellierung – Typ A

In der deterministischen Modellierung wird jeder Zeitschritt daraufhin untersucht, welcher als Startzeitpunkt, in Summe mit den folgenden benötigten Perioden um die Energiemenge zu erreichen, der kostengünstigste ist. Zu jedem Zeitschritt darf zusätzlich nur eine bestimmte Menge an Energie verbraucht werden.
In Abbildung 8 ist der Codeausschnitt zu sehen, der dies modelliert. Dabei stellt „len_day“ die Länge des Tages in Stunden, „period“ die benötigten Perioden, um die Energiemenge zu erreichen, „prices“ die Kosten zur jeweiligen Stunde und „powers“ die Verbrauchsentscheidungen dar.

```
len_day = len(price_real) # Stunden innerhalb eines Tages

if verbose:
    print('.', end="")
costs = np.zeros(len_day - period)
for t in range(len_day - period):
    costs[t] = sum(prices[t:t + period])*power
time = np.argmin(costs)
powers = np.zeros(len_day)
powers[time:time + period] = power
cost_optimal = costs[time]
#print (cost_optimal)

cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten
```

Abbildung 8: Codeausschnitt deterministische Modellierung Typ A
Quelle: Eigene Ausarbeitung

3.4.1.2 Stochastische Modellierung – Typ A

Die stochastische Modellierung basiert auf der deterministischen von Typ A. Es wird wiederum jeder Zeitschritt daraufhin untersucht, welcher in Summe der günstigste ist, den Verbraucher zu starten. Da der Strompreis in zukünftigen Schritten aber nicht bekannt ist, wird anhand des knn-Verfahrens die wahrscheinliche Preisentwicklung bestimmt. Solange entschieden wird den Verbraucher nicht zu starten, wird dies zu jedem Zeitschritt erneut durchgeführt. Wird der Verbraucher gestartet, bricht die Schleife ab und der aktuelle Zeitpunkt wird für die Kostenberechnung heran gezogen.

In Abbildung 9 ist der Codeausschnitt zu sehen, der dies modelliert. Dabei stellen zusätzlich zu den bereites erklärten Beschriftungen „knn_numb“ die Anzahl der nächsten Nachbarn und „search_aktiv“ das Finden eines Startzeitpunkts dar. Mit der „if“-Bedingung unterhalb des Kommentars „# Stundenindexabhänger Preis bestimmen“ wird ausgewählt, welche knn-Methode verwendet wird. Die timeline-Methode wird unter Punkt 4.4.2 noch genauer erläutert.

```
price_real = price_sel[day_index] # Preisverlauf des aktuellen Tages
len_day = len(price_real) # Stunden innerhalb eines Tages
knn_numb = 3 # Anzahl der nächsten Nachbarn, die herangezogen werden
costs = np.zeros(len_day - period)
search_aktiv = True
time_line_OFF = True # Auswahl knn-Methode

if verbose:
        print ('.', end="")

for i in range(len_day - period):
    if search_aktiv:

        # Stundenindexabhänger Preis bestimmen
        if time_line_OFF == True:
            price_moment, knn_price_array = knn_probability_fast(knn_numb, i+1, price_history, price_real)
        else:
            price_real_prev = price_sel[day_index - 1]
            price_moment, knn_price_array = knn_probability_timeline(knn_numb, i+1, price_history, price_real, price_real_prev)

        knn_number_szenarios= knn_price_array.shape[0]
        times = np.zeros(knn_number_szenarios)

        for j in range(knn_number_szenarios):

            price_moment = price_history[knn_price_array[j][0]].copy()
            price_moment[0:i+1] = price_real[0:i+1].copy()

            for t in range(len_day - period):
                costs[t] = sum(price_moment[t:t + period])*power
            time_loop = np.argmin(costs)
            if time_loop <= i:
                times[j] = 1 * knn_price_array[j][2] # Einschalteentscheidung der jeweiligen Szenarien

        if np.sum(times) >= 0.5: # Ist die Szenarien gewichtete Entscheidung größer 50% --> Verbraucher starten
            search_aktiv = False
            powers = np.zeros(len_day)
            powers[i:i + period] = power


cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten
```

Abbildung 9: Codeausschnitt stochastische Modellierung Typ A
Quelle: Eigene Ausarbeitung

3.4.2 Verschiebbare, unterbrechbare Verbraucher – Typ B

Dieser Verbrauchertyp wird innerhalb von 24 Stunden zu einem bestimmten Zeitpunkt eingeschaltet, kann aber unterbrechen, um die Arbeit später fertig zu stellen. Ein Beispiel hierfür ist eine Industriemaschine, welche Produkte bearbeitet, die auch längere Zeit lagern können ohne Schaden zu nehmen. Abbildung 10 stellt diesen Verbrauchertyp schematisch dar.

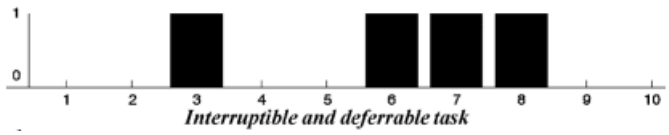

Abbildung 10: Verschiebbarer, unterbrechbarer Verbraucher
Quelle: [3]

3.4.2.1 Deterministische Modellierung – Typ B

Die deterministische Modellierung beschreibt ein lineares Optimierungsproblem, in dem die geforderte Energiemenge innerhalb von 24 Stunden konsumiert werden muss, und dies möglichst zu minimalen Kosten. Zusätzlich darf pro Zeitschritt nur eine begrenzte Menge an Energie verbraucht werden. Folgend ist das lineare Optimierungsproblem beschrieben.

$$\min. \sum_{t=1}^{24} c_t * E_t \tag{7}$$

$$s.t. 0 \leq E_t \leq E_{max} \tag{8}$$

$$\sum_{t=1}^{24} E_t = E_{gesamt} \tag{9}$$

Formel (7) beschreibt die Zielfunktion, die minimiert werden soll. Dabei sind c_t die Kosten zum jeweiligen Zeitschritt und E_t die Energiemenge, die dabei konsumiert wird. E_t stellt somit die Entscheidungsvariable dar. Formel (8) definiert, dass pro Zeitschritt nur eine bestimmte Energiemenge E_{max} konsumiert werden kann und Formel (9) besagt, dass die geforderte Gesamtenergiemenge E_{gesamt} innerhalb von 24 Stunden konsumiert werden muss.

In Abbildung 11 ist der Codeausschnitt zu sehen, der dieses Optimierungsproblem modelliert. Dabei stellt „prices“ die Kosten zur jeweiligen Stunde, „power“ die maximale Energie pro Zeitschritt, „period“ die benötigten Perioden um die Energiemenge zu erreichen und somit „power*period“ die Gesamtenergiemenge dar. Die Matrix A wird durch die beiden Matrizen A1 und A2 und der Vektor b durch die beiden Vektoren b1 und b2 erstellt. Durch G und h wird die obige Formel (9) beschrieben. Sind die Zielfunktion und die Nebenbedingungen definiert, werden die optimalen Verbrauchsentscheidungen x anhand des Solvers berechnet.

```
# Zusammenstellen der Zielfunktion und der Nebenbedingungen
c  = prices
A1 = np.eye(len(prices))
b1 = np.ones(len(prices))*power
A2 = -np.eye(len(prices))
b2 = np.zeros(len(prices))
A  = np.vstack((A1,A2))
b  = np.concatenate((b1,b2))
G  = np.ones((1,len(prices)))
h  = power*period

[x,s,z,y,status] = lp_solve(c,A,b,G,h)
# print("Solver status = %s" %status)
powers = x
cost_optimal = np.dot(c.T,x)[0] #Berechnung der optimal möglichen Kosten

cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten
```

Abbildung 11: Codeausschnitt deterministische Modellierung Typ B
Quelle: Eigene Ausarbeitung

3.4.2.2 Stochastische Modellierung – Typ B

Die stochastische Modellierung basiert auf der deterministischen von Typ B. Die Zielfunktion und die Nebenbedingungen bleiben im Grunde die gleichen. Da der Strompreis in zukünftigen Schritten aber nicht bekannt ist, wird anhand des knn-Verfahrens die wahrscheinliche Preisentwicklung zu jedem Zeitschritt neu bestimmt. Daher werden auch die Zielfunktion und die Nebenbedingungen zu jeder Stunde neu formuliert, unter Berücksichtigung der bereits verbrauchten Energiemenge, was mittels „energy_used“ realisiert wird. Anschließend wird die Optimierung vom aktuellen Zeitpunkt bis zum Ende durchlaufen, um die optimale Entscheidung zum aktuellen Zeitpunkt zu erhalten. In Abbildung 12 ist der Codeausschnitt zu sehen, der dieses Modell beschreibt. Mit der „for“-Schleife zu Beginn, wird die Preisfunktion anhand der Szenarien aufgebaut. Mit der „for“-Schleife, welche die Variablen G und h umschließt, werden die Nebenbedingungen anhand der Szenarien gebildet.

```
knn_numb = 3 #Anzahl der nahesten Nachbarn, die herangezogen werden
price_real = price_sel[day_index] # Preisverlauf des aktuellen Tages
len_day = len(price_real) # Stunden innerhalb eines Tages
energy_used = 0
powers = np.zeros(len_day)
time_line_OFF = True # Auswahl knn-Methode

if verbose:
        print('.', end="")

for i in range(len_day-1):

    # Stundenindexabhänger Preis bestimmen
    if time_line_OFF == True:
        price_moment, knn_price_array = knn_probability_fast(knn_numb, i+1, price_history, price_real)
    else:
        price_real_prev = price_sel[day_index - 1]
        price_moment, knn_price_array = knn_probability_timeline(knn_numb, i+1, price_history, price_real, price_real_prev)

    knn_number_szenarios= knn_price_array.shape[0]

    # Zusammenstellen der Zielfunktion und der Nebenbedingungen
    c  = np.array([price_real[i]])
    for j in range(knn_number_szenarios):
        c_temp = price_history[knn_price_array[j][0]]
        weight_factor = knn_price_array[j][2]
        c_moment = c_temp[i+1:len_day] * weight_factor
        c = np.concatenate((c,c_moment))

    A1 = np.eye(c.size)   # oder np.eye(1+((len_day-(i+1))*knn_numb))
    b1 = np.ones(c.size)*power
    A2 = -np.eye(c.size)
    b2 = np.zeros(c.size)
    A  = np.vstack((A1,A2))
    b  = np.concatenate((b1,b2))

    step_size = c_moment.size # Länge eines Szenarios in Stunden bis zum Ende des Tages ohne den bekannten Wert
    for j in range(knn_number_szenarios): # Für G: 1er an die Stellen setzen, die für die jeweiligen Szenarios benötigt werden
                            # Für h: Den Vektor für alle Szenarien mit dem Restenergiebeadarf füllen
        G_temp = np.zeros((1,c.size))
        G_temp[0][0] = 1
        G_temp[0][1+j*step_size:1+(j+1)*step_size] = 1
        h_temp = power*period-energy_used
        if j == 0:
            G = G_temp
            h = h_temp
        else:
            G = np.vstack((G,G_temp))
            h = np.vstack((h,h_temp))

    [x,s,z,y,status] = lp_solve(c,A,b,G,h)
    #print("Solver status = %s" %status)

    energy_used += round(x[0],3)  # Aufsummieren der Energieverbräuche, damit für die folgenden Optimierungsdurchläufe
                                  # der erlaubte Restverbrauch angepasst wird. x entspricht der Verbrauch im aktuellen Schritt
    powers[i] = x[0] # Verbrauchsentscheidung des aktuellen Optimierungsdurchlauf zuweisen

powers[len_day-1] = power*period-energy_used

cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten
```

Abbildung 12: Codeausschnitt stochastische Modellierung Typ B
Quelle: Eigene Ausarbeitung

3.4.3 Verschiebbare, unterbrechbare Verbraucher mit Wirkungsgrad – Typ C

Dieser Verbrauchertyp ist grundsätzlich der gleiche wie Typ B. Daher gilt auch hier die schematische Darstellung aus Punkt 3.4.2. Er unterscheidet sich nur darin, dass die konsumierte Energiemenge zum aktuellen Zeitpunkt, also die Entscheidungsvariable, nicht zu 100% umgesetzt werden kann und daher mit einem Wirkungsgrad behaftet ist. Ein Beispiel hierfür wäre ein Pumpspeicherkraftwerk. Die gewünschte Nettogesamtendenergiemenge, welche innerhalb von 24 Stunden konsumiert werden soll, bleibt aber dieselbe.

3.4.3.1 Deterministische Modellierung – Typ C

Das Optimierungsproblem, welches gegenüber Typ B nur durch den Wirkungsgrad ergänzt wurde, sieht nun wie folgt aus.

$$\min. \sum_{t=1}^{24} c_t * E_t \tag{10}$$

$$s.t. 0 \leq E_t \leq E_{max} \tag{11}$$

$$\sum_{t=1}^{24} E_t \eta = E_{gesamt} \tag{12}$$

Die Änderung gegenüber Typ B ist in Formel (12) zu sehen. Es wird die eingesetzte Energie zum aktuellen Zeitpunkt mit dem Wirkungsgrad η multipliziert. Daher können nicht 100% der Energiemenge E_t umgesetzt werden.

In Abbildung 13 ist der Codeausschnitt zu sehen, der dieses Optimierungsproblem modelliert. Die Änderung gegenüber Typ B ist in der Zeile zu sehen, in der h definiert wird. Der Wirkungsgrad wurde nur auf die andere Seite der Gleichung dividiert.

```
# Zusammenstellen der Zielfunktion und der Nebenbedingungen
c  = prices
A1 = np.eye(len(prices))
b1 = np.ones(len(prices))*power_load_max
A2 = -np.eye(len(prices))
b2 = np.zeros(len(prices))
A  = np.vstack((A1,A2))
b  = np.concatenate((b1,b2))
G  = np.ones((1,len(prices)))
h  = energy_provided/efficiency

[x,s,z,y,status] = lp_solve(c,A,b,G,h)
# print("Solver status = %s" %status)
powers = x
cost_optimal = np.dot(c.T,x)[0] #Berechnung der optimal möglichen Kosten

cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten
```

Abbildung 13: Codeausschnitt deterministische Modellierung Typ C
Quelle: Eigene Ausarbeitung

3.4.3.2 Stochastische Modellierung – Typ C

Die stochastische Modellierung basiert auf der deterministischen von Typ C und ähnelt der stochastischen Modellierung von Typ B. Die einzige Änderung ist wieder die ergänzte Multiplikation mit dem Wirkungsgrad η. In Abbildung 14 ist der Codeausschnitt zu sehen, der dieses Modell beschreibt.

```
knn_numb = 10 #Anzahl der nahesten Nachbarn, die herangezogen werden
price_real = price_sel[day_index] # Preisverlauf des aktuellen Tages
len_day = len(price_real) # Stunden innerhalb eines Tages
energy_used = 0
powers = np.zeros(len_day)
time_line_OFF = True # Auswahl knn-Methode

if verbose:
        print('.', end="")

for i in range(len_day-1):

    # Stundenindexabhänger Preis bestimmen
    if time_line_OFF == True:
        price_moment, knn_price_array = knn_probability_fast(knn_numb, i+1, price_history, price_real)
    else:
        price_real_prev = price_sel[day_index - 1]
        price_moment, knn_price_array = knn_probability_timeline(knn_numb, i+1, price_history, price_real, price_real_prev)

    knn_number_szenarios= knn_price_array.shape[0]

    # Zusammenstellen der Zielfunktion und der Nebenbedingungen
    c  = np.array([price_real[i]])
    for j in range(knn_number_szenarios):
        c_temp = price_history[knn_price_array[j][0]]
        weight_factor = knn_price_array[j][2]
        c_moment = c_temp[i+1:len_day] * weight_factor
        c = np.concatenate((c,c_moment))

    A1 = np.eye(c.size)
    b1 = np.ones(c.size)*power_load_max
    A2 = -np.eye(c.size)
    b2 = np.zeros(c.size)
    A  = np.vstack((A1,A2))
    b  = np.concatenate((b1,b2))

    step_size = c_moment.size # Länge eines Szenarios in Stunden bis zum Ende des Tages ohne den bekannten Wert
    for j in range(knn_number_szenarios): # Für G: Ier an die Stellen setzen, die für die jeweiligen Szenarios benötigt werden
                # Für h: Den Vektor für alle Szenarien mit dem Restenergiebeadarf füllen
        G_temp = np.zeros((1,c.size))
        G_temp[0][0] = 1
        G_temp[0][1+j*step_size:1+(j+1)*step_size] = 1
        h_temp = energy_provided/efficiency-energy_used
        if j == 0:
            G = G_temp
            h = h_temp
        else:
            G = np.vstack((G,G_temp))
            h = np.vstack((h,h_temp))

    [x,s,z,y,status] = lp_solve(c,A,b,G,h)
    #print("Solver status = %s" %status)

    energy_used += round(x[0],3)  # Aufsummieren der Energieverbräuche, damit für die folgenden Optimierungsdurchläufe
                                  # der erlaubte Restverbruach angepasst wird. x entspricht der Verbrauch im aktuellen Schritt
    #print(energy_used)
    powers[i] = x[0] # Verbrauchsentscheidung des aktuellen Optimierungsdurchlauf zuweisen

powers[len_day-1] = energy_provided/efficiency-energy_used

cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten
```

Abbildung 14: Codeausschnitt stochastische Modellierung Typ C
Quelle: Eigene Ausarbeitung

3.4.4 Verschiebbare, unterbrechbare Verbraucher mit Verlusten – Typ D

Der Verbrauchertyp D ist mit stetigen Verlusten behaftet. Beispiel hierfür ist ein Boiler, dessen Wasser auf Umgebungstemperatur abkühlt, sobald keine Energie mehr investiert wird. Wie schnell diese Abkühlung voranschreitet, hängt von der Materialkonstante der Isolierung ab. Die Eigenschaften dieses Verbrauchertyps basieren grundsätzlich auf dem Newton'schen Abkühlungsgesetz. Folgend wird dieser Zusammenhang gezeigt.

$$c_1 m\dot{T}_{Wasser}(t) = P - c_2\left(T_{Wasser}(t) - T_{Umgebung}\right)$$
$$mit\ c_1 = \frac{4{,}182\ kJ}{3600\ kg * K} = \frac{1{,}162 * 10^{-3} kWh}{kg * K} \qquad (13)$$
$$mit\ c_2 = \frac{1{,}1972\ W}{1000\ K} = \frac{1{,}1972 * 10^{-3}\ kW}{K}$$

Dabei stellt c_1 die Wärmespeicherkapazität des Wassers umgerechnet in kWh/(kg K) dar. m beschreibt die Masse des Wassers (in vorliegender Simulation 150kg) und $\dot{T}_{Wasser}$ die zeitliche Ableitung über die Wassertemperatur. P stellt die Leistung des Boilers und c_2 den Wärmeleitwert der Isolierung mit Konvektion und Konduktion (Wert aus der Literatur [2] übernommen) umgerechnet in kW/K dar. $T_{Wasser}(t)$ beschreibt die aktuelle Temperatur des Wassers und $T_{Umgebung}$ die Umgebungstemperatur. In den folgenden Formelzeilen wird die zugehörige Differentialgleichung auf Energiebasis hergeleitet. Die erste Vereinfachung ist, dass $T_{Wasser}(t) - T_{Umgebung}$ als T(t) definiert wird. Daraus folgt:

$$c_1 m\dot{T}(t) = P - c_2 T(t)$$
$$mit\ E(t) = c_1 m T(t)\ und\ \lambda = \frac{c_2}{c_1 m} \quad folgt \qquad (14)$$
$$\dot{E}(t) = P - \lambda E(t)$$

Nach Berücksichtigung der Anfangsbedingung E(0) = E_0 und Lösen der Differentialgleichung (Lösen der homogenen und der partikulären Diff-Gleichung) folgt nun das Ergebnis, welches in Formel (15) ersichtlich ist, wobei „dt" einer Stunde entspricht.

$$\mathrm{E(t)} = E_0 e^{-\lambda dt} + P\frac{1 - e^{-\lambda * dt}}{\lambda} \qquad (15)$$

Am Ende des Tages, nach 24 Stunden, muss eine entsprechende Energiemenge im Speicher vorhanden sein.

3.4.4.1 Deterministische Modellierung – Typ D

Nach der Herleitung aus Punkt 3.4.4 wird das lineare Optimierungsproblem nun deterministisch modelliert. Um die Ergebnisse zu den anderen Verbrauchertypen vergleichbar zu machen, ist die Energie im Speicher zu Beginn 0, daher ist $E_0 = 0$. Pro Zeitschritt kann wiederum nur eine begrenzte Menge an Energie konsumiert werden. Weiters ist die Energiemenge am Ende des Tages entscheidend, daher wird die konsumierte Energie zum aktuellen Zeitpunkt mit entsprechender Abkühlung zum Ende des Tages hin berechnet. Dies wird wie folgt umgesetzt.

$$\mathrm{P} = E_t * \frac{1 - e^{-\lambda * dt}}{\lambda} * e^{-\lambda(24-t)} \tag{16}$$

Somit wird der Verbrauchertyp D wie folgt deterministisch modelliert.

$$\min. \sum_{t=1}^{24} c_t * E_t \tag{17}$$

$$s.t. 0 \leq E_t \leq E_{max} \tag{18}$$

$$\sum_{t=1}^{24} E_t * \frac{1 - e^{-\lambda * dt}}{\lambda} * e^{-\lambda(24-t)} = E_{gesamt} \tag{19}$$

Formel (17) und (18) sind gleich wie bei Typ B und Typ C. Formel (19) beschreibt die Bedingung, dass am Ende des Tages eine entsprechende Energiemenge im Speicher vorhanden sein muss. Dabei wird ausgehend von der Energiemenge des aktuellen Zeitpunkts berechnet, wie viel davon am Ende des Tages durch Verluste noch übrig ist.
In Abbildung 15 ist der Codeausschnitt zu sehen, der dieses Optimierungsproblem modelliert. Dabei stellt „prices“ die Kosten zur jeweiligen Stunde, „power“ die maximale Energie pro Zeitschritt, „period“ die benötigten Perioden, um die Energiemenge zu erreichen und somit „energy_provided=power*period“ die Gesamtenergiemenge dar. Die Matrix A wird durch die beiden Matrizen A1 und A2 und der Vektor b durch die beiden Vektoren b1 und b2 erstellt. Durch G und h wird die obige Formel (19) beschrieben. Sind die Zielfunktion und die Nebenbedingungen definiert, werden die optimalen Verbrauchsentscheidungen x anhand des Solvers berechnet.

```
# Zusammenstellen der Zielfunktion und der Nebenbedingungen
dt = 1 # dt ist eine Stunde, da ich auf Stundenbasis arbeite --> gilt für die Abkühlungsverluste
c  = prices
A1 = np.eye(len(prices))
b1 = np.ones(len(prices))*power_load_max
A2 = -np.eye(len(prices))
b2 = np.zeros(len(prices))
A  = np.vstack((A1,A2))
b  = np.concatenate((b1,b2))

if lbd == 0: f=1 # Für den Fall, dass lbd = 0 --> f = 1 --> das Ergebnis muss dem Verbrauchertyp B entsprechen
else: f  = (1 - np.exp(-lbd*dt))/lbd
G  = f*np.exp(-lbd*(times[-1] - np.array([times])))

h  = energy_provided

[x,s,z,y,status] = lp_solve(c,A,b,G,h)
# print("Solver status = %s" %status)
powers = x
cost_optimal = np.dot(c.T,x)[0] #Berechnung der optimal möglichen Kosten

cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten
```

Abbildung 15: Codeausschnitt deterministische Modellierung Typ D
Quelle: Eigene Ausarbeitung

3.4.4.2 Stochastische Modellierung – Typ D

Die stochastische Modellierung basiert auf der deterministischen von Typ D. Die Zielfunktion und die Nebenbedingungen bleiben im Grunde die gleichen. Da der Strompreis in zukünftigen Schritten aber nicht bekannt ist, wird anhand des knn-Verfahrens die wahrscheinliche Preisentwicklung zu jedem Zeitschritt neu bestimmt. Daher werden auch die Zielfunktion und die Nebenbedingungen zu jeder Stunde neu formuliert, unter Berücksichtigung der bereits verbrauchten Energiemenge. Anschließend wird die Optimierung vom aktuellen Zeitpunkt bis zum Ende durchlaufen, um die optimale Entscheidung zum jetzigen Zeitpunkt zu erhalten. In Abbildung 16 ist der Codeausschnitt zu sehen, der dieses Modell beschreibt. Die Preisvariable c wird wiederum mit der oberen „for“-Schleife anhand der Szenarien gebildet. Die Variablen G und h für die Nebenbedingungen, werden in der zweiten „for“-Schleife anhand der Szenarien zusammengesetzt.

```
knn_numb = 3 #Anzahl der nahesten Nachbarn, die herangezogen werden
price_real = price_sel[day_index] # Preisverlauf des aktuellen Tages
len_day = len(price_real) # Stunden innerhalb eines Tages
energy_used = 0
powers = np.zeros(len_day)
time_line_OFF = True # Auswahl knn-Methode

if verbose:
        print('.', end="")

for i in range(len_day-1):

    # Stundenindexabhänger Preis bestimmen
    if time_line_OFF == True:
        price_moment, knn_price_array = knn_probability_fast(knn_numb, i+1, price_history, price_real)
    else:
        price_real_prev = price_sel[day_index - 1]
        price_moment, knn_price_array = knn_probability_timeline(knn_numb, i+1, price_history, price_real, price_real_prev)

    times_moment = times[0:len_day-i]

    knn_number_szenarios= knn_price_array.shape[0]

    # Zusammenstellen der Zielfunktion und der Nebenbedingungen
    dt = 1
    c  = np.array([price_real[i]])
    for j in range(knn_number_szenarios):
        c_temp = price_history[knn_price_array[j][0]]
        weight_factor = knn_price_array[j][2]
        c_moment = c_temp[i+1:len_day] * weight_factor
        c = np.concatenate((c,c_moment))

    A1 = np.eye(c.size)
    b1 = np.ones(c.size)*power_load_max
    A2 = -np.eye(c.size)
    b2 = np.zeros(c.size)
    A  = np.vstack((A1,A2))
    b  = np.concatenate((b1,b2))

    if lbd == 0: f=1 # Für den Fall, dass lbd = 0 --> f = 1 --> das Ergebnis muss dem Verbrauchertyp B entsprechen
    else: f  = (1 - np.exp(-lbd*dt))/lbd
    G_temp_whole  = f*np.exp(-lbd*(times_moment[-1] - np.array([times_moment])))

    if (energy_provided-energy_used) < 0:
        h_temp = 0
    else:
        h_temp  = energy_provided-energy_used

    step_size = c_moment.size # Länge eines Szenarios in Stunden bis zum Ende des Tages ohne den bekannten Wert
    for j in range(knn_number_szenarios): # Für G: 1er an die Stellen setzen, die für die jeweiligen Szenarios benötigt werden
                            # Für h: Den Vektor für alle Szenarien mit dem Restenergiebeadarf füllen
        G_temp = np.zeros((1,c.size))
        G_temp[0][0] = G_temp_whole[0][0]
        G_temp[0][1+j*step_size:1+(j+1)*step_size] = G_temp_whole[0][1:1+step_size]
        if j == 0:
            G = G_temp
            h = h_temp
        else:
            G = np.vstack((G,G_temp))
            h = np.vstack((h,h_temp))

    [x,s,z,y,status] = lp_solve(c,A,b,G,h)
    #print("Solver status = %s" %status)

    energy_used += (np.ceil(x[0]*1000)/1000)*f*np.exp(-lbd*(times_moment[-1])) # ceil ist zum Aufrunden, damit solver kein Prob
                                # Aufsummieren der Energieverbräuche, damit für die folgenden Optimierungsdurchläufe
                                # der erlaubte Restverbruach angepasst wird. x entspricht der Verbrauch im aktuellen Schritt
    #print(energy_used)
    powers[i] = x[0] # Verbrauchsentscheidung des aktuellen Optimierungsdurchlauf zuweisen

powers[len_day-1] = (energy_provided-energy_used)/f  # Benötigte Leistung der letzten Stunde zuweisen

cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten
```

Abbildung 16: Codeausschnitt stochastische Modellierung Typ D
Quelle: Eigene Ausarbeitung

3.5 Preisbasis der Optimierungsverfahren

Bei allen Optimierungsverfahren wird derselbe Tag herangezogen, um die Endkosten zu berechnen, und damit die Ergebnisse vergleichbar machen zu können. Der Unterschied liegt darin, welcher Preis zur Entscheidungsfindung herangezogen wird, also wie viel Energie zu welchem Zeitschritt konsumiert wird.

3.5.1 Heuristische Opt. anhand konstantem Strompreis – KONST

In diesem Verfahren wird aus dem gewählten Tag der Durchschnitt des volatilen Strompreises gebildet, um einen konstanten Strompreis zu erhalten. Dieser stellt gleichzeitig die Basis der Endkostenberechnung dar.

3.5.2 Deterministische Opt. anhand volatilem Mittelwert-Strompreis – DET_{mittel}

Die Preisbasis für dieses deterministische Optimierungsverfahren legt die historischen Preisverläufe von einem Jahr zugrunde. Aus diesen 365 Verläufen wird zu jeder Stunde der Mittelwert gebildet. Alle Mittelwerte aneinander gereiht ergibt die Mittelwert-Strompreiskurve.

3.5.3 Deterministische Opt. anhand volatilem Einzel-Strompreis – DET_{einzel}

Dieser Preisverlauf ist der jeweilig ausgewählte Tag. Das heißt bei diesem deterministischen Optimierungsdurchlauf wird angenommen, dass der wirkliche Strompreisverlauf bereits zu Beginn des Tages bekannt ist. Somit wird das Ergebnis dieser Optimierung die niedrigsten möglichen Kosten liefern und damit die optimalen Entscheidungsvariablen. Dieser Preisverlauf dient nur zum Vergleich, da das Wissen über die wirkliche Preisentwicklung zu Beginn des Tages natürlich unmöglich ist.

3.5.4 Stochastische Opt. anhand volatilem Einzel-Strompreis – STOCH

In der stochastischen Optimierung wird der Preisverlauf zu jeder Stunde neu bestimmt. Dies geschieht anhand des knn-Verfahrens, welches k ähnlichste Preisverläufe aus den 365 historischen Verläufen, anhand der bereits bekannten Strompreise, herausfiltert. Abhängig der Anzahl k, werden entsprechend viele Szenarien in der stochastischen Optimierung gebildet, welche aber zusätzlich mit den jeweiligen Wahrscheinlichkeiten gewichtet sind.

3.5.5 Stochastische Opt. anhand volatilem Einzel-Strompreis – $STOCH_{Sohm}$

Im Zuge der Arbeit entstand aus einem Versuch eine neue Variante der Preisbildung. Hierbei wird gleich wie bei der Methode unter Punkt 3.5.4 zu jeder Stunde der Preisverlauf neu bestimmt. Der Unterschied liegt aber in der Bildung der Szenarien. Es fließen nicht k Szenarien in die Optimierung mit ein, sondern der mit der Wahrscheinlichkeit gemittelte Strompreis aus den k Szenarien. Somit entsteht nur ein gemitteltes Szenario pro Zeitschritt.

3.6 Ergebnismatrix

Nachdem in Python nun alle Verbrauchertypen und auch für alle Optimierungsverfahren die Preisbasen modelliert wurden, kann jeder Verbrauchertyp auf jedes Optimierungsverfahren angewendet werden. Um letztendlich einen Eintrag in der Ergebnis-Matrix machen zu können, muss zuerst klargestellt werden, bei welcher Preisbasis der Optimierungsverfahren, welcher Preis für die Kostenberechnung herangezogen wird, denn die Einträge in der Ergebnismatrix sind letztendlich Kosten, die miteinander verglichen werden. In Tabelle 1 ist ersichtlich, welche Preise für die Optimierung der jeweiligen Preisbasis herangezogen werden, und welche für die Kostenberechnung.

	Preis für die Optimierung	Preis für die Kostenberechnung
KONST	Preisbasis von KONST	Preisbasis von KONST
DET_{mittel}	Preisbasis von DET_{mittel}	Preisbasis von DET_{einzel}
DET_{einzel}	Preisbasis von DET_{einzel}	Preisbasis von DET_{einzel}
STOCH	Preisbasis von STOCH	Preisbasis von DET_{einzel}
$STOCH_{SOHM}$	Preisbasis von $STOCH_{SOHM}$	Preisbasis von DET_{einzel}

Tabelle 1: Optimierungspreis vs. Kostenberechnungspreis
Quelle: Eigene Ausarbeitung

Die Preisbasis des volatilen, deterministischen Einzelpreises stellt den jeweiligen ausgewählten Tag dar, also die realen Kosten des aktuellen Tages. Daher werden auch diese Preise für die Kostenberechnung aller Optimierungen herangezogen, welche basierend auf einem volatilen Strompreis durchgeführt wurden. Die einzige Ausnahme stellt die Preisbasis KONST dar. Hier handelt es sich um den konstanten Mittelwert des ausgewählten Tages. Da natürlich vor der Optimierung das Wissen über einen konstanten oder volatilen Strompreis bereits vorhanden ist, muss auch der Preis für die Kostenberechnung in diesem Fall der konstante Strompreis sein. Ansonsten wäre die Grundlage für einen fairen Kostenvergleich dieser Preisbasis nicht gewährleistet.

Somit ist definiert, wie die Kosteneinträge der jeweiligen Optimierungsverfahren entstehen. Um die Unterschiede der Verfahren deutlich ersichtlich zu machen und somit einen schnellen Vergleich zu ermöglichen, wird nun die Ergebnis-Matrix mit den Resultaten aus den jeweiligen Optimierungen befüllt. In Tabelle 2 ist diese Matrix zu sehen.

Preisbasis der Optimierung / Verbrauchertypen	KONST	DET_{mittel}	DET_{einzel}	STOCH	$STOCH_{SOHM}$
Verbrauchertyp A					
Verbrauchertyp B					
Verbrauchertyp C					
Verbrauchertyp D					

Tabelle 2: Ergebnismatrix nicht befüllt
Quelle: Eigene Ausarbeitung

Um nun die Resultate der jeweiligen Einträge bestimmen zu können, wird ein komplettes Jahr simuliert und anschließend die durchschnittlichen Tageskosten errechnet. Diese Tageskosten stellen die Einträge der Ergebnismatrix dar.

Anhand der Matrix ist somit gut ersichtlich, welche Preisbasis eines Optimierungsverfahrens in Bezug auf einen entsprechenden Verbrauchertyp die optimale

Lösung darstellt. Respektive welche Optimierung am nächsten an die Lösung mittels optimalem Wissen, also die Optimierung mittels DET_{einzel}, herankommt.

4. Resultate und Varianten

In diesem Kapitel werden die Resultate aus den Simulationen dokumentiert und miteinander verglichen. Zu Beginn wird zunächst ein Simulationsdurchlauf eines Tages erklärt, um die Systematik einer Simulation deutlich zu machen.

Am Ende des Kapitels werden unterschiedliche Parameter variiert, um auch Ergebnisse aus Varianten zu bekommen.

4.1 Resultaterläuterung anhand einer Tagessimulation

Im Folgenden wird die Simulation einer Tages anhand des Verbrauchertyps B, dem verschiebbaren, unterbrechbaren Verbraucher ohne Verluste, erklärt. Dabei muss eine bestimmte Menge an Energie im Laufe eines Tages umgesetzt werden, und dies zu minimalen Kosten. Das heißt für diesen verlustlosen Verbrauchertyp, dass Brutto-Energiemenge gleich Netto-Energiemenge ist, und die Energie auch möglichst in den Stunden konsumiert wird, in denen der Strompreis am billigsten ist. Folgende Parameter sind Basis dieser Simulation:

- Preisdaten basieren auf EEX Datenbank
    - Historische Daten aus dem Jahr 2011
    - Ausgewählter Tag ist der 29. Tag aus dem Jahr 2012, also der 29. 01. 2012
- Energiemenge, welche umgesetzt werden muss; beträgt 10kWh
- Max. Energiemenge, welche pro Stunde konsumiert werden kann; beträgt 2kWh

Wird nun der gewählte Tag mit obigen Parametern simuliert, resultieren folgende Ergebnisse, welche immer mit den niedrigsten möglichen Kosten, also den Kosten aus der Optimierung mittels der Preisbasis DET_{einzel}, verglichen werden. In Tabelle 3 sind diese Ergebnisse zu sehen.

	Tageskosten	Abweichung zur optimalen Entscheidung
Optimierung mittels Preisbasis KONST	41,78 Cent	9,51 Cent entsprechen 22,8%
Optimierung mittels Preisbasis DET_{mittel}	36,46 Cent	4,19 Cent entsprechen 11,5%
Optimierung mittels Preisbasis DET_{einzel}	32,28 Cent	-
Optimierung mittels Preisbasis STOCH	35,14 Cent	2,86 Cent entsprechen 8,2%
Optimierung mittels Preisbasis $STOCH_{SOHM}$	35,14 Cent	2,86 Cent entsprechen 8,2%

Tabelle 3: Kostenvergleich der Tagessimulation
Quelle: Eigene Ausarbeitung

Zu sehen ist, dass die höchsten Kosten durch die Optimierung mittels der Preisbasis KONST entstehen. Die zweithöchsten Kosten anhand der Optimierung mittels DET_{mittel}. Die besten Näherungen an die minimalen Kosten stellen an diesem Tag jeweils die stochastischen Optimierungen mittels der Preisbasis STOCH bzw. $STOCH_{SOHM}$ dar. Der EVPI ist in diesem Fall 2,86 Cent.

Um dies zu verdeutlichen, werden nacheinander die zeitlichen Verläufe, welche die Verbrauchsentscheidungen (roter Verlauf mit zugehöriger roter Beschriftung auf der rechten Achse), die Preisbasis für die Simulation (grüner Verlauf mit zugehöriger blauer Beschriftung auf der linken Achse) und die realen Kosten zum gewählten Tag (blauer Verlauf mit zugehöriger blauer Beschriftung auf der linken Achse) beinhalten, dargestellt.

Abbildung 17 stellt die Verläufe dar, welche durch die Optimierung anhand der Preisbasis KONST entstanden sind.

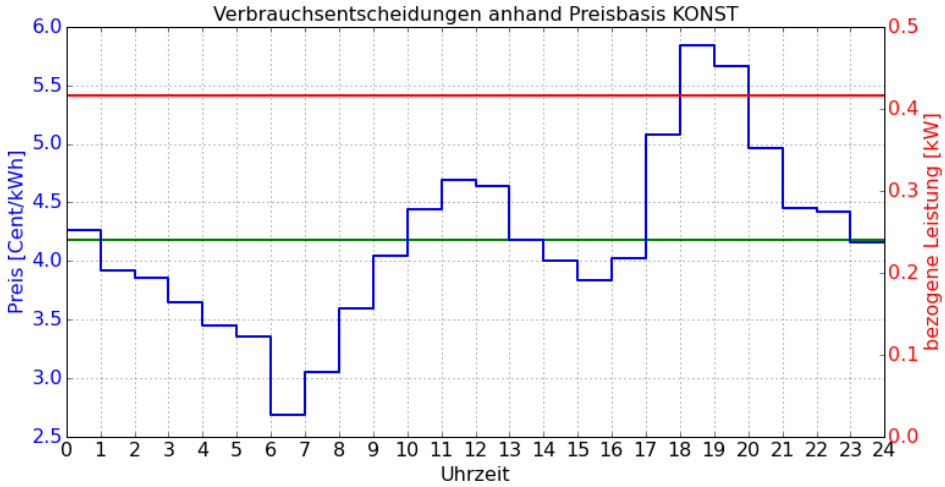

Abbildung 17: Tagesverlauf der Optimierung mittels Preisbasis KONST
Quelle: Eigene Ausarbeitung

Zu sehen ist, dass die konstante Preiskurve (grün), anhand dieser optimiert wird, den Mittelwert des realen Preisverlaufs am gewählten Tag (blau) darstellt. Da die verbrauchten Energiemengen bei diesem Verbrauchertyp B keinem Verlust unterzogen sind, ist egal zu welcher Tageszeit die Energie konsumiert wird. Deshalb verteilt der Solver die Verbrauchsentscheidungen (rot) auch gleichmäßig über den Tag.

Abbildung 18 stellt die Optimierung anhand der Preisbasis DET_{mittel} dar.

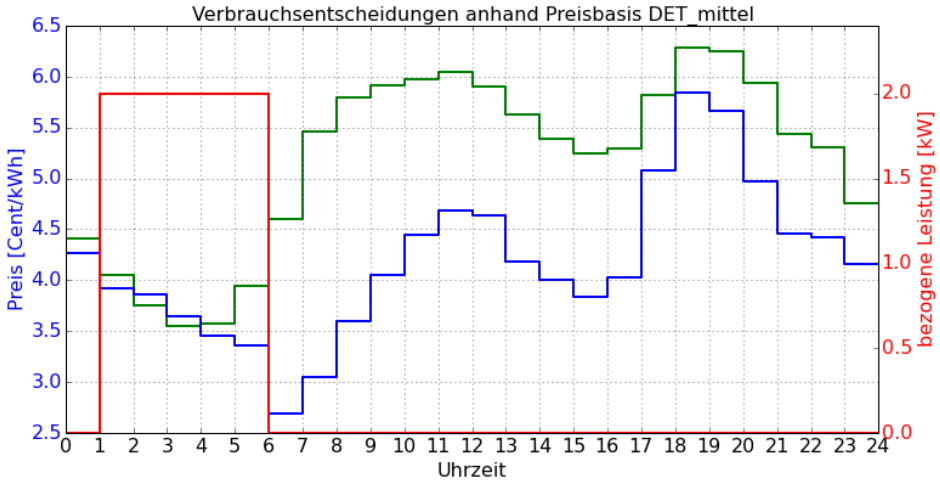

Abbildung 18: Tagesverlauf der Optimierung mittels Preisbasis DET_{mittel}
Quelle: Eigene Ausarbeitung

Es ist deutlich zu erkennen, dass die Optimierung anhand der grünen Kurve durchgeführt wurde, weil die Verbrauchsentscheidungen (rot) genau in die Preistäler dieses Verlaufes gelegt werden. Die Preistäler des tatsächlichen Preises sind leicht nach rechts verschoben dargestellt (blau). Daher entsteht der Preisversatz zu den optimal niedrigen Kosten.

Abbildung 19 stellt die Verläufe dar, welche durch die Optimierung anhand der Preisbasis DET_{einzel} entstanden sind.

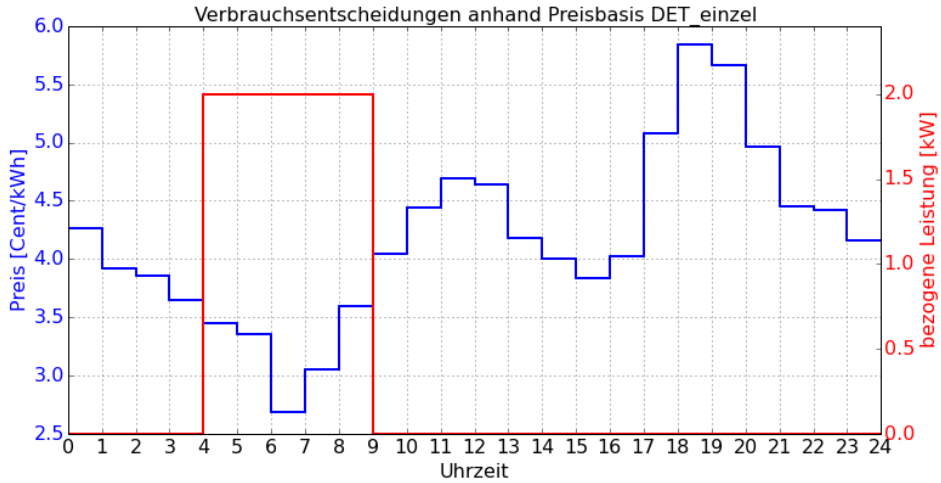

Abbildung 19: Tagesverlauf der Optimierung mittels Preisbasis DET_{einzel}
Quelle: Eigene Ausarbeitung

Obige Abbildung stellt die optimalen Verbrauchsentscheidungen (rot) dar, welche genau in die Preistäler des realen Preisverlaufes (blau) gelegt werden. Da dieser Verlauf die optimalen Verbrauchsentscheidungen darstellt, existiert auch kein grüner Verlauf, weil sich dieser mit dem blauen Verlauf deckt.

Abbildung 20 stellt die Optimierungen anhand der Preisbasen $STOCH_{SOHM}$ und STOCH dar.

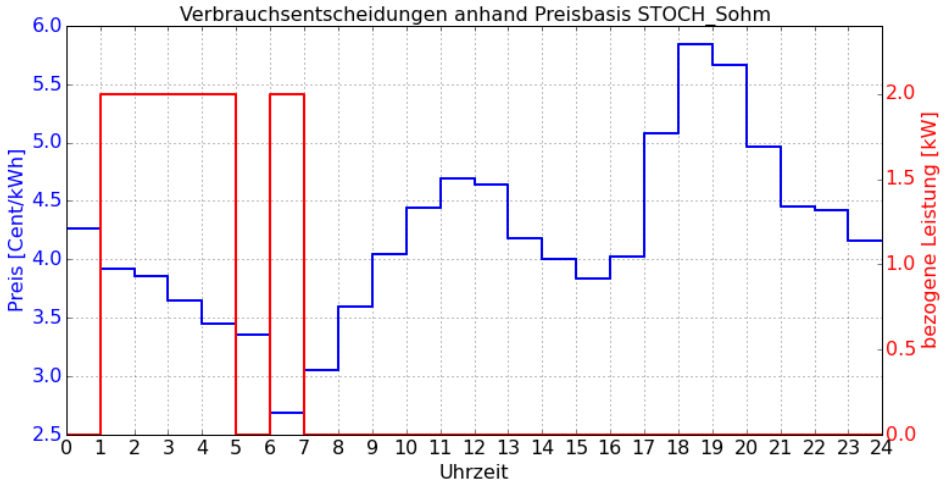

Abbildung 20: Tagesverlauf der Opt. mittels Preisbasen $STOCH_{SOHM}$ und STOCH
Quelle: Eigene Ausarbeitung

An diesem Tag liefern beide stochastischen Optimierungen dieselben Kosten und auch dieselben Verbrauchsentscheidungen (rot). Daher wird hier eine Abbildung stellvertretend für beide Optimierungen dargestellt. Diese beiden Optimierungen liefern aber nicht immer die gleichen Endergebnisse, da bei der Preisbasis $STOCH_{SOHM}$ zu jeder Stunde ein gewichtetes Mittelwertszenario und bei der Preisbasis STOCH mehrere Szenarien für die Entscheidungsfindung herangezogen werden. Was aber für beide Verläufe gilt ist, dass nicht nur ein grüner Verlauf, anhand dem optimiert wird, dargestellt werden kann, da die zukünftige Preisentwicklung, anhand der stochastischen Methode und des knn-Verfahrens, zu jeder Stunde wieder neu ermittelt wird.

4.2 Jahressimulation anhand der EEX Preisdaten

In diesem Abschnitt werden für die Simulationen die Preise der EEX Datenbank verwendet, welche One-Day-Ahead Preise sind. Anschließend werden die Tagessimulationen, welche in Punkt 4.1 erläutert wurden, auf ein ganzes Jahr angewandt und anschließend die Mittelwert-Tageskosten bestimmt. Diese gemittelten Tageskosten werden in einer Ergebnismatrix miteinander verglichen. Folgende Simulationsparameter liegen den Verbrauchertypen zugrunde:

- Preisdaten basieren auf EEX Datenbank
    - Historische Daten aus dem Jahr 2011
    - Simuliertes Jahr ist das Jahr 2012
- Energiemenge, welche umgesetzt werden muss; beträgt 10kWh (Bedeutet für die Verbrauchertypen C & D die Nettoenergiemenge, die umgesetzt werden muss. Im Bsp. Boiler muss diese Energiemenge am Ende des Tages in Form von Warmwasser zur Verfügung stehen)
- Max. Energiemenge, welche pro Stunde konsumiert werden kann; beträgt 2kWh
- Wirkungsgrad η für Verbrauchertyp C; beträgt 80%
- Konstanten c_1 und c_2 für Verbrauchertyp D; sind dieselben wie unter Punkt 3.4.4 beschrieben
- Szenarienbildung für die Preisbasis STOCH anhand der Büschel-Methode
- Anzahl k der knn-Methode; beträgt 3

Mit Hilfe der folgenden Abbildung 21 wird anhand eines Beispiels erläutert, wie die durchschnittlichen Tageskosten aus der Tageskostenverteilung übers Jahr entstehen. Dabei werden die Optimierungsergebnisse der Preisbasen DET_{einzel} und STOCH aus Verbrauchertyp B zusammen aufgetragen und verglichen.

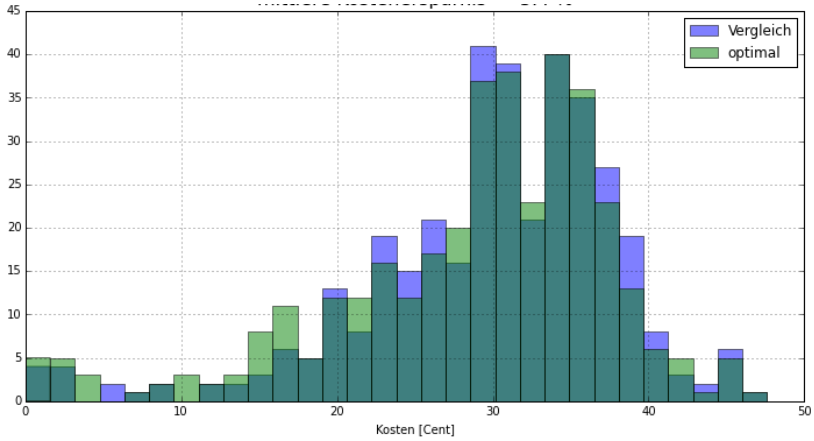

Abbildung 21: Tageskostenverteilung Verbrauchertyp B - STOCH vs. DET_{einzel}
Quelle: Eigene Ausarbeitung

Die vertikale Achse beschreibt die Häufigkeit, mit der entsprechende Tageskosten übers Jahr gesehen vorkommen. Die horizontale Achse beschreibt die Höhe der Tageskosten. Grün ersichtlich ist die Optimierung anhand der Preisbasis DET_{einzel}, welche die optimale Verteilung beschreibt, blau dargestellt ist die Optimierung anhand der Preisbasis STOCH. Wo sich beide Verläufe überschneiden, entsteht eine Mischfarbe (dunkelgrün). Aus der obigen Abbildung ist gut ersichtlich, dass die Balken der grünen Verteilung im Mittel weiter links angeordnet sind. Damit ergeben sich die niedrigeren durchschnittlichen Tageskosten.

4.2.1 Ergebnismatrix anhand der EEX Preisdaten

Die Resultate für die Ergebnismatrix werden anhand der unter Punkt 4.2 beschriebenen Parameter errechnet. Für die stochastischen Optimierungen mittels der Preisbasis STOCH werden die Szenarien anhand der Büschelmethode gebildet. Nachfolgend ist in Tabelle 4 diese Matrix zu sehen, deren Einträge in Cent angegeben sind.

Preisbasis der Optimierung / Verbrauchertypen	KONST	DET_{mittel}	DET_{einzel}	STOCH	$STOCH_{SOHM}$
Verbrauchertyp A	43,34	29,44	28,94	29,85 EVPI = 0,91	29,56 EVPI = 0,62
Verbrauchertyp B	43,34	29,44	28,73	29,76 EVPI = 1,03	29,74 EVPI = 1,01
Verbrauchertyp C	54,18	38,42	37,15	38,62 EVPI = 1,47	38,63 EVPI = 1,48
Verbrauchertyp D	44,11	34,51	33,09	34,61 EVPI = 1,52	34,61 EVPI = 1,52

Tabelle 4: Ergebnismatrix anhand EEX Daten
Quelle: Eigene Ausarbeitung

Grün hinterlegt sind die minimal möglichen Kosten, gelb hinterlegt sind für jeden Verbrauchertyp die Kosten, welche am nächsten an die minimalen Kosten herankommen. In dieser Jahressimulation ist jeweils die beste Entscheidung anhand der Mittelwertpreise der historischen Daten zu optimieren. Die zweitbeste Entscheidung wäre eine stochastische Optimierung zu wählen, wobei die $STOCH_{SOHM}$ noch minimal bessere Ergebnisse liefert.

Hingegen weit abgeschlagen ist die Optimierung anhand der Preisbasis KONST. Die Kosten dieser Optimierung sind im Vergleich zur optimalen Lösung 25-35% schlechter.

4.2.2 Ergebnismatrix anhand der timeline-Methode

Die stochastische Optimierung basiert darauf, die zukünftigen Preisverläufe anhand bereits bekannter Strompreisdaten so gut wie möglich vorbestimmen zu können. Bei der momentanen Umsetzung der stochastischen Preisbasen STOCH und $STOCH_{SOHM}$ ist zur Stunde 1 nur ein Wert bekannt, zur Stunde 2 nur zwei Werte usw.

Da am Tagesbeginn normalerweise auch das Wissen über die Preise des Vortages vorhanden ist, wurde eine Methode implementiert, welche auch die Preisdaten des Vortages für die Szenarienbildung berücksichtigt. Diese Methode wird timeline-Methode genannt, weil die Preise des Vortages vor den realen Preis angestückelt werden und somit eine Zeitlinie gebildet wird. Sie ermöglicht dem knn-Verfahren eine einstellbare Anzahl an Stunden in die Vergangenheit zu blicken und diese Preisdaten für die Szenarienbildung heran zu ziehen.

Für die folgende Simulation gelten dieselben Parameter wie unter Punkt 4.2 beschrieben. Der einzige Unterschied liegt darin, dass für die Preisbasen der stochastischen Optimierungen die timeline-Methode angewandt wird, bei der 6 Stunden zurückliegende Preise mit berücksichtigt werden.

In der nachstehenden Tabelle 5 sind die Resultate aus dieser Jahressimulation in Cent Beträgen dargestellt.

Preisbasis der Optimierung / Verbrauchertypen	KONST	DET_{mittel}	DET_{einzel}	STOCH	$STOCH_{SOHM}$
Verbrauchertyp A	43,34	29,44	28,94	29,85 EVPI = 0,91	29,62 EVPI = 0,68
Verbrauchertyp B	43,34	29,44	28,73	29,83 EVPI = 1,10	29,89 EVPI = 1,16
Verbrauchertyp C	54,18	38,42	37,15	38,53 EVPI = 1,38	38,61 EVPI = 1,46
Verbrauchertyp D	44,11	34,51	33,09	34,50 EVPI = 1,41	34,55 EVPI = 1,46

Tabelle 5: Ergebnismatrix anhand EEX Daten und timeline-Methode
Quelle: Eigene Ausarbeitung

Grün hinterlegt sind wiederum die minimal möglichen Kosten und gelb wieder die Kosten je Verbrauchertyp, welche am nächsten an die minimalen Kosten herankommen. Die Werte in den ersten drei Spalten sind gegenüber der vorigen Jahressimulation gleich geblieben, da sich bei den deterministischen Optimierungen nichts geändert hat. Beide stochastischen Optimierungen zeigen gegenüber der vorigen Jahressimulation minimale Verschlechterungen im Zehntel-Cent-Bereich.

4.2.3 Zusammenfassung der Ergebnisse anhand der EEX Preisdaten

In beiden Jahressimulationen resultieren für alle Verbrauchertypen die niedrigst möglichen, realen Kosten, wenn die Optimierung anhand der Preisbasis DET_{mittel} durchgeführt wird. Die Möglichkeit bei den stochastischen Optimierungen auch Preise des Vortages mit in die Szenarienbildung einfließen zu lassen, bringt keine Verbesserung mit sich.

4.3 Jahressimulation anhand der EPEX Preisdaten

In diesem Kapitel werden für die Simulationen die Preise der EPEX Datenbank verwendet, welche Intraday Preise sind. Die Jahressimulationen werden mit Ausnahme der Strompreisdaten anhand der gleichen Parameter wie unter Punkt 4.2 durchgeführt. Die gemittelten Tageskosten werden anschließend in einer Ergebnismatrix miteinander verglichen. Nachfolgend sind die Simulationsparameter nochmals vollständig aufgelistet:

- Preisdaten basieren auf EPEX Datenbank
    - Historische Daten aus den Jahren 2012 und 2013
    - Simuliertes Jahr ist das Jahr 2014
- Energiemenge, welche umgesetzt werden muss; beträgt 10kWh (Bedeutet für die Verbrauchertypen C & D die Nettoenergiemenge, die umgesetzt werden muss. Im Bsp. Boiler muss diese Energiemenge am Ende des Tages in Form von Warmwasser zur Verfügung stehen)
- Max. Energiemenge, welche pro Stunde konsumiert werden kann; beträgt 2kWh
- Wirkungsgrad η für Verbrauchertyp C; beträgt 80%
- Konstanten c_1 und c_2 für Verbrauchertyp D sind dieselben wie unter Punkt 3.4.4 beschrieben
- Szenarienbildung für die Preisbasis STOCH anhand der Büschel-Methode
- Anzahl k der knn-Methode; beträgt 3

4.3.1 Ergebnismatrix anhand der EPEX Preisdaten

Die Resultate für die folgende Ergebnismatrix werden anhand der unter Punkt 4.3 beschriebenen Parameter errechnet. In Tabelle 6 ist diese Matrix zu sehen, deren Einträge in Cent angegeben sind.

Preisbasis der Optimierung / Verbrauchertypen	KONST	DET_{mittel}	DET_{einzel}	STOCH	$STOCH_{SOHM}$
Verbrauchertyp A	33,00	22,97	21,23	23,49 EVPI = 2,26	23,26 EVPI = 2,03
Verbrauchertyp B	33,00	22,97	20,87	23,97 EVPI = 3,10	23,62 EVPI = 2,75
Verbrauchertyp C	41,25	29,72	27,13	30,75 EVPI = 3,62	30,44 EVPI = 3,31
Verbrauchertyp D	33,58	26,85	23,91	27,50 EVPI = 3,59	27,36 EVPI = 3,45

Tabelle 6: Ergebnismatrix anhand EPEX Daten
Quelle: Eigene Ausarbeitung

Grün hinterlegt sind die minimal möglichen Kosten, gelb hinterlegt sind für jeden Verbrauchertyp die Kosten, welche am nächsten an die minimalen Kosten herankommen. In der Jahressimulation anhand der EPEX Preisdaten ist gleich wie bei den EEX Daten jeweils die beste Entscheidung anhand der Mittelwertpreise der historischen Daten zu optimieren. Die zweitbeste Entscheidung wäre ebenso eine stochastische Optimierung, wobei die $STOCH_{SOHM}$ wiederum minimal bessere Ergebnisse liefert. Die Optimierung anhand der Preisbasis KONST liefert im Vergleich zur optimalen Lösung 28-35% höhere Kosten. Somit schneidet diese Optimierungsvariante wiederum deutlich schlechter ab.

4.3.2 Ergebnismatrix anhand der timeline-Methode

Auch für EPEX Preisdaten wird die timeline-Methode, wie in Punkt 4.2.2 beschrieben, als Vergleich heran gezogen. Sie dient wiederum der Szenarienbildung anhand der Preisdaten des Vortages. Somit stehen auch zu Tagesbeginn bereits vergangene Preise für die knn-Methode zur Verfügung.

Für die folgende Simulation gelten dieselben Parameter wie unter Punkt 4.3 beschrieben. Der einzige Unterschied liegt darin, dass für die Preisbasen der stochastischen Optimierungen die timeline-Methode angewandt wird, bei der 6 Stunden zurückliegende Preise mit berücksichtigt werden.

In der nachstehenden Tabelle 7 sind die Resultate aus dieser Jahressimulation in Cent Beträgen dargestellt.

Preisbasis der Optimierung / Verbrauchertypen	KONST	DET_{mittel}	DET_{einzel}	STOCH	$STOCH_{SOHM}$
Verbrauchertyp A	33,00	22,97	21,23	23,39 EVPI = 2,16	23,14 EVPI = 1,91
Verbrauchertyp B	33,00	22,97	20,87	23,99 EVPI = 3,12	23,60 EVPI = 2,73
Verbrauchertyp C	41,25	29,72	27,13	30,77 EVPI = 3,64	30,53 EVPI = 3,40
Verbrauchertyp D	33,58	26,85	23,91	27,69 EVPI = 3,78	27,49 EVPI = 3,58

Tabelle 7: Ergebnismatrix anhand EPEX Daten und timeline-Methode
Quelle: Eigene Ausarbeitung

Grün hinterlegt sind wiederum die minimal möglichen Kosten und gelb hinterlegt die Kosten je Verbrauchertyp, welche am nächsten an die minimalen Kosten herankommen. Die Werte in den ersten drei Spalten sind gegenüber der vorigen Jahressimulation gleich geblieben, da sich bei den deterministischen Optimierungen nichts geändert hat. Die stochastischen Optimierungen zeigen gegenüber der vorigen Jahressimulation minimale Unterschiede im Zehntel-Cent-Bereich, welche teilweise besser oder auch schlechter sind.

4.3.3 Zusammenfassung der Ergebnisse anhand der EPEX Preisdaten

Gleich wie bei den Resultaten anhand der EEX Preisdaten resultieren in beiden Jahressimulationen für alle Verbrauchertypen die niedrigst möglichen, realen Kosten, wenn die Optimierung anhand der Preisbasis DET_{mittel} durchgeführt wird. Die Möglichkeit bei den stochastischen Optimierungen auch Preise des Vortages in die Szenarienbildung miteinfließen zu lassen, bringt keine Verbesserung mit sich.

4.4 Varianten der Jahressimulationen

Um eine Aussagekraft über die gewählten Parameter der bereits durchgeführten Simulationen zu bekommen, werden verschiedene Simulationsparameter variiert und die Ergebnisse mit den ursprünglichen Simulationen verglichen. In den folgenden Unterpunkten sind die Ergebnisse zu diesen Varianten dokumentiert.

4.4.1 Anzahl der k-nächsten Nachbarn variieren

Wird die Anzahl der nächsten Nachbarn der knn-Methode erhöht, werden die Ergebnisse der stochastischen Optimierungen besser. Das heißt, sie nähern sich immer mehr den Resultaten der Optimierung mittels der Preisbasis DET_{mittel} an. Ab 100 nächsten Nachbarn sind die Ergebnisse teilweise sogar im Hundertstel-Cent-Bereich besser. Sie sind aber eben nicht stets besser, sondern sie schwanken im Hundertstel-Cent-Bereich um die DET_{mittel} Resultate. Dies erklärt sich, da durch ein Erhöhen der Anzahl der nächsten Nachbarn sich die Preisbasis der stochastischen Optimierungen immer mehr an die Preisbasis DET_{mittel} annähert, nur eben gewichtet durch die Wahrscheinlichkeiten der Nachbarn. Der Nachteil, der dabei entsteht, ist, dass durch die erhöhte Anzahl an Nachbarn und die Tatsache, dass bei den stochastischen Optimierungen zu jeder Stunde die Preiskurve für die Optimierung neu errechnet wird, dies sehr zu Lasten der Rechenzeit geht. Bei der Preisbasis STOCH wird pro knn-Nachbarn ein Szenario mehr im Solver berechnet, was dazu führt, dass bei 100 Nachbarn eine Tagessimulation ca. 5 Minuten dauert. Eine Jahressimulation würde daher ca. 30 Stunden dauern. Eine höhere Anzahl als 20 Nachbarn ist deswegen nicht zu empfehlen.

Die Kernaussage dieser Variante ist, dass eine Erhöhung der Nachbarn die Resultate zwar verbessert, sie aber dennoch nicht besser als die Ergebnisse mittels der Preisbasis DET_{mittel} werden.

4.4.2 Anzahl Stunden der timeline – Methode variieren

Mittels der timeline-Methode kann bei der Suche nach den k-nächsten Nachbarn für die stochastischen Optimierungen eine bestimmte Anzahl an Stunden in die Vergangenheit geblickt werden. Damit stehen bei dieser Methode auch zu Beginn des Tages bekannte Werte vom Vortag zur Verfügung.

Das Variieren der Stunden liefert über alle Verbrauchertypen hinweg im Vergleich zur Einstellung mit 6 Stunden, minimale Unterschiede im Zehntel-Cent-Bereich mit meist

schlechteren Ergebnissen. Zusätzlich muss erwähnt werden, dass die Resultate aus den Jahressimulationen mit angepassten Vergangenheitsstunden nicht an die Resultate der Optimierung anhand der Preisbasis DET_{mittel} heran kommen.

4.4.3 Szenarienbildung anhand der knn-Baummethode

Die Szenarienbildung mittels der knn-Baummethode betrifft nur die Optimierung anhand der Preisbasis STOCH. Bei dieser Methode werden alle 6 Stunden wieder k neue Szenarien gebildet, wie bei einem Ast, der sich erneut verzweigt. Das heißt, wenn k = 3 ist, werden zu Beginn des Tages 3^4 Szenarien für die stochastischen Optimierungen gebildet, also 81. Das führt wiederum zu einer sehr langen Rechenzeit.

Zusätzlich zeigen diese Ergebnisse über alle Verbrauchertypen hinweg die schlechtesten stochastischen Optimierungsergebnisse. Daher ist diese Methode nicht zu empfehlen.

5. Diskussion und Zusammenfassung

In den folgenden Unterkapiteln werden die verwendeten Methoden nochmals hinsichtlich ihrer Genauigkeit und der nicht berücksichtigten Aspekte diskutiert. Zusätzlich werden die Ergebnisse der Arbeit zusammengefasst. Abschließend wird ein Ausblick über mögliche folgende Aufgaben geboten.

5.1 Annahmen

5.1.1 Energieverbrauch

Bei jeder Optimierung wird eine konstante zu verbrauchende Menge an Energie vorausgesetzt, welche innerhalb von 24 Stunden konsumiert werden bzw. bei Speichern wie dem Boiler am Ende zur Verfügung stehen muss. Dies ist nicht ganz realistisch, weil damit spontane Verbräuche nicht berücksichtigt werden. Es ist damit aber möglich, die unterschiedlichen Optimierungsmethoden gut miteinander vergleichen zu können.

5.1.2 Stochastik nur in Bezug auf den Strompreis

Die stochastische Variable in allen Optimierungen bezieht sich nur auf den volatilen Strompreis. Dies hat den Grund die stochastische Optimierung einfacher implementieren zu können und trotzdem gut vergleichbare Ergebnisse zu erhalten. Möglich wäre auch die stochastische Einbindung des Verbrauchs. Dies wäre ein möglicher Anknüpfungspunkt für weitere Forschungen im Themengebiet.

5.2 Umsetzung der stochastischen Optimierung

Im Zuge der Implementierung der stochastischen Optimierung mittels der Two-Stage Methode ergab ein Versuch einer Variante eine abgewandelte, vereinfachte Version der stochastischen Optimierung. Bei dieser fließt für die „second stage desicion“ nicht jedes Szenario einzeln in die Optimierung ein, sondern es entsteht aus den Szenarien jeweils ein mit der Wahrscheinlichkeit, welche aus der knn-Methode kommt, gewichteter Strompreisverlauf pro Zeitschritt. Diese stochastische Optimierungsvariante wurde im Zuge der Arbeit mit in die Ergebnismatrix aufgenommen und mit $STOCH_{SOHM}$ bezeichnet. Ein Vorteil dieses Verfahrens ist, dass sie weniger Rechenleistung benötigt und somit schneller gerechnet werden kann. Außerdem liefert sie bessere Ergebnisse als die Variante STOCH.

5.3 Verwendete Strompreisdaten

5.3.1 One-Day-Ahead vs. Intraday in Bezug auf Optimierungssituation

Die Datenpakete, welche von der EEX- bzw. EPEX-Datenbank verwendet wurden, waren jeweils Stundenpreise. Die One-Day-Ahead Preise werden jeweils für den nächsten Tag bekannt gegeben. Die Intraday Preise können sogar noch bis 45 Minuten vor Auslieferung des Energiepakets verhandelt werden. Dies macht einen Unterschied in Bezug auf die Optimierungssituation. Generell wurde in dieser Arbeit die stochastische Optimierung je Stunde mit einem neuen Preis gefüttert. Dies entspricht der Situation, in der Intraday Daten vorhanden sind. Die deterministischen Optimierungen, welche als Vergleich zu den stochastischen Optimierungen hergenommen wurden, entsprechen genau der Situation, bei der die Daten am Vortag bereits bekannt sind, was die One-Day-Ahead Preisdaten betrifft.

Im Endeffekt wurden beide Datensätze auf alle Optimierungen angewandt, um eine signifikante und vergleichbare Aussage treffen zu können. Das heißt bei den One-Day-Ahead Preisen wurde für die stochastische Optimierung pro Stunde immer nur ein neuer Preis bekannt gegeben und anhand dieser Daten die Intraday Preise simuliert. Die Intraday Preise wurden für die deterministischen Optimierungen im Vorhinein als bekannt vorgegeben und somit als One-Day-Ahead Preise simuliert.

Letztendlich sind die Verhältnisse der Ergebnisse zueinander gleich geblieben, weil die Preisverläufe übers Jahr gesehen meistens in den gleichen Stunden die jeweiligen Höchst- bzw. Tiefstwerte vorwiesen.

5.3.2 Preisbasis für deterministische Optimierungen

Die Preisbasis KONST diente als einfachste mögliche Preiskurve, welche jeweils den Durchschnitt über den aktuellen Tag darstellt. Diese Kurve soll eine Aussage darüber geben, wie gut die Resultate im Vergleich abschneiden, wenn es egal ist, wann der Verbraucher eingeschaltet wird. Dies gilt natürlich nicht für den Verbrauchertyp D: da dieser mit Verlusten behaftet ist, ist ein Einschalten am Ende des Tages immer kostengünstiger.

Die Preisbasis DET_{mittel} liefert eine einfache Möglichkeit der Preisgenerierung, welche bereits auf die volatilen Preisverläufe, nicht aber auf die aktuelle Entwicklung des Preises während des Tages eingeht. Die Konsumierung der Energiemenge wird dadurch immer in den gleichen Zeitbereich des Tages gelegt, weil dort für jeden Tag das Preistief erwartet

wird. Da die verwendeten Preisdaten sehr gleichmäßig sind, liefert die Optimierung anhand dieser Preisbasis für alle Verbrauchertypen die besten Ergebnisse.

Um eine Aussage darüber treffen zu können, wie gut das Resultat der Optimierung anhand der Preisbasen KONST, DET_{mittel}, STOCH und $STOCH_{SOHM}$ war, wurde auch die Optimierung anhand der Preisbasis DET_{einzel} durchgeführt. Somit kann anhand dieser Resultate abgeschätzt werden, wie gut eine Optimierung an die perfekte Lösung heran gekommen ist.

5.3.3 Preisbasen für stochastische Optimierungen

Die Preisbasis STOCH, welche die Optimierung laut Birge [7] darstellt, wurde in dieser Arbeit nach der Büschel- und der Baummethode umgesetzt. Die Baummethode lieferte zum einen schlechtere Ergebnisse als die Büschelmethode, zum anderen wurde aufgrund der höheren Anzahl an Szenarien auch die Rechenzeit deutlich länger. Daher war die Büschelmethode jene, welche für die Berechnungen herangezogen wurde.

Im Laufe der Arbeit entstand auch die Idee, jeweils zur aktuellen Stunde eine mit den Wahrscheinlichkeiten gewichtete Mittelwertkurve zu verwenden. Diese kann Rechenzeit einsparen, da nur ein Szenario berechnet werden muss, während gleichzeitig auf die aktuelle Preisentwicklung eingegangen wird. Diese Optimierungsvariante zeigt anhand der verwendeten Preisdaten sogar bessere Resultate als die Optimierung laut Birge.

5.4 Modelle der Verbrauchertypen

Die Modellierung der vier Verbrauchertypen sollte genau betrachtet werden. Der Grundgedanke der Arbeit ist sehr einfache Modelle von Verbrauchern zu verwenden. Dies ermöglicht es einen Verbraucher durch Abändern weniger Parameter in einen anderen zu verwandeln. Der neue Verbraucher kann dann mittels der Optimierungssequenzen sofort berechnet werden. Ein Beispiel hierfür ist der Verbrauchertyp D, hier müssten nur die Konstanten c_1, c_2 und m angepasst werden und aus dem Boiler kann beispielsweise eine Batterie gemacht werden. Ein weiteres Beispiel liefert der Verbrauchertyp A. Hier muss nur die Leistung des Gerätes und die benötigte Gesamtenergiemenge für die Fertigstellung der Aufgabe angepasst werden und schon kann anstelle einer Waschmaschine ein Geschirrspüler simuliert werden.

Das Ziel der Arbeit lag also nicht darin einen Verbraucher bis ins letzte Detail zu modellieren, um mit den Simulationsergebnissen genau die Ergebnisse der Wirklichkeit zu treffen. Diese Verbrauchertypen sollten vielmehr derart modelliert werden, dass die

Funktionsweise richtig abgebildet wird, sie aber trotzdem einfach und verständlich gehalten werden. Wichtig ist, dass die prinzipielle Funktionsweise und die Ergebnisse im Vergleich zu anderen Verbrauchertypen und in Bezug auf die unterschiedlichen Strompreisdaten vergleichbar sind und damit eine schnelle Aussage getroffen werden kann.

5.5 Ausblick

5.5.1 Tag-Nacht Preisbasis

Die Preisbasis KONST liefert eine Aussage darüber, welche Kosten resultieren, wenn die Verbraucher (bis auf den Verbrauchertyp D aufgrund seiner Verluste) zu beliebigen Zeitpunkten eingeschaltet werden. Was in dieser Arbeit nicht untersucht wurde, ist die Kostenentwicklung mittels der Optimierung anhand einer auf Tag-Nacht-Strom basierenden Preisbasis. Diese würde eine realistische Aussage in Bezug auf unser aktuelles in Österreich verwendetes Preismodell liefern. Was aber mitberücksichtigt werden muss ist, dass die meisten Verbraucher dennoch tagsüber arbeiten. Somit wäre auch eine Anpassung der Nebenbedingungen der Verbrauchertypen notwendig.

5.5.2 Kombinierte Preisbasis aus det. Mittelwert und Stochastik

Varianten von Preisbasen wurden im Zuge dieser Arbeit nicht untersucht. Da der Verlauf der Preisbasis DET_{mittel} sehr nahe an die meisten Tagesverläufe heran kommt, liefert die Optimierung anhand dieser Preisbasis die besseren Ergebnisse als die stochastischen Optimierungen. Es gibt jedoch Tage, an denen die deterministische Mittelwert-Preisbasis deutlich von den aktuellen Tagesverläufen abweicht. Daher wäre eine weitere interessante Variante für eine mögliche Untersuchung, das Einbeziehen von 50% der Preisbasis DET_{mittel} bei jeder stochastischen Szenarienbildung. Somit würde man sich nahe am Mittelwert bewegen, mit der Stochastik aber dennoch auf unerwartete Schwankungen reagieren können.

5.5.3 Simulation anhand synthetischer Preisdaten

Die verwendeten Preisdaten von EEX und EPEX sind Preise, die aktuell so an der Strombörse gehandelt werden. Diese Daten sind aufgrund des heutigen Strommixes im Mittel sehr ähnlich, weshalb die Optimierung anhand der Preisbasis DET_{mittel}, bis auf die Optimierung mittels perfekten Preiswissens, auch die besten Resultate liefert. Was in dieser

Arbeit nicht behandelt wurde, sind Ergebnisse aus Optimierungen anhand unregelmäßiger Strompreisdaten. Dafür müssten synthetische Preisdaten generiert werden, bei welchen sich das deterministische Mittelwertszenario an den meisten Tagen hinsichtlich der Verlaufsform nicht ähnelt. Erwartungsgemäß müsste eine Optimierung anhand einer stochastischen Methode in diesem Kontext bessere Ergebnisse liefern als eine Optimierung anhand der Preisbasis DET_{mittel}.

5.5.4 Übergreifende Verbrauchersteuerung

Was in dieser Arbeit behandelt wurde, ist die Optimierung einzelner Verbrauchertypen. Die Fragen, die dadurch offen bleiben, sind: Wie wird der Strompreisverlauf beeinflusst, wenn alle Verbraucher zum selben Zeitpunkt einschalten wollen? Und wie wirkt sich das auf die Netzqualität und -stabilität aus? Eine ähnliche Situation ergab sich durch die Sonnenfinsternis am 20. März 2015, jedoch umgekehrt. Folgende Zahlen, welche die Ausmaße in Deutschland darstellen, besagen, dass um ca. 9:00 Uhr MEZ+1 innerhalb von 90 Minuten 5 GW weniger Leistung aus Photovoltaik und weitere 90 Minuten später zum Ende der Sonnenfinsternis wieder 12GW mehr Leistung vorhanden waren. Diese enormen Leistungsgradienten mussten mittels Wasser- und Gasspeicherkraftwerken und der Steuerung großer Verbraucher kompensiert werden.

Falls in Zukunft dem Endnutzer ein volatiler Strompreis angeboten wird, muss sich dieser Gedanken über die übergreifende Steuerung von Verbrauchern machen. Eine Möglichkeit wäre Regionen bezogene Tarifmodelle zu bilden. Ob dies aber wirklich ein gangbarer Weg wäre, müsste dann genau untersucht werden.

5.6 Zusammenfassung

Ziel dieser Arbeit war es eine stochastische Optimierung an vier Verbrauchertypen durchzuführen und damit durch optimale Steuerung minimale Kosten zu generieren. Dies wird auch Demand Side Management (DMS) genannt. Eine grundlegende Annahme war, dass der Endverbraucher einen volatilen Strompreis angeboten bekommt. Als Vergleich wurden konstante Preise, deterministische Mittelwertpreise und auch die Optimierung mittels perfekten Preiswissens herangezogen. Nachdem die Verbrauchermodelle und auch die Preisbasen für die Optimierungen mittels der Sprache Python umgesetzt wurden, konnten die Jahressimulationen gestartet werden. Es resultierte aus jeder Simulation ein über das Jahr gemittelter Tagespreis, welcher einen Eintrag in der Ergebnismatrix darstellte. Im Zuge der Arbeit entstand noch eine Variante der stochastischen Optimierung anhand der Preisbasis $STOCH_{SOHM}$, welche zusätzlich in die Ergebnismatrix mitaufgenommen wurde. Nach ersten Simulationen wurden ebenso Varianten der knn-Methode implementiert, wie auch Parameter bestehender Methoden variiert und anschließend simuliert.

Ergebnis aus den Simulationen ist, dass die Optimierung anhand der Preisbasis DET_{mittel} am nächsten an die Optimierung mittels perfektem Preiswissen herankommt und somit die niedrigsten und damit besten Resultate liefert. Grund dafür ist, dass die Tagespreisverläufe übers Jahr gesehen in der Form sehr ähnlich sind, und damit die Preistäler meistens an denselben Stellen haben. Somit ist auch die Preisbasis DET_{mittel} sehr ähnlich zu den meisten Tagesverläufen, was dazu führt annähern optimale Verbrauchsentscheidungen aus dieser Optimierung zu generieren.

Wird in Zukunft der Anteil an regenerativen Energien am Markt stärker und damit der Tagespreisverlauf möglicherweise unregelmäßiger, so kann die stochastische Optimierung auf diese Unregelmäßigkeiten besser reagieren und im Vergleich zur Preisbasis DET_{mittel} die niedrigeren Kosten generieren. Dies konnte auch an einzelnen Tagesimulationen nachgewiesen werden, deren Tagespreisverlauf unregelmäßiger gegenüber dem deterministischen Mittel war.

6. Glossar

- **DSM** … Demand Side Management, optimale Steuerung verbraucherseitiger Geräte
- **EEX** … European Energy Exchange AG
- **EPEX** … European Power Exchange
- **EVPI** … Expected value of perfect information, ist eine Möglichkeit zur Bewertung einer stochastischen Optimierung zum bestmöglichen Optimierungsergebnis
- **knn** … k-nearest neighbourhood, ist ein Verfahren um die k nächsten Nachbarn zum aktuellen, bekannten Preisverlauf zu finden
- **MEZ** … Mitteleuropäische Zeit
- **Timeline-Methode** … Methode zur Verwendung von Preisen des Vortages, dabei wird der Vortag wie in einer Zeitlinie dem aktuellen Tag voran gestückelt

Literaturverzeichnis

[1] Kall, Wallace (2003): Stochastic Programming 2nd edition

[2] Kepplinger, Huber, Amann, Rheinberger, Petrasch (2014): Active Demand Side Management with Domestic Hot Water Heaters Using Binary Integer Programming

[3] Chen, Wu, Fu (2012): Real-Time Price-Based Demand Response Management for Residential Appliances via Stochastic Optimization and Robust Optimization

[4] Strbac (2008): Demand side management: Benefits and challenges

[5] Kollmann (2014): Lastverschiebung in Haushalt, Industrie, Gewerbe und kommunaler Infrastruktur – Potenzialanalyse für Smart Grids – LOADSHIFT

[6] Publish-industry Verlag (2014): Energie 2.0 Kompendium 2014

[7] Birge (2011): Introduction to Stochastic Programming - 2nd edition

[8] Scheibmayer (2008): Schnelle K-Nearest Neighbor-Algorithmen auf der Basis von Simulated Annealing

[9] Yang, Li, Zhang, Hu; Department of Computing & Information Technology, Fudan University, School of Management, Fudan University (2007): A Fast KNN Algorithm Based on Simulated Annealing

[10] Kumar Nath (2005): An Enhancement of k-Nearest Neighbor Classification using Genetic Algorithm

[11] Siddharth Deokar (2009): Weighted k nearest neighbor

[12] Informationen über EEX, EPEX, One-Day-Ahead und Intraday Handel

https://www.next-kraftwerke.de/wissen/strommarkt/

Abgerufen am 11.06.2015

[13] BP Plc, London (2013): „Energy Outlook 2030“

Anhang

A. Codeumsetzung knn-Methode

```
def knn_probability_fast(knn_anzahl, Stunden, price_history, price_select):

    # Differenzen der einzelnen Spalten bilden
    Diff = price_history[:,0:Stunden]-price_select[0:Stunden]
    # Betrag über die Zeilen bilden (also über die einzelnen Tage)
    dist = LA.norm(Diff, axis=1)
    # Sortieren der Abstände von klein nach groß
    dist_ord = np.sort(dist)
    dist_ord = dist_ord[0:knn_anzahl]
    # Indizes sortiert bekommen
    indizes = np.argsort(dist)
    indizes = indizes[0:knn_anzahl]
    #Kehrwert der sortierten Distanzen bilden
    kehr = 1/(1+dist_ord)
    # Wahrscheinlichkeiten der sortieren Distanzen errechnen
    prob = kehr/(np.sum(kehr))

    # Arrays zusammensetzen
    knn_array_fast_trans = np.vstack((indizes, dist_ord, prob))
    knn_array_fast = knn_array_fast_trans.T

    # Berechnung des gewichteten Strompreis abhängig der Wahrscheinlichkeiten --> für die Sohm'sche Stochastik
    knn_price_fast = np.zeros(np.size(price_select))
    for i in range(knn_anzahl):
        knn_price_fast += price_history[knn_array_fast[i][0]] * knn_array_fast[i][2]

    return knn_price_fast, knn_array_fast
```

Abbildung 22: Codeumsetzung knn-Methode
Quelle: Eigene Ausarbeitung

B. Codeumsetzung stochastische Optimierung laut Sohm

B.1. Codeumsetzung Typ A

```
def optimize_shifting_stoch_sohm(day_index, price_history, price_sel, power, period, verbose=False):
    """
    nicht unterbrechbare verschiebbare Lasten
    mehrfacher Durchlauf für die stochastische Optimierung
    """

    price_real = price_sel[day_index] # Preisverlauf des aktuellen Tages
    len_day = len(price_real) # Stunden innerhalb eines Tages
    knn_numb = 3 # Anzahl der nächsten Nachbarn, die herangezogen werden
    costs = np.zeros(len_day - period)
    search_aktiv = True
    time_line_OFF = True # Auswahl knn-Methode

    if verbose:
            print ('.', end="")

    for i in range(len_day - period):
        if search_aktiv:

            # Stundenindexabhänger Preis bestimmen
            if time_line_OFF == True:
                price_moment, knn_price_array = knn_probability_fast(knn_numb, i+1, price_history, price_real)
            else:
                price_real_prev = price_sel[day_index - 1]
                price_moment, knn_price_array = knn_probability_timeline(knn_numb, i+1, price_history, price_real, price_real_prev)

            price_moment[0:i+1] = price_real[0:i+1].copy()

            for t in range(len_day - period):
                costs[t] = sum(price_moment[t:t + period])*power
            time = np.argmin(costs)
            if time <= i:
                search_aktiv = False
                powers = np.zeros(len_day)
                powers[time:time + period] = power


    cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten

    return powers, cost_real
```

Abbildung 23: Codeumsetzung Typ A laut Sohm
Quelle: Eigene Ausarbeitung

B.2. Codeumsetzung Typ B

```
def optimize_interrupt_stoch_sohm(day_index, price_history, price_sel, power, period, verbose=False):
    """
    unterbrechbare verschiebbare Lasten
    mehrfacher Durchlauf für die stochastische Optimierung
    LP:
    ZF: min dot(prices, powers)
    NBs:
        0 <= powers <= power
        sum(powers) = power*period
    """

    knn_numb = 3 #Anzahl der nahesten Nachbarn, die herangezogen werden
    price_real = price_sel[day_index] # Preisverlauf des aktuellen Tages
    len_day = len(price_real) # Stunden innerhalb eines Tages
    energy_used = 0
    powers = np.zeros(len_day)
    time_line_OFF = True # Auswahl knn-Methode

    if verbose:
            print('.', end="")

    for i in range(len_day-1):

        # Stundenindexabhänger Preis bestimmen
        if time_line_OFF == True:
            price_moment, knn_price_array = knn_probability_fast(knn_numb, i+1, price_history, price_real)
        else:
            price_real_prev = price_sel[day_index - 1]
            price_moment, knn_price_array = knn_probability_timeline(knn_numb, i+1, price_history, price_real, price_real_prev)

        price_moment[0:i+1] = price_real[0:i+1]

        # Zusammenstellen der Zielfunktion und der Nebenbedingungen
        c  = price_moment[i:len_day]
        A1 = np.eye(len_day-i)
        b1 = np.ones(len_day-i)*power
        A2 = -np.eye(len_day-i)
        b2 = np.zeros(len_day-i)
        A  = np.vstack((A1,A2))
        b  = np.concatenate((b1,b2))
        G  = np.ones((1,len_day-i))
        h  = power*period-energy_used

        [x,s,z,y,status] = lp_solve(c,A,b,G,h)
        #print("Solver status = %s" %status)

        energy_used += round(x[0],3)  # Aufsummieren der Energieverbräuche, damit für die folgenden Optimierungsdurchläufe
                                      # der erlaubte Restverbruach angepasst wird. x entspricht der Verbrauch im aktuellen Schritt
        powers[i] = x[0] # Verbrauchsentscheidung des aktuellen Optimierungsdurchlauf zuweisen

    powers[len_day-1] = power*period-energy_used

    cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten

    return powers, cost_real
```

Abbildung 24: Codeumsetzung Typ B laut Sohm
Quelle: Eigene Ausarbeitung

B.3. Codeumsetzung Typ C

```
def optimize_storage_simple_stoch_sohm(day_index, price_history, price_sel, energy_provided, power_load_max,
                                       efficiency=1, verbose=False):
    """
    unterbrechbare verschiebbare Lasten bzw. Speicher mit Wirkungsgrad
    mehrfacher Durchlauf für die stochastische Optimierung
    LP:
    ZF: min dot(prices, powers)
    NBs:
        0 <= powers <= power_load_max
        sum(powers)*efficiency = energy_provided
    """

    knn_numb = 3 #Anzahl der nahesten Nachbarn, die herangezogen werden
    price_real = price_sel[day_index] # Preisverlauf des aktuellen Tages
    len_day = len(price_real) # Stunden innerhalb eines Tages
    energy_used = 0
    powers = np.zeros(len_day)
    time_line_OFF = True # Auswahl knn-Methode

    if verbose:
            print('.', end="")

    for i in range(len_day-1):

        # Stundenindexabhänger Preis bestimmen
        if time_line_OFF == True:
            price_moment, knn_price_array = knn_probability_fast(knn_numb, i+1, price_history, price_real)
        else:
            price_real_prev = price_sel[day_index - 1]
            price_moment, knn_price_array = knn_probability_timeline(knn_numb, i+1, price_history, price_real, price_real_prev)

        price_moment[0:i+1] = price_real[0:i+1]

        # Zusammenstellen der Zielfunktion und der Nebenbedingungen
        c  = price_moment[i:len_day]
        A1 = np.eye(len_day-i)
        b1 = np.ones(len_day-i)*power_load_max
        A2 = -np.eye(len_day-i)
        b2 = np.zeros(len_day-i)
        A  = np.vstack((A1,A2))
        b  = np.concatenate((b1,b2))
        G  = np.ones((1,len_day-i))
        h  = energy_provided/efficiency-energy_used

        [x,s,z,y,status] = lp_solve(c,A,b,G,h)
        #print("Solver status = %s" %status)

        energy_used += round(x[0],3)  # Aufsummieren der Energieverbräuche, damit für die folgenden Optimierungsdurchläufe
                                      # der erlaubte Restverbruach angepasst wird. x entspricht der Verbrauch im aktuellen Schritt
        #print(energy_used)
        powers[i] = x[0] # Verbrauchsentscheidung des aktuellen Optimierungsdurchlauf zuweisen

    powers[len_day-1] = energy_provided/efficiency-energy_used

    cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten

    return powers, cost_real
```

Abbildung 25: Codeumsetzung Typ C laut Sohm
Quelle: Eigene Ausarbeitung

B.4. Codeumsetzung Typ D

```
def optimize_storage_exp_stoch_sohm(times, day_index, price_history, price_sel, energy_provided, power_load_max,
                                    lbd=0, verbose=False):
    """
    Speicher mit DGL $\dot{E} + \lambda E = P$ with $E(0)=0$
    mehrfacher Durchlauf für die stochastische Optimierung
    LP:
    ZF: min dot(prices, powers)
    NBs:
        0 <= powers <= power_load_max
        sum(energies_end) = energy_provided
    """

    knn_numb = 3 #Anzahl der nahesten Nachbarn, die herangezogen werden
    price_real = price_sel[day_index] # Preisverlauf des aktuellen Tages
    len_day = len(price_real) # Stunden innerhalb eines Tages
    energy_used = 0
    powers = np.zeros(len_day)
    time_line_OFF = True # Auswahl knn-Methode

    if verbose:
        print('.', end="")

    for i in range(len_day-1):

        # Stundenindexabhänger Preis bestimmen
        if time_line_OFF == True:
            price_moment, knn_price_array = knn_probability_fast(knn_numb, i+1, price_history, price_real)
        else:
            price_real_prev = price_sel[day_index - 1]
            price_moment, knn_price_array = knn_probability_timeline(knn_numb, i+1, price_history, price_real, price_real_prev)

        price_moment[0:i+1] = price_real[0:i+1]
        times_moment = times[0:len_day-i]

        # Zusammenstellen der Zielfunktion und der Nebenbedingungen
        dt = 1
        c  = price_moment[i:len_day]
        A1 = np.eye(len_day-i)
        b1 = np.ones(len_day-i)*power_load_max
        A2 = -np.eye(len_day-i)
        b2 = np.zeros(len_day-i)
        A  = np.vstack((A1,A2))
        b  = np.concatenate((b1,b2))

        if lbd == 0: f=1 # Für den Fall, dass lbd = 0 --> f = 1 --> das Ergebnis muss dem Verbrauchertyp B entsprechen
        else: f  = (1 - np.exp(-lbd*dt))/lbd
        G  = f*np.exp(-lbd*(times_moment[-1] - np.array([times_moment])))

        if (energy_provided-energy_used) < 0: h = 0
        else: h  = energy_provided-energy_used

        [x,s,z,y,status] = lp_solve(c,A,b,G,h)
        #print("Solver status = %s" %status)

        energy_used += (np.ceil(x[0]*1000)/1000)*f*np.exp(-lbd*(times_moment[-1])) # ceil ist zum Aufrunden, damit solver kein Prob
                                    # Aufsummieren der Energieverbräuche, damit für die folgenden Optimierungsdurchläufe
                                    # der erlaubte Restverbruach angepasst wird. x entspricht der Verbrauch im aktuellen Schritt
        #print(energy_used)
        powers[i] = x[0] # Verbrauchsentscheidung des aktuellen Optimierungsdurchlauf zuweisen

    powers[len_day-1] = (energy_provided-energy_used)/f  # Benötigte Leistung der letzten Stunde zuweisen

    cost_real = np.dot(price_real.T,powers) #Berechnung der realen Kosten

    return powers, cost_real
```

Abbildung 26: Codeumsetzung Typ D laut Sohm
Quelle: Eigene Ausarbeitung

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dornbirn, am 31. Juli 2015 Thomas Sohm