Lehrstuhl für Energieverbundtechnik

Masterarbeit

Modellierung des elektrischen Energiesystems eines Elektrostahlwerkes

Aaron Maximilian Marschnig, BSc

Januar 2021

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich die Richtlinien des Senats der Montanuniversität Leoben zu "Gute wissenschaftliche Praxis" gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 08.01.2021

Aaron Marschnig

Unterschrift Verfasser/in
Aaron Maximilian Marschnig

KURZFASSUNG

Durch die nahezu vollständige Recyclingfähigkeit von Stahl steigt der Anteil der Stahlproduktion über die Schrott-Elektrolichtbogenofenroute weltweit tendenziell an. Verglichen mit anderen Prozessrouten stellt sich die Produktionsroute Schrott-Elektrolichtbogenofen deutlich umweltverträglicher dar, da diese neben dem geringeren spezifischen Energieverbrauch auch einen geringeren Emissionsausstoß aufweist. Um weitere Optimierungsmöglichkeiten für diese Prozessroute, speziell im Hinblick auf die Energieeffizienz und das Demand Side Management (DSM), ableiten zu können, benötigt man Lastprofile der Verbraucher für die unterschiedlichen betrieblichen Lastsituation.

Das Ziel dieser Arbeit ist es, ein Energiesystemmodell der elektrischen Verbraucher für das Stahlwerk der Breitenfeld Edelstahl AG zu entwickeln. Dieses Modell soll die Möglichkeit zur Analyse und Bewertung von Optimierungspotentialen bieten. Dazu werden für die energie- und leistungsmäßig relevanten Aggregate im Werk die betrieblichen Leistungsdaten erfasst. Diese realen elektrischen Lastszenarien werden analysiert und aus den spezifischen Daten der einzelnen Aggregate synthetische Lastprofile, die eine zeitliche Auflösung für die Wirk-, Blind- und Scheinleistung von einer Minute aufweisen, gebildet. Zur Modellierung der synthetischen Lastprofile werden einheitliche Konzepte ermittelt und angewandt, in denen unterschiedliche stochastische wie auch statistische Methoden angewendet werden. Durch Verknüpfung der einzelnen synthetischen Lastprofile erhält man das Energiesystemmodell der elektrischen Verbraucher, das den Standort des Stahlwerks abbildet.

Abschließend erfolgt eine Validierung der synthetisch erzeugten Lastprofile für die jeweiligen Aggregate sowie für das gesamte Energiesystemmodell. Dafür werden die aus der Simulation erzeugten synthetischen Leistungsdaten den realen Leistungsdaten gegenübergestellt und die Ergebnisse hinsichtlich ihrer Genauigkeit analysiert und bewertet.

ABSTRACT

The share of steel production via the scrap-electric arc furnace route tends to increase worldwide, since steel can be recycled almost completely. In addition to the lower specific energy consumption and lower emissions, the scrap-electric arc furnace route is more environmentally friendly than other steel production routes. In order to be able to derive further optimization possibilities for this process route regarding energy efficiency and demand side management, load profiles of the consumers are required for the different operational load situations in order to be able to evaluate further optimization options for this process route.

The aim of this master's thesis is to develop an electric energy system model for the Breitenfeld Edelstahl AG steel mill. This model is aimed to offer the possibility of analyzing and evaluating potentials for demand side management and optimization options. For this purpose, process parameters and power are recorded for the respective electrical units in the steel mill. These real electrical load scenarios are analyzed, and synthetic load profiles are created for active, reactive and apparent power with a temporal resolution of one minute. For modeling the synthetic load profiles, a uniform concept is established and different stochastic as well as statistical methods are used. By combining the individual synthetic load profiles, the energy system model is obtained, which depicts the energy system of the steel mill.

Finally, the synthetically generated load profiles for the respective units as well as the overall energy system are validated. For this purpose, the data generated from the simulation is compared with the measured data and the results are analyzed and evaluated regarding to their accuracy.

VORWORT

Die vorliegende Arbeit entstand während meiner Tätigkeit als studentischer Mitarbeiter am Lehrstuhl für Energieverbundtechnik der Montanuniversität Leoben im Rahmen des NEFI-Projekts OxySteel. Ich möchte mich an dieser Stelle bei all jenen bedanken, die mich bei der Erstellung dieser Masterarbeit unterstützt haben.

Mein besonderer Dank gilt Herrn Univ.-Prof. Dipl.-Ing. Dr.techn. Thomas Kienberger für die Ermöglichung dieser Masterarbeit. Herrn Dipl.-Ing. Johannes Dock danke ich für die angenehme und unkomplizierte Zusammenarbeit und für die hervorragende fachliche Betreuung.

Ein weitere Dank gilt meinen Kommilitonen vor allem für den guten Zusammenhalt während des Studiums und für die Hilfsbereitschaft auch in schwierigen Zeiten. Bei meinen Freunden und meiner Freundin bedanke ich mich besonders für den emotionalen Rückhalt während des Studiums.

Abschließend möchte ich mich bei meiner Familie, insbesondere meinen Eltern, für die Ermöglichung meines Studiums und für die Unterstützung all meiner Entscheidungen bedanken.

INHALTSVERZEICHNIS

NOMENKLATUR

Abkürzungen

BOF	Basic Oxygen Furnace (dt. Linz-Donawitz Verfahren)
CO	Kohlenstoffmonoxid
CO_2	Kohlenstoffdioxid
COREX-Verfahren	Schmelzreduktionsverfahren
cosφ	Leistungsfaktor
DRI	Direct Reduced Iron (dt. Eisenschwamm)
E	Elektrische Energie
EAF	Electric Arc Furnace (dt. Elektrolichtbogenofen)
EU 28	Mitgliedsstaaten der Europäischen Union
FINEX-Verfahren	Schmelzreduktionsverfahren
H_2	Wasserstoff
HBI	Hot Briquetted Iron (dt. heißbrikettierter Eisenschwamm)
HYL-Verfahren	Direktreduktionsverfahren
LF	Ladle Furnace (dt. Pfannenofen)
LS	Liquid Steel (dt. Flüssiger Stahl)
MIDREX-Verfahren	Direktreduktionsverfahren
NG	Natural Gas (dt. Erdgas)
P	Wirkleistung
Q	Blindleistung
S	Scheinleistung
Tap-to-tap-Zeit	Zeitspanne zwischen zwei Abstichen
VD-Verfahren	Vacuum Degassing (dt. Vakuumentgasung)
VOD-Verfahren	Vacuum Oxygen Decarburisation (dt. Entkohlungsprozess)
UHP-Furnace	Ultra High Power Furnace (dt. Hochleistungsofen)

Indizes

°C	Grad Celsius
GJ	Gigajoule
kg	Kilogramm
kW	Kilowatt
kWh	Kilowattstunde
m^3	Kubikmeter
Mio.	Million(en)
Mrd.	Milliarde(n)
MW	Megawatt
MWh	Megawattstunde
Nm^3	Normkubikmeter
t	Tonne
Tsd.	Tausend

ABBILDUNGSVERZEICHNIS

Abbildungsverzeichnis

TABELLENVERZEICHNIS

1 EINLEITUNG

Um die von der europäischen Union gesetzten Klimaziele zu erreichen, steht speziell die produzierende Industrie vor großen Herausforderungen. Um diese Herausforderungen zu meistern sind in der EU-Verordnung 2018/1999 für die Energieunion fünf Dimensionen festgelegt worden, von denen die Dimensionen Dekarbonisierung und Energieeffizienz speziell in der Stahlindustrie eine wichtige Rolle tragen, da die Stahlindustrie zu den energie- und CO_2-intensiven Industriesektoren zählt. [1] Um die geforderten Ziele zu erfüllen, muss vor allem eine Steigerung der Energieeffizienz, die Erhöhung des Anteils bzw. der vollständige Umstieg auf Energie aus erneuerbaren Quellen und die Reduktion der prozessbedingten Treibhausgasemissionen erfolgen. Für die Umsetzung der in der EU-Verordnung geforderten Ziele wird im integrierten nationalen Energie- und Klimaplan für Österreich von sogenannten Break-Through-Technologien für die Industrie gesprochen. Bei diesen Technologien handelt es sich um hocheffiziente intelligente Industrieprozesse, die beispielsweise Potentiale zur Rückgewinnung von Energie oder die Möglichkeit der Teilnahme am sektorgekoppelten Energieverbund bieten. Mit diesen Technologien soll bei einem gleichbleibenden Produktoutput eine Reduktion von Rohstoff- und Energieverbräuchen erreicht werden und die Dekarbonisierung industrieller Prozesse und Produkte erfolgen, beispielsweise eine CO_2-neutrale Stahlerzeugung. [2]

Konkret wird in dieser Arbeit das Elektrostahlwerk der Breitenfeld Edelstahl AG mit Standort im Mürztal betrachtet, bei dem Stahl über die Schrott-Elektrolichtbogenofenroute produziert wird. Bei dieser Art der Stahlproduktion wird Schrott in einem Elektrolichtbogenofen (EAF – Electric Arc Furnace), der ein Fassungsvermögen von 65 Tonnen aufweist, chargiert und aufgeschmolzen. Bei einer Standardcharge erfolgt nach dem Abstich am Elektrolichtbogenofen das Aufheizen und Homogenisieren der Stahlschmelze am Pfannenofen. Ist die Stahlschmelze homogenisiert und weist die gewünschte Temperatur auf, erfolgt die Vakuumentgasung, auch VD (Vacuum Degassing)-Verfahren. Durch das VD-Verfahren wird die Stahlreinheit verbessert und gelöste Gase werden entfernt. Vor dem Blockguss erfolgt ein weiteres Aufheizen der Stahlschmelze am Pfannenofen, bis die Gusstemperatur erreicht ist. Beim Blockguss wird der flüssige Stahl in Dauerformen, die verschiedene Querschnitte annehmen können, gegossen. Nach dem Guss erfolgen noch weitere Behandlungen, bei denen die gewünschten Eigenschaften des Stahlprodukts eingestellt werden können. Neben Standard-Stählen können mittels VOD (Vacuum Oxygen Decarburisation)-Verfahren auch Stähle mit hohem Chrom- und niedrigen Kohlenstoffgehalten am Standort im Mürztal hergestellt werden. [3]

Die Stahlproduktion über die Schrott-Elektrolichtbogenofenroute weist im Vergleich zur integrierten Produktionsroute über den Hochofen und Konverter viele Vorteile auf. Zu diesen Vorteilen zählen vor allem die Schonung von natürlichen Ressourcen und Rohstoffen, der geringere spezifische Energieverbrauch und die dadurch verbundene Reduktion der prozessbedingten und energiebezogenen Emissionen. Dennoch ist die Stahlproduktion über die Schrott-Elektrolichtbogenofenroute sehr energieintensiv. Der hohe Energieverbrauch muss dabei auch, teilweise wegen prozessbedingter Faktoren, durch fossile Brennstoffe gedeckt werden. Um nun Potentiale in Hinblick auf Optimierungsmöglichkeiten identifizieren und bewerten zu können, benötigt man ein Energiesystemmodell, dass die realen Prozesse und die damit verbundenen Parameter des Stahlwerks wiedergibt.

Die vorliegende Arbeit ist in acht Kapitel gegliedert. In Kapitel 1 erfolgt eine Einleitung in die Thematik dieser Arbeit und die Begründung, warum die in dieser Arbeit behandelte Forschungsfrage relevant ist. In Kapitel 2 wird die Aufgabenstellung definiert und die Herangehensweise zur Lösung und Erreichung der festgelegten Ziele beschrieben. Um die Relevanz dieser Thematik zu begründen, erfolgt in Kapitel 3 ein kurzer Überblick über den aktuellen Stand der Technik in der Stahlproduktion. Bei diesem Überblick wird das Hauptaugenmerk auf die Stahlerzeugung mittels der Schrott-Elektrolichtbogenofenroute gelegt. Dabei erfolgt ein Vergleich mit anderen Prozessrouten. Des Weiteren wird die Entwicklung dieser Stahlproduktionsroute und die umweltrelevanten Aspekte, wie der spezifische Energieverbrauch und die prozessbedingten und energiebezogenen CO_2-Emissionen, thematisiert. In Kapitel 4 ist die Modellierung des Energiesystemmodells der elektrischen Verbraucher beschrieben. Dabei wird in den verschiedenen Abschnitten erläutert, mit welchen Methoden aus den Leistungsdaten und Prozessparametern der realen Lastprofile die synthetischen Lastprofile erstellt werden. In Kapitel 5 erfolgt der Vergleich der realen und synthetischen Lastprofile. Dazu werden diese graphisch gegenübergestellt und Dauerlinien der Wirkleistung für unterschiedliche Betrachtungszeiträume entwickelt und verglichen. In diesem Kapitel werden auch die Abweichungen zwischen dem Energiesystemmodell und dem realen Gesamtlastprofil bewertet. In Kapitel 6 erfolgt eine Zusammenfassung der vorliegenden Arbeit und eine Diskussion der Ergebnisse.

2 AUFGABENSTELLUNG

Das zentrale Ziel der vorliegenden Arbeit war die Entwicklung eines Energiesystemmodells, welches die elektrischen Verbraucher des Stahlwerks der Breitenfeld Edelstahl AG abbildet. Das Energiesystemmodell soll die Möglichkeit der Evaluierung von Optimierungspotentialen bieten. Konkret handelt es sich bei den betrachteten Potentialen um Demand Side Management Maßnahmen und Flexibilitätsoptionen um eine Integration von fluktuierenden, erneuerbaren Energieträgern zu ermöglichen. Generell muss also den Fragen nachgegangen werden, ob es überhaupt Flexibilitäten gibt und welche Möglichkeiten sich zum Einsatz von Demand Side Management bieten. Des Weiteren soll das Energiesystemmodell so flexibel wie möglich gestaltet werden, damit auch zukünftige Fragestellungen und Szenarien mit diesem Modell simuliert und bewertet werden können.

Um diese Forschungsfragen zu beantworten, wurden vor Ort die Leistungsdaten der Hauptenergieverbraucher des Stahlwerks, angefangen mit dem Elektrolichtbogenofen, nacheinander erhoben. Die erhobenen Lastprofile der jeweiligen Aggregate wurden charakterisiert und statistisch ausgewertet. Danach wurden zwei Grundkonzepte zur Modellierung der synthetischen Lastprofile der elektrischen Verbraucher erstellt. Diese zwei Modellierungskonzepte sind für sämtliche im Energiesystemmodell beschriebenen elektrischen Verbraucher anwendbar. Mit diesen Grundkonzepten wurden aus den Betriebsparametern und Daten der realen Lastprofile der verschiedenen Aggregate des Stahlwerks, synthetische Lastprofile modelliert und miteinander zu einem gesamten Energiesystemmodell verknüpft. Die Entwicklung des Energiesystemmodells erfolgte mit der Programmiersprache Python in der Entwicklungsumgebung PyCharm. Parallel zum Prozess der Modellentwicklung erfolgte die Literaturrecherche zu den Themenbereichen, die für diese Arbeit relevant sind.

3 STAND DER TECHNIK DER EISEN- UND STAHLPRODUKTION

Aufgrund seiner technischen Eigenschaften ist Stahl eines der anpassungsfähigsten und vielseitigsten Materialien. Die vielen Einsatzbereiche wie beispielsweise die Automobil- und Schifffahrtsindustrie, der Hoch- und Tiefbau, die Luft- und Raumfahrttechnik, der Kraftwerksbau und die Energiewirtschaft machen Stahl zum vielseitigsten Industriematerial der Welt. Dadurch, dass Stahl eine nahezu vollständige Recyclingfähigkeit aufweist, ist dieser auch ein Schlüsselmaterial der Kreislaufwirtschaft. Stahl ist als technischer Grundstoff auch ein wesentlicher Faktor für die Entwicklung und den Einsatz innovativer, CO_2-reduzierender Technologien, die Verbesserung der Ressourceneffizienz und die Förderung einer nachhaltigen Entwicklung in Europa. [4]

Die europäische Stahlindustrie ist ein wesentlicher Bestandteil der europäischen Wirtschaft und ist auch weltweit führend in Bezug auf Innovation und ökologische Nachhaltigkeit. Mit Stand 2018 umfasste die Stahlindustrie in Europa knapp 2,6 Mio. Arbeitsplätze, von denen 320.000 direkt in der Stahlindustrie angesiedelt waren. Die restlichen Arbeitsplätze teilen sich in indirekte und induzierte Arbeitsplätze auf. Unter Berücksichtigung aller durch mit der Stahlindustrie verknüpften Arbeitsplätze betrug für das Jahr 2018 die Bruttowertschöpfung 148 Mrd. Euro und mit der Produktion wurden 17,2 Prozent der weltweiten Rohstahlproduktion abgedeckt. Das bedeutet, dass durchschnittlich 170 Mio. Tonnen Stahl pro Jahr an mehr als 500 Produktionsstandorten in Europa produziert werden. 2018 wurden davon 20,6 Mio. Tonnen Stahl an Stahlendprodukten exportiert und im Gegenzug 29,3 Mio. Tonnen Stahlendprodukte importiert. Jährlich fallen über 40 Mio. Tonnen an Schlacke als Nebenprodukt der Stahlerzeugung an. Durch neue Technologien können auch die Schlacken der Stahlproduktion für unterschiedliche Bereiche, von der Bauindustrie bis hin zum Düngemittel, verwendet werden. [5]

Laut Norm DIN EN 10 200 wird Stahl als die Werkstoffgruppe, dessen Hauptelement Eisen ist, bezeichnet. Das charakteristische Legierungselement von Stahl ist Kohlenstoff, welches auch zur Einteilung der Stähle herangezogen wird. Dafür wird ein Massenanteil von zwei Prozent Kohlenstoff als Grenzwert festgelegt. Unter zwei Prozent Kohlenstoffanteil spricht man von Stahl und darüber von Gusseisen. Einige Chromstähle können jedoch mehr als zwei Prozent Kohlenstoff enthalten. Je nach Einsatzgebiet und erwünschter Qualität werden dem Stahl Legierungselemente hinzugefügt. Durch die Vielzahl der Legierungselemente und der vielen unterschiedlichen Kombinationsmöglichkeiten sind heute etwa 2500 Stahlsorten verfügbar. Grundsätzlich werden Stähle nach ihrer chemischen Zusammensetzung in die drei Hauptgüteklassen eingeteilt: unlegierte Stähle, nichtrostende Stähle und legierte Stähle. [6]

3.1 Überblick der Prozessrouten der Eisen- und Stahlherstellung

Die Erzeugung von Stahl kann in die vier in Abbildung 3-1 dargestellten Hauptprozessrouten eingeteilt werden. Eine weitere Einteilung kann in primäre und sekundäre Erzeugung von Stahl erfolgen. Die integrierte Route sowie die Route der Schmelz- und Direktreduktion zählen zur primären Stahlerzeugung. Die Route Schrott-Elektrolichtbogenofen gilt als sekundäre Stahlerzeugung. Diese vier Routen werden in diesem Abschnitt beschrieben und im weiteren Verlauf der Arbeit wird die Stahlerzeugung aus Schrott über die Elektrolichtbogenofenroute genauer betrachtet.

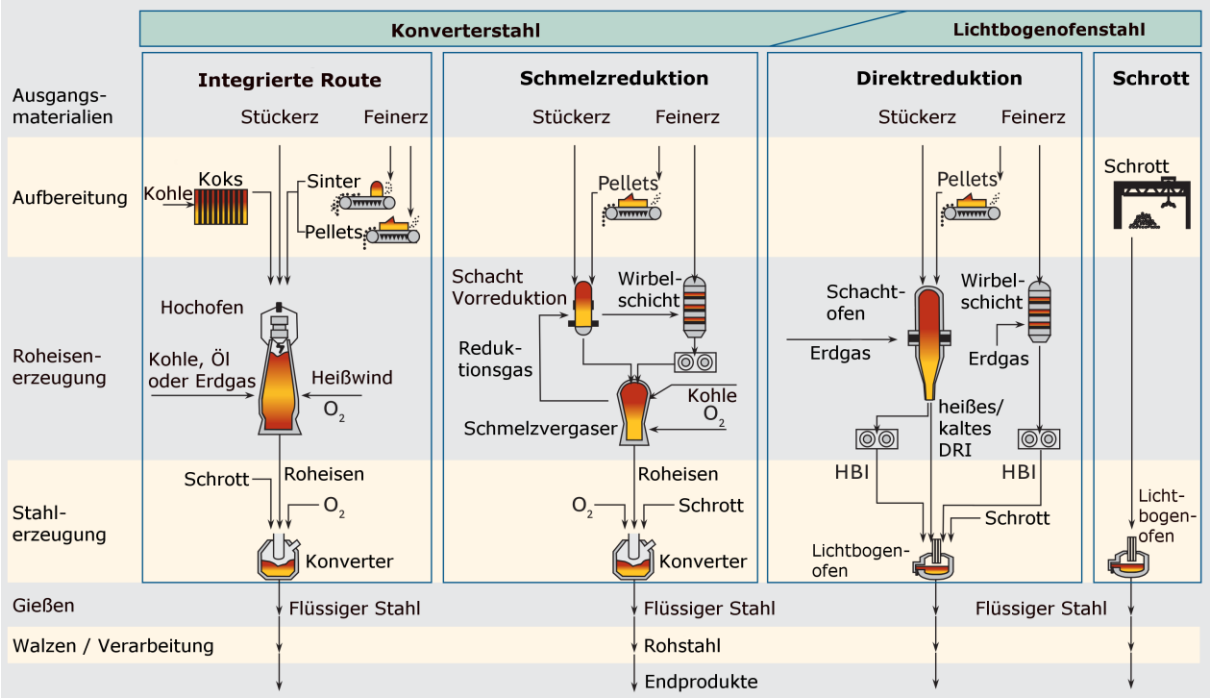

Abbildung 3-1: Prozessrouten der Eisen- und Stahlherstellung [4]

3.1.1 Integrierte Hochofenroute

Bei der integrierten Route über den Hochofen dienen Stückerz, Feinerz und Kohle als Ausgangsmaterialien. Das Feinerz kann vor Ort pelletiert werden, jedoch erfolgt das Pelletieren meist direkt beim Erzlieferanten. Vor dem Einsatz im Hochofen wird das Feinerz gesintert. Dazu wird das Feinerz mit Koksgrus und weiteren Zuschlägen in Bandsinteranlagen, durch Zündung der Oberfläche, gebacken. Als Reduktionsmittel dient Koks, der hauptsächlich aus gemahlener Steinkohle gewonnen wird. [4], [6]

Die Roheisenerzeugung findet im Hochofen statt, einem im Gegenstrom betriebenen schachtförmigen Reaktor. Betrieb im Gegenstrom bedeutet, dass die chargierten Feststoffe auf der oberen Seite eingebracht werden und nach unten wandern und die Gasphase sich entgegen dieser Richtung bewegt. Die Feststoffe werden dabei chemisch und physikalisch

verändert. Im Winderhitzer wird Luft auf eine möglichst konstante Temperatur von ca. 1200 Grad Celsius erhitzt. Über eine Ringleitung über Düsen wird die erhitzte Luft (Wind) in den Hochofen eingeblasen. Der eingeblasene Sauerstoff vergast den Koks zu CO, was zu einer starken Temperaturerhöhung führt. Kohlestaub, Öl und Erdgas können als teilweiser Ersatz für das Reduktionsmittel Koks verwendet werden. Die aus den Einsatzstoffen durch chemische und physikalische Reaktionen am Hochofen produzierte Hauptprodukte sind flüssiges Roheisen, flüssige Schlacke und Gichtgas. [4], [6]

Die Stahlerzeugung findet bei der integrierten Hochofenroute im Konverter statt. Dafür wird das flüssige Roheisen in den Konverter chargiert und reiner Sauerstoff auf die Schmelze aufgeblasen, um die verbliebenen unerwünschten Verunreinigungen zu entfernen. Weiters wird in diesem Schritt der Kohlenstoffgehalt auf das benötigte Niveau der erforderlichen Qualität, typischerweise unter zwei Prozent, eingestellt. Da die Prozesse am Konverter stark exotherm sind und somit viel Energie freisetzen, muss eine Kühlung der Stahlschmelze erfolgen. Zum Kühlen wird in den meisten Fällen Schrott, aber manchmal auch kaltes Roheisen oder direkt reduziertes Eisen hinzugefügt. [4], [6]

3.1.2 Integrierte Schmelzreduktionsroute

Beim integrierten Schmelzreduktionsverfahren wird der Hochofenprozess, also die Aufgabe der Roheisenerzeugung, durch einen zweistufigen Prozess ersetzt. Die Schmelzreduktion kombiniert den Prozess der Direktreduktion, also der Vorreduktion von Eisen zu Eisenschwamm, mit einem Schmelzprozess, der Hauptreduktion. Bei den Schmelzreduktionsverfahren unterscheidet man zwischen COREX- und FINEX-Verfahren. Beim COREX-Verfahren wird Stückerz oder pelletiertes Erz in einem Reaktionsschacht im Gegenstrom mit dem Prozessgas aus dem Schmelzvergaser vorreduziert. Dadurch wird als Produkt Eisenschwamm erhalten, welcher zum Schmelzvergaser transportiert und dort im folgenden Schritt eingeschmolzen wird. Die partielle Verbrennung der zugeführten Kohle erzeugt die benötigte Wärme zum Schmelzen des Eisenschwamms und durch direkten Kontakt mit Kohlenstoff erfolgt die weitere Reduktion des Erzes. Gleichzeitig werden die meisten unerwünschten Elemente der Eisenerze und der Kohlenasche verschlackt. Durch Vergasung von Kohle mit Sauerstoff wird im oberen Teil des Einschmelzvergasers das benötigte Reduktionsgas für den Schachtreaktor gebildet. Nach dem Entwässern und der Entgasung bildet sich im Einschmelzvergaser ein Fest- oder Wirbelbett aus. Ähnlich wie beim Hochofen wird die Schlacke und das Roheisen am Boden abgeführt. Beim FINEX-Verfahren kann das Feinerz direkt ohne Pelletierung im Prozess eingesetzt werden. Das Feinerz wird anstelle des Schachtreaktors, in einem mehrstufigen Wirbelschichtreaktor vorreduziert. Wie beim COREX-Verfahren wird das Reduktionsgas aus dem Einschmelzvergaser abgezogen. Der

restliche Prozess gestaltet sich wie beim COREX-Verfahren. [4], [7] Bei beiden Verfahren erfolgt die Stahlerzeugung aus dem Roheisen gleich wie bei der integrierten Hochofenroute im Konverter.

3.1.3 Integrierte Direktreduktionsroute

Die Ausgangsstoffe bei der integrierten Direktreduktionsroute sind dieselben wie bei der Schmelzreduktionsroute. Das MIDREX- und das HYL-Verfahren sind die meistverbreiteten Verfahren der Direktreduktionsroute. Bei beiden Verfahrensprinzipien wird das Eisenerz in einem Schachtofen reduziert und nicht aufgeschmolzen. Zusätzlich zu diesen Verfahren gibt es andere Routen, die beispielsweise wie in der obigen Abbildung Wirbelschichtreaktoren anwenden. [4]

Beim MIDREX-Verfahren wird einem Schachtofen, der ein geschichtetes Bett aus Erzpellets oder Stückerz enthält, Erdgas, das als Reduktionsmittel dient, im Gegenstrom zugeführt. Zuvor wird dafür das Erdgas in einem Gasreformer zu einem Synthesegas, bestehend aus H_2 und CO, aufgespalten. Bei der Reduktion des Erzes mit dem Synthesegas wird der Sauerstoff aus dem Erz gebunden und dabei entstehen H_2O und CO_2. Ein Vorteil der Reduktion mit Erdgas ist, dass weniger CO_2 als bei der Reduktion mit Koks entsteht. Das aus dem Schachtofen erhaltene Produkt ist direkt reduziertes Eisen (DRI – Direct Reduced Iron), welches direkt dem Elektrolichtbogenofen zugeführt werden kann. Aufgrund seiner pyrophoren Eigenschaften wird das DRI aber oft zu heißbrikettiertem Eisen (HBI – Hot Briquetted Iron) weiterverarbeitet, da es sich besser für Überseetransporte eignet. [8]

Die Stahlerzeugung erfolgt bei der Direktreduktionsroute am Elektrolichtbogenofen. Hierzu wird DRI oder HBI durch Einsatz von elektrischer Energie, die über Elektroden in thermische Energie umgewandelt wird, aufgeschmolzen. Durch Zugabe nichtmetallischer Einsatzstoffe wird am Elektrolichtbogenofen eine Schlacke gebildet, welche die unerwünschten Elemente bindet. Der Stahl kann am Elektrolichtbogenofen legiert werden, jedoch erfolgt die Sekundärmetallurgie meist in einem weiteren Verfahrensschritt an einem Pfannenofen. [6]

3.1.4 Elektrolichtbogenofenroute

Bei der Stahlerzeugung über die Elektrolichtbogenofenroute wird Stahlschrott im Elektrolichtbogenofen aufgeschmolzen. Dabei erfolgt der Wärmeeintrag hauptsächlich über Wärmestrahlung zwischen den Elektroden und dem eingesetzten Schrott. Ein zusätzlicher thermischer Energieeintrag kann durch das Zuführen von Sauerstoff bzw. den Einsatz von Sauerstoffbrennern erfolgen. [9]

Die Elektrolichtbogenofenroute ist die Einzige der vier beschriebenen Routen, die die Herstellung von Spezialstählen ermöglicht. Die mögliche Stahlproduktion wird bei diesem Verfahren durch die allgemeine Verfügbarkeit von Schrott innerhalb eines Landes oder einer Region limitiert. Weiters schränkt die Qualität des verfügbaren Schrotts den Bereich der Stahlqualitäten, die im Prozess erzeugt werden können, ein. Deswegen findet Elektrostahl aus Schrott hauptsächlich in der Bauindustrie Anwendung. Für Industrien, die höhere Qualitätsanforderungen an den Stahl stellen, müssen wegen der Schrottunreinheiten zusätzlich noch Eisenschwamm oder Roheisen als Einsatzgüter verwendet werden, wodurch diese Einsatzgüter als limitierende Faktoren auftreten können. Beispiele für hochqualitative Elektrostähle sind Kohlenstoffstähle für die Automobilindustrie. [4], [6]

3.2 Stahlproduktion im weltweiten Vergleich

Die Rohstahlproduktion in Österreich basiert vollständig auf der integrierten Hochofenroute und der Schrott-Elektrolichtbogenofenroute. In Tabelle 3-1 sind die Produktionsmengen der Stahlindustrie für Österreich, die Europäische Union und der weltweit produzierten Menge in Tsd. Tonnen dargestellt. Im Jahr 2017 entfielen auf die integrierte Hochofenroute, also die Stahlproduktion über den Konverter (BOF) 91,1 Prozent. Die restlichen 8,9 Prozent wurden über die Schrott-Elektrolichtbogenofenroute (EAF) hergestellt. Im europäischen Vergleich wird in Österreich deutlich mehr Stahl, prozentuell gemessen an der inländischen Gesamtproduktion, über die Hochofen-Konverter-Route hergestellt. Der Anteil an Elektrostahl ist in Europa, verglichen mit dem weltweiten Anteil um ca. 12 Prozent höher. Andere Verfahren wie die Direkt- oder Schmelzreduktion haben in Europa nur eine äußerst geringe Bedeutung und weltweit betrachtet nur einen sehr geringen Anteil an der produzierten Stahlmenge. [10]

Tabelle 3-1: Vergleich der Stahlproduktion nach Prozessverfahren für das Jahr 2017 [10]

	BOF		**EAF**		**Total**
	[Tt]	[%]	[Tt]	[%]	[Tt]
Österreich	7 412	91,1	723	8,9	8 135
Europäische Union (28)	100 408	59,7	67 870	40,3	168 298
Weltweit	1 206 963	71,6	471 778	28,0	1 686 763

Für die weitere Entwicklung kann angenommen werden, dass der Anteil des produzierten Stahls über die Elektrolichtbogenofenroute in den EU 28 steigen wird. Weltweit betrachtet stagniert dieser Trend vorerst, da in China die Kapazitäten der integrierten Stahlwerke in den letzten Jahren stark erweitert wurden. Es kann jedoch angenommen werden, dass durch ein höheres Schrottaufkommen in den nächsten Jahren auch der Ausbau der Kapazitäten zur Herstellung von Elektrostahl steigen wird. [10]

In Österreich wird Stahl über die integrierte Hochofenroute an den zwei Voestalpine-Standorten in Linz und Donawitz produziert. Die drei Standorte bei denen Stahl über die Schrott-Elektrolichtbogenofenroute hergestellt, wird sind die Marienhütte in Graz, Böhler Edelstahl in Kapfenberg und Breitenfeld Edelstahl in Mitterdorf im Mürztal. [6]

3.3 Motivation zur Produktion von Stahl über die Schrott-Elektrolichtbogenofenroute

In Tabelle 3-2 ist eine Gegenüberstellung der Stahlproduktion mittels BOF und EAF dargestellt. Um die beiden Prozessrouten miteinander vergleichen zu können, werden als Inputindikatoren der Ressourcen- und Energieverbrauch und als Outputindikatoren die mit der Produktion verbundenen Emissionen und Nebenprodukte für die Produktion von einer Tonne flüssigen Stahl (LS) gewählt.

Tabelle 3-2: Umweltbezogene Input- und Outputindikatoren der Stahlherstellung mittels BOF und EAF [11]

Input	**BOF**	**EAF**	Units	Output	**BOF**	**EAF**	Units
Rohstoffe				**Produkte**			
Eisenerz	0,02-19,4	0	kg/t LS	Flüssiger Stahl	1000	1000	kg
Roheisen	788-931	0-18,8	kg/t LS	**Emissionen**			
Schrott	101-297	1009-1499	kg/t LS	CO2	22,6-174	82,4-180,7	kg/t LS
Metalle	0-60	1027-1502	kg/t LS	CO	393-7200	0,05-5,5	kg/t LS
Koks	0-0,36	15,4-19,4	kg/t LS	NOx	10-143	4-500	g/t LS
Kalk	30-67	25-140	kg/t LS	Staub	10-143	4-500	g/t LS
Dolomit	0-28,4	0-24,5	kg/t LS	Cr	0,01-0,08	0,003-4,3	g/t LS
Legierungen	1,3-33	14,4-25,9	kg/t LS	Fe	45,15	0	g/t LS
Kohle/Anthrazit	0	0,9-91	kg/t LS	Pb	0,17-0,98	0,075-2,85	g/t LS
Graphitelektroden	0	2-6	kg/t LS	Sox	0	3,2-252	g/t LS
Feuerfest-Materialien	0	3-38	kg/t LS	PAH	10	9-970	mg/t LS
Energie				**Energie**			
Elektrische Energie	9,5-60	440-748	kWh/t LS	BOF Gas	97-194	0	kWh/t LS
Erdgas	12-203	14-417	kWh/t LS	Dampf	34-93	0	kWh/t LS
Koksofengas	0-16	0	kWh/t LS	**Nebenprodukte**			
Dampf	3,6-42	9,2-70	kWh/t LS	Schlacken	101-206	70-343	kg/t LS
BF Gas	0,55-5,26	0	m^3/t LS	Stäube	0,75-24	10-30	kg/t LS
Druckluft	8-26,0	0	Nm^3/t LS	Stahlspritzer	2,8-15	0	kg/t LS
Gase				Schutt	0,05-6,4	0	kg/t LS
Sauerstoff	49,5-54,5	5-65	m^3/t LS	Walzzunder	2,3-7,7	0	kg/t LS
Stickstoff	0,55-1,1	5,9-12	m^3/t LS	Feuerfestausbruch	0	1,6-22,8	kg/t LS
Argon	2,3-18,2	0,79-1,45	m^3/t LS	Stahlschlämme	0	4,3	kg/t LS
Wasser	0,8-41,7	3,75-42,8	m^3/t LS	Abwasser	0,3-6	0	m^3/t LS

Vergleicht man die heutigen Produktionsdaten mit den Daten der 1970er- und 1980er-Jahre, kann man feststellen, dass der Rohstoffeinsatz von durchschnittlich 144 kg auf 115 kg zur Produktion von 100 kg flüssigen Stahl reduziert werden konnte. Diese Steigerung der Ressourceneffizienz von 21 Prozent lässt sich aufgrund von hohen Investitionen in der Forschung und Entwicklung erklären. Die technologischen Entwicklungen wurden aber in erster Linie aus wirtschaftlichen Gründen und nicht aus Umweltaspekten durchgeführt. Vergleicht man nun die beiden in Tabelle 3-2 dargestellten Verfahren, scheint es auf den ersten Blick so, dass die EAF-Route energieintensiver ist. Bei der BOF-Route muss man aber berücksichtigen, dass das Roheisen erst hergestellt werden muss. Der energieintensivste Prozess dabei ist die Reduktion des Eisenerzes im Hochofen. Im Hochofen werden ungefähr 72 Prozent der Gesamtenergie, die zur Stahlerzeugung über die BOF-Route benötigt wird, verbraucht. Insgesamt beträgt die Energieintensität der primären Stahlerzeugung über die integrierte Hochofenroute zwischen 26,4 und 41,6 GJ je produzierter Tonne Stahl. Bei der sekundären Stahlerzeugung über die Schrott-Elektrolichtbogenofenroute beträgt der Energiebedarf 9,1 bis 12,5 GJ je produzierter Tonne Stahl. Damit ist die Stahlherstellung aus Schrott weitaus weniger energieintensiv und zusätzlich können gegenüber der BOF-Herstellung CO_2-Emissionen und Ressourcen eingespart werden. Diese Aspekte machen die EAF-Stahlproduktion im Vergleich zur BOF-Route deutlich umweltfreundlicher. [11]

3.4 Die Rolle der Kreislaufwirtschaft in der Stahlerzeugung

Die Kreislaufwirtschaft bezieht sich auf einen Übergang von linearen Geschäftsmodellen zu Kreislaufgeschäftsmodellen. Dieses Konzept verfolgt, dass aus Rohstoffen hergestellte Produkte nicht weggeworfen werden, sondern Produkte oder Teile davon repariert, wiederverwendet und recycelt werden. Dieses Konzept basiert auf der allgemeinen Definition der Nachhaltigkeit, bei der ökonomische, ökologische und soziale Ziele gleichermaßen verfolgt werden sollten. Stahl hat in einer gut strukturierten Kreislaufwirtschaft erhebliche Vorteile gegenüber konkurrierenden Materialien, aber eine nachhaltige Zukunft bringt auch viele Herausforderungen mit sich. [12] Die grundsätzlichen Aspekte der Kreislaufwirtschaft von Stahl sind das Recycling, das Nutzen von Nebenprodukten, die Verbesserung der Energieeffizienz und der Einsatz und die Entwicklung neuer umweltschonenderer Technologien.

3.4.1 Recycling von Stahl

Durch die Verwendung von Stahlschrott werden die Kohlenstoffemissionen des gesamten Stahllebenszyklus reduziert. Des Weiteren kann Stahl nahezu verlustfrei recycelt werden, was auch in den letzten Jahrzehnten zu einer starken Erhöhung der Recyclingquote geführt

hat, die aber durch die hohe Lebensdauer von Stahl und die Verfügbarkeit von Schrott begrenzt ist. Durch Richtlinien, die speziell auf die Recyclingfähigkeit von Produkten und somit dem Design der Demontage dieser abzielen, könnte die Recyclingquote zukünftig weiter erhöht werden. [12]

Die Schrott-Elektrolichtbogenofenroute ist das wichtigste Verfahren, um Schrott zu recyceln. In Europa werden dazu oft bis zu 100 Prozent Stahlschrott im EAF eingesetzt. In Ländern, in denen Schrott in den erforderlichen Mengen nicht immer verfügbar ist, wird der EAF oft auch mit DRI, HBI, Roheisen und Flüssigstahl, zusätzlich zum Schrott, beladen. Bei der Betrachtung der weltweiten Situation deckt Schrott ca. 75 Prozent der eingesetzten Metalle ab, DRI und HBI ca. 15 Prozent und der Rest wird mit Roheisen und Flüssigstahl abgedeckt. Neben der Aufgabe des Recyclings von Stahlschrott dienen Elektrolichtbogenöfen auch zur Verwertung und Rückgewinnung von internen Produktionsabfällen wie Schlacken, feuerfesten Materialien und Lichtbogenofenstaub. Für die Stahlherstellung im Elektrolichtbogenofen kann der Schrott vorab je nach Herkunft in die folgenden drei Klassen unterteilt werden: [13]

- Veralteter Schrott: Autos, Geräte, Maschinen
- Industrieabfall: produktionsbedingter Abfall
- Interner Schrott: Ausschuss, aus Schlacke gewonnener Stahl (Stahlbären)

Durch die vermehrte Nutzung von Schrott aus recycelten Autos sind immer mehr Begleitelemente im Schrott zu finden. Diese Begleitelemente werden als Elemente definiert, die dem Stahl nicht absichtlich zugesetzt und durch einfache metallurgische Verfahren auch nicht entfernt werden können. Eines dieser Begleitelemente ist Kupfer, das hauptsächlich aus den elektrischen Leitungen in Kraftfahrzeugen stammt. Ein weiteres Begleitelement ist Zinn, das von Lötverbindungen und verzinnten Produkten stammt. In Edelstahlprodukten enthaltene Begleitelemente sind Nickel, Chrom, Molybdän und Kobalt. Weiters ist auch der Zinkgehalt im Schrott hoch, was von Vorteil ist, da die Nachfrage nach verzinktem Stahl zum Schutz vor Korrosion stetig steigt. [9]

3.4.2 Verwendung von Nebenprodukten

Die Stahlproduktion führt zur Erzeugung von Nebenprodukten, die den Emissionsausstoß reduzieren können, indem sie Ressourcen in anderen Branchen ersetzen. Stahlschlacken werden auch als Zuschlagstoffe in der Bauindustrie verwendet, wodurch natürliche Ressourcen geschont und die Umweltbelastung verringert werden kann. [12]

Elektrolichtbogenofen-Schlacke

EAF-Schlacke wird bei der Kohlenstoffstahlproduktion während des Schmelzprozesses von Stahlschrott durch Zugabe von Kalkstein bzw. Dolomit als Flussmittel bei Temperaturen um 1600 °C erzeugt. Die EAF-Schlacke setzt sich aus Kalzium-, Eisen- und Magnesiumoxid sowie Siliziumdioxid und geringen Mengen anderer Oxide zusammen. Durchschnittlich können je produzierte Tonne flüssigen Stahl (Tabelle 3-2), zwischen 70 und 343 kg an Schlacke anfallen. Die Schlacke wird im EAF zur Abscheidung von unerwünschten Begleitmaterialien benötigt. Beim Abschlackstand gelangt auch regelmäßig flüssiger Stahl mit in die Schlackenmulde, der dann in Form der sogenannten Stahlbären vorliegt. Diese Stahlbären werden aussortiert, analysiert und der entsprechenden Schrottgruppe zugeordnet. Dadurch konnten beispielsweise im Kalenderjahr 2019 im Stahlwerk der Breitenfeld Edelstahl AG aus 27.571 Tonnen Schlacke 2.417 Tonnen Stahlbären aussortiert und dem Stahlerzeugungsprozess wieder zugeführt werden. [14] Die weitere Nutzung von stabilisierter Schlacke, die als inertes Nebenprodukt gilt, findet hauptsächlich in der Bau- und Zementindustrie statt. [13]

Elektrolichtbogenofen-Staub

Während des Schmelzprozesses bildet sich zusätzlich zur EAF-Schlacke je produzierter Tonne flüssigem Stahl zwischen 10 und 30 kg Staub. Dieser EAF-Staub ist hauptsächlich aus Eisenoxiden, Kalzium- und Zinkoxid zusammengesetzt und wird in Filtern, die der Abgasnachbehandlungsanlage nachgeschalten sind, gesammelt. Elektrolichtbogenofen-Staub muss nachbehandelt werden, da die Entsorgung wegen der Gefahr der Auslaugung von Zink, Cadmium und Blei in den meisten Ländern nicht erlaubt ist. Für die Nachbehandlung und Wertmetallrückgewinnung von EAF-Staub gibt es unterschiedliche Verfahren und Methodiken. [15] Ein Verfahren dafür ist die Reduktion am Metallbad, bei der EAF-Staub zusammen mit einem Reduktionsmittel auf ein Metallbad aufgegeben wird. Dabei werden die Oxide reduziert und die einzelnen Wertmetalle im darunterliegenden Metallbad angereichert. Zink hingegen verflüchtigt sich und somit bleibt als Produkt stabilisierte Schlacke zurück, die in der Zement- oder Bauindustrie weiterverwendet werden kann. [16]

Feuerfest-Materialien

Der Ausbruch der Feuerfest-Materialien vom Ofengefäß beträgt je produzierter Tonne flüssigen Stahls zwischen 1,6 und 22,8 kg. Dieser Feuerfestausbruch besteht vor allem aus Ziegelfragmenten, Ziegeln und Metallverunreinigungen. Die Ziegel werden von den Herstellern der Feuerfestmaterialien zurückgekauft bzw. können die Ziegelfragmente zerkleinert und als Schlackenkonditionierer weiterverwendet werden. [14]

3.5 Technologische Entwicklungen des Elektrolichtbogenofens

In den letzten 50 Jahren konnte der Energieverbrauch der Stahlindustrie pro Tonne produzierten Stahl um 60 Prozent gesenkt werden. Dies wurde vor allem durch den Einsatz von technologischen Verbesserungen und der optimalen Ausnutzung der eingesetzten Energie ermöglicht. Beispiel dafür ist die Nutzung von Abwärme für andere Prozesse. Heutzutage liegt die durchschnittliche Energieintensität der Stahlproduktion bei ca. 20 GJ je Tonne Rohstahl. Es wird davon ausgegangen, dass nach wie vor Verbesserungspotentiale zur Steigerung der Energieeffizienz von 15 bis 20 Prozent in der Stahlproduktion vorhanden sind. Festzuhalten ist, dass die Energieintensität von älteren Anlagen, die nicht über die neueste Ausrüstung verfügen, durch den Einsatz von softwaretechnischen Optimierungen und einer optimalen Betriebsweise der Anlage gesenkt werden können. [12]

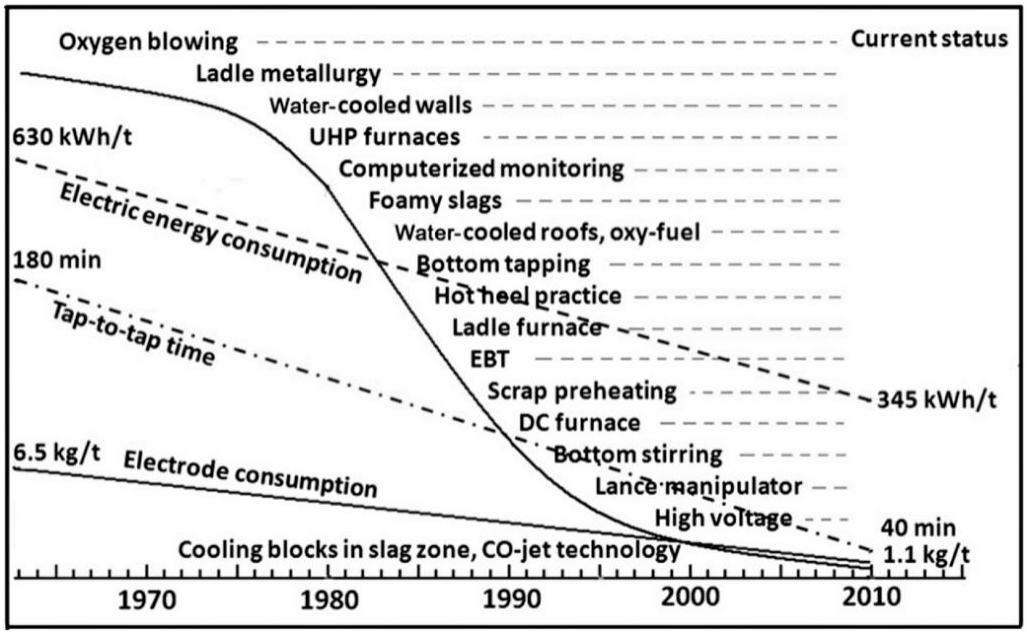

Abbildung 3-2: Technologische Entwicklungen des Elektrolichtbogenofen [13]

In Abbildung 3-2 sind die technologischen Entwicklungen des Elektrolichtbogenofens vom Jahr 1965 bis 2010 dargestellt. Diese Entwicklungen ermöglichen vor allem einen geringeren Energieverbrauch, eine Verkürzung der Tap-to-tap-Zeit, die der Zeitdauer zwischen zwei Abstichen entspricht, und eine Reduktion des eingesetzten Elektrodenmaterials. Diese stetigen Verbesserungen und Optimierungen zur Steigerung der Produktivität gehen einher mit der Verringerung des spezifischen Energieverbrauchs und können daher als energieeffizienzsteigernde Maßnahmen angesehen werden.

Einige dieser technologischen Entwicklungen, die einen großen Einfluss auf die Entwicklung des Elektrolichtbogenofens hinsichtlich der Energieeffizienz genommen haben, werden hier im Folgenden beschrieben.

Hochleistungs-Öfen

Ein wichtiger technologischer Entwicklungsschritt war die Einführung von im Hochleistungsbereich betriebenen Öfen (UHP – Ultra High Power). Um die Tap-to-tap-Zeiten zu verkürzen, werden größere Ofentransformatoren installiert. Auf die Energieeffizienz nimmt das keinen großen Einfluss, jedoch können die Tap-to-tap-Zeiten deutlich verkürzt werden. Dadurch kann die Produktivität erhöht und der spezifische Elektrodenverbrauch sowie das spezifische Abgasvolumen verringert werden. Durch die hohen Leistungen kommt es jedoch zu einem höheren Verschleiß der feuerfesten Ofenauskleidung. [17]

Wassergekühlte Ofenwände und Dächer

Um die UHP-Ofentechnologie nutzen zu können, werden in erster Linie wassergekühlte Ofenwände benötigt. Dafür werden die Ofenwände und das Ofendach mit wassergekühlten Paneelen ausgekleidet. Weiters konnte neben der Möglichkeit der Nutzung des UHP-Betriebs auch feuerfestes Material eingespart werden und Maßnahmen zur Energierückgewinnung gesetzt werden. Bei den Möglichkeiten zur Energierückgewinnung wird grundsätzlich zwischen zwei Techniken unterschieden. Dabei gibt es die kalte oder warme Kühlung, bei der Kühlwasser durch Rohrleitungen fließt. Eine Erhöhung der Kühlwassertemperatur führt dabei zu Leistungsverlusten am Lichtbogenofen. Die zweite Methode ist die Verdunstungskühlung, bei der durch Verdampfung von Kühlwasser die Strahlungswärme des Lichtbogenprozesses abgeführt wird. Die Paneele müssen vor thermischer Überlastung geschützt werden. Dies erfolgt meist durch softwaretechnische Regelungen des Schmelzprozesses. [18]

Computergestützte Prozesssteuerung

Seit 1982 wird die computergestützte Steuerung von Elektrolichtbogenöfen verwendet, da die hohe geforderte Produktivität und die damit hohen Durchsätze von Materialien verwaltet und koordiniert werden müssen. Durch diese Methodik kann der Energieeinsatz optimiert und die Emissionen verringert werden. Durch den Einsatz von effizienter Leistungselektronik kann die Stromversorgung des EAFs deutlich verbessert werden, wodurch auch die vorgeschalteten elektrischen Netze weniger stark belastet werden. Des Weiteren wurde gezeigt, dass eine optimierte Stromversorgung zu einer Erhöhung der Produktivität von bis zu sieben Prozent führen kann. Diese Aspekte führen somit auch zur Erhöhung der Energieeffizienz. [19]

Schaumschlackenpraxis

Durch Erzeugung von Schaumschlacke kann innerhalb des Ofens eine deutlich bessere Wärmeübertragung stattfinden. Ein weiterer Vorteil ist, dass Beanspruchungen der

Feuerfestauskleidungen des Ofens geringer ausfallen. Durch die Schaumschlackenpraxis erhält der Lichtbogen eine höhere Stabilität und aufgrund der geringeren Strahlungseffekte sinkt der Energieverbrauch, der Verbrauch von Elektrodenmaterial und der Geräuschpegel. Außerdem hat die Schaumschlacke positive Effekte auf mehrere metallurgischen Reaktionen unter Ausschluss von negativen Auswirkungen. Für Edelstähle und andere hochlegierte Stähle ist die Schaumschlackenpraxis nicht anwendbar. [20]

Abstich am Boden des Elektrolichtbogenofengefäßes

Seit 1983 verwendet und heute weit verbreitet, ist der Abstich am Boden des EAF-Gefäßes. Der Hauptvorteil des Abstichs am Boden des Ofengefäßes ist, dass die Verschleppung der oxidativen Schlacke minimiert wird. Durch den Abstich am Boden kann auch der flüssige Stahl schneller abgezogen werden, wodurch die Energieverluste geringer bleiben. Dämpfe können durch das geschlossene Obergefäß außerdem einfacher abgezogen werden. Die meisten Öfen der Edelstahlproduktion sind nach wie vor mit Ausgüssen ausgestattet. Dies liegt vor allem in der einfacheren Handhabung und der günstigeren Instandhaltung der Systeme mit Ausguss. [19]

Pfannen- und Sekundärmetallurgie

Lange Zeit wurden die sekundären Prozessschritte der Stahlherstellung, wie das Entschwefeln, das Zuführen von Legierungen und die chemische Homogenisierung im Ofengefäß des EAFs durchgeführt. Ab 1985 wurden diese Prozesse auf andere Aggregate verlagert. Allein durch das Nutzen von Pfannen und Pfannenöfen konnten Nettoenergieeinsparungen von 10 bis 30 kWh pro Tonne produzierten Stahl erzielt werden. Weiters erfolgte einer Reduzierung der Tap-to-tap-Zeit von bis zu 20 Minuten und die Temperatur konnte im Pfannenofen für den Strangguss deutlich besser kontrolliert und gehalten werden. Durch die kürzeren Prozesszeiten und die Auslagerung der Sekundärmetallurgie reduziert sich auch der Verbrauch des Elektrodenmaterials am EAF. Die Emissionen am EAF selbst werden verringert, aber durch Zunahme der anderen Emissionsquellen braucht es höhere Investitionen in Luftreinhalteeinrichtungen, Rauchgasabscheidungen und Hauben für die jeweiligen Aggregate. [21]

Autogenbrenner und Sauerstofflanzen

Durch den Einsatz von Sauerstofflanzen wird das gleichmäßige Schmelzen des Schrotts gefördert. Elektrische Lastspitzen können durch den Einsatz von Autogenbrennern und Sauerstofflanzen reduziert werden, da der elektrische Gesamtenergiebedarf durch den zusätzlichen Energieeintrag reduziert wird. [19]

3.6 Energiebilanz des Elektrolichtbogenofens

Um weitere Einsparpotentiale im Sinne der Energieeffizienz und der Reduktion des Gesamtenergieverbrauchs zu ermitteln, betrachten wir die vollständige Energiebilanz des Elektrolichtbogenofens. Dafür ist in Abbildung 3-3 ein einfaches System des Elektrolichtbogenofens dargestellt. In dieser Systemdarstellung ist der EAF zur Bildung einer Energiebilanz abgegrenzt und sämtliche auf den EAF wirkenden Einflussgrößen dargestellt. Die genaue Bestimmung des Gesamtenergieeintrags in einen EAF ist jedoch komplex, da dem EAF Energie aus mehreren Quellen zugeführt wird. Diese Quellen sind elektrische Energie sowie chemische Energie. Letztere wird bei der Verbrennung von Erdgas, Flüssiggas oder Öl frei. Aufgrund von Oxidation der Elemente Kohlenstoff, Silizium, Aluminium, Eisen, Chrom und Mangan wird in der Schmelze zusätzlich Energie freigesetzt. Die freigesetzte Energie dieser Reaktionen hängt nicht nur vom Sauerstoffeintrag, sondern auch von der chemischen Zusammensetzung des Schrotts und der gebildeten Schlacke ab. [20]

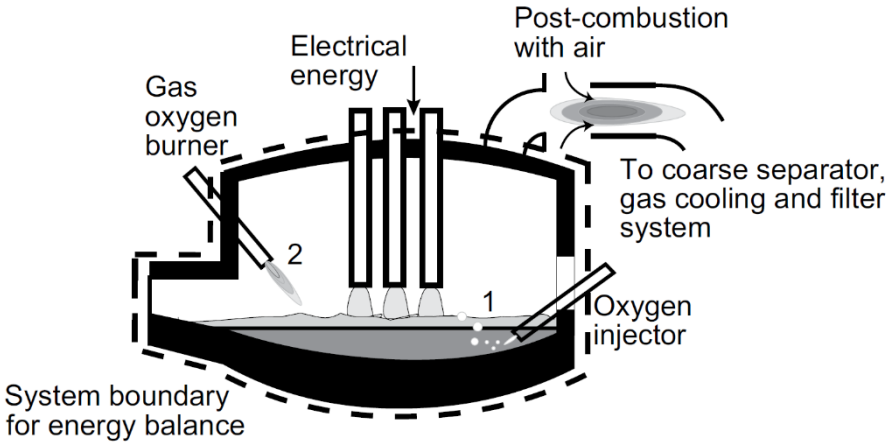

Abbildung 3-3: Systemdarstellung eines Elektrolichtbogenofens [20]

Der Eintrag der elektrischen Energie in den EAF erfolgt über die Graphitelektroden, welchen ein Ofentransformator vorgeschalten ist. Thermische Energie wird über die Verbrennung von Erdgas eingebracht. Mit der Nutzung der Erdgasbrenner kann einerseits verhindert werden, dass sich kalte Stellen im Ofengefäß bilden, die sich negativ auf die Haltbarkeit auswirken würden. Andererseits wird durch das Einbringen der chemischen Energie über diese Brenner, der Energieeintrag in den EAF erhöht, wodurch die Schmelzzeit des Schrotts bei gleichbleibender Transformatorleistung verkürzt werden kann. Findet ein zusätzlicher Wärmeintrag statt, beispielsweise durch den Eintrag von heiß brikettiertem Eisen, direkt reduziertem Eisen oder vorgeheiztem Schrott, müsste dieser Wärmeeintrag in der Bilanz berücksichtigt werden. In der verwendeten Darstellung des Ofensystems wird davon ausgegangen, dass Schrott gemäß der Schrott-Elektrolichtbogenofenroute ohne Vorheizung

in Umgebungstemperatur aufgeschmolzen wird. Für diese Annahmen kann die Gesamtenergiebilanz für den Elektrolichtbogenofen wie in Gleichung ***(3-1)*** formuliert werden. Für die Gesamtenergiebilanz wird eine zeitliche Integration der Enthalpie- und Wärmeströme für das zeitliche Intervall zwischen dem Beladen des Elektrolichtbogenofens und dem Abstich, also der Tap-to-tap-Zeit, durchgeführt. [20]

$$E_{Total} = \int_{Charging}^{Tapping} P_{Electric}dt + \Delta H_{Oxygen\ injection} + \Delta H_{NG\ burners} \tag{3-1}$$

$$= \Delta H_{Steel} + \Delta H_{Slag} + \int \Delta \dot{H}_{G-off}dt + \int \Delta \dot{Q}_C dt + \int \Delta \dot{Q}_{Rad,other}dt$$

Dabei ist $P_{Electric}$ die elektrische Leistung, die aus den Betriebsdaten des Ofentransformators bestimmt werden kann und nach der Tap-to-tap-Zeit integriert die elektrische Energie ergibt. Bei Berechnung der elektrischen Leistung ist auf die ohmschen Verluste zwischen dem Ofentransformator und dem Lichtbogen, die einige Prozente ausmachen, zu berücksichtigen. $\Delta H_{Oxygen\ injection}$ ist die Enthalpie des eingeblasenen Sauerstoffs und $\Delta H_{NG\ burners}$ die Enthalpie des Brenngases. Die beiden Enthalpien stellen den chemischen Energieeintrag in das System dar. Einen großen Anteil der abgeführten Energie in der Energiebilanz stellen die Stahl- und Schlackeenthalpien, ΔH_{Steel} und ΔH_{Slag}, dar. Diese hängen vor allem von den chemischen Zusammensetzungen des chargierten Schrotts und dessen spezifischer Enthalpie ab. Die weiteren abgeführten Energien sind die Wärmeverluste des Abgases ($\Delta \dot{H}_{G-off}$) und die Verluste, die durch Kühlung der Paneele in den Seitenwänden und im Dach des Ofengefäßes auftreten ($\Delta \dot{Q}_C$). Mit $\Delta \dot{Q}_{Rad,other}$ werden die Strahlungsverluste, die hauptsächlich beim Chargieren auftreten und andere Verluste berücksichtigt.

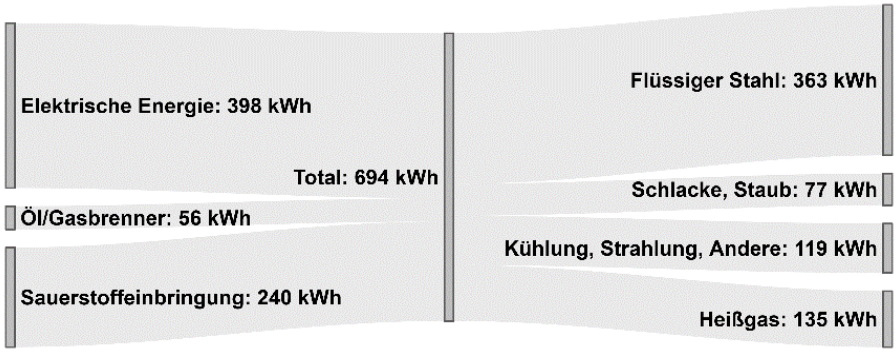

Abbildung 3-4: Sankey-Diagramm für die Produktion einer Tonne flüssigen Stahl mit dem EAF-Verfahren

Ein Beispiel für die nach der obigen Gleichung ***(3-1)*** beschriebenen Energieflüsse sind in der Abbildung 3-4 in Form eines Sankey-Diagramms für die Stahlproduktion mit einem EAF ohne

Schrottvorwärmung dargestellt. Dafür wurden Durchschnittswerte für die Produktion von einer Tonne flüssigen Stahl für die jeweiligen Energieströme aus der Literatur ermittelt. [20]

3.7 CO_2-Emissionen des Elektrolichtbogenofens

In der Europäischen Union sanken die gesamten CO_2-Emissionen der Stahlindustrie zwischen 1990 und 2010 von 298 Mio. Tonnen auf 223 Mio. Tonnen. Bei diesen Daten ist jedoch zu berücksichtigen, dass die Rohstahlproduktion in diesem Zeitraum von 197 Mio. Tonnen auf 173 Mio. Tonnen zurückging. Unter Berücksichtigung des Stahlherstellungsverfahrens, also ob der Rohstahl über die integrierte Hochofenroute oder die Schrott-Elektrolichtbogenofenroute produziert wurde, entspricht das einer Reduktion der gesamten CO_2-Emissionen von 14 Prozent. Die von der Stahlerzeugung über die Schrott-Elektrolichtbogenofenroute spezifischen CO_2-Emissionen konnten im Betrachtungszeitraum von 667 auf 455 kg CO_2 pro Tonne Rohstahl gesenkt werden. Das entspricht einer Reduktion von 32 Prozent. Diese große Emissionsreduktion beim EAF-Verfahren ist vor allem auf den Rückgang der indirekten Emissionen der elektrischen Energie zurückzuführen. Von 1990 bis 2010 konnten die spezifischen CO_2-Emissionen der elektrischen Energie in Europa von 585 auf 429 g CO_2 pro kWh gesenkt werden. Da der Energieverbrauch für die Rohstahlerzeugung mittels EAF in etwa zur Hälfte mit elektrischer Energie gedeckt wird (Abbildung 3 4), geht damit eine starke Reduktion der spezifischen CO_2-Emissionen bei der EAF-Stahlproduktion einher. Somit können die bei der Stahlherstellung über die Schrott-Elektrolichtbogenofenroute verursachten CO_2-Emissionen zu gleichen Teilen in energiebezogene und prozessbedingte Emissionen aufgeteilt werden. [4]

Energiebezogene Emissionen:

- Indirekte Emissionen zur Erzeugung und Bereitstellung von elektrischer Energie
- Öl- bzw. Gasbrenner

Prozessbedingte Emissionen:

- Entkohlung des flüssigen Stahls durch Einblasen von Sauerstoff
- Einbringen von Kohlenstoff zur Schaumschlackenbildung
- Verbrauch der Graphitelektroden

Neben den indirekten energiebezogenen CO_2-Emissionen des elektrischen Energiebedarfs, kommt es durch die Verbrennung von Erdgas im Elektrolichtbogenofen zusätzlich zu weiteren energiebezogenen CO_2-Emissionen. Der effiziente Einsatz von Erdgasbrennern kann in modernen EAFs jedoch dazu führen, dass trotz Erhöhung der Energieintensität die gesamten energiebezogenen Emissionen gleichbleiben bzw. reduziert werden können. Dabei

kommt es vor allem auf die spezifischen CO_2-Emissionen der eingesetzten elektrischen Energie an. Die prozessbedingten CO_2-Emissionen ergeben sich vor allem durch den Eintrag von Kohle in den EAF zur Schaumschlackenbildung, die zur Verringerung der Chromoxidation dient. Bei der Herstellung von hochlegierten Stählen wird auch zusätzlich Kohlenstoff eingebracht. Des Weiteren wird Kohlenstoff mit dem Schrott, meist in Form von Öl- und Fettkontaminationen, in den EAF eingeführt. Für die gewünschte Stahlqualität muss dieser Kohlenstoff mit Sauerstoff auf ein bestimmtes Niveau oxidiert werden. Die durch den Verbrauch der Graphitelektroden entstehenden CO_2-Emissionen fallen im Vergleich zu den übrigen prozessbedingten Emissionen sehr gering aus. Eine Reduktion der spezifischen CO_2-Emissionen durch den Abbrand der Graphitelektroden kann im Grunde nur durch eine Verkürzung der Tap-to-tap-Zeit erreicht werden. [20]

4 ENTWICKLUNG DES ENERGIESYSTEMMODELLS

In diesem Kapitel wird beschrieben, wie das elektrische Energiesystemmodell entwickelt wird und welche Charakteristik dieses aufweist. Dazu folgt eine detaillierte Beschreibung der Vorgehensweise, wie aus den erhobenen Daten die synthetischen Lastprofile erstellt und zu einem Energiesystemmodell verknüpft werden. Zur Erstellung des Energiesystemmodells wurden die energietechnisch relevanten Verbraucher am Standort der Breitenfeld Edelstahl AG erfasst und evaluiert. Das elektrische Energiesystem, das im Zuge dieser Arbeit erstellt und modelliert wird, umfasst davon folgende Verbraucher:

- Elektrolichtbogenofen
- Pfannenöfen
- Entstaubungsgebläse:
    - Elektrolichtbogenofen Primär
    - Elektrolichtbogenofen Sekundär
    - Halle 1
    - Halle 9
- Grundlast
- Andere Verbraucher

4.1 Datenerhebung und Genauigkeit

Die Datenerhebung erfolgte vor Ort am Standort des Unternehmens Breitenfeld Edelstahl. Die Leistungsdaten des Elektrolichtbogenofens, eines Pfannenofens und der sechs Gebläse wurden für mehrere Wochen aufgezeichnet, um für die Modellierung aussagekräftige Datensätze zu erhalten. Bei den Messungen wurden die Wirk-, Blind- und Scheinleistung sowie die Netzfrequenz mit einem Power Analyzer AM 15 PLOG aufgezeichnet. Je nach Aggregat wurden noch zusätzliche Daten, auf die in den jeweiligen Modellierungen eingegangen wird, aufgezeichnet.

Die Messdaten haben eine Auflösung von einer Minute und liefern dadurch eine äußerst genaue Abbildung der realen Betriebsweise der Aggregate. Weiters wurden von Breitenfeld Edelstahl das elektrische Gesamtlastprofil der Jahre 2018 und 2019 mit einer Auflösung von 15 Minuten zur Verfügung gestellt. Aus diesen Datensätzen wurden die für das Modell relevanten Daten und Parameter für das Energiesystemmodell extrahiert und weiterverarbeitet.

4.2 Modellierungsmethodik

In diesem Abschnitt werden die im Modell angewandten Methodiken, die zur Modellierung der synthetischen Lastprofile eingesetzt wurden, erklärt. Dabei werden stochastischen Methoden wie Markov-Ketten und das Rechnen mit Verteilungen herangezogen, da es sich bei der Änderung von diversen Prozessparametern der modellierten Aggregate um zufällige-, also stochastische Prozesse handelt. Mit diesen stochastischen Methoden können die realen Prozessabläufe sehr realitätsnah nachgebildet werden. Des Weiteren wird der Aufbau des Modells und die Entwicklungsumgebung beschrieben.

4.2.1 Markov-Ketten

Eine Markov-Kette ist ein mathematisches Modell, mit dem zeitliche Entwicklungen zeit-diskreter stochastischer Prozesse beschrieben werden können. Unter einem stochastischen Prozess versteht man im Allgemeinen eine Menge von Zufallsvariablen. Die Definition der Markov-Kette ist in der Gleichung ***(4-1)*** beschrieben. Dabei ist $(X_n)_{n\in N0}$ der zeit-diskrete stochastische Prozess der Markov-Kette mit abzählbaren Zustandsraum I, für alle Zeitpunkte $n \epsilon N_0$ und alle Zustände $i_0, \ldots, i_{n-1}, i_n, i_{n+1} \in I$. Die bedingte Wahrscheinlichkeit $P(X_{n+1} = i_{n+1} \mid X_n = i_n)$, mit der bei Vorliegen von i_n der Nachfolgezustand i_{n+1} angenommen wird, heißt Übergangswahrscheinlichkeit des Prozesses. [22]

$$P(X_{n+1} = i_{n+1} | X_{0=i_o}, \ldots, X_{n-1} = i_{n-1}, X_n = i_n) = P(X_{n+1} = i_{n+1} \mid X_n = i_n) \quad (4\text{-}1)$$

Die Markov-Kette zeichnet sich durch ihre „Gedächtnislosigkeit“, die sogenannte Markov-Eigenschaft, aus. Aufgrund dieser Markov-Eigenschaft fließt immer nur der zuletzt beobachtete Zustand in die zukünftige Entwicklung des Prozesses ein. Somit kann gesagt werden, dass sich die zeitliche Entwicklung der Markov-Kette durch ihre Anfangswahrscheinlichkeiten und ihre Übergangswahrscheinlichkeiten vollständig beschreiben lässt. [22]

In Bezug auf das Modell werden diese Eigenschaften genutzt, um physikalische Vorgänge wie die Leistungsaufnahme am Elektrolichtbogenofen und die Leistungsaufnahme an den Pfannenöfen zu beschreiben. In diesen Prozessen ist die Veränderung der Leistungsaufnahme von einen auf den nächsten Zeitschritt an eine bestimmte Wahrscheinlichkeit gebunden, die in der Übergangswahrscheinlichkeitsmatrix eingetragen ist. Diese Übergangswahrscheinlichkeitsmatrizen werden vorab mit Hilfe eines eigenen Python-Modells aus den gemessenen Lastprofilen des Elektrolichtbogenofens bzw. des Pfannenofens erstellt. Dafür werden aus den gemessenen Daten die beobachteten

Sequenzen der Leistungsaufnahme der jeweiligen Prozessphase ermittelt. Aus diesen Beobachtungssequenzen kann die Übergangswahrscheinlichkeitsmatrix erstellt werden. Die erstellten Übergangswahrscheinlichkeitsmatrizen sind im Anhang der Arbeit dargestellt.

Um nun eine Markov-Kette konstruieren zu können, werden die Wahrscheinlichkeiten für jeden Ausgangszustand, das heißt für jede Zeile der Matrix, kumuliert. Wird nun ein Ausgangszustand als Startpunkt vorgeben und eine Zufallszahl z ($0 \leq z \leq 1$) gezogen, gelangt man über die Übergangswahrscheinlichkeitsmatrix in den nächsten Zustand, der dann als neuer Ausgangszustand für den nächsten Zeitschritt dient. Durch Aneinanderreihen dieser Zustände erhält man eine synthetische Zeitreihe, die der Verteilung der beobachteten Sequenz gehorcht. Bei jedem Aufruf der Markov-Kette nimmt diese Sequenz eine andere Form an. [23]

Diese Prinzipien werden in dieser Arbeit auf die Modellierung des Elektrolichtbogenofens, der Pfannenöfen und zur Modellierung des Lastprofils der sonstigen Verbraucher angewandt. Dabei wird für jede Prozessphase, aufgrund von unterschiedlichen Leistungscharakteristiken eine eigene Übergangswahrscheinlichkeitsmatrix ermittelt und aus einem Excel-File in das Modell importiert.

4.2.2 Stochastische Ermittlung der Prozessparameter

Die am Standort der Breitenfeld Edelstahl AG gemessenen Prozessparameter wurden zur Implementierung im Modell vorab ausgewertet. Diese Prozessparameter, zu denen beispielsweise die Prozesszeiten, die Schrottmenge und der Energieverbrauch unterschiedlicher Aggregate zählen, weisen teilweise sehr hohe Streuungen auf. Aus den gemessenen Daten werden Verteilungen ermittelt, die auf das Modell übertragen werden und dadurch für realitätsnahe Simulationsergebnisse sorgen. Dafür wird zur Charakterisierung der gemessenen physikalischen Größen in der Modellierung des Energiesystems auf die realen Häufigkeitsverteilungen zurückgegriffen. Dabei werden die gemessenen Größen in Klassen eingeteilt und deren absolute Klassenhäufigkeit ermittelt. Um diese Verteilungen im Modell implementieren zu können, müssen aus den absoluten Häufigkeiten die relativen Häufigkeiten ermittelt werden. Dafür wird, wie in Gleichung ***(4-2)***, der Stichprobenwert n_i der absoluten Häufigkeit durch die Anzahl aller Stichprobenwerte n dividiert.

$$h_i = \frac{n_i}{n} \tag{4-2}$$

Aus den Klassen werden nun Mittelwerte gebildet. Diesen Klassenmitten x_i werden die relativen Häufigkeiten h_i zugewiesen. Wichtig ist darauf zu achten, dass die kumulierte

Häufigkeit dabei 1 bzw. 100 Prozent entspricht. Die für die Simulation relevanten Werte werden im Modell, wie hier in Tabelle 4-1 dargestellt, implementiert. [24]

Tabelle 4-1: Häufigkeitsverteilung

Klassenmitte x_i	x_1	x_2	x_3	...	x_k
relative Klassenhäufigkeit h_i	h_1	h_2	h_3	...	h_k

In diesen Listen der Häufigkeitsverteilungen sind die realen Messwerte hinterlegt. Diese Verteilungen, die im Modell als Eingangsgrößen dienen, liefern wahrheitsgetreue Ergebnisse, jedoch gibt es dann keine Möglichkeit, die Eingangsgrößen zu verändern. Um das Modell flexibler zu gestalten und Veränderungen der Eingabegrößen zu ermöglichen, können die diskreten Häufigkeitsverteilungen mit einer stetigen Verteilungsdichtefunktion beschrieben werden. Die Dichtefunktion wird nach der Gleichung ***(4-3)*** berechnet und entspricht der Normalverteilung. [24]

$$f(x) = \frac{1}{s\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x-\bar{x}}{s}\right)^2} \tag{4-3}$$

Dabei entspricht $\bar{x}$ dem Mittel- oder Erwartungswert der Grundgesamtheit und s der Standardabweichung. Damit die Normalverteilung die bestmögliche Genauigkeit im Vergleich zur realen Häufigkeitsverteilung aufweist wurden der Mittelwert und die Standardabweichung nach den auftretenden Häufigkeiten gewogen. Der gewogene arithmetische Mittelwert berechnet sich nach Gleichung ***(4-4)***.

$$\bar{x}_{gew} = \frac{\sum h_i x_i}{n} \tag{4-4}$$

Dabei ist $\bar{x}_{gew}$ der gewogene arithmetische Mittelwert, für den das Summenprodukt aus der auftretenden Häufigkeit h_i und der Klassenmitte x_i durch die Gesamtanzahl der Häufigkeiten n dividiert wird. Mit diesen Parametern wird auch die gewogene Standardabweichung nach Gleichung ***(4-5)*** berechnet. [25]

$$s_{gew} = \sqrt{\frac{n\sum h_i {x^2}_i - (\sum h_i x_i)^2}{n(n-1)}} \tag{4-5}$$

In Abbildung 4-1 ist verdeutlicht, welcher Unterschied im Hinblick auf die Ergebnisse der Simulation erreicht wird, wenn gewogene Mittelwerte und Standardabweichungen verwendet werden. Die reale Häufigkeitsverteilung der gemessenen Daten ist mit grauen

Balken als Histogramm dargestellt. Die rot strichlierte Linie stellt die Normalverteilung bei Verwendung des normalen Mittelwerts und der normalen Standardabweichung dar. Dabei ist deutlich zu erkennen, dass die Standardabweichung sehr hoch und dadurch der abgedeckte Bereich zu breit ist. Weiters ist feststellbar, dass der nicht gewogene Mittelwert vom gewogenen Mittelwert deutlich abweicht. Dies würde in der Simulation zu einer Verfälschung der Ergebnisse führen. Die blaue Linie, welche der Normalverteilung mit dem gewogenen Mittelwert und der gewogenen Standardabweichung entspricht, beschreibt die reale Häufigkeitsverteilung besser. Dass liegt vor allem an der geringeren Standardabweichung.

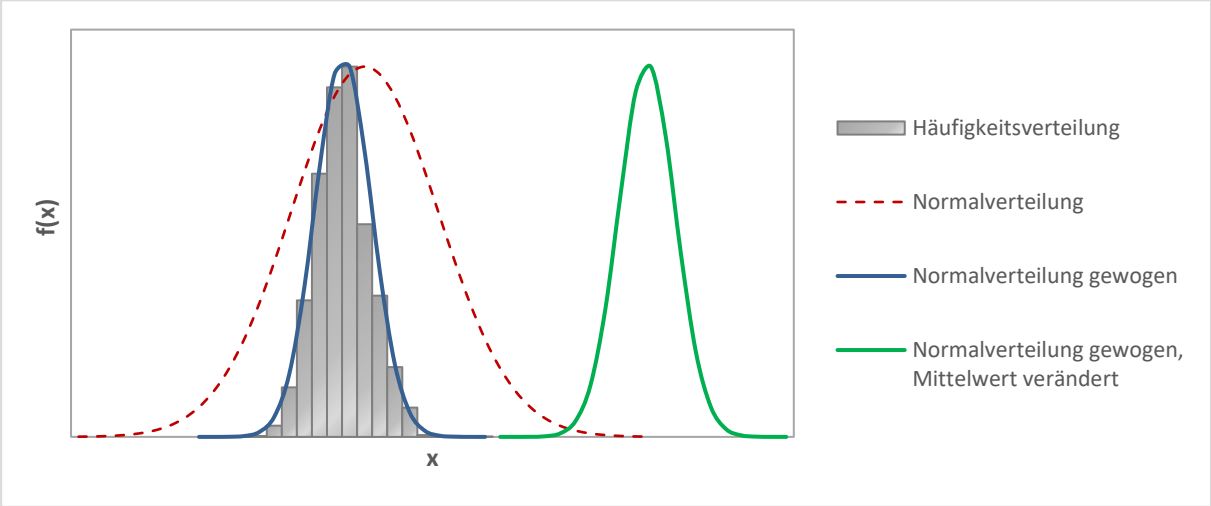

Abbildung 4-1: Übersicht der unterschiedlichen Verteilungen

Die grüne Linie stellt nun den Fall dar, wenn für ein Aggregat ein Parameter verändert wird. Durch Eingabe eines spezifischen Wertes nimmt die Normalverteilung mit dem gewogenen Mittelwert und der gewogenen Standardabweichung den eingegebenen Wert als neuen Mittelwert an und dadurch wird die Verteilung auf diesen neuen Mittelwert verschoben.

Diese Methodik ist vor allem für die Flexibilität des Modells relevant, da somit auch bei der Änderung von Parametern das Modell mit den jeweiligen Verteilungen rechnet. Will man im Modell die mittlere Tap-to-tap-Zeit nachträglich verändern, da diese beispielsweise durch Verbesserungsmaßnahmen im Stahlwerk verringert werden konnte, liefert das Modell wahrheitsgetreue Berechnungen, wenn die gemessene gewichtete Normalverteilung um diesen Mittelwert verschoben wird und der neue Wert nicht als statischer Wert definiert wird. Würde man den neuen Mittelwert als statischen Wert definieren, würden die Ergebnisse nicht der Realität entsprechen. Diese Methodik wird im Modell für die veränderlichen Eingangsparameter der unterschiedlichen Aggregate angewandt. Die Verteilungen für dieses statistischen Verteilungen sind dafür in den jeweiligen Abschnitten graphisch dargestellt.

4.2.3 Berechnung der Leistungen und des Energieverbrauchs

In diesem Kapitel wird die Berechnung der Schein- und Blindleistung sowie der elektrischen Energie im Modell beschrieben. Die Scheinleistung ist die vektorielle Summe aus Wirk- und Blindleistung und ist die relevante Größe für die Dimensionierung des Netzes. Die Blindleistung ist aus netztechnischer Sicht nicht erwünscht, da diese sich nicht in Nutzenergie umwandeln lässt aber das Netz trotzdem belastet. Der Blindleistungsanteil im Netz steigt bei der unregelmäßigen Einspeisung von Energie, wie es bei der Energieerzeugung aus fluktuierenden erneuerbaren Energiequellen der Fall ist. Blindleistung wird benötigt, um Magnetfelder in Transformatoren, Generatoren und Elektromotoren aufzubauen. Im Falle der Breitenfeld Edelstahl AG werden für die Transformatoren der Öfen große Blindleistungsbezüge, die starken Schwankungen in sehr kurzen Zeiträumen unterliegen, benötigt. Mit dem Modell kann die Blindleistung für die einzelnen Aggregate und das gesamte Stahlwerk ermittelt werden. Mit diesen Daten können dann Vorhersagen über die Belastung des vorgelagerten Netzes getroffen und Optimierungsmöglichkeiten identifiziert werden, wie die Abschätzung notwendiger Kompensationsmaßnahmen, welche die Energieerzeugeranlagen und Energieübertragungseinrichtungen entlasten. [26]

Für sämtliche Aggregate des Energiesystemmodells wurde die Wirk-, Blind- und Scheinleistung mit der Auflösung von einer Minute gemessen. Der Leistungsfaktor $cos\varphi$, der dem Verhältnis zwischen Wirk- und Scheinleistung entspricht, wurde für die einzelnen Aggregate, gemäß der Gleichung ***(4-6)*** berechnet. [26] Das Modell rechnet mit arithmetisch gemittelten Leistungsfaktoren, die für jedes Aggregat aus den gemessenen Leistungsdaten der Wirk- und Scheinleistung bestimmt wurden. Für die Gebläse wurden für den Teil- und Vollastbetrieb unterschiedliche mittlere Leistungsfaktoren berechnet und dem Modell übergeben.

$$cos\varphi = \frac{P}{S} \tag{4-6}$$

Zur Modellierung der synthetischen Lastprofile wurde die Wirkleistung gewählt. Dadurch muss eine Umrechnung auf die Blind- und Scheinleistung erfolgen. Durch Umformen der obigen Gleichung erhält man die Gleichung ***(4-7)***, mit der die Scheinleistung aus dem erstellten Lastprofil der Wirkleistung berechnet wird.

$$S = \frac{P}{cos\varphi} \tag{4-7}$$

Da im Modell der Leistungsfaktor cosφ hinterlegt ist, wird die Gleichung zur Berechnung der Blindleistung ***(4-8)*** dementsprechend angepasst.

$$Q = P\ tan\varphi = P \tan\left(\arccos\left(cos\varphi\right)\right) \tag{4-8}$$

Die Berechnung der verbrauchten elektrischen Energie, die für Beurteilung des Modells von großer Bedeutung ist, erfolgt über die Gleichung ***(4-9)***.

$$E = P\ t \tag{4-9}$$

4.2.4 Simulationsprogramm und Aufbau des Energiesystemmodells

Zur Erstellung des Energiesystemmodells wurde die integrierte Entwicklungsumgebung PyCharm des Unternehmens JetBrains verwendet. Mit der Community-Version bietet JetBrains eine kostenlose Plattform an, mit der Projekte in der Programmiersprache Python erstellt werden können.

Nachdem ein Projekt erstellt wird, muss für dieses Projekt ein „Python Interpreter“ festgelegt werden. Im Falle dieser Arbeit wurde die Version Python 3.8 genutzt. Um gewisse Funktionen nutzen zu können, werden zusätzlich Programmbibliotheken, sogenannte Packages, in PyCharm installiert. Die für diese Arbeit erforderlichen Pakete und deren Aufgaben sind in der folgenden Liste angeführt:

- Matplotlib: Visualisieren von Daten
- Math: Mathematische Funktionen
- Numpy: Grundlegendes Paket zur Array-Programmierung
- Openpyxl: Lesen und Schreiben von Excel 2010 xlsx/xlsm-Dateien
- Pandas: Leistungsstarke Datenstrukturen für Datenanalyse und Statistiken
- Scipy: Wissenschaftliche Bibliothek
- Tkinter: Erstellen von graphischen Benutzeroberflächen (GUI)
- Xlrd: Extrahieren von Daten aus Microsoft Excel-Tabellenkalkulationsdateien

In PyCharm müssen die benötigten Programmbibliotheken für jedes Modul am Seitenanfang importiert werden, damit diese genutzt werden können. In der Abbildung 4-2 ist der Aufbau des Energiesystemmodells mit den verwendeten Modulen dargestellt. Im Input-Modul sind alle Eingabegrößen, die für die Simulation notwendig sind, hinterlegt bzw. werden die Eingabegrößen aus der GUI, der graphischen Benutzeroberfläche, abgefragt. Die Steuerung des Modells erfolgt über die Module Main, LoadProfiles, Days und SecondaryMetallurgy. Die Simulation wird über das Main-Modul gestartet und dabei öffnet sich das Fenster der graphischen Benutzeroberfläche, in dem sämtliche veränderbaren Parameter des Modells

angeführt sind. Im Main-Modul erfolgt die Initialisierung der Lastprofile und die Konstruktion dieser. In der GUI kann auch die Ausgabe der Simulation festgelegt werden. Das Modul Days dient als zeitliches Koordinationselement, um Simulationen die länger als eine Woche dauern, durchzuführen. In diesem Modul sind außerdem die Betriebszeiten der Aggregate definiert. Im LoadProfiles-Modul werden die Lastprofile erzeugt und die Werte dieser gespeichert. Zusätzlich werden für diverse Parameter noch leere Listen erzeugt. Im Modul SecondaryMetallurgy werden die Prozessschritte, die zwischen dem Abstich am Lichtbogenofen und dem Blockguss stattfinden, festgelegt.

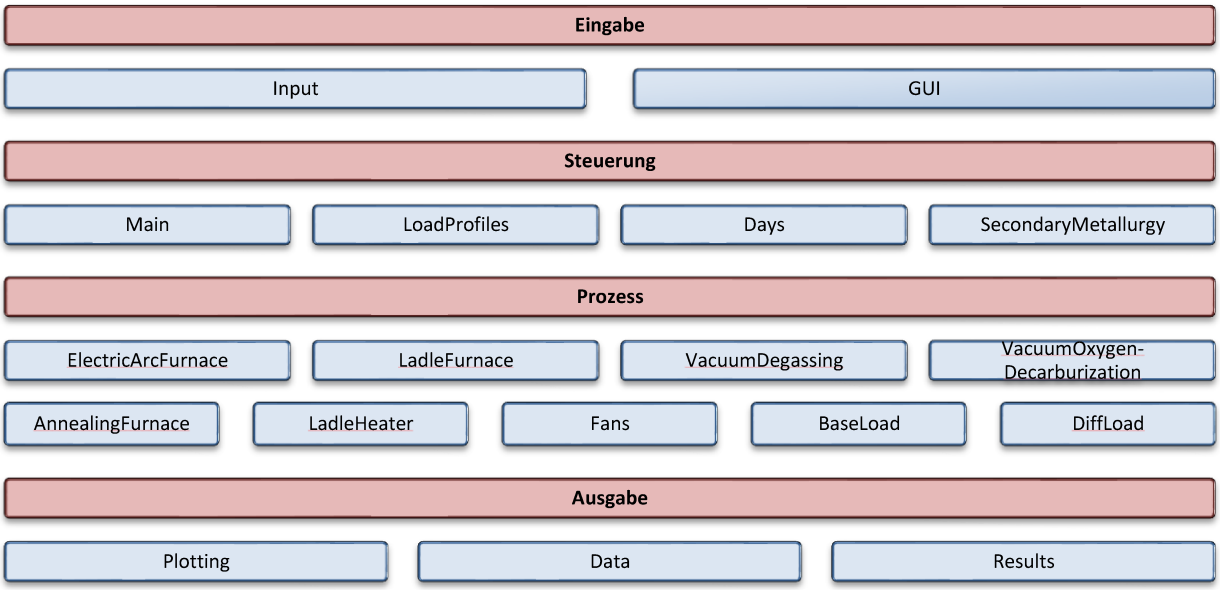

Abbildung 4-2: Aufbau des Energiesystemmodells in PyCharm

Im Modul LadleHeater werden die Pfannen und die logistischen Prozesse, die diese durchlaufen, definiert. Die Module VacuumDegassing und VacuumOxygenDegassing dienen der Berechnung des Dampf- bzw. Sauerstoffverbrauchs des VD- und VOD-Verfahrens. In den Modulen ElectricArcFurnace, LadleFurnace, Fans, BaseLoad und DiffLoad werden die synthetischen Lastprofile der jeweiligen Komponenten modelliert. Im Modell sind die synthetischen Lastprofile der einzelnen Gebläse in den Modulen FanEafPrim, FanEafSec, FanId1 und FanId9 modelliert. Zur besseren Übersicht wurden diese einzelnen Module der Gebläse in der Abbildung 4-2 zum Modul Fans zusammengefasst.

Die Ausgabe der generierten Lastprofile erfolgt in den Modulen Plotting, Data und Results. Im Modul Plotting werden die gewünschten Darstellungen der Lastprofile modelliert und ausgegeben. In Data erfolgt die Ausgabe der Ergebnisse in die gleichnamige Excel-Datei. Nachdem sämtliche Plots geschlossen sind, öffnet sich ein Fenster mit den abschließenden KPIs und Ergebnissen der Simulation. Die Darstellung und die Ausgabe dieses Fensters wird im Modul Results durchgeführt.

4.3 Modellierung des Elektrolichtbogenofens

In diesem Abschnitt wird die Herangehensweise zur Ermittlung und Erstellung des synthetischen Lastprofils für den Elektrolichtbogenofen beschrieben.

4.3.1 Analyse des realen Lastprofils

Aus den am Elektrolichtbogenofen gemessenen Daten lässt sich ein hochaufgelöstes reales elektrisches Lastprofil wie in Abbildung 4-3 darstellen. Deutlich zu erkennen ist dabei, dass der Betrieb des Elektrolichtbogenofens starken Lastschwankungen unterliegt. Diese führen in weiterer Folge auch zu hohen Lastschwankungen am gesamten Produktionsstandort. Der Elektrolichtbogenofen ist mit knapp 50 Prozent des durchschnittlichen elektrischen Energieverbrauchs der größte elektrische Energieverbraucher des Standorts.

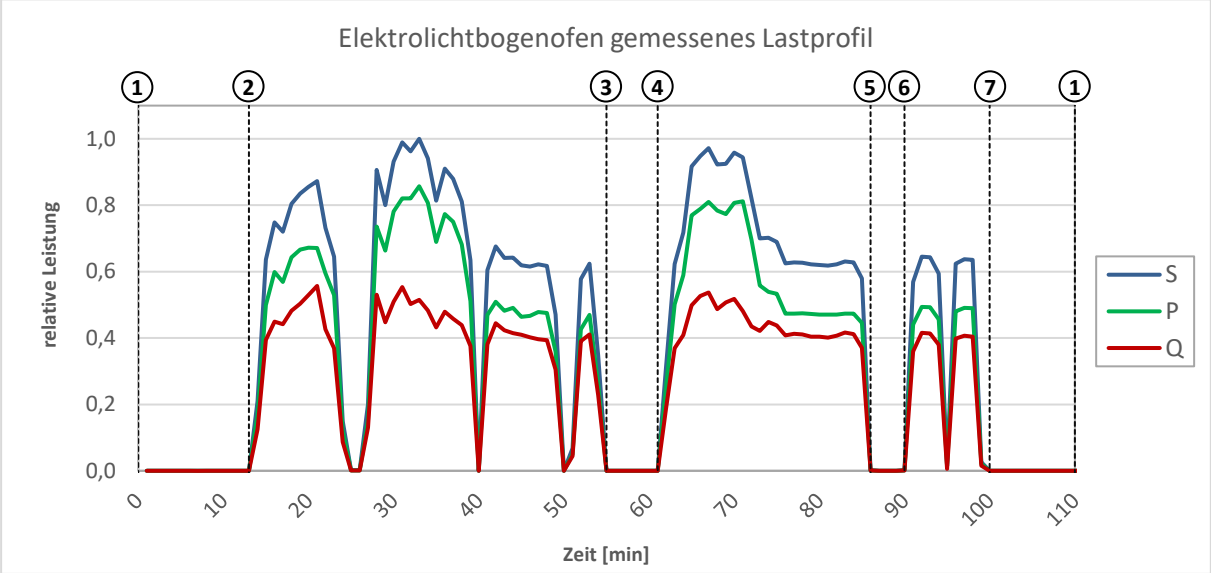

Abbildung 4-3: Gemessenes Lastprofil des Elektrolichtbogenofens

Beim Schmelzprozess von Stahlschrott zu flüssigem Stahl kommen dabei folgende Prozessschritte für jeweils eine Charge zur Anwendung:

1 → 2: Vorbereiten und Chargieren 1. Korb
2 → 3: Schmelzen 1. Korb
3 → 4: Chargieren 2. Korb
4 → 5: Schmelzen 2. Korb
5 → 6: Wartezeit
6 → 7: Frischen
7 → 1: Abstich und Vorbereitung für nächste Charge

In Abhängigkeit der zu produzierenden Menge an Stahl wiederholen sich die Prozessschritte Chargieren und Schmelzen so oft, bis die gewünschte Produktionsmenge erreicht ist und sich eine flüssige Stahlschmelze gebildet hat.

In der Abbildung 4-3 sind die gemessenen Minutenwerte der Wirk-, Blind- und Scheinleistung für die Dauer einer Charge und den dazugehörigen oben beschriebenen Prozessschritten der Stahlerzeugung mit einem Elektrolichtbogenofen dargestellt. Der Energieeintrag erfolgt dabei in den Schmelzphasen und während des Frischens.

4.3.2 Statistische Daten und Parameter

Neben den aufgezeichneten Leistungsdaten des Elektrolichtbogenofens wurden auch das Chargengewicht, die Zeit zwischen den einzelnen Abstichen, die sogenannte Tap-to-tap-Zeit, sowie das eingeblasene Sauerstoffvolumen bei Normbedingungen erfasst. Diese erfassten Daten wurden analysiert und ausgewertet und sind im Energiesystemmodell mit ihren jeweiligen Verteilungen hinterlegt und somit im Modell als Eingabegrößen definiert.

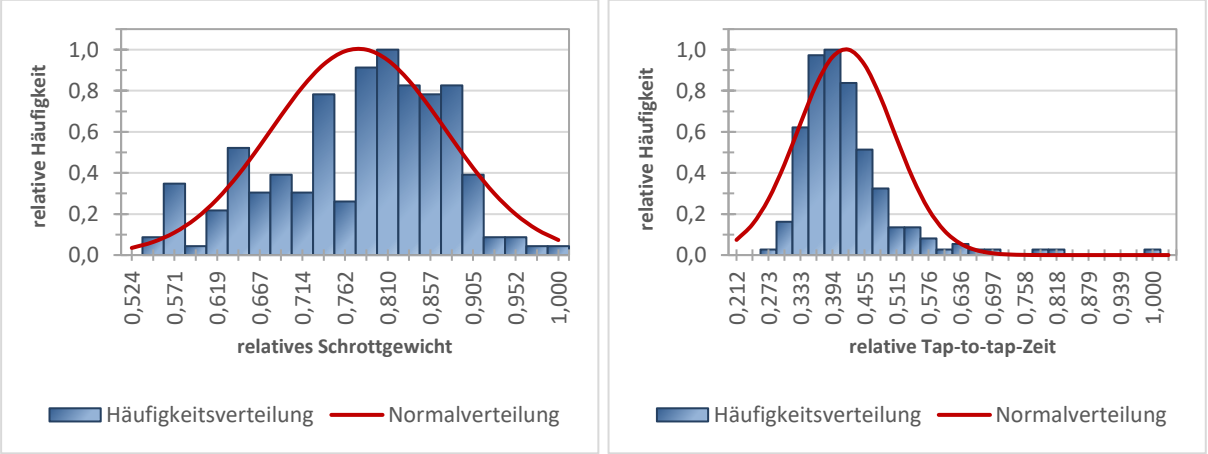

Abbildung 4-4: Verteilungen des Schrottgewichts und der „Tap to tap"-Zeiten

In Abbildung 4-4 sind die Verteilungen des Chargengewichts und der Tap-to-tap-Zeit dargestellt. Auf der linken Seite der Abbildung ist das aufgeschmolzene Schrottgewicht je Charge in Relation zum höchsten erfassten Wert im Messzeitraum, dargestellt. Die Darstellung rechts zeigt die Häufigkeitsverteilung der unterschiedlichen Tap-to-tap-Zeiten. Anhand dieser Verteilung der Tap-to-tap-Zeiten lässt sich erkennen, warum die Ermittlung der Häufigkeiten wichtig ist. Aufgrund von technischen Problemen und Kapazitätsengpässen kann es auch vorkommen, dass hohe Verzögerungen zwischen den einzelnen Abstichen auftreten. Durch Implementierung dieser Daten in das synthetische Lastprofil werden damit auch solche Unregelmäßigkeiten in der Produktion im Energiesystemmodell berücksichtigt. Um das Modell flexibel zu halten und die durchschnittlichen Schrottgewichte und Tap-to-

tap-Zeiten skalieren zu können, wurden zu den realen Häufigkeitsverteilungen zusätzlich die dargestellten Normalverteilungen implementiert.

4.3.3 Erstellung des synthetischen Lastprofils

Um das Lastprofil berechnen zu können, müssen unterschiedliche Eingangsgrößen festgelegt werden. Dazu zählen die Verteilungen der chargierten Schrottmenge, der Tap-to-tap-Zeiten sowie des Sauerstoffbedarfs. Die weiteren Eingangsgrößen sind die Übergangswahrscheinlichkeitsmatrizen, die vorab auf Basis der am Elektrolichtbogenofen gemessenen Lastprofile erstellt werden.

Mit Hilfe dieser Eingangsgrößen werden im nächsten Schritt im Modell weitere für den Prozess relevante Größen berechnet. Aus den oben genannten Verteilungen der chargierten Schrottmenge, der Tap-to-tap-Zeit und des Sauerstoffbedarfs wird nun für jede Charge ein Wert aus den realen Verteilungen gezogen und zugewiesen. Damit wird sichergestellt, dass das synthetische Lastprofil dieselbe Charakteristik wie das reale Lastprofil aufweist. Aus dem gezogenen Wert der Tap-to-tap-Zeit wird im Weiteren für jede einzelne simulierte Charge die Zeitdauer, welche angibt, ob sich der Elektrolichtbogenofen im Schmelz- oder Frischprozess befindet, festgelegt. Eine weitere relevante Größe ist die vom Ofen pro Charge aufgenommene elektrische Energie, deren Vorgabe sicherstellt, dass die zugeführte Schrottmenge auch erschmolzen werden kann und das Modell somit auch physikalisch sinnvolle Ergebnisse liefert. Dies kommt dann zum Tragen, wenn beispielsweise die Prozessdauer verkürzt wird und dadurch die durchschnittlich aufgenommene Leistung des Ofens erhöht werden muss, um die erforderliche Energiemenge, die zum Schmelzen des Schrotts notwendig ist, bereitzustellen.

Nach Ermittlung dieser Größen erfolgt eine Abfrage, ob der Elektrolichtbogenofen produzieren soll oder nicht. Dies erfolgt über die Festlegung der realen wöchentlichen Produktionsdauer. Ist der Elektrolichtbogenofen in Betrieb wird für jede Prozessphase die zuvor für die Schrottmenge benötigte Energiemenge auf eine Markov-Kette umgelegt. Entspricht die Länge der aneinandergereihten Ketten der festgelegten Tap-to-tap Zeit, wird das Lastprofil der Charge in die Zeitreihe übernommen. Wenn das nicht der Fall ist, wird das Lastprofil der Charge mit den gleichen Eingangsgrößen solange neu berechnet, bis es mit der festgelegten Tap-to-tap-Zeit übereinstimmt.

Das Lastprofil der Charge wird dann gespeichert und die Lastprofile der folgenden Charge werden hinzugereiht. Die Simulation bietet auch die Möglichkeit, beispielsweise die Tap-to-tap-Zeit zu verändern. Ist dies der Fall, wird die Eingabe in der GUI als Mittelwert

interpretiert, und die Kurve der Normalverteilung wird auf diesen verschoben. Dies gilt auch für das Schrottgewicht und den Sauerstoffbedarf.

4.4 Modellierung des Pfannenofens

Für die Modellierung des Pfannenofens wurden die Daten eines der vier am Standort befindlichen Pfannenöfen erhoben. Ähnlich wie beim Elektrolichtbogenofen werden diese Daten analysiert und ausgewertet. Danach erfolgt die Erstellung des synthetischen Lastprofils.

4.4.1 Analyse des realen Lastprofils

Der Pfannenofen wird zum Aufheizen der Stahlschmelze und zu deren Homogenisierung benötigt. Im Vergleich zum gemessen Lastprofil des Elektrolichtbogenofens, der starke Leistungsschwankungen aufweist, ist Leistungsaufnahme des Pfannenofens für das gemessene unten dargestellte Lastprofil in Abbildung 4-5 relativ gleichförmig. Die Dauer des Energieeintrags erfolgt hingegen unterschiedlich lang und ist in erster Linie von den am Pfannenofen ablaufenden sekundärmetallurgischen Prozessen abhängig. Die Prozesse, welche am Pfannenofen stattfinden, können in folgende Teilprozesse eingeteilt werden:

1 → 2: Zeitdauer von Abstich am EAF bis Beginn der Wärmebehandlung am LF
2 → 3: Prozessphase 1: Aufheizen und Homogenisieren der Stahlschmelze
3 → 4: Entgasen
4 → 5: Prozessphase 2: Aufheizen und Homogenisieren der Stahlschmelze
5 → 6: Zeitdauer bis zum Blockguss

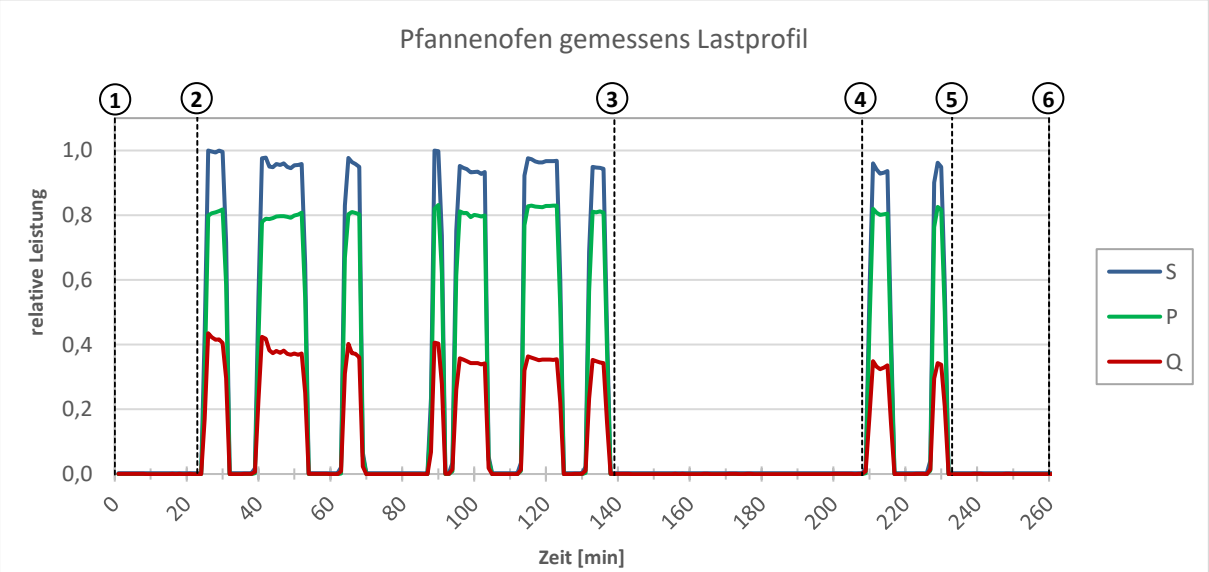

Abbildung 4-5: Gemessenes reales elektrisches Lastprofil einer Charge am Pfannenofen

Das in Abbildung 4-5 dargestellte Lastprofil des Pfannenofens zeigt den Verlauf der Wirk-, Blind- und Scheinleistung für die einzelnen Prozessschritte, die eine Charge am Pfannenofen durchläuft. Für die Modellierung des Pfannenofens wurden von den gemessenen Daten 20 sogenannte Referenzchargen ausgewählt und analysiert.

4.4.2 Auswertung der Daten und Festlegen der Simulationsparameter

Für die Modellierung des Pfannenofens wurden 20 repräsentative Referenzchargen analysiert. Aus diesen Referenzchargen wurden die unterschiedlichen Prozesszeiten ermittelt und überprüft, welche Zusammenhänge und Abhängigkeiten feststellbar sind. Die gewählten Referenzchargen sind mit aufsteigender Prozessdauer in der Abbildung 4-6 dargestellt. Im Gegensatz zum Elektrolichtbogenofen kann hier keine Korrelation zwischen Prozessdauer und Energieverbrauch nachgewiesen werden. Beispielsweise hat die Charge 13 die zweitlängste Prozessdauer, aber dabei nur einen relativen Energieverbrauch von unter 50 Prozent, gemessen am Maximum der Referenzchargen, welcher deutlich unter dem arithmetischen Mittel der 20 Chargen liegt.

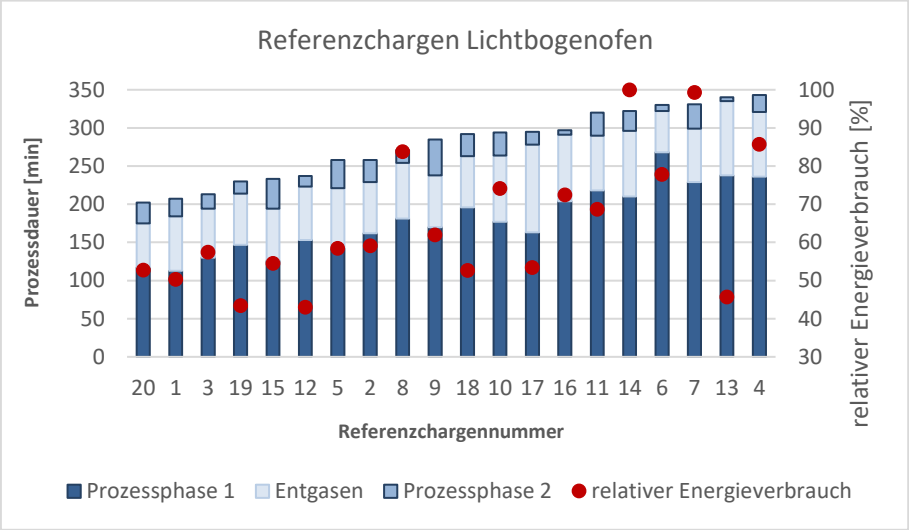

Abbildung 4-6: Darstellung der Prozessdauer der einzelnen Prozessschritte und des jeweiligen Energieverbrauchs der 20 Referenzchargen

Weiters kann auch keine Korrelation zwischen der Dauer der ersten Prozessphase und der Dauer der zweiten Prozessphase hergestellt werden. Um den notwendigen Energieeintrag bestimmen zu können, müsste man zusätzlich zu den gemessenen Daten die Temperaturwerte der Stahlschmelze zur Verfügung haben, da der Stahl nach den sekundärmetallurgischen Prozessen auf Halte- bzw. Gusstemperatur aufgeheizt werden muss. Damit könnte man die benötigte Energie, die notwendig ist, um die Stahlschmelze auf gewünschter Temperatur zu halten, berechnen und eine Korrelation zwischen der Prozessdauer und Leistung des Pfannenofens herstellen.

Für die Erstellung des synthetischen Lastprofils wurden somit Intervalle für die Zeitdauer der unterschiedlichen Prozessphasen ermittelt. Das heißt, dass für die 20 Referenzchargen die Minimal- und Maximalwerte für die jeweiligen Prozessphasen ermittelt wurden.

4.4.3 Erstellung des synthetischen Lastprofils

Vorab müssen wieder die Eingangsgrößen für die Erstellung des synthetischen Lastprofils festgelegt werden. Dazu zählen die unterschiedlichen Zeitdauern der einzelnen Prozessschritte, die Nennleistung des jeweiligen Pfannenofens und die Übergangswahrscheinlichkeitsmatrizen für die jeweiligen Prozessschritte. Weiters werden nun Startbedingungen festgelegt. Dabei wird abgefragt, ob der Abstich am Elektrolichtbogenofen erfolgte und ob der Pfannenofen belegt oder frei ist. Die Zeitdauer zwischen dem Abstich am Elektrolichtbogenofen und dem Start des Aufheizprozesses am Pfannenofen wird in der graphischen Benutzeroberfläche als Zeitspanne eingegeben. Aus dieser Zeitspanne wird dann für jede einzelne Charge ein Wert zufällig bestimmt.

Für die Erstellung des synthetischen Lastprofils werden im Energiesystemmodell in PyCharm zwei Module verwendet. Im Modul „SecondaryMetallurgy“ sind die Prozessphasen definiert. In diesem Modul folgt auch die Unterscheidung zwischen VD- und VOD-Chargen. Die aus den Referenzchargen ermittelten Minimal- und Maximalwerte der einzelnen Prozessschritte sind auch im Modul „SecondaryMetallurgy“ als Intervalle hinterlegt. Diese können aber auch in der graphischen Benutzeroberfläche verändert werden. Für jede einzelne Charge werden nun individuell die Zeitdauern für die einzelnen Prozessschritte per Zufallszahl aus den vorgegebenen Intervallen gezogen. Im nächsten Schritt wird überprüft, ob es sich um eine Standard- oder um eine VOD-Charge handelt. Der prozentuale Anteil an VOD-Chargen ist mit sieben Prozent standardmäßig festgelegt. Bei einer VOD-Charge verändert sich der Prozess insofern, dass im Modell die Zeit der Dauer der VOD-Behandlung zusätzlich noch vor dem erstmaligen Beginn des Aufheizens addiert wird und dieser Zeitschritt somit länger wird. Daraus folgt, dass der gesamte sekundärmetallurgische Prozess länger dauert.

Im zweiten Modul „LadleFurnace“ wird nun das synthetische Lastprofil erzeugt. Dazu wird über die Chargennummer festgelegt, auf welchen Ofen die Pfanne verteilt wird und somit die Nennleistung des jeweiligen Pfannenofens und der Leistungsfaktor festgelegt. Befindet sich die Pfanne nun im Status der Prozessphase 1 oder der Prozessphase 2, wird ähnlich wie beim Lichtbogenofen der aktuelle Wert der Übergangswahrscheinlichkeitsmatrix mit der jeweiligen Nennleistung des Pfannenofens multipliziert. Daraus erhält man die Wirkleistung des Pfannenofens. Mit dem vorab ermittelten Leistungsfaktor wird die Blind- und Scheinleistung berechnet. Dies erfolgt für jeden Zeitschritt und durch Aneinanderreihen

dieser Leistungen erhält man die Markov-Kette. Abschließend erfolgt die Berechnung der verbrauchten elektrischen Energie durch Aufsummieren der Leistungen über die Prozessdauer.

4.5 Modellierung der Gebläse

In diesem Abschnitt wird die Modellierung der jeweiligen Gebläse beschrieben. Die Vorgehensweise orientiert sich an der Modellierung des Elektrolichtbogenofens und des Pfannenofens.

4.5.1 Modellierung der Primär- und Sekundärgebläse des Lichtbogenofens

Zur Analyse der Primär- und Sekundärentstaubung des Elektrolichtbogenofens wurden jeweils über mehrere Wochen die Daten der Wirk-, Blind- und Scheinleistung der jeweiligen Gebläse gemessen. Für die Modellierung des synthetischen Lastprofils werden diese Daten im folgenden Kapitel ausgewertet. Methodisch kommen hier Häufigkeitsverteilungen und Normalverteilungen zur Anwendung.

4.5.1.1 Analyse des realen Lastprofils

Die in Abbildung 4-7 dargestellten Lastprofile zeigen den realen Verlauf der Wirk-, Blind- und Scheinleistung der Primär- und Sekundärgebläse für einen Zeitraum von vier Stunden. Aus den beiden Profilen lässt sich die Betriebsweise der Gebläse erkennen. Dabei kann man den Betrieb der Gebläse in die zwei Leistungsstufen, Teil- und Volllast, unterteilen.

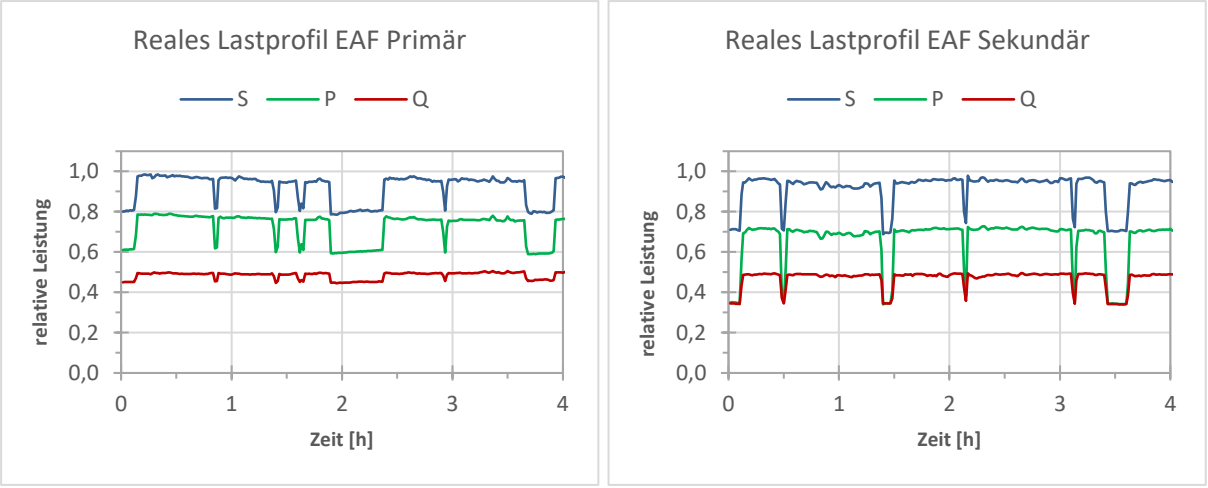

Abbildung 4-7: Gemessenes Lastprofil des Primär- und Sekundärgebläses des Elektrolichtbogenofens

Die Gebläse werden in Volllast betrieben, wenn am Lichtbogenofen Schrott erschmolzen wird. Dieser Vorgang lässt sich in der linken Darstellung des Primärgebläses erkennen. Zwischen null und zwei Stunden wird hier am Elektrolichtbogenofen eine Charge hergestellt. Dabei sind auch deutlich die unterschiedlichen Prozesse, welche während der

Stahlherstellung am Elektrolichtbogenofen durchgeführt werden, zu erkennen. Nach der ersten Charge wird das Gebläse wieder in Teillast betrieben, bis am Elektrolichtbogenofen mit der nächsten Charge begonnen wird. Ähnlich äußert sich die Situation in der Darstellung des sekundären Entstaubungsgebläses auf der rechten Seite der Abbildung 4-7. Hier sind auch zwei Chargen, bei denen die Prozessschritte am Lichtbogenofen erkennbar sind, dargestellt.

4.5.1.2 Auswertung der gemessenen Daten

Da die Primär- und Sekundärgebläse an den Elektrolichtbogenofenprozess gekoppelt sind, müssen für die Modellierung keine Prozesszeiten, sondern lediglich die Leistungsdaten der Voll- und Teillastbereiche ermittelt werden. In der Abbildung 4-8 ist zu erkennen, dass die Leistungsschwankungen in den jeweiligen Betriebsweisen, ob für Primär- oder Sekundärgebläse, sehr gering ausfallen.

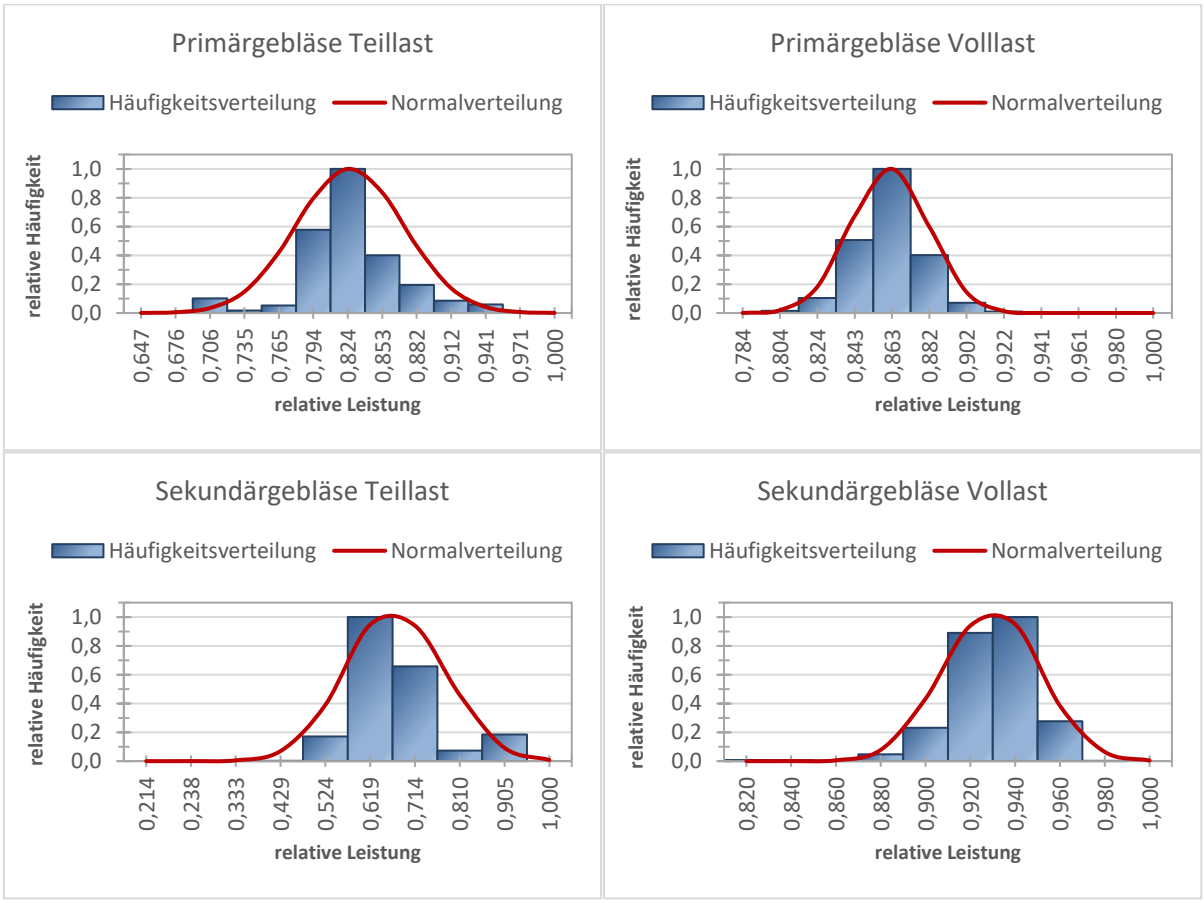

Abbildung 4-8: Verteilungen der Wirkleistung unter Teil- und Volllast der Primär- und Sekundärgebläse des Elektrolichtbogenofens

Wie in Abbildung 4-8 dargestellt, können die Leistungsverteilungen über Normalverteilungen mit gewichteten Mittelwerten und gewichteten Standardverteilungen beschrieben werden. Diese Häufigkeitsverteilungen sowie die dazugehörigen Normalverteilungen werden im

Modell hinterlegt, sodass man sich bei der Simulation entscheiden kann, welche Verteilung für die Lastprofilberechnung herangezogen wird. Für die Betriebsbereiche Teil- und Volllast, werden zusätzlich die Leistungsfaktoren für das Primär- und Sekundärgebläse aus den gemessenen Daten bestimmt. Im Modell werden die arithmetischen Mittelwerte der Leistungsfaktoren für jeden Betriebsbereich als konstante Werte hinterlegt und können nachträglich verändert werden.

4.5.1.3 Erstellung des synthetischen Lastprofils

Die Erstellung des synthetischen Lastprofils erfolgt für das Primär- und Sekundärgebläse jeweils in einem eigenen Modul. Dies dient nur zur besseren Übersicht, denn beide Modellierungen werden genau gleich durchgeführt.

Zuerst werden die stochastischen Leistungsdaten festgelegt. Dazu werden die Klassen der Wirkleistungen mit ihren zugehörigen Häufigkeiten als Listen in das Modell eingefügt. Danach erfolgt die Initialisierung der Gebläse.

Wie oben schon kurz erwähnt, ist das Primär- und Sekundärgebläse an die Prozessphasen des Lichtbogenofens gekoppelt. Deswegen erfolgt hier im nächsten Schritt eine erste Abfrage, ob sich der Elektrolichtbogenofen gerade im Status „Produktion“ befindet, was an Werktagen der Fall ist. Bei der nächsten Abfrage wird überprüft, ob der Lichtbogenofen gerade in Betrieb ist und ob die bezogene Leistung größer als null ist. Trifft das zu, wird das Primär- und Sekundärgebläse in Volllast betrieben. Im Betrieb in Volllast wird dem Lastprofil aus der realen Häufigkeitsverteilung ein Wirkleistungswert zugewiesen und im gleichen Schritt über den Leistungsfaktor die Blind- und Scheinleistung berechnet. Alternativ kann hier auch ein anderer Wert für die Wirkleistung in der GUI festgelegt werden, falls sich die installierte Gebläseleistung ändert. Dieser Wert wird dann als Mittelwert interpretiert und mit der oben dargestellten Normalverteilung verteilt. Das Modell bietet außerdem die Möglichkeit einen konstanten Wert für die Wirkleistung festzulegen.

Befindet sich der Lichtbogenofen zwar nicht im Betrieb, jedoch im Status „Produktion“, wird das Primär- und Sekundärgebläse in Teillast betrieben. Dazu wird wieder für den aktuellen Zeitschritt ein Wert für die Wirkleistung aus der vorgegebenen Liste der realen Häufigkeitsverteilung zugewiesen und die Blind- und Scheinleistung berechnet. Auch hier kann dem Modell wieder ein alternativer oder statischer Wert für die Wirkleistung vorgegeben werden.

Abschließend erfolgt noch die Berechnung des elektrischen Energieverbrauchs. Diese Abläufe werden für jeden Zeitschritt der Simulation für das Primär- und Sekundärgebläse durchgeführt.

4.5.2 Modellierung des Entstaubungsgebläses der Halle 9

Die Wirk-, Blind- und Scheinleistung des Entstaubungsgebläses der Halle 9 wurde für einen Zeitraum von über drei Wochen aufgezeichnet. Zur Modellierung des synthetischen Lastprofils werden diese Daten analysiert und ausgewertet, um einen Überblick über die Betriebsweise zu erhalten.

4.5.2.1 Analyse des realen Lastprofils

Ein Auszug aus dem gemessenen Lastprofil für die Wirk-, Blind- und Scheinleistung für zwölf Stunden ist in Abbildung 4-9 dargestellt. In diesem Lastprofil kann man zwei Betriebszustände definieren. Ersterer ist der Betrieb, in dem das Entstaubungsgebläse in Teillast, also mit ca. 50 Prozent der Spitzenleistung betrieben wird. Der zweite Betriebszustand beschreibt den Betrieb des Entstaubungsgebläses in Volllast.

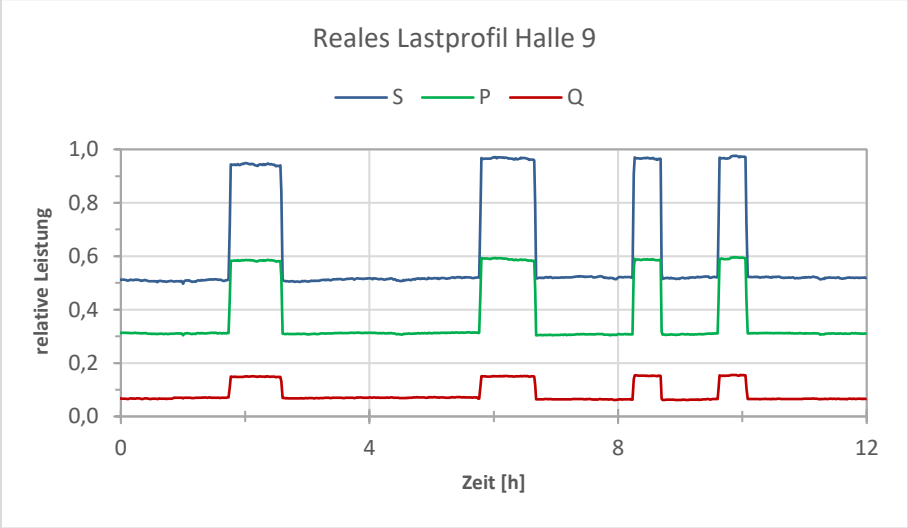

Abbildung 4-9: Gemessenes Lastprofil des Entstaubungsgebläses der Halle 9 für den Zeitraum von zwölf Stunden

Aus dem gemessenen Profil kann man gut erkennen, dass in den jeweiligen Betriebszuständen kaum Leistungsschwankungen auftreten. Das Entstaubungsgebläse der Halle 9 wird immer dann auf Volllast geschalten, wenn der Blockguss erfolgt. Das heißt, die Zeitdauer, in der das Gebläse unter Volllast betrieben wird, setzt sich aus der Gussdauer sowie der Vor- und Nachlaufzeit zusammen.

4.5.2.2 Auswertung der gemessenen Daten

Aus der Analyse kann man nun ableiten, dass für das Modell die Leistungen der Voll- und Teillastbereiche sowie die Betriebsdauer unter Volllast relevant sind. Die in der Abbildung 4-10 dargestellte Häufigkeitsverteilung stellt die Wirkleistung unter Teillast dar. Weiters ist auch eine Normalverteilung mit einem gewichteten Mittelwert und einer gewichteten

Standardabweichung angeführt. In diesem Diagramm kann man noch deutlicher die geringe Varianz der Wirkleistung des Entstaubungsgebläses erkennen. Um die Nennleistung verändern zu können, wird auch die Normalverteilung im Modell hinterlegt.

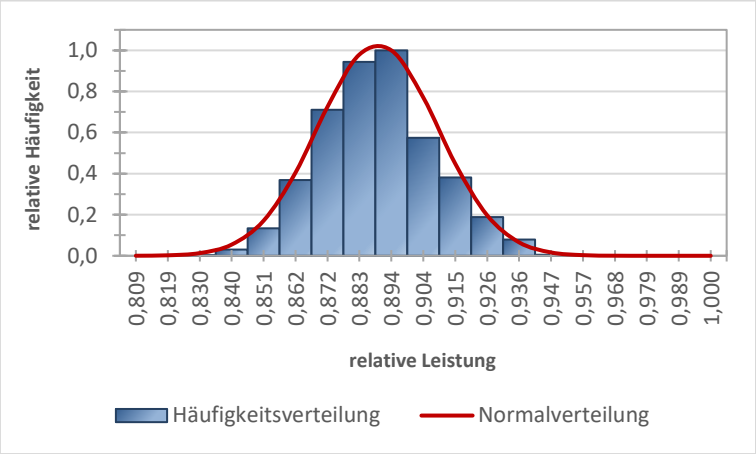

Abbildung 4-10: Verteilungen der Wirkleistung unter Teillast des Entstaubungsgebläses der Halle 9

In Abbildung 4-11 sind die Verteilungen für die Betriebsweise unter Volllast dargestellt. Ähnlich wie bei der Häufigkeitsverteilung der Teillast kann auch hier eine Normalverteilung zur Beschreibung der Verteilung der Wirkleistung herangezogen werden. Im Gegensatz dazu trifft das auf die Verteilung der Zeitdauer, welche angebt, wie lang das Gebläse in Volllast betrieben wird, nicht zu. Hier ist ein eindeutiges Maximum bei 43 Prozent des Absolutwertes der maximalen Zeitdauer zu finden. In der Simulation werden die Werte mit ihren dazugehörigen Häufigkeiten im Modell hinterlegt

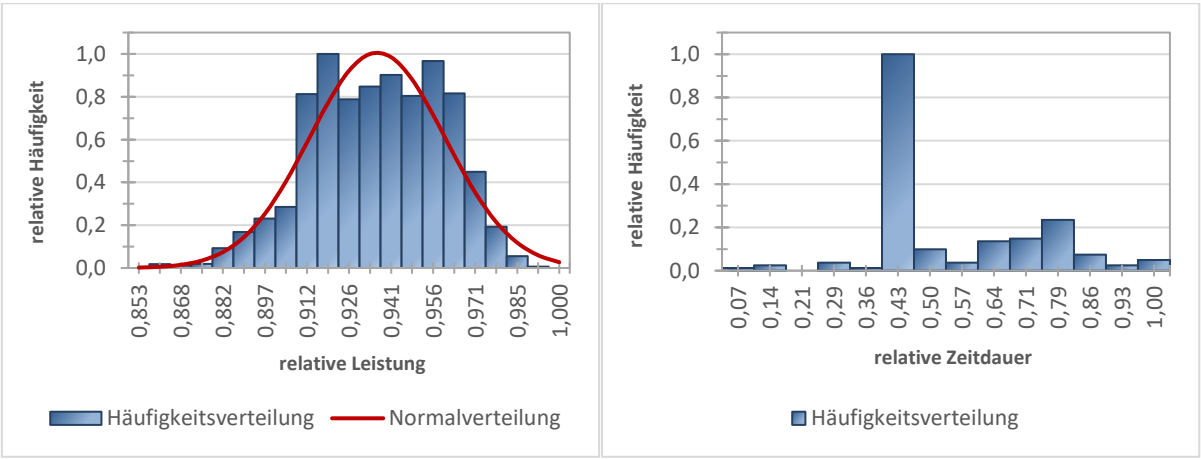

Abbildung 4-11: Verteilungen der Wirkleistung und der Zeitdauer unter Volllast des Entstaubungsgebläses der Halle 9

Zusätzlich zu diesen stochastischen Parametern werden aus den gemessenen Daten noch die Leistungsfaktoren für den Betrieb in Voll- und Teillast separat ermittelt.

4.5.2.3 Erstellung des synthetischen Lastprofils

Die Eingangsgrößen werden zur Erstellung des synthetischen Lastprofils wieder vorab festgelegt. Dazu zählt die Wirkleistung in Teil- und Volllast sowie die Zeitdauer für den Vollastbetrieb. Diese Größen werden im Modell mit Hilfe der oben bestimmten jeweiligen Häufigkeitsverteilungen hinterlegt. Wie schon beim Entstaubungsgebläse des Elektrolichtbogenofens kann man auch hier entscheiden, ob man die Wirkleistung, egal ob Teil- oder Volllast, verändern will. Diese wird dann mithilfe der zuvor berechneten Normalverteilung verschoben. Diese Möglichkeit gibt es für die Zeitdauer der Volllast nicht, da sich die Gusszeiten nicht verändern würden, falls ein Gebläse mit einer anderen Nennleistung installiert wird.

Im nächsten Schritt wird eine Prozesszeit definiert, die für den Zeitraum zwischen dem Ende der Behandlung am Pfannenofen und dem Beginn des Blockgusses steht. Diese wird als Intervall in der GUI definiert und aus diesem Intervall wird dann eine Zufallswert generiert.

Nun wird überprüft, ob sich die Anlage gerade in Betrieb befindet. Das erfolgt über das Modul „Days“, indem festgelegt ist, ob es sich um einen Werktag handelt. Ist diese Bedingung erfüllt, wird die Wirkleistung der Teillast für den jeweiligen Zeitschritt berechnet. Durch Umrechnung mit dem Leistungsfaktor der Teillast werden im gleichen Zug die Blind- und Scheinleistung berechnet. Entspricht der aktuelle Zeitschritt gerade dem Status „Wochenende“, werden die Werte für die Wirk-, Blind- und Scheinleistung auf null gesetzt.

Im nächsten Schritt wird der Volllastanteil des Entstaubungsgebläses erzeugt. Dazu erfolgt die Abfrage, ob die Pfanne sich gerade im Status „Guss“ befindet. Dieser Status wird im Modul „SecondaryMetallurgy“ definiert und folgt direkt auf die Behandlung am Pfannenofen. Befindet sich nun die Pfanne im Status „Guss“, wird eine Dauer, Vor- und Nachlaufzeit sowie Gussdauer des gesamten Gussprozess aus den eingegebenen Daten generiert. Für diesen generierten Zeitraum wird nun für jeden einzelnen Zeitschritt ein Wirkleistungswert für die Volllastleistung des Gebläses festgelegt. Wie zuvor erfolgt im selben Zeitschritt die Berechnung der Blind- und Scheinleistung mit dem Leistungsfaktor der Volllast. Weiters wird noch die verbrauchte elektrische Energie vom Gebläse berechnet.

Diese Schritte und Abläufe werden dann für jeden Zeitschritt durchgeführt und durch aneinanderreihen dieser erhält man das synthetische Lastprofil für das Entstaubungsgebläse der Halle 9.

4.5.3 Modellierung des Entstaubungsgebläses der Halle 1

Für die Modellierung der Entstaubungsgebläse der Halle 1 wurden die Werte der Wirk-, Blind- und Scheinleistung für ca. acht Tage aufgezeichnet. Bei der Modellierung wurde ähnlich der Modellierung der Teillast des Entstaubungsgebläses der Halle 9 vorgegangen.

4.5.3.1 Analyse des realen Lastprofils

Das reale gemessene Lastprofil, das in Abbildung 4-12 für einen Zeitraum von zwölf Stunden dargestellt ist, stellt einen typischen Werktag dar, an dem das Entstaubungsgebläse in Betrieb ist. Deutlich zu erkennen ist, dass die Lastschwankungen im Betrieb sehr gering ausfallen und nur geringe Abweichungen aufweisen. An Wochenenden ist das Gebläse nicht in Betrieb und für diese Zeitdauer wird im Modell ein konstanter Wert, der sich aus den Messdaten ergibt, festgelegt.

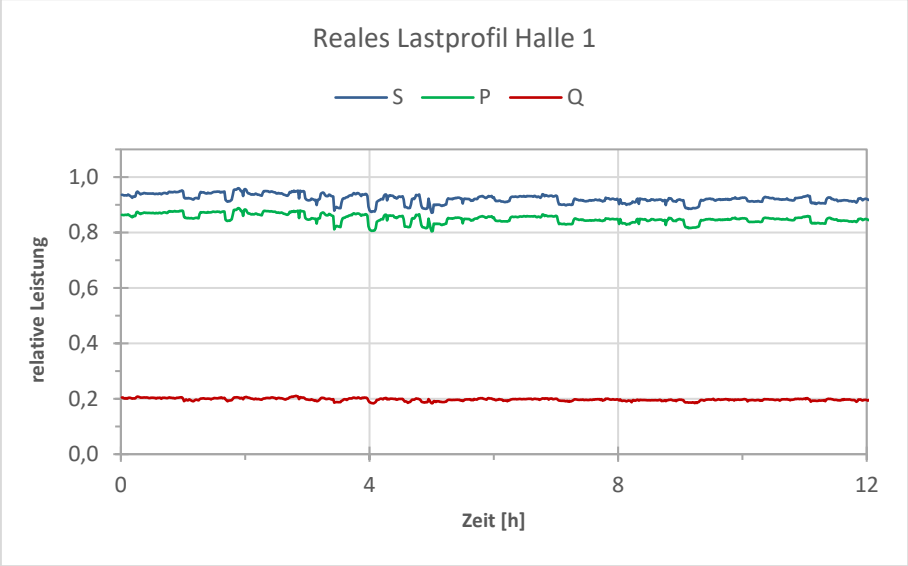

Abbildung 4-12: Gemessenes Lastprofil des Entstaubungsgebläses der Halle 1 für den Zeitraum von zwölf Stunden

4.5.3.2 Auswertung der gemessenen Daten

Da das Entstaubungsgebläse der Halle 1 an den Werktagen durchgehend in Betrieb ist, muss hier keine weitere Bestimmung der Prozesszeiten durchgeführt werden. Für die auftretenden unterschiedlichen Wirkleistungen wurde jedoch eine Häufigkeitsverteilung erstellt. Die in Abbildung 4-13 dargestellte Häufigkeitsverteilung zeigt die Verteilung der Wirkleistungen. Dabei ist zu erkennen, dass diese einer Normalverteilung entsprechen. Im Modell sind einerseits die realen Häufigkeitsverteilungen hinterlegt, aber andererseits kann die der Häufigkeitsverteilung entsprechende Normalverteilung um den Mittelwert verschoben werden. Dadurch kann die Wirkleistung des Entstaubungsgebläses verändert

werden. Weiters wurde für das Modell der Leistungsfaktor aus den gemessenen Daten ermittelt, um im Modell die Blind- und Scheinleistung berechnen zu können.

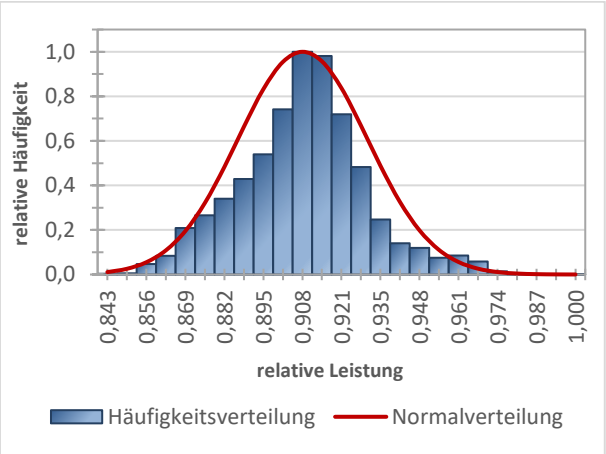

Abbildung 4-13: Verteilungen der Wirkleistung des Entstaubungsgebläses der Halle 1

4.5.3.3 Erstellung des synthetischen Lastprofils

Für die Erstellung des synthetischen Lastprofils erfolgt vorab die Festlegung der stochastischen Parameter. Dazu werden die Klassen der Wirkleistungen mit deren jeweiligen Häufigkeiten im Programm hinterlegt. Weiters ist auch eine Normalverteilung, die der Häufigkeitsverteilung entspricht, im Programm hinterlegt. Wird diese Option ausgewählt, wird die Verteilung um den Mittelwert verschoben. Eine weitere Option stellt die Eingabe eines konstanten Wertes dar.

Nach der Initialisierung des Lastprofils erfolgt die Abfrage, ob das Entstaubungsgebläse gerade in Betrieb ist. Diese Zeiträume werden im Python Modul „Days“ festgelegt und werden dann für die gesamte Simulation konstant angenommen, da die gemessenen Daten nur ein Wochenende enthielten und somit nur ein Zeitraum bekannt ist.

Ergibt nun die Abfrage, dass das Gebläse in Betrieb ist, wird ermittelt, welche Option für die Festlegung der Leistung gewählt wurde. Danach wird für diesen Zeitschritt die Wirkleistung zugewiesen. Aus dieser zugewiesenen Wirkleistung wird anschließend der Energieverbrauch berechnet. Dieser Ablauf erfolgt für jeden Zeitschritt, das heißt, für jeden Minutenwert der Simulation wird dann ein eigener Wert der Wirkleistung zugewiesen. Im Falle, dass das Gebläse nicht in Betrieb ist, wird der Wert der Wirkleistung auf null gesetzt, was zur Folge hat, dass die Blind- und Scheinleistung auch den Wert von Null aufweisen. Somit erhält man dann für jeden Zeitpunkt der Simulation einen Wirkleistungswert. Über den aus den gemessenen Daten ermittelten Leistungsfaktor werden dann auch die Beträge der Blind und Scheinleistung berechnet.

4.6 Modellierung der Grundlast

Zur Modellierung der Grundlast wurde das Gesamtlastprofil des Standorts Breitenfeld vom Jahr 2019 herangezogen. Dieses wurde analysiert und der niedrigste Wert der Wirkleistung als Grundlastwert für das Energiesystemmodell festgelegt. Dafür wurde darauf geachtet, dass dieser Wert außerhalb der Revision liegt. Für den Leistungsfaktor cosφ wurde ein Wert von 0,95 angenommen.

Für die Erstellung des synthetischen Lastprofils der Grundlast wird ein eigenes Modul erstellt. Für jeden Zeitschritt der Simulation wird der ermittelte konstante Wert der Wirkleistung des Lastprofils zugewiesen. Mit dem Leistungsfaktor wird die Blind- und Scheinleistung berechnet. Abschließend wird der Energieverbrauch der Grundlast ermittelt.

4.7 Modellierung der übrigen Verbraucher

Um die übrigen Verbraucher zu modellieren, wurde ein Lastprofil aus den gegebenen Datensätze ermittelt. Dafür wurde aus dem Gesamtlastprofil eine Woche gewählt, in der der Elektrolichtbogenofen und der Pfannenofen gemessen wurden. Für die restlichen Komponenten wurde auch jeweils eine Referenzwoche ausgewählt. Da das Gesamtlastprofil eine zeitliche Auflösung von 15 Minuten aufweist, wurden die gemessenen Lastprofile auf dieses Intervall angepasst. Dazu wurden die Minutenwerte in Viertelstunden-Mittelwerte umgerechnet. Dieses Vorgehen erfolgte für alle gemessenen Lastprofile.

Um nun das Lastprofil der übrigen Verbraucher zu ermitteln, wurden die bekannten Lastprofile vom Gesamtlastprofil subtrahiert. Das Ergebnis dieser Subtraktion stellt das Lastprofil der übrigen Verbraucher am Standort Breitenfeld dar. Dieses Lastprofil wurde dann analysiert und ausgewertet. Die Analyse hat gezeigt, dass die Last während der Betriebszeit des Elektrolichtbogenofens und der sekundärmetallurgischen Anlagen eine andere Charakteristik aufweist, als wenn dieser wochenends nicht in Betrieb sind.

Zur Erstellung des synthetischen Lastprofils erfolgt somit erst eine Abfrage in welchen Status sich das Stahlwerk befindet. Werktags wird das Lastprofil über eine zuvor erstellte Markov-Kette mit Multiplikation der Nennleistung erstellt. Am Wochenende wird für jeden Zeitschritt ein Wert aus einer festgelegten Liste mit Wirkleistungen und der dazugehörigen Häufigkeiten zugewiesen. Über einen angenommenen Leistungsfaktor cosφ von 0,95 wird die Blind- und Scheinleistung berechnet. Abschließend erfolgt die Ermittlung der verbrauchten Energiemenge.

5 VERGLEICH DER REALEN UND SYNTHETISCHEN LASTPROFILE

In diesem Kapitel werden die gemessenen Daten mit den synthetischen, aus dem Modell erzeugten Daten verglichen. Dazu werden mit dem Modell Simulationen für die jeweiligen Aggregate durchgeführt, bei denen die erzeugten synthetischen Daten in einem Excel-File gespeichert und ausgewertet werden. Die Ergebnisse der Auswertung werden den realen Daten gegenübergestellt.

5.1 Elektrolichtbogenofen

Für den graphischen Vergleich zwischen dem realen und dem synthetischen Lastprofil wurden aus dem Modell die Leistungsdaten für eine beispielhafte Charge ausgewertet. In Abbildung 5-1 ist dieses synthetische Lastprofil des EAFs für eine Charge in der Abbildung rechts dargestellt. Dabei wird schon wie in den Abbildungen der Lastprofile zuvor die Wirk-, Blind- und Scheinleistung dargestellt. Als Vergleich wurde dasselbe reale Lastprofil wie in Abbildung 4-3 verwendet. Wie im realen Lastprofil sind auch im synthetischen Lastprofil die einzelnen Prozessschritte, die eine Charge am EAF durchläuft, erkennbar. Das Aufschmelzen des ersten Korbs erfolgt im Bereich von 5 bis 47 Minuten. Danach wird der zweite Korb chargiert. Das Aufschmelzen des zweiten Korbs erfolgt dabei in den Minuten 49 bis 77. Danach erfolgt noch der Vorgang des Frischens.

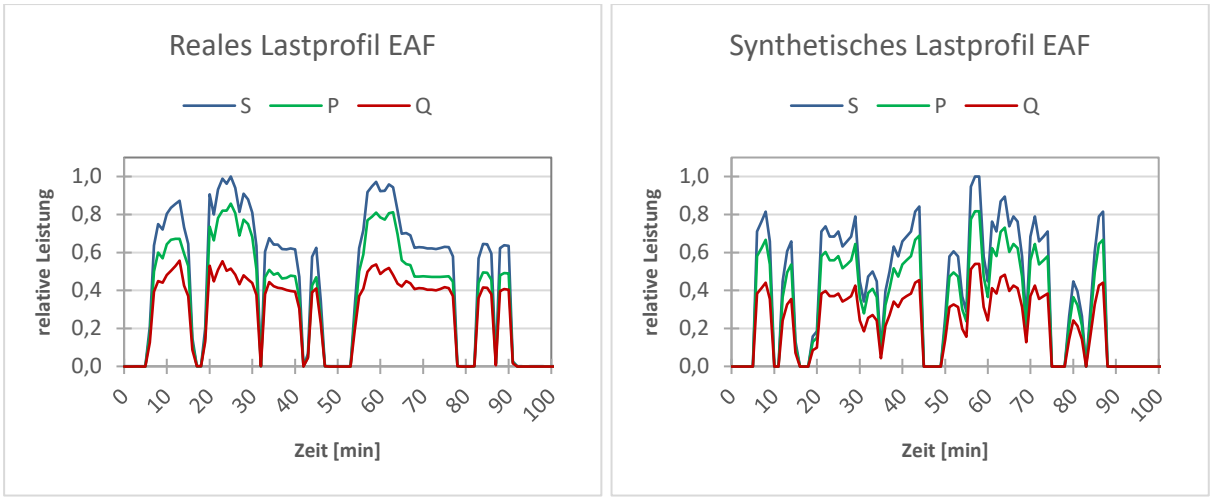

Abbildung 5-1: Vergleich des realen und synthetischen Lastprofils des Elektrolichtbogenofens

Zum graphischen Vergleich der beiden Lastprofile wurde speziell ein synthetisches Lastprofil ausgewählt, dass eine ähnliche Charakteristik wie das reale Lastprofil aufweist. An dieser Stelle muss gesagt werden, dass es bei den realen Lastprofilen auch keine eindeutige Charakteristik gibt, da es sich hierbei um stochastische Prozesse handelt. Das bedeutet, dass die Leistungsverläufe in den Prozessphasen immer unterschiedliche Formen bzw. Verläufe

annehmen. Aus diesem Grund stellt sich der rein graphische Vergleich für den Elektrolichtbogenofen als nicht sehr aussagekräftig dar.

Um das synthetische Lastprofil des Elektrolichtbogenofens besser bewerten zu können, werden die Wirkleistungen des Lastprofils als geordnete Dauerlinien dargestellt. In Abbildung 5-2 sind die geordneten Dauerlinien für die realen und die synthetischen Wirkleistungen des Elektrolichtbogenofens abgebildet. Die beiden Dauerlinien sind auf der Ordinate auf die maximale gemessene Leistung des EAFs normiert. Auf der Abszisse erfolgt die Normierung auf die reale Betriebszeit des EAFs, das heißt, wenn am Ofentransformator Leistung bezogen wird, für eine durchschnittliche Produktionswoche. Zur Bildung der realen geordneten Dauerlinie wurden die am EAF gemessenen Wirkleistungsdaten für drei Produktionswochen betrachtet. Dafür werden die Wirkleistungen geordnet und aus den Daten der drei Wochen ein Durchschnitt gebildet. Dieser Durchschnitt der realen Wirkleistungsdaten ist in der Abbildung als blaue Linie dargestellt.

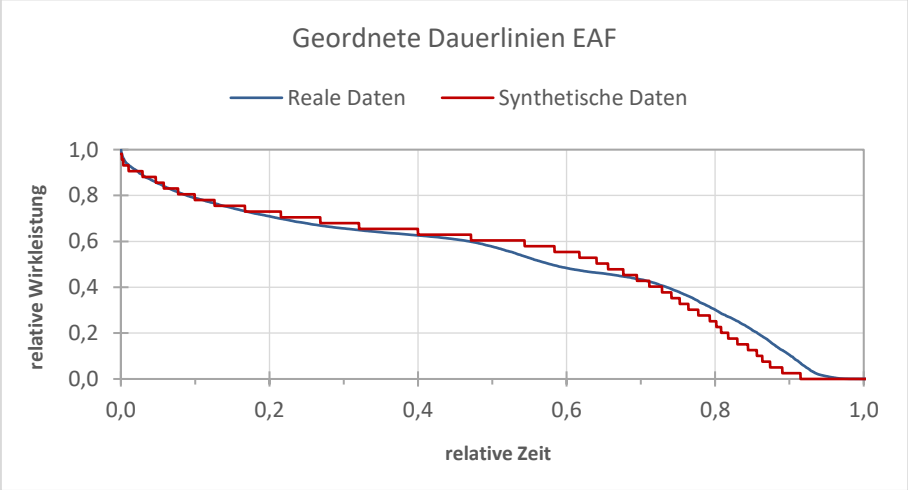

Abbildung 5-2: Vergleich der geordneten Dauerlinien der realen und synthetischen Wirkleistung des Elektrolichtbogenofens

Für die Bildung der synthetischen Dauerlinie wurden mit dem Modell drei Simulationen für jeweils einen Zeitraum von einer Woche durchgeführt. Die daraus erzeugten Wirkleistungsdaten wurden geordnet und für die Dauerlinie wurde ein Durchschnittswert gebildet. Die synthetische Dauerlinie weist gegenüber der realen Dauerlinie ein stufenförmiges Profil auf. Dass liegt daran, dass die Leistung des EAFs im Modell in Megawatt-Schritten definiert ist. Für die Genauigkeit des Modells spielt das jedoch keine besondere Rolle, da für jeden Minutenschritt ein anderer Leistungswert ermittelt wird und das Modell somit eine sehr hohe Auflösung aufweist. Wenn man nun die beiden Dauerlinien vergleicht, erkennt man, dass die synthetische Linie im mittleren Zeitbereich eine höhere relative Wirkleistung aufweist und dafür am Ende des Zeitbereichs gegenüber der realen

Dauerlinie abfällt. Das kann daran liegen, dass die aus den Verteilungen zufällig gezogenen Tap-to-tap-Zeiten im Schnitt niedriger ausfallen, als sie in der Realität sind. Bei einer höheren Auflösung der Übergangswahrscheinlichkeitsmatrix des Elektrolichtbogenofens könnte man dieser sehr geringen Ungenauigkeit zu Lasten der Rechenzeit auch entgegenwirken.

Um nun eine konkrete Aussage über die Genauigkeit des Modells treffen zu können, kann man den Energieverbrauch, der die Fläche unter der Dauerlinie darstellt, vergleichen. Dabei muss beachtet werden, dass die Abweichung des Energieverbrauchs hauptsächlich von der Anzahl an produzierten Chargen abhängt und weniger vom chargierten Schrottgewicht, obwohl die Auswertung gezeigt hat, dass dieses für das untersuchte Stahlwerk starken Schwankungen unterliegt. Vergleicht man nun den realen mit dem synthetischen Energieverbrauch, zeigt sich, dass der durchschnittliche absolute Energieverbrauch des synthetischen Modells des EAFs für drei simulierte Wochen lediglich um 0,4 Prozent höher ist als der Energieverbrauch, der bei den Messungen am EAF vor Ort aufgezeichnet wurde. Somit kann gesagt werden, dass die vom Modell erzeugten synthetischen Leistungsdaten für den EAF sehr gut mit den realen Leistungsdaten übereinstimmen und den realen EAF wahrheitsgetreu abbilden.

5.2 Pfannenofen

Ähnlich wie beim EAF wird auch für den Pfannenofen ein graphischer Vergleich dargestellt. In Abbildung 5-3 ist dazu auf der rechten Seite das synthetische Lastprofil der Wirk-, Blind- und Scheinleistung dem realen Lastprofil gegenübergestellt.

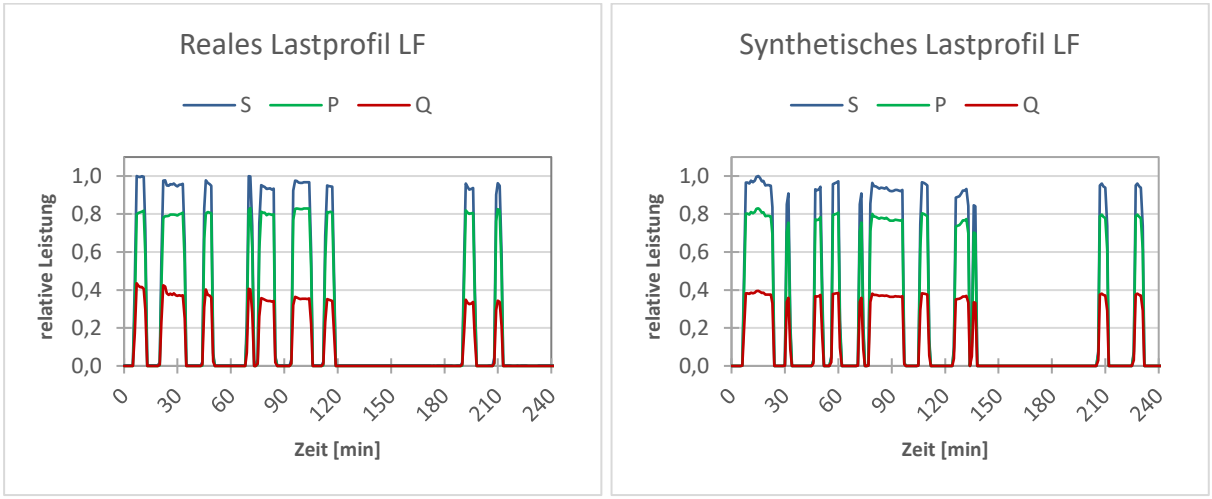

Abbildung 5-3: Vergleich des realen und synthetischen Lastprofils des Pfannenofens

Für diesen Vergleich wurde ein synthetisches Lastprofil, dessen Charakteristik in Hinblick auf die einzelnen Zeitdauern der Prozessschritte Ähnlichkeiten zum realen Lastprofil aufweist, ausgewählt. In der Darstellung des synthetischen Lastprofils sind wie beim gemessenen

Lastprofil die einzelnen Prozessschritte eindeutig zu erkennen. Von 5 bis 140 Minuten dauert der erste Schritt der Behandlung des flüssigen Stahls am Pfannenofen. Dann erfolgt von den Minuten 140 bis 206 der Prozessschritt, bei dem der flüssige Stahl entgast wird. Abschließend erfolgt noch die Vorbereitung und Einstellung der benötigten Temperatur für den Blockguss. Wie in der Realität nimmt auch das synthetische Lastprofil des Pfannenofens für jede Charge eine andere Form an bzw. weist eine andere Charakteristik auf.

Darum wird auch hier die Darstellung der geordneten Dauerlinien zur Evaluierung des Modells verwendet. In Abbildung 5-4 sind die geordneten Dauerlinien der Wirkleistung für das gemessene und synthetische Lastprofil des Pfannenofens dargestellt. Auf der Ordinate erfolgt eine Normierung auf den höchsten gemessen Wert der Wirkleistung und auf der Abszisse erfolgt die Normierung auf die höchste gemessene Betriebszeit des Pfannenofens für eine Charge. Die Dauerlinien stellen somit die durchschnittlichen Wirkleistungswerte von den 20 ausgewählten Referenzchargen und von 20 im Modell erstellten synthetischen Chargen dar.

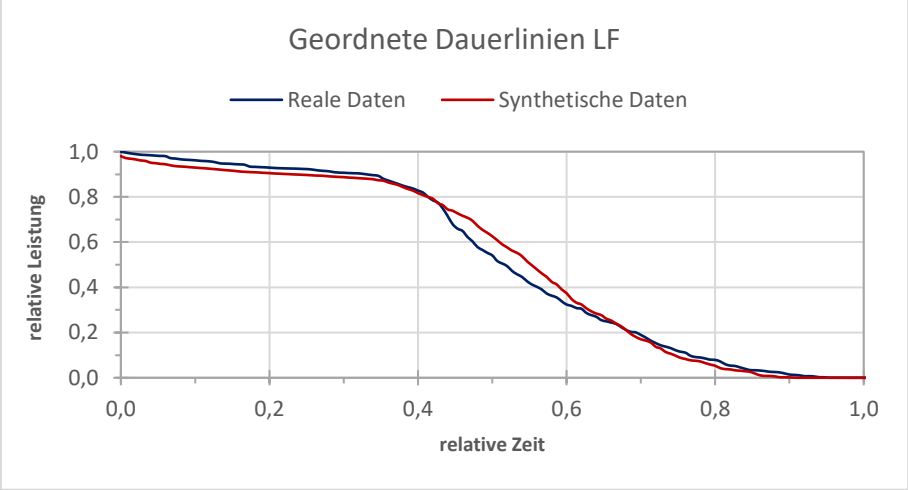

Abbildung 5-4: Vergleich der geordneten Dauerlinien der realen und synthetischen Wirkleistung des Pfannenofens

Das synthetische Lastprofil weist im Bereich der mittleren Leistung und der mittleren Zeitdauern einen Überhang auf. In den Bereichen höherer Leistung und den Bereichen niedriger Leistung stimmen die synthetisch erzeugten Leistungsdaten aber gut mir den gemessenen Leistungsdaten überein. Für den Vergleich der 20 Referenzchargen mit den 20 simulierten Chargen ist hier nur eine sehr kleine Abweichung der jeweiligen Energieverbräuche von 0,2 Prozent vorhanden. Jedoch kann diese Abweichung deutlich höher ausfallen, da die Schwankung des Energieverbrauchs zwischen den einzelnen Chargen teilweise über 100 Prozent beträgt. Während längerer Simulationszeiten, beispielsweise über eine Woche, werden diese starken Schwankungen jedoch sehr gut ausgeglichen,

wodurch das Modell realitätsnahe Ergebnisse liefert. Kurze Simulationszeiten können aber dazu führen, dass der synthetisch ermittelte Energieverbrauch für den Pfannenofen auf den ersten Blick zu hoch oder zu gering erscheinen kann, obwohl das nicht der Fall ist. Das liegt daran, dass der Pfannenofen, wie schon der Elektrolichtbogenofen, im Modell als stochastischer Prozess betrachtet wird.

5.3 Entstaubungsgebläse

In diesem Abschnitt werden sämtliche modellierte Leistungsdaten der Gebläse den realen gemessenen Daten gegenübergestellt. Dazu erfolgt ein graphischer wie auch ein Vergleich mit geordneten Dauerlinien für die Wirkleistung.

5.3.1 Primär- und Sekundärentstaubung des Elektrolichtbogenofens

In Abbildung 5-5 ist der Vergleich des realen und des synthetischen Lastprofils für das Primärgebläse des Elektrolichtbogenofens dargestellt. Dabei sind die Wirk-, Blind- und Scheinleistung auf die Ordinate und die Zeit auf die Abszisse aufgetragen.

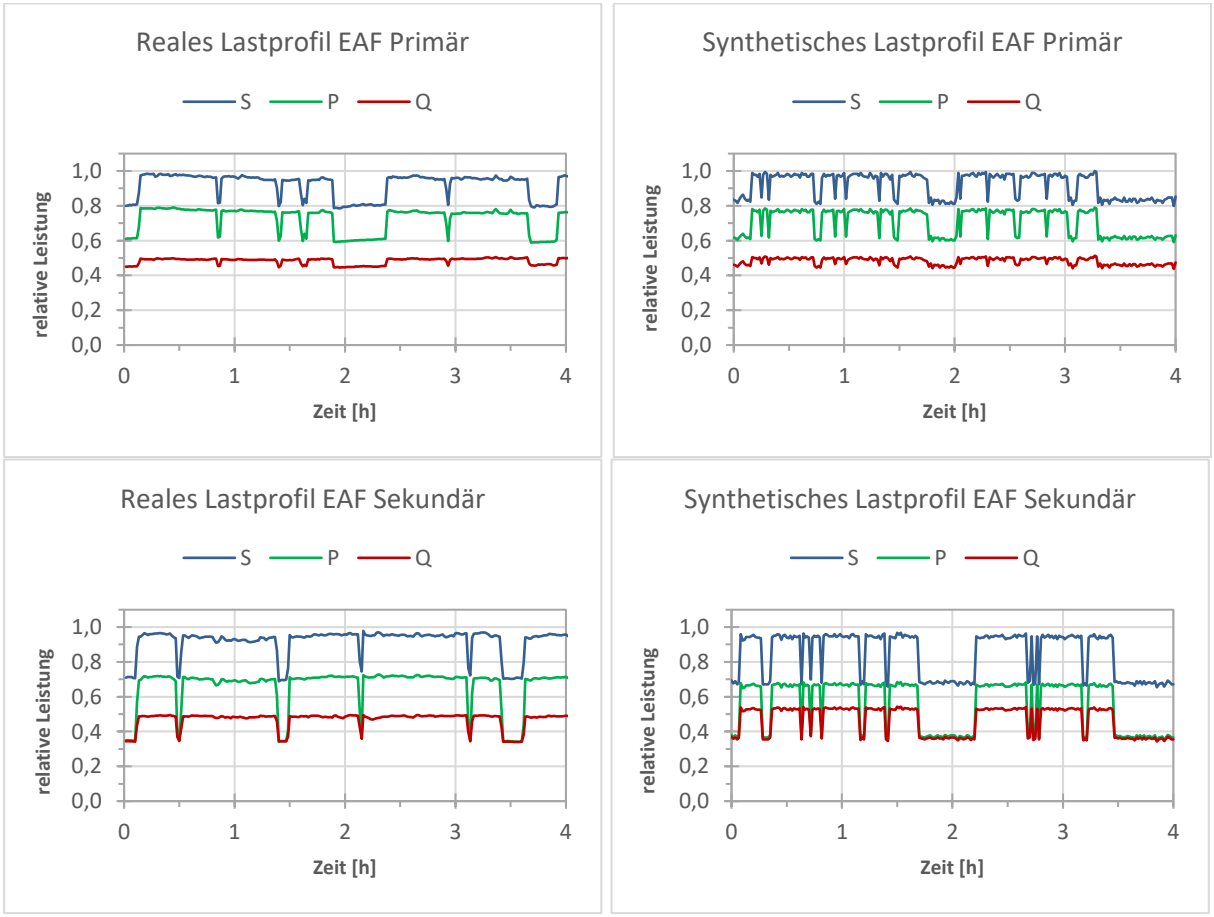

Abbildung 5-5: Vergleich des realen und synthetischen Lastprofils des primären- und sekundären Entstaubungsgebläses des Elektrolichtbogenofens

Um eine Vergleichbarkeit zu ermöglichen, ist auch das synthetische wie das reale Lastprofil für die Dauer von vier Stunden dargestellt. In den synthetischen Lastprofilen sind die Prozessschritte des EAFs für jeweils zwei produzierte Chargen erkennbar. Dadurch, dass im Modell für jeden Zeitschritt ein neuer Wert für die Wirkleistung aus den hinterlegten Häufigkeitsverteilungen zugewiesen wird, sind die synthetischen ermittelten Leistungsdaten der primären sowie sekundären Entstaubungsgebläse des EAFs auf den jeweiligen Leistungsniveaus nicht so glatt wie die Leistungsdaten der realen Lastprofile. Hinsichtlich der Genauigkeit des Modells spielt dieses Verhalten keine Rolle.

Um die synthetischen Lastprofile hinsichtlich des Energieverbrauchs zu vergleichen, wurde die Darstellung über geordnete Dauerlinien gewählt. Dafür werden die Wirkleistungen nach dem Betrag geordnet und auf der Ordinate aufgetragen. Auf der Abszisse wurde für das Primär- und Sekundärgebläse eine jeweils aussagekräftige Betriebsdauer gewählt.

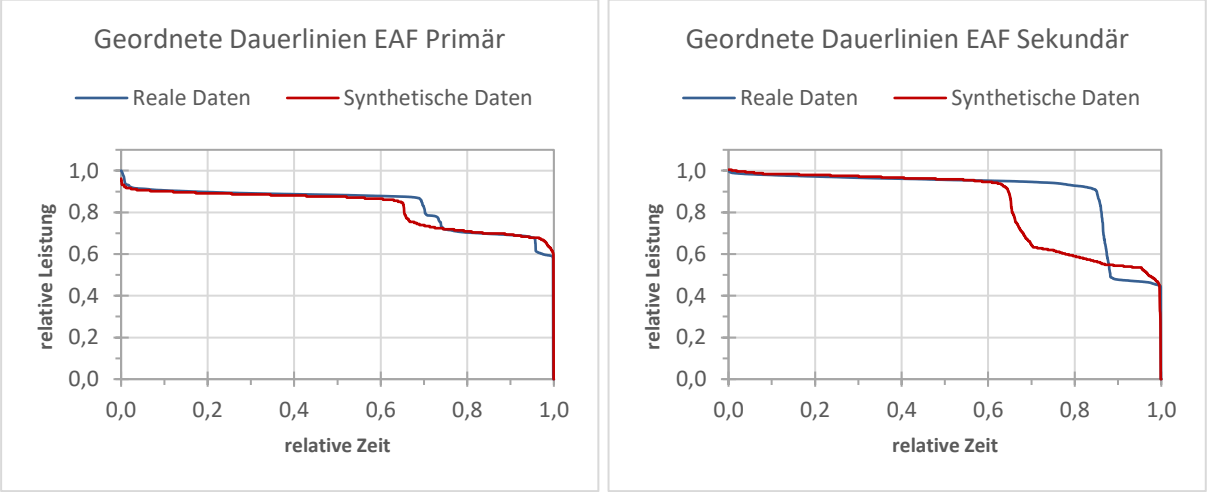

Abbildung 5-6: Vergleich der geordneten Dauerlinien der realen und synthetischen Wirkleistung des Primär- und Sekundärgebläses des Elektrolichtbogenofens

Bei Betrachtung der Dauerlinien für das primäre Entstaubungsgebläse des EAFs erkennt man, dass die realen und synthetisch ermittelten Leistungswerte gut übereinstimmen. Die absolute Abweichung des Energieverbrauchs, der die darunterliegende Fläche darstellt, beträgt dabei 1,1 Prozent.

Betrachtet man nun die geordneten Dauerlinien für das Sekundärgebläse, lässt sich eine starke Abweichung zwischen den realen und errechneten Leistungsdaten erkennen. Die Abweichung des Energieverbrauchs beträgt dabei 7,3 Prozent. Im Modell wurde das primäre- und sekundäre Gebläse so programmiert, dass sich das jeweilige Gebläse nur in Volllast betrieben wird, wenn auch der Elektrolichtbogenofen in Betrieb ist. Für das primäre Entstaubungsgebläse passt diese Kopplung zwischen EAF und Gebläse sehr gut, jedoch führt es beim sekundären Entstaubungsgebläse dazu, dass das Gebläse im Modell kürzere Zeit im

Volllastbereich betrieben wird. Dazu muss gesagt werden, dass dieser Fehler, der hier am sekundären Entstaubungsgebläse auftritt, nur sehr marginale Auswirkungen auf das Gesamtergebnis des Modells hat, da die Gebläse im Vergleich zu den Pfannenöfen oder den Elektrolichtbogenofen nur einen sehr geringen Energieverbrauch aufweisen.

5.3.2 Entstaubungsgebläse Halle 9

Das synthetische Lastprofil des Entstaubungsgebläses der Halle 9 ist in Abbildung 5-7 auf der rechten Seite dargestellt. Dabei ist die Wirk-, Blind- und Scheinleistung in Relation zum Maximalwert der Scheinleistung auf der Ordinate aufgetragen. Als Abbildungsdauer wurde wie schon für das reale Lastprofile zwölf Stunden gewählt. Das Gebläse der Halle 9 wird während des Blockgusses in Volllast betrieben.

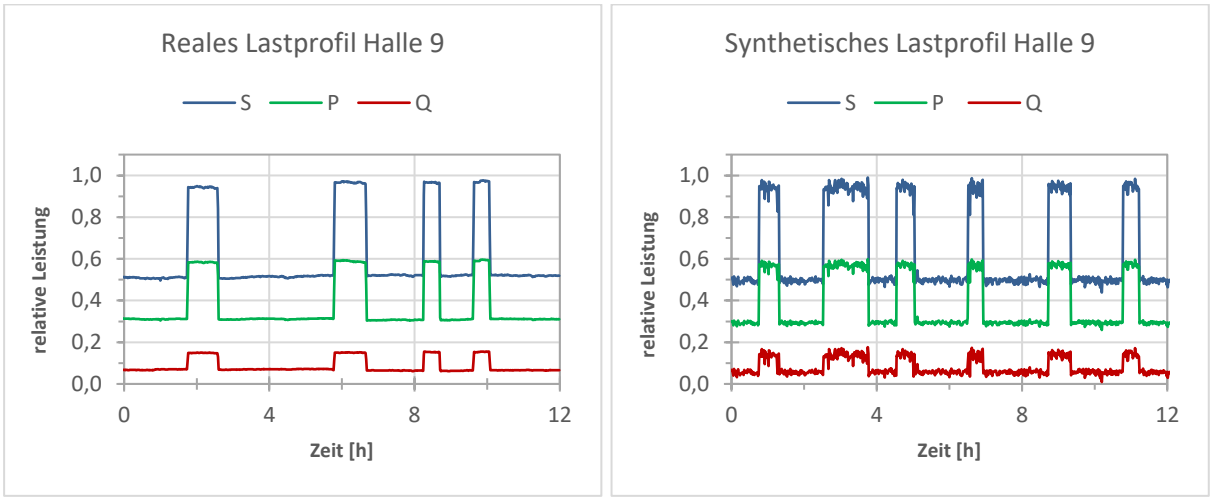

Abbildung 5-7: Vergleich des realen und synthetischen Lastprofils des Entstaubungsgebläses der Halle 9

Im dargestellten synthetischen Lastprofil ist wie im realen Lastprofil der Unterschied zwischen Teillast und Volllast deutlich zu erkennen. Weil im Modell für jeden Zeitschritt ein Leistungswert berechnet wird, sind die Leistungswerte des synthetischen Lastprofils dementsprechend verrauscht. Klar zu erkennen ist jedoch, dass die unterschiedlichen Niveaus der Leistungen mit den realen Leistungen sehr gut übereinstimmen. Die Schwankungen im Volllastbetrieb entstehen durch die verwendeten Häufigkeiten, die aus den realen Daten ermittelt wurden. Bei Simulation mehrerer Chargen bzw. bei Simulation über einen längeren Zeitraum werden diese geglättet und nehmen dabei keinen Einfluss auf das Ergebnis des Gesamtmodells. Für die Breite der Vollastbereiche, also der Zeitdauer, konnte keine Häufigkeitsverteilung ermittelt werden und somit werden diese Zeitdauern zufällig aus einem Intervall gezogen. Das kann dazu führen, dass das synthetische Lastprofil einen höheren Volllastbereich aufweist, als das in Realität der Fall ist. Dieser Effekt ist jedoch

nur bei sehr kurzen Simulationszeiten von Bedeutung und auch dann im Hinblick des Gesamtenergieverbrauchs vernachlässigbar.

In Abbildung 5-8 sind die geordneten Dauerlinie der Wirkleistung für das Entstaubungsgebläse der Halle 9 dargestellt. Wie gerade erwähnt, tritt im Volllastbereich ein sehr geringer höherer Energieverbrauch im Modell auf. Dieser ist im ersten Fünftel des abgebildeten Zeitbereichs ersichtlich.

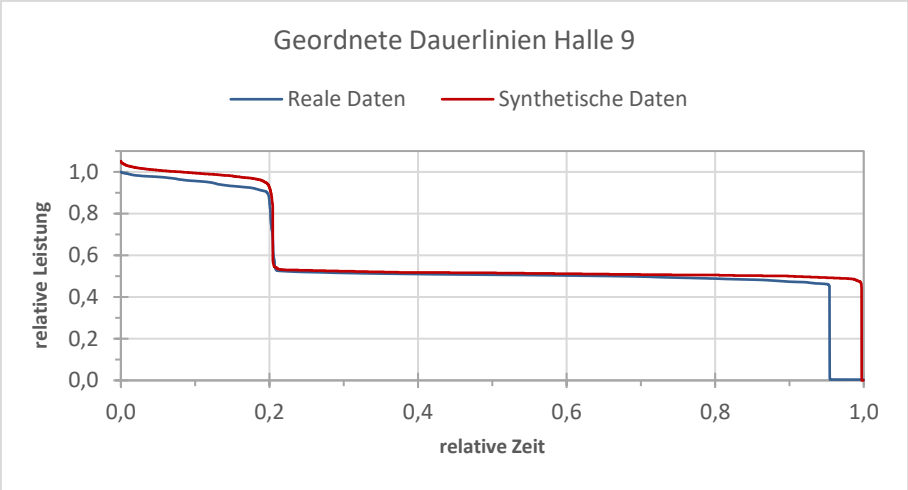

Abbildung 5-8: Vergleich der geordneten Dauerlinien der realen und synthetischen Wirkleistung des Entstaubungsgebläses der Halle 9

Im mittleren Zeitbereich stimmen die synthetischen Wirkleistungsdaten mit den realen Wirkleistungsdaten sehr gut überein. Der Fehler zwischen 95 und 100 Prozent der relativen Zeit kommt daher, dass es nicht genau bekannt ist, wie lange das Entstaubungsgebläse der Halle 9 in Betrieb ist. Dieser Wert der Betriebszeit ändert sich auch für jede Produktionswoche. Unter Berücksichtigung, dass das modellierte Gebläse länger in Betrieb war, weicht der absolute Energieverbrauch um 6,7 Prozent ab. Betrachtet man nur den Energieverbrauch zwischen 0 und 95 Prozent der relativen Zeit, dann beträgt die absolute Abweichung des Energieverbrauchs nur mehr 3,7 Prozent, was in diesem Fall ein sehr guter Wert ist, da die kumulierte Dauer des Volllastbereichs auch deutlich höher ausfallen kann und somit die Abweichung größer wäre.

5.3.3 Entstaubungsgebläse Halle 1

Das synthetische Lastprofil des Entstaubungsgebläses der Halle 1 wird wie das der Halle 9 für einen Betrachtungszeitraum von zwölf Stunden dargestellt. In Abbildung 5-9 wird dafür die Wirk-, Blind- und Scheinleistung bezogen auf den Maximalwert der Scheinleistung auf die Ordinate aufgetragen. Das synthetische Lastprofil des Entstaubungsgebläses der Halle 1 weist durch die Zuweisung der Leistung für jeden Zeitschritt ein starkes Rauschen auf. Für

die Genauigkeit des Modells spielt das keine Rolle, da die mittleren Niveaus der Wirk-, Blind- und Scheinleistung mit den gemessenen realen Leistungen übereinstimmen.

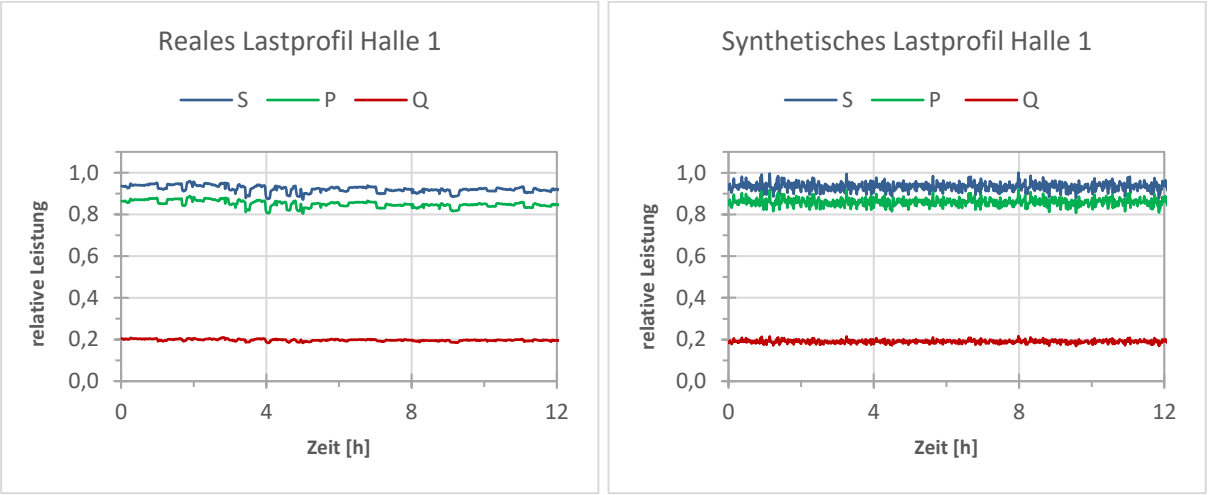

Abbildung 5-9: Vergleich des realen und synthetischen Lastprofils des Entstaubungsgebläses der Halle 1

Gegenteilig der geordneten Dauerlinie der synthetischen Wirkleistung der Halle 9 ist hier, in Abbildung 5-10 dargestellt, ab ca. 92 Prozent der dargestellten Zeitdauer ein Überhang der realen Dauerlinie gegenüber der Dauerlinie der synthetischen Wirkleistung vorhanden. Das bedeutet, dass das Entstaubungsgebläse in der Realität länger in Betrieb war als in der Simulation. Dieser Wert der Betriebsdauer ändert sich jedoch wöchentlich.

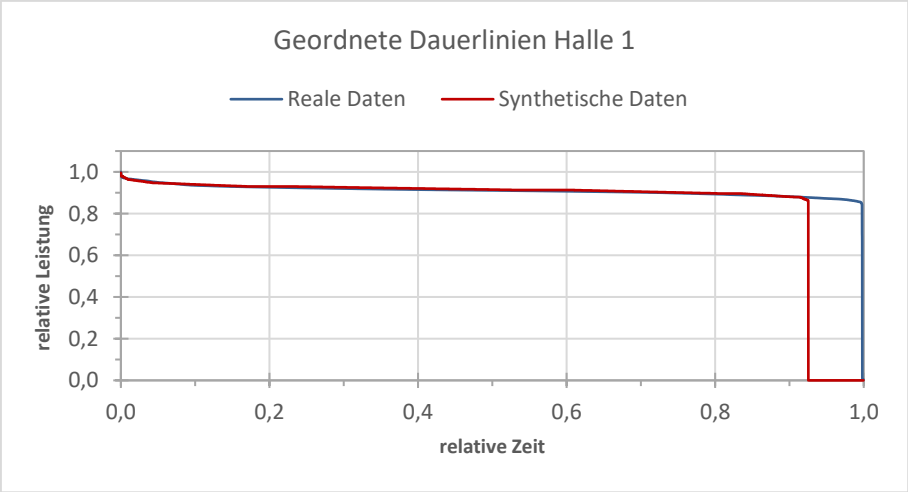

Abbildung 5-10: Vergleich der geordneten Dauerlinien der realen und synthetischen Wirkleistung des Entstaubungsgebläses der Halle 1

Vergleicht man die jeweiligen Energieverbräuche, so weist das Modell eine absolute Abweichung von 6,6 Prozent auf. Vergleicht man die beiden Energieverbräuche im Intervall zwischen 0 und 92 Prozent, beträgt die Abweichung zwischen dem realen und dem synthetisch ermittelten Energieverbrauch lediglich 0,4 Prozent. Da sich die Produktionszeit

und somit die Betriebsdauer des Gebläses wöchentlich ändert, wurde im Modell ein mittlerer Wert für die Einschaltdauer des Entstaubungsgebläses vorgegeben.

5.4 Energiesystemmodell

In diesem Abschnitt werden die Analyse und die Bewertung des vollständigen Energiesystemmodells durchgeführt. Um die Ergebnisse des Energiesystemmodells vergleichen zu können, werden aus dem Datensatz des Gesamtenergieverbrauchs der Breitenfeld Edelstahl AG die Wirkleistungsdaten der Produktionswochen vom Jahr 2019 herangezogen. Daraus folgt, dass mit dem Modell auch jeweils Simulationen für eine Zeitdauer von einer Woche durchgeführt und ausgewertet werden. In Abbildung 5-11 ist dafür ein reales Lastprofil einer typischen Produktionswoche einem synthetischen Lastprofil für dieselbe Simulationsdauer gegenübergestellt. Um das synthetische Lastprofil übersichtlicher zu gestalten, ist hier nur die Wirkleistung dargestellt.

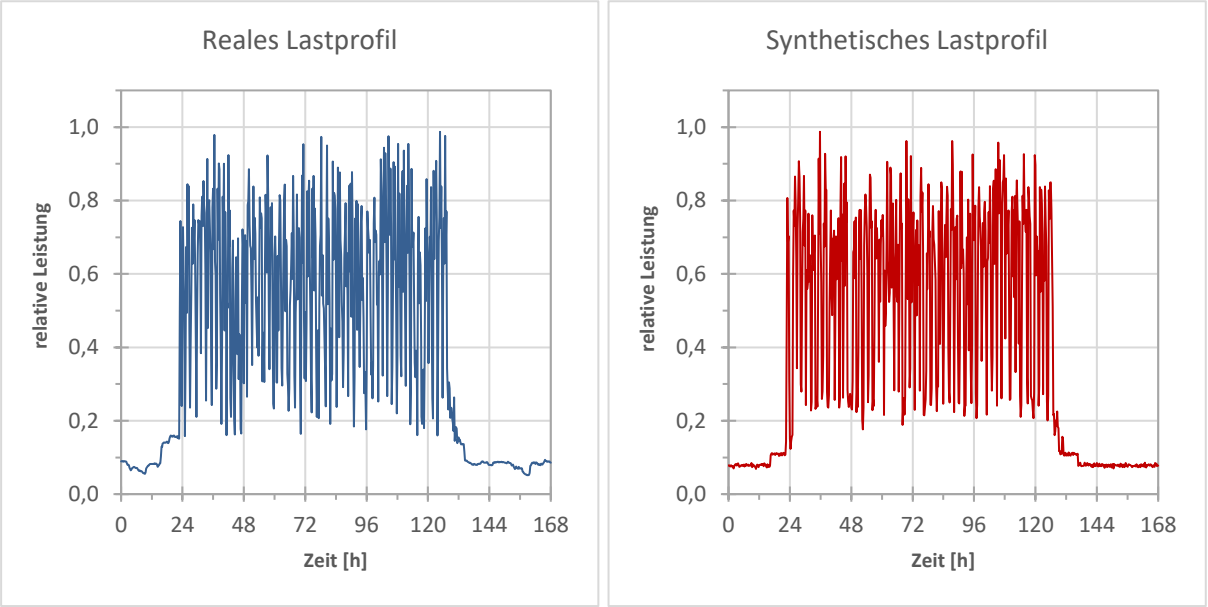

Abbildung 5-11: Vergleich des realen und synthetischen Lastprofils der Wirkleistung für das gesamte Energiesystemmodell

Die relativen Wirkleistungswerte sind auf den maximal aufgetretenen Wert der Wirkleistung des Jahres 2019 bezogen. Für die weitere Auswertung muss gesagt werden, dass der Datensatz des Gesamtenergieverbrauchs des Standorts nur die Wirkleistung beinhaltet und somit keine Möglichkeit des Vergleichs der Blind- und Scheinleistungsdaten besteht. Das reale Lastprofil weist eine Auflösung von 15 Minuten auf. Aus diesem Grund wurde auch das synthetische Lastprofil mit einer Auflösung von 15 Minuten für den graphischen Vergleich herangezogen. Würde man für das synthetische Lastprofil die Minutenwerte des Modells heranziehen, wären einzelne Spitzenleistungen deutlich höher. In diesem Fall werden diese

kurzen Lastspitzen, die vor allem durch den Elektrolichtbogenofen hervorgerufen werden und auch in der Realität auftreten, durch die niedrigere Auflösung von 15 Minuten geglättet.

Die beiden in Abbildung 5-11 dargestellten Lastprofile kann man grob in zwei Bereiche einteilen. Der erste Bereich stellt die Vorbereitung auf die kommende Produktionswoche bzw. das Ende der Produktionswoche dar. Dieser Bereich ist in beiden Darstellungen zwischen 0 und ca. 24 Stunden bzw. 130 bis 168 Stunden erkennbar. Der zweite Bereich kann als Produktionsbereich, also der Bereich zwischen dem Beginn des Aufschmelzens des ersten Korbs Schrott am Elektrolichtbogenofen bis hin zum Blockguss der letzten Wochencharge, definiert werden. Dieser Bereich ist in beiden Darstellungen zwischen ca. 24 und 130 Stunden erkennbar.

Vergleicht man nun den Produktionsbereich der beiden Lastprofile, erkennt man, dass die Leistungsniveaus der Spitzenlast sowie der minimalen Last des synthetischen Lastprofils in diesem Bereich mit den realen Leistungsniveaus sehr gut übereinstimmen. Auch das Abfertigen der letzten Chargen am Pfannenofen ist am Ende des Produktionsbereichs des synthetischen Lastprofils zu erkennen und stimmt mit dem realen Lastprofil überein. Bei Betrachtung des Bereichs außerhalb der Produktion, kann man im synthetischen Profil den Start und Nachlauf der Gebläse erkennen. Des Weiteren wirkt der Lastgang des synthetischen im Vergleich mit dem realen sehr gleichförmig, was daran liegt, dass im Modell für jeden Zeitschritt ein Wert für die Last der übrigen Verbraucher zugewiesen wird.

Um eine bessere Vergleichbarkeit zwischen dem realen Gesamtlastprofil und dem synthetischen Lastprofil des Energiesystemmodells zu erhalten, werden in Abbildung 5-12 die geordneten Dauerlinien der Wirkleistung für einen Betrachtungszeitraum von einer Produktionswoche für das gesamten Energiesystemmodell dargestellt. Aus dem Datensatz des Gesamtenergieverbrauchs wurden dafür Dauerlinien für die minimal und maximal auftretenden Energieverbräuche einer Produktionswoche ermittelt und dargestellt. Des Weiteren wurde aus allen Daten von 2019 eine mittlere Dauerlinie gebildet. Für die Bildung der synthetischen Dauerlinien wurden dieselbe Anzahl an Daten, die der reale Datensatz enthält, erstellt. Diese synthetisch erstellten Daten weisen wie der reale Datensatz eine Auflösung von 15 Minuten auf. Für die synthetischen Daten wurde wie für die realen Dauerlinien der minimale-, maximale und mittlere Energieverbrauch bestimmt. Diese werden wie die realen Dauerlinien auch als geordnete Dauerlinien in der Abbildung 5-12 dargestellt. Der reale minimale und maximale Energieverbrauch kann somit als unterer bzw. oberer Grenzwert herangezogen und der Bereich zwischen den beiden geordneten Dauerlinien als Toleranz angesehen werden.

Für die einfachere Beschreibung und Auswertung der Dauerlinien wird die Abbildung in folgenden drei Bereiche eingeteilt:

- Hoher Leistungsbereich: 50 bis 100 Prozent der relativen Leistung
- Mittlerer Leistungsbereich: 10 bis 50 Prozent der relativen Leistung
- Niedriger Leistungsbereich: 0 bis 10 Prozent der relativen Leistung

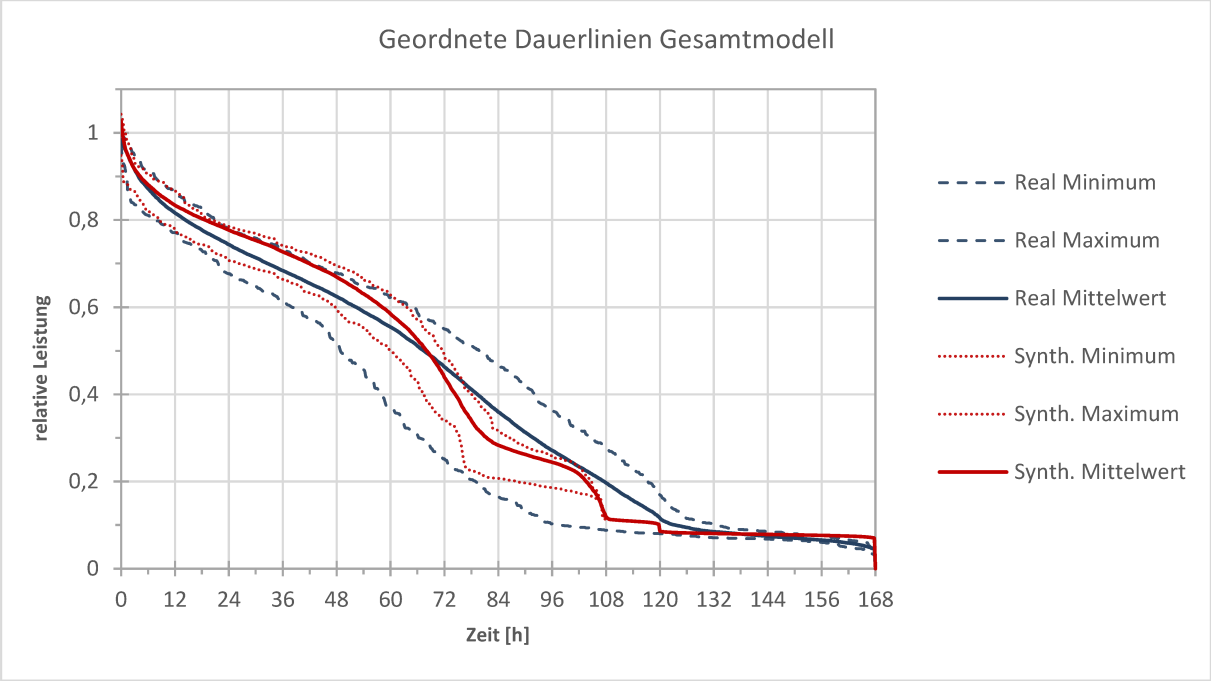

Abbildung 5-12: Vergleich der geordneten Dauerlinien der realen und synthetischen Wirkleistung des Gesamtmodells

Im hohen Leistungsbereich ist der mittlere Leistungsbezug des Modells höher als die mittlere reale Leistung. Dieser höhere Energieverbrauch wird rein vom Elektrolichtbogenofen verursacht und auf die Problematik wurde oben bei der Beschreibung des EAFs bereits eingegangen. Der maximale synthetische Energieverbrauch überschreitet im Bereich der hohen relativen Leistungen den maximalen realen Energieverbrauch. Das kann daran liegen, dass bei der Simulation durch die zufällige Zuweisung von kurzen Tap-to-tap-Zeiten ein bis zwei Chargen mehr als in einer typischen realen Produktionswoche erstellt wurden und dadurch ein höherer Leistungsbezug am Elektrolichtbogenofen aufgetreten ist.

Im mittleren Leistungsbereich weist die mittlere synthetische Dauerlinie einen deutlich geringeren Energieverbrauch als die mittlere reale Dauerlinie auf. Das kann vor allem daran liegen, dass hier andere nicht erfasste elektrische Verbraucher des Standorts im Modell nicht berücksichtigt werden konnten, ein Beispiel dafür ist die Elektroschlacke-Umschmelzanlage (ESU). Diese nicht erfassten Aggregate können zu dem in der Realität höher ausfallenden Energieverbrauch beitragen. Ein Faktor kann auch die Leistung der anderen drei

Pfannenöfen, die nicht Teil der Datenerhebung waren, darstellen. Die Leistung dieser Pfannenöfen kann im Modell jedoch jederzeit im Eingabefenster eingestellt werden.

Im niedrigen Leistungsbereich treten Abweichungen vor allem durch den unbekannten Energieverbrauch der übrigen Verbraucher am Standort auf. Auch die Einschaltdauer bzw. Betriebszeit der unterschiedlichen Gebläse kann hier zu Abweichungen führen. Bei Betrachtung der Abbildung der geordneten Dauerlinien kann aber gesagt werden, dass diese Abweichungen im niedrigen Leistungsbereich nur äußerst gering sind. Vergleicht man nun die beiden Mittelwerte des realen und synthetischen Leistungsbezugs, beträgt die absolute Abweichung lediglich 0,2 Prozent.

Um die Ergebnisse und die Genauigkeit des erstellten Energiesystemmodells einordnen zu können, ist in Abbildung 5-13 eine Häufigkeitsverteilung für die realen wöchentlichen Energieverbräuche vom Jahr 2019 sowie für die über das Modell ermittelten wöchentlichen Energieverbräuche dargestellt. Die ermittelten Energieverbräuche wurden dabei auf den Mittelwert aller wöchentlichen realen Energieverbräuche vom Jahr 2019 normiert.

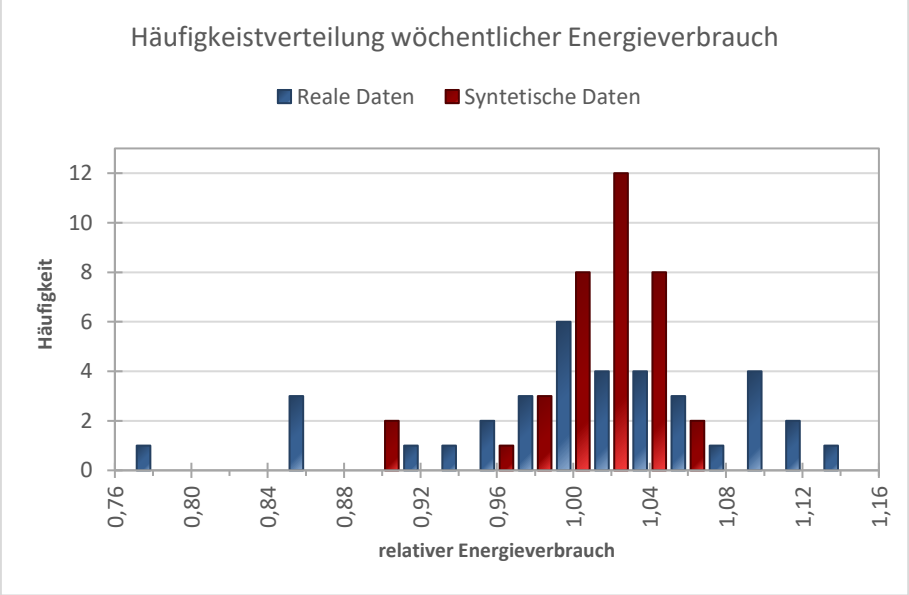

Abbildung 5-13: Vergleich des realen und synthetischen wöchentlichen Energieverbrauchs

Die synthetisch ermittelten, wöchentlichen Energieverbräuche des Energiesystemmodell weisen gegenüber den realen Energieverbräuchen eine deutlich geringere Streuung um den Mittelwert der realen Wochenverbräuche auf. Das liegt vor allem daran, dass im Modell ungeplante, zufällig auftretende Produktionsunterbrechungen nicht berücksichtigt werden. Des Weiteren wurden die modellierten Aggregate nur für ein paar Wochen und nicht über den gesamten Zeitraum der vorhandenen realen Daten vermessen.

Um nun das Ergebnis des Energiesystemmodells in absoluten Zahlen darzustellen, erfolgt eine Auswertung der Verteilung des Energieverbrauchs nach den einzelnen Komponenten,

die in Abbildung 5-14 dargestellt ist. Da jede Produktionswoche in der Realität starke Schwankungen des Energieverbrauchs aufweist, wurde zum Vergleich der Verteilung des Energieverbrauchs eine Woche vom Jahr 2019 gewählt, für die zur selben Zeit Leistungsdaten am Elektrolichtbogenofen und am Pfannenofen erhoben wurden. Um einen korrekten Vergleich zwischen der realen Verteilung des Energieverbrauchs und der synthetischen Verteilung des Energieverbrauchs zu ermöglichen, wurde eine simulierte Woche mit einem identischen Energieverbrauch, herangezogen.

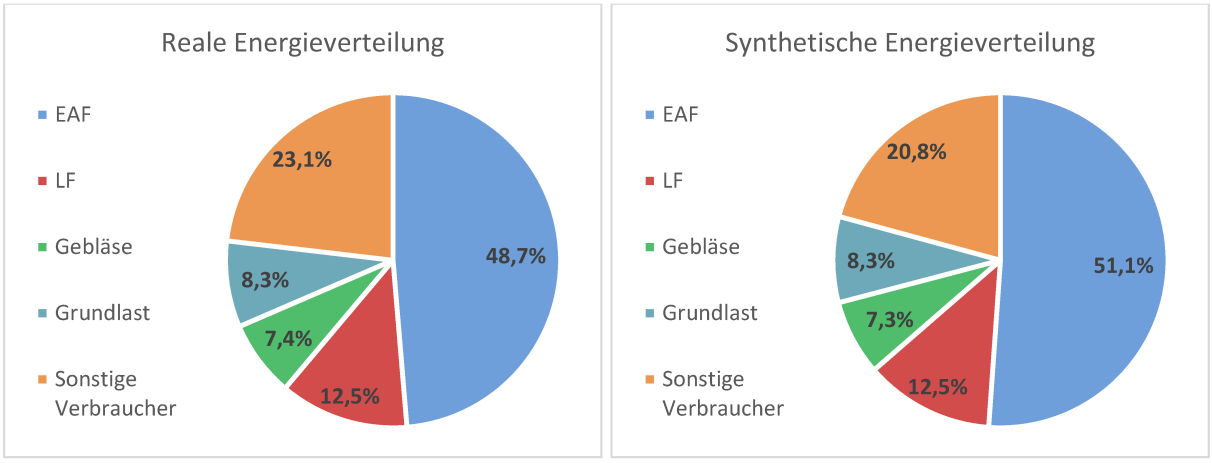

Abbildung 5-14: Vergleich der realen und synthetischen Verteilung des Energieverbrauchs

Betrachtet man die in Abbildung 5-14 dargestellten Verteilungen, lässt sich deutlich erkennen, dass der modellierte EAF bei gleichem Gesamtenergieverbrauch der synthetisch modellierten und realen Produktionswoche einen höheren Energieverbrauch aufweist. Dieses Problem lässt sich einerseits auf die zufällig aus der Häufigkeitsverteilung bestimmten Eingangsgrößen, die Tap-to-tap-Zeit und das Schrottgewicht, zurückführen. Andererseits wird die für das Aufschmelzen einer Charge benötigte Energie im Modell für jede Charge mit denselben konstanten spezifischen Energieverbrauch je Tonne Schrott ermittelt. In der Realität schwankt der spezifische Energieverbrauch aber je nach Zusammensetzung des Schrotts und darum kann es auch zu dieser Abweichung führen.

Die Verteilung der synthetischen Energieverbräuche der Pfannenöfen, der Gebläse und der Grundlast stimmen mit den realen Daten sehr gut überein. Jedoch muss hier gesagt werden, dass im Betrachtungszeitraum lediglich die Leistungen an einen Pfannenofen gemessen wurden. Da von den anderen drei Pfannenöfen die jeweiligen Leistungswerte nicht bekannt sind, können diese hier als Unsicherheitsfaktor gesehen werden. Das Gleiche betrifft die Verteilung der Kleinverbraucher, da hier einige Annahmen gemacht werden mussten, um diese zu ermitteln. Eine Messung dieser Kleinverbraucher kommt aber Aufgrund des damit verbundenen Aufwands nicht in Frage.

6 ZUSAMMENFASSUNG UND AUSBLICK

Das in dieser Arbeit entwickelte Energiesystemmodell eignet sich zur Generierung repräsentativer Lastprofile für einzelne Verbraucher sowie für das gesamte Elektrostahlwerk. Um das Modell so flexibel und anwenderfreundlich wie möglich zu gestalten, wurden die veränderbaren Parameter des Modells in die graphische Nutzeroberfläche integriert. Dabei wurden für gewisse Parameter die realen Verteilungen so hinterlegt, dass sich diese bei einer Änderung des jeweiligen Parameters anpassen. Dadurch wird sichergestellt, dass das Energiesystemmodell kein statisches, sondern ein dynamisches Verhalten aufweist.

Der Elektrolichtbogenofen stellt den größten einzelnen elektrischen Verbraucher des Modells dar und hat somit den anteilsmäßig größten Einfluss auf das Ergebnis des Energiesystemmodells. Der Elektrolichtbogenofenprozess ist durch drei Prozessphasen, in denen der stark schwankende Leistungsbezug erfolgt, charakterisiert. Diese Charakteristik entspricht einem stochastischen Prozess und kann mittels Markov-Ketten modelliert werden. Die Eingangsgrößen für das Modell sind die Schrottmenge, die Tap-to-tap-Zeit, der Sauerstoffbedarf und der spezifische Energieverbrauch, der zum Aufschmelzen des Schrotts benötigt wird. Das synthetische Lastprofil sowie der Energieverbrauch des Elektrolichtbogens stimmen mit den gemessenen Daten sehr gut überein und Ungenauigkeiten treten lediglich im Bereich mittlerer Leistungen auf.

Der Pfannenofen stellt nach dem Elektrolichtbogenofen den zweitgrößten elektrischen Verbraucher des Modells dar. Die Charakteristik des Pfannenofens lässt sich durch zwei Prozessphasen, in denen der Energieeintrag über die Graphitelektroden in die Stahlschmelze erfolgt, beschreiben. Zwischen den zwei Prozessphasen erfolgt die Vakuumbehandlung. Der Energieeintrag erfolgt mit einer konstanten Leistung für kurze Zeitintervalle und stellt einen stochastischen Prozess dar, der auch mittels Markov-Ketten modelliert wird. Als Prozessparameter werden für die Erstellung des synthetischen Lastprofils die Prozesszeiten der einzelnen Prozessphasen sowie die Nennleistung und der Leistungsfaktor des Pfannenofens vorgegeben. Das synthetische Lastprofil des Pfannenofens stimmt mit dem realen Lastprofil gut überein und der Energieverbrauch des Modells befindet sich in einem realistischen Rahmen. In der Realität ist der Energieverbrauch jedoch von den Prozesszeiten, der Stahlmenge, der Stahlsorte und der Menge an Legierungszusätzen abhängig.

Um das Energiesystemmodell genauer zu gestalten, könnten die einzelnen Modelle des Elektrolichtbogenofens, des Pfannenofens und die Last der übrigen Verbraucher verbessert bzw. weiterentwickelt werden. Beim Modell des Elektrolichtbogenofens könnte man die Übergangswahrscheinlichkeitsmatrizen mit den bis jetzt gewonnenen Erkenntnissen

überarbeiten, um eine bessere Übereinstimmung speziell im mittleren Leistungsbereich zu erreichen. Die Prozessparameter des Elektrolichtbogenofens, die mit den realen Verteilungen hinterlegt sind, müssen nicht überarbeitet werden bzw. haben nur einen marginalen Einfluss auf das Ergebnis. Jedoch würde die Integration einer realen Verteilung anstelle des konstanten Werts für den spezifischen Energieverbrauch, der zum Aufschmelzen des Schrotts benötigt wird, eine Verbesserung der Genauigkeit des Modells schaffen. Für das Modell des Pfannenofens wäre es interessant, wenn sich die Möglichkeit zur Integration von den zuvor beschriebenen Prozessdaten, die Einfluss auf den Energieverbrauch haben, bieten würde. Weiters könnten noch Verteilungen für die unterschiedlichen Prozesszeiten hinterlegt werden, da derzeit diese Zeitparameter aus einem vorgegebenen Intervall zufällig gezogen und festgelegt werden. Im Zuge dieser Arbeit wurde festgestellt, dass die Modelle der Gebläse vollständig entwickelt sind. Abweichungen treten lediglich wegen der Unterschiede der wöchentlichen Betriebsdauern auf. Diese Betriebsdauern können aber jederzeit geändert werden und somit stellen die geringen Abweichungen kein Problem dar. Die wichtigste Weiterentwicklung des Modells, welche darauf abzielt, um den gesamten Standort und nicht nur das Stahlwerk realitätsgetreu nachzubilden, wäre somit eine Weiterentwicklung der Modellierung der Last der übrigen Verbraucher. Den nächsten Modellierungsschritt bilden die Prozessgase und die Erdgasverbraucher.

Zusammenfassend kann festgehalten werden, dass mit dem Energiesystemmodell reale Lastszenarien simuliert und ausgewertet werden können. Entscheidend für die Genauigkeit und Vergleichbarkeit der Ergebnisse ist dabei, dass im Modell längere Simulationszeiträume, beispielsweise von zwei Tagen und länger, gewählt werden. Das liegt vor allem an der starken Schwankung des Energieverbrauchs je Charge, der auch in der Realität auftritt. Durch längere Simulationszeiten glätten sich diese Schwankungen. Das Energiesystemmodell kann aber nicht als Prognose beispielsweise für eine Produktionswoche dienen, da unerwartete Produktionsstopps und Wartungsarbeiten nicht im Modell berücksichtigt werden. Für den industriellen Chargenbetrieb ist es generell schwierig genaue Prognosen zu treffen, da diese Produktionsweise einen sehr flexiblen und diskontinuierlichen Zeitablauf aufweist.

Zukünftig kann das Energiesystemmodell zur Identifikation und Bewertung von Optimierungspotentialen genutzt werden. Bei diesen Potentialen handelt es sich konkret um Flexibilitäten zur Integration von Energie aus erneuerbaren Energiequellen und der Ableitung von Maßnahmen des Demand Side Managements. Außerdem kann das Modell zur Berechnung und Vorhersage der Belastung des vorgelagerten Netzes herangezogen werden und dadurch bei der Auslegung von Kompensationsanlagen helfen.

7 Literaturverzeichnis

[1] DAS EUROPÄISCHE PARLAMENT UND DER RAT DER EUROPÄISCHEN UNION: *Verordnung (EU) 2018/1999 des Europäischen Parlaments und des Rates über das Governance-System für die Energieunion und den Klimaschutz*, Amtsblatt L 328/2018

[2] BUNDESMINISTERIUM FÜR KLIMASCHUTZ, UMWELT, ENERGIE, MOBILITÄT, INNOVATION UND TECHNOLOGIE: *Integrierter nationaler Energie- und Klimaplan für Österreich* 18.12.2019

[3] BREITENFELD EDELSTAHL AG: *Infrastruktur Stahlwerk*. URL https://breitenfeld.at/infrastruktur-stahlwerk/. – Aktualisierungsdatum: 2020-12-07

[4] THE BOSTON CONSULTING GROUP: *Steel's Contribution to a Low-Carbon Europe 2050* : *Technical and Economic Analysis of the Sector's CO2 Abatement Potential*. 2014

[5] EUROFER; Axel Eggert (Mitarb.): *European Steel in Figures*. 2019

[6] PETER PULM, Harald Raupenstrauch: *Roadmap Industrie* : *Energieeffizienz in der Eisen- und Stahlindustrie*. 2014

[7] Future of Process Metallurgy. In: *Treatise on Process Metallurgy* : Elsevier, 2014, S. 1563–1726

[8] BATTLE, Thomas ; SRIVASTAVA, Urvashi ; KOPFLE, John ; HUNTER, Robert ; MCCLELLAND, James: The Direct Reduction of Iron. In: *Treatise on Process Metallurgy* : Elsevier, 2014, S. 89–176

[9] ZULFIADI ZULHAN: *EAF-Based Flat-Steel Production Applying Secondary Metallurgical Processes*. 2006

[10] WORLD STEEL ASSOCIATION: *Steel Statistical Yearbook 2018*. 2018

[11] YELLISHETTY, Mohan ; MUDD, Gavin M. ; RANJITH, P. G. ; THARUMARAJAH, A.: *Environmental life-cycle comparisons of steel production and recycling: sustainability issues, problems and prospects*. In: *Environmental Science & Policy* 14 (2011), Nr. 6, S. 650–663

[12] WORLD STEEL ASSOCIATION: *Steel's Contribution to a Low Carbon Future and Climate Resilient Societies* : *Worldsteel Position Paper*. 2017

[13] MADIAS, Jorge: Electric Furnace Steelmaking. In: *Treatise on Process Metallurgy* : Elsevier, 2014, S. 271–300

[14] BREITENFELD EDELSTAHL AG: *Kreislaufwirtschaft & Ressourcenschonung*. URL https://breitenfeld.at/kreislaufwirtschaft-ressourcenschonung/. – Aktualisierungsdatum: 2020-11-25

[15] *Treatise on Process Metallurgy* : Elsevier, 2014

[16] THOMÉ-KOZMIENSKY, Karl J. (Hrsg.): *Aschen, Schlacken, Stäube* : *Aus Abfallverbrennung und Metallurgie*. Neuruppin : TK Thomé-Kozmiensky, 2013

[17] HEINEN, Karl-Heinz (Hrsg.): *Elektrostahl-Erzeugung*. 4., völlig neu bearb. und erw. Aufl. Düsseldorf : Stahleisen, 1997

[18] H. PFEIFER, M. Kirschen: *Thermodynamic analysis of EAF energy efficiency and comparison with a statistical model of electric energy demand*. 2002

[19] REMUS, Rainer: *Best available techniques (BAT) reference document for iron and steel production* : *Industrial emissions directive 2010/75/EU (integrated pollution prevention and control)*. Luxembourg : Publications Office of the European Union, 2013 (Scientific and technical research series 25521)

[20] KIRSCHEN, Marcus ; RISONARTA, Victor ; PFEIFER, Herbert: *Energy efficiency and the influence of gas burners to the energy related carbon dioxide emissions of electric arc furnaces in steel industry*. In: *Energy* 34 (2009), Nr. 9, S. 1065–1072

[21] HOLAPPA, Lauri: Secondary Steelmaking. In: *Treatise on Process Metallurgy* : Elsevier, 2014, S. 301–345

[22] WALDMANN, Karl-Heinz ; STOCKER, Ulrike M.: *Stochastische Modelle* : *Eine anwendungsorientierte Einführung*. Berlin : Springer, 2004 (EMILeA-stat - Medienreihe zur angewandten Statistik)

[23] STEWART, William J.: *Probability, Markov chains, queues, and simulation* : *The mathematical basis of performance modeling*. Princeton, Oxford : Princeton University Press, 2009

[24] PAPULA, Lothar: *Vektoranalysis, Wahrscheinlichkeitsrechnung, Mathematische Statistik, Fehler- und Ausgleichsrechnung*. 7., überarbeitete und erweiterte Auflage. Wiesbaden : Springer Vieweg, 2016 (Lehrbuch / Lothar Papula ; Band 3)

[25] HEDDERICH, Jürgen ; SACHS, Lothar: *Angewandte Statistik* : *Methodensammlung mit R*. 16., überarbeitete und erweiterte Auflage. Berlin, Germany : Springer Spektrum, 2018

[26] TKOTZ, Klaus ; BASTIAN, Peter: *Fachkunde Elektrotechnik*. 24., überarb. und erw. Aufl., Dr. 1. Haan-Gruiten : Verl. Europa-Lehrmittel Nourney Vollmer, 2004 (Europa-Fachbuchreihe für elektrotechnische Berufe)

8 ANHANG

Im Anhang dieser Arbeit sind die für das Modell relevanten Prozessparameter, die Übergangswahrscheinlichkeitsmatrizen und der Code des Energiesystemmodells für die einzelnen Module dargestellt.

In den folgenden Tabellen sind die Prozessparameter, die aus den gemessenen Daten erhoben bzw. von der Breitenfeld Edelstahl AG zur Verfügung gestellt wurden, abgebildet. Dabei wird der Minimal- und Maximalwert sowie teilweise das gewichtete arithmetische Mittel des jeweiligen Parameters angegeben.

In Tabelle 8-1 sind die Prozessparameter des Elektrolichtbogenofens dargestellt. Die Werte der Wirkleistung und des Leistungsfaktors stammen aus den gemessenen Daten und die restlichen Daten dieser Tabelle wurden von der Breitenfeld Edelstahl AG zur Verfügung gestellt.

Tabelle 8-1: Prozessparameter des Elektrolichtbogenofens

Prozessparameter EAF	Einheit	Minimum	Maximum	Gewichteter Mittelwert
Wirkleistung	[MW]	0	40,75	25,23
Leistungsfaktor cosφ	[-]	0,15	0,99	0,81
Chargierte Schrottmenge	[t]	47,0	78,5	65,3
Tap-to-tap-Zeit	[min]	81	329	140
Sauerstoffverbrauch	[Nm^3]	100	2000	1179
Abstichtemperatur	[°C]	1647	1789	1724
Ausbringung	[%]	72,2	105,1	89,0

Sämtliche Prozessparameter des Pfannenofens, die in Tabelle 8-2 abgebildet sind, wurden aus den gemessenen Leistungsdaten ermittelt.

Tabelle 8-2: Prozessparameter des Pfannenofens

Prozessparameter LF	Einheit	Minimum	Maximum	Gewichteter Mittelwert
Wirkleistung	[MW]	0	5,94	4,98
Leistungsfaktor cosφ	[-]	0,02	0,94	0,81
Prozessphase 1	[min]	71	268	185
Entgasen	[min]	54	115	81
Prozessphase 2	[min]	5	75	35
Energieverbrauch pro Charge	[MWh]	4,08	11,55	7,86

Die in Tabelle 8-3 dargestellten Daten, die den zeitlichen Ablauf zwischen den Prozessen der Stahlherstellung beschreiben, wurden teils an den gemessenen Daten ermittelt und teils

angenommen. Diese zeitlichen Abläufe können jedoch jederzeit im Eingabefenster des Modells verändert bzw. angepasst werden.

Tabelle 8-3: Parameter der ablaufspezifischen Prozesszeiten

Prozessparameter	Einheit	Minimum	Maximum
Zeitdauer Abstich bis Behandlung Pfannenofen bei VD	[min]	45	60
Zeitdauer Abstich bis Behandlung Pfannenofen bei VOD	[min]	100	120
Zeitdauer Ende Behandlung Pfannenofen bis Blockguss	[min]	10	20

In Tabelle 8-4 sind sämtliche Prozessparameter der Entstaubungs- und Hallengebläse aufgelistet. Sämtliche in dieser Tabelle aufgelisteten Prozessparameter können im Modell im Eingabefenster verändert werden. Zusätzlich zu den Leistungsdaten ist hier auch für das Entstaubungsgebläse der Halle 9 die Betriebsdauer unter Volllast beim Blockguss angegeben.

Tabelle 8-4: Prozessparameter der Gebläse

Prozessparameter	Einheit	Minimum	Maximum	Gewichteter Mittelwert
EAF Primärgebläse Teillast				
Wirkleistung	[kW]	0	160	138
Leistungsfaktor cosφ	[-]	0,00	0,79	0,78
EAF Primärgebläse Volllast				
Wirkleistung	[kW]	160	214	174
Leistungsfaktor cosφ	[-]	0,28	0,83	0,79
EAF Sekundärgebläse Teillast				
Wirkleistung	[kW]	0	180	120
Leistungsfaktor cosφ	[-]	0,44	0,65	0,51
EAF Sekundärgebläse Volllast				
Wirkleistung	[kW]	180	240	230
Leistungsfaktor cosφ	[-]	0,61	0,76	0,74
Halle 9 Teillast				
Wirkleistung	[kW]	0	400	333
Leistungsfaktor cosφ	[-]	0,12	0,62	0,51
Halle 9 Volllast				
Wirkleistung	[kW]	400	672	631
Leistungsfaktor cosφ	[-]	0,60	0,62	0,61
Betriebsdauer in Volllast	[min]	4	83	41
Halle 1				
Wirkleistung	[kW]	706	1066	969
Leistungsfaktor cosφ	[-]	0,23	0,93	0,88

Abschließend sind in Tabelle 8-5 die Prozessparameter für die Grundlast und die Last der übrigen Verbraucher dargestellt. Für den Wirkleistungswert der Grundlast wurde der niedrigste Viertelstundenwert aus dem Gesamtjahreslastprofil 2019 herangezogen. Die Werte der Leistungsfaktoren wurden für die Grundlast und die Last der übrigen Verbraucher angenommen.

Tabelle 8-5: Prozessparameter der Grundlast und der Last der übrigen Verbraucher

Prozessparameter	Einheit	Minimum	Maximum	Gewichteter Mittelwert
Last der übrigen Verbraucher				
Wirkleistung	[kW]	0	12689	5614
Leistungsfaktor cosφ	[-]	0,95	0,95	0,95
Grundlast				
Wirkleistung	[kW]	1440	1440	1440
Leistungsfaktor cosφ	[-]	0,95	0,95	0,95

Im weiteren Anhang folgen die Übergangswahrscheinlichkeitsmatrizen und der Code der einzelnen Module. Für eine bessere Übersicht ist für den Code, der nach den Übergangswahrscheinlichkeitsmatrizen folgt, ein Verzeichnis angeführt. Um das Modell nachbilden zu können sind nur die relevanten Module angegeben.

Python-Code

Übergangswahrscheinlichkeitsmatrix Elektrolichtbogenofen Prozessphase 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	0,34	0,36	0,42	0,46	0,51	0,53	0,56	0,6	0,63	0,66	0,71	0,72	0,75	0,78	0,78	0,8	0,84	0,87	0,88	0,91	0,91	0,93	0,96	0,97	0,97	0,97	0,97	0,99	0,99	1	1	1	1	1	1	1	1	1	1	1
1	0,5	0,5	0,5	0,5	0,5	0,5	0,6	0,6	0,6	0,6	0,7	0,8	0,8	0,8	0,8	0,8	0,8	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	1	1	1	1	1	1	1	1	1	1	1
2	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,31	0,31	0,31	0,31	0,31	0,31	0,38	0,38	0,38	0,56	0,63	0,63	0,63	0,63	0,63	0,63	0,88	0,94	0,94	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0,13	0,13	0,13	0,13	0,19	0,19	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,38	0,44	0,63	0,63	0,69	0,81	0,81	0,81	0,81	0,81	0,94	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	0,2	0,2	0,2	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,4	0,47	0,6	0,67	0,67	0,67	0,73	0,73	0,8	0,8	0,87	0,87	0,87	0,87	0,87	0,87	0,87	0,87	0,87	0,87	1	1	1	1	1	1	1	1
5	0,14	0,14	0,14	0,14	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,43	0,43	0,43	0,57	0,57	0,57	0,57	0,57	0,57	0,71	0,71	0,86	0,86	0,86	0,86	1	1	1	1	1	1	1	1	1	1	1	1
6	0,23	0,23	0,23	0,31	0,31	0,31	0,31	0,31	0,31	0,31	0,31	0,31	0,31	0,38	0,46	0,46	0,54	0,54	0,62	0,69	0,69	0,69	0,69	0,69	0,69	0,85	0,85	0,92	0,92	0,92	0,92	0,92	0,92	0,92	1	1	1	1	1	1
7	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,27	0,27	0,27	0,27	0,27	0,36	0,36	0,55	0,55	0,55	0,73	0,73	0,73	0,73	0,73	0,73	0,82	0,82	0,82	0,82	0,82	0,91	0,91	1	1	1	1	1	1	1	1	1	1
8	0,33	0,33	0,33	0,33	0,33	0,47	0,47	0,47	0,47	0,47	0,47	0,53	0,6	0,67	0,67	0,67	0,67	0,67	0,87	0,87	0,87	0,87	0,87	0,87	0,87	0,87	0,87	0,93	0,93	1	1	1	1	1	1	1	1	1	1	1
9	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,19	0,25	0,25	0,31	0,31	0,38	0,38	0,38	0,44	0,5	0,5	0,56	0,63	0,69	0,69	0,69	0,75	0,81	0,88	0,88	0,88	0,94	0,94	0,94	0,94	0,94	0,94	0,94	0,94	0,94	1
10	0,14	0,14	0,29	0,29	0,29	0,29	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,43	0,43	0,43	0,43	0,43	0,5	0,5	0,64	0,64	0,71	0,71	0,86	0,93	0,93	0,93	0,93	0,93	1	1	1	1	1	1	1	1
11	0,33	0,33	0,42	0,42	0,42	0,42	0,42	0,42	0,5	0,58	0,58	0,58	0,75	0,75	0,83	0,83	0,92	0,92	0,92	0,92	0,92	0,92	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	0,21	0,21	0,21	0,21	0,21	0,21	0,21	0,32	0,32	0,37	0,37	0,37	0,37	0,42	0,42	0,47	0,58	0,63	0,63	0,68	0,79	0,79	0,79	0,84	0,84	0,84	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	1	1	1	1
13	0,09	0,14	0,18	0,23	0,27	0,27	0,27	0,27	0,32	0,32	0,36	0,36	0,45	0,55	0,64	0,64	0,68	0,68	0,68	0,73	0,82	0,86	0,86	0,91	0,91	0,91	0,95	0,95	0,95	0,95	0,95	0,95	0,95	0,95	1	1	1	1	1	1
14	0,07	0,07	0,07	0,13	0,13	0,17	0,17	0,2	0,2	0,23	0,23	0,3	0,3	0,33	0,37	0,5	0,5	0,6	0,77	0,83	0,93	0,93	0,93	0,93	0,93	0,93	0,97	1	1	1	1	1	1	1	1	1	1	1	1	1
15	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,12	0,12	0,18	0,18	0,18	0,29	0,41	0,47	0,47	0,53	0,59	0,65	0,65	0,71	0,71	0,71	0,76	0,88	0,88	0,88	0,88	0,88	0,94	0,94	0,94	0,94	0,94	1	1	1	1
16	0	0	0	0	0,04	0,04	0,04	0,04	0,07	0,11	0,11	0,11	0,11	0,15	0,15	0,22	0,33	0,41	0,52	0,59	0,78	0,85	0,89	0,89	0,93	0,93	0,93	0,93	0,96	0,96	1	1	1	1	1	1	1	1	1	1
17	0,11	0,15	0,15	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,26	0,33	0,41	0,56	0,63	0,74	0,78	0,85	0,89	0,89	0,89	0,89	0,89	0,89	0,93	0,96	0,96	1	1	1	1	1
18	0,07	0,07	0,07	0,11	0,11	0,11	0,11	0,11	0,11	0,14	0,14	0,18	0,18	0,21	0,21	0,21	0,32	0,32	0,36	0,39	0,54	0,64	0,71	0,82	0,89	0,89	0,89	0,89	0,96	0,96	0,96	0,96	0,96	0,96	0,96	1	1	1	1	1
19	0,03	0,03	0,03	0,03	0,07	0,07	0,13	0,13	0,13	0,13	0,17	0,17	0,17	0,17	0,17	0,17	0,17	0,2	0,23	0,47	0,53	0,63	0,73	0,77	0,83	0,87	0,9	0,9	0,93	0,93	0,93	1	1	1	1	1	1	1	1	1
20	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,14	0,19	0,24	0,24	0,3	0,32	0,32	0,38	0,46	0,51	0,62	0,73	0,73	0,78	0,86	0,89	0,95	0,95	0,97	1	1	1	1	1	1	1	1
21	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,11	0,11	0,11	0,11	0,11	0,11	0,15	0,15	0,15	0,15	0,15	0,22	0,26	0,44	0,56	0,74	0,78	0,85	0,85	0,85	0,89	0,93	0,93	0,96	1	1	1	1	1	1
22	0,03	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,11	0,11	0,11	0,11	0,13	0,13	0,13	0,18	0,21	0,21	0,21	0,24	0,24	0,24	0,32	0,53	0,66	0,71	0,76	0,89	0,97	1	1	1	1	1	1	1	1	1	1
23	0,06	0,06	0,06	0,06	0,06	0,08	0,08	0,08	0,1	0,12	0,12	0,14	0,14	0,14	0,2	0,2	0,24	0,24	0,24	0,24	0,29	0,29	0,29	0,41	0,53	0,67	0,76	0,78	0,84	0,92	0,94	0,96	0,96	0,98	0,98	0,98	1	1	1	1
24	0,06	0,06	0,06	0,07	0,07	0,07	0,08	0,08	0,08	0,08	0,08	0,09	0,11	0,13	0,14	0,14	0,14	0,17	0,17	0,19	0,21	0,21	0,23	0,29	0,6	0,78	0,84	0,87	0,87	0,92	0,97	0,98	0,99	1	1	1	1	1	1	1
25	0,05	0,05	0,07	0,07	0,07	0,07	0,08	0,09	0,11	0,12	0,12	0,13	0,13	0,13	0,15	0,15	0,15	0,17	0,2	0,21	0,22	0,23	0,29	0,32	0,39	0,66	0,77	0,83	0,89	0,95	0,98	1	1	1	1	1	1	1	1	1
26	0,05	0,06	0,06	0,06	0,06	0,07	0,07	0,07	0,1	0,1	0,11	0,12	0,14	0,14	0,16	0,16	0,16	0,16	0,16	0,16	0,17	0,19	0,21	0,25	0,31	0,42	0,79	0,87	0,9	0,93	0,95	0,98	0,99	0,99	1	1	1	1	1	1
27	0,06	0,07	0,07	0,07	0,08	0,08	0,1	0,1	0,1	0,1	0,1	0,11	0,11	0,13	0,14	0,14	0,14	0,15	0,15	0,17	0,17	0,19	0,19	0,19	0,24	0,29	0,44	0,74	0,81	0,86	0,9	0,96	0,97	1	1	1	1	1	1	1
28	0,03	0,03	0,03	0,04	0,05	0,05	0,05	0,05	0,07	0,07	0,08	0,08	0,11	0,12	0,12	0,13	0,14	0,14	0,14	0,14	0,16	0,16	0,18	0,22	0,28	0,32	0,38	0,47	0,68	0,76	0,87	0,93	0,96	0,97	0,99	1	1	1	1	1
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0,01	0,03	0,03	0,03	0,03	0,03	0,03	0,06	0,06	0,07	0,07	0,07	0,13	0,18	0,24	0,38	0,56	0,72	0,79	0,85	0,91	0,94	0,97	1	1	1	1
30	0,03	0,03	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,07	0,09	0,09	0,09	0,09	0,1	0,1	0,1	0,1	0,12	0,12	0,12	0,14	0,16	0,22	0,28	0,33	0,34	0,4	0,45	0,59	0,69	0,79	0,84	0,9	0,93	0,98	0,98	1	1	1
31	0	0	0	0,02	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,07	0,07	0,07	0,07	0,07	0,09	0,09	0,11	0,16	0,22	0,22	0,31	0,42	0,56	0,64	0,84	0,87	0,91	0,93	0,93	1	1	1
32	0	0,03	0,03	0,03	0,03	0,03	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,08	0,08	0,08	0,08	0,08	0,08	0,11	0,16	0,24	0,34	0,39	0,42	0,47	0,61	0,76	0,82	0,89	0,97	0,97	1	1
33	0	0	0,03	0,05	0,05	0,05	0,05	0,05	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,14	0,16	0,19	0,19	0,19	0,19	0,19	0,22	0,22	0,24	0,3	0,32	0,35	0,41	0,51	0,7	0,73	0,81	0,95	1	1	1
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,08	0,08	0,08	0,08	0,08	0,19	0,27	0,27	0,31	0,38	0,54	0,65	0,85	0,92	0,96	1	1
35	0	0	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,05	0,08	0,14	0,16	0,24	0,27	0,3	0,32	0,32	0,46	0,57	0,76	0,97	1	1	1
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,03	0,03	0,03	0,03	0,03	0,03	0,06	0,06	0,06	0,06	0,06	0,08	0,11	0,14	0,28	0,33	0,36	0,36	0,47	0,67	0,86	0,97	1	1
37	0	0	0	0	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,13	0,13	0,13	0,2	0,2	0,27	0,27	0,27	0,27	0,27	0,47	0,73	0,87	1	1
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,17	0,17	0,17	0,17	0,17	0,5	0,67	0,67	1
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,67	0,67	0,67	0,67	1	1

Tabelle 8-6: Übergangswahrscheinlichkeitsmatrix des Elektrolichtbogenofens für die erste Prozessphase

Übergangswahrscheinlichkeitsmatrix Elektrolichtbogenofen Prozessphase 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
0	0,95	0,95	0,95	0,95	0,95	0,96	0,96	0,96	0,96	0,96	0,96	0,96	0,96	0,97	0,97	0,97	0,97	0,97	0,97	0,98	0,98	0,98	0,98	0,98	0,99	0,99	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0,33	0,33	0,33	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,67	0,83	0,83	0,83	0,83	0,83	0,83	0,83	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,75	0,75	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,86	0,86	0,86	1	1	1	1	1	1	1	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,67	0,67	0,67	0,67	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	0,43	0,43	0,43	0,57	0,57	0,57	0,57	0,57	0,57	0,57	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,71	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,75	0,75	0,75	0,75	0,75	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,67	0,67	0,67	0,67	0,67	0,83	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	0,33	0,33	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	0,33	0,33	0,33	0,33	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,67	0,83	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,75	0,88	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83	1	1	1	1	1	1	1	1	1	1	1
15	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,67	0,67	0,67	0,67	0,67	0,67	0,83	0,83	0,83	0,83	0,83	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,83	0,83	1	1	1	1	1	1	1	1	1	1	1	1
18	0,15	0,15	0,15	0,15	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,31	0,31	0,31	0,31	0,38	0,46	0,69	0,69	0,69	0,69	0,69	0,69	0,69	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,67	0,83	1	1	1	1	1	1	1	1	1	1	1	1	1
20	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,67	1	1	1	1	1	1	1	1	1	1	1	1	1
21	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,8	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,63	0,75	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	0,13	0,13	0,13	0,19	0,31	0,38	0,38	0,38	0,38	0,38	0,38	0,38	0,44	0,44	0,44	0,44	0,44	0,44	0,44	0,44	0,44	0,44	0,44	0,69	0,81	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	0,05	0,09	0,09	0,09	0,12	0,12	0,12	0,12	0,14	0,16	0,19	0,21	0,21	0,23	0,26	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,35	0,86	0,98	0,98	1	1	1	1	1	1	1	1	1	1	1	1
25	0,03	0,06	0,08	0,08	0,08	0,08	0,08	0,09	0,09	0,12	0,14	0,14	0,14	0,17	0,22	0,22	0,22	0,25	0,25	0,28	0,28	0,31	0,35	0,37	0,42	0,83	0,97	0,98	0,98	1	1	1	1	1	1	1	1	1	1
26	0	0	0,02	0,02	0,05	0,05	0,05	0,09	0,12	0,12	0,12	0,12	0,12	0,14	0,14	0,14	0,14	0,14	0,16	0,16	0,19	0,21	0,21	0,21	0,26	0,37	0,88	0,98	0,98	0,98	0,98	1	1	1	1	1	1	1	1
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0,05	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,15	0,2	0,25	0,45	0,95	1	1	1	1	1	1	1	1	1	1	1
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,2	0,2	0,2	0,4	0,8	1	1	1	1	1	1	1	1	1	1
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	1	1	1	1	1	1	1	1	1
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabelle 8-7: Übergangswahrscheinlichkeitsmatrix des Elektrolichtbogenofens für die zweite Prozessphase

Übergangswahrscheinlichkeitsmatrix Elektrolichtbogenofen Prozessphase 3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0,6	0,7	0,8	0,8	0,8	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	0,9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	0,25	0,25	0,5	0,5	0,63	0,75	0,75	0,88	0,88	0,88	0,88	0,88	0,88	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	0,09	0,18	0,55	0,73	0,82	0,82	0,82	0,82	0,91	0,91	0,91	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	0	0,05	0,1	0,14	0,19	0,71	0,71	0,71	0,76	0,76	0,76	0,76	0,76	0,81	0,86	0,86	0,9	0,9	0,9	0,9	0,9	0,9	0,95	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	0	0	0	0,1	0,1	0,24	0,66	0,69	0,69	0,69	0,69	0,72	0,72	0,76	0,76	0,79	0,83	0,86	0,86	0,86	0,9	0,9	0,93	0,93	0,93	0,97	0,97	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0,06	0,06	0,11	0,39	0,67	0,78	0,78	0,83	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,94	0,94	0,94	0,94	0,94	0,94	0,94	0,94	0,94	0,94	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	0	0	0	0	0,07	0,07	0,14	0,43	0,71	0,79	0,79	0,86	0,86	0,86	0,86	0,86	0,86	0,86	0,86	0,93	0,93	0,93	0,93	0,93	0,93	0,93	0,93	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	0	0	0	0	0,1	0,1	0,1	0,2	0,2	0,3	0,3	0,3	0,4	0,4	0,4	0,4	0,5	0,5	0,5	0,5	0,7	0,8	0,8	0,8	0,8	0,8	0,9	0,9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0,14	0,14	0,29	0,29	0,29	0,29	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,57	0,57	0,57	0,57	0,71	0,71	0,86	0,86	0,86	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	0	0	0	0	0	0	0	0	0	0	0	0,14	0,29	0,29	0,29	0,43	0,43	0,43	0,43	0,43	0,57	0,57	0,71	0,86	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	0	0	0	0	0	0	0,22	0,22	0,33	0,44	0,44	0,44	0,56	0,56	0,56	0,56	0,56	0,56	0,56	0,56	0,56	0,67	0,67	0,67	0,67	0,78	0,78	0,89	0,89	0,89	0,89	0,89	0,89	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,2	0,4	0,6	0,6	0,6	0,6	0,8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0,2	0,2	0,2	0,2	0,4	0,4	0,6	0,6	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,25	0,25	0,25	0,25	0,5	0,5	0,75	0,75	0,75	0,75	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0,08	0,08	0,17	0,17	0,17	0,17	0,17	0,17	0,25	0,25	0,67	0,75	0,75	0,75	0,75	0,75	0,92	0,92	0,92	0,92	0,92	0,92	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	0	0	0	0	0	0	0	0	0	0,2	0,4	0,4	0,4	0,4	0,4	0,4	0,6	0,6	0,6	0,6	0,6	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	0	0	0	0	0	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,6	0,6	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	0	0	0	0	0	0	0	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,5	0,5	0,5	0,75	0,75	0,75	0,75	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	0	0	0	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,22	0,22	0,44	0,67	0,67	0,67	0,78	0,78	0,78	0,78	0,78	0,78	0,78	0,78	0,78	0,78	0,89	0,89	1	1	1	1	1	1	1	1	1	1	1	1
20	0	0	0	0	0	0	0,2	0,3	0,3	0,3	0,3	0,4	0,4	0,4	0,4	0,4	0,5	0,5	0,5	0,5	0,5	0,5	0,6	0,7	0,8	0,9	0,9	0,9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	0	0	0	0	0	0	0	0,08	0,17	0,17	0,17	0,25	0,33	0,42	0,42	0,42	0,42	0,42	0,42	0,42	0,42	0,5	0,75	0,75	0,75	0,83	0,92	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	0	0	0	0	0	0	0	0,08	0,08	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,33	0,33	0,33	0,33	0,33	0,42	0,5	0,58	0,58	0,58	0,58	0,83	0,83	0,92	0,92	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	0	0	0	0	0	0	0	0,03	0,03	0,06	0,09	0,12	0,18	0,18	0,18	0,18	0,18	0,18	0,21	0,21	0,21	0,21	0,21	0,29	0,76	0,82	0,91	0,94	0,94	0,94	0,97	0,97	0,97	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	0	0	0	0	0	0	0	0	0	0	0,06	0,06	0,09	0,09	0,09	0,12	0,12	0,15	0,15	0,15	0,15	0,15	0,18	0,18	0,44	0,79	0,82	0,85	0,88	0,88	0,88	0,88	0,91	0,94	0,97	0,97	1	1	1	1	1	1	1	1	1	1	1	1
25	0	0	0	0	0	0	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,15	0,2	0,55	0,65	0,75	0,75	0,8	0,85	0,9	0,95	0,95	0,95	0,95	0,95	1	1	1	1	1	1	1	1	1	1	1
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,36	0,57	0,79	0,86	0,93	0,93	0,93	0,93	0,93	0,93	0,93	0,93	0,93	0,93	1	1	1	1	1	1	1	1	1	1
27	0	0	0	0	0	0	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,07	0,13	0,13	0,13	0,13	0,13	0,13	0,2	0,4	0,6	0,67	0,67	0,73	0,73	0,8	0,87	0,87	1	1	1	1	1	1	1	1	1	1	1	1	1
28	0	0	0	0	0	0	0	0	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,25	0,38	0,38	0,38	0,5	0,88	0,88	0,88	0,88	0,88	1	1	1	1	1	1	1	1	1	1	1	1	1
29	0	0	0	0	0	0	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,17	0,17	0,42	0,5	0,58	0,83	0,83	1	1	1	1	1	1	1	1	1	1	1	1	1	1
30	0	0	0	0	0	0	0	0	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,17	0,17	0,25	0,58	0,75	0,92	0,92	0,92	0,92	1	1	1	1	1	1	1	1	1	1	1
31	0	0	0	0	0	0	0	0	0	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,17	0,17	0,17	0,17	0,17	0,17	0,17	0,33	0,33	0,42	0,42	0,5	0,58	0,75	0,83	0,83	0,83	0,83	0,83	0,92	0,92	1	1	1	1	1
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,06	0,12	0,12	0,12	0,12	0,12	0,12	0,12	0,12	0,12	0,24	0,35	0,41	0,47	0,65	0,82	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	1
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,22	0,22	0,22	0,33	0,67	0,89	1	1	1	1	1	1	1	1	1	1	1	1
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,07	0,07	0,07	0,07	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,2	0,2	0,27	0,33	0,53	0,6	0,8	0,8	0,93	0,93	0,93	0,93	0,93	0,93	0,93	1	1	1	1
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,13	0,13	0,25	0,38	0,38	0,38	0,5	0,63	0,63	0,63	0,88	1	1	1	1	1	1	1	1
36	0	0	0	0	0	0	0	0	0	0,14	0,14	0,14	0,14	0,14	0,14	0,14	0,14	0,14	0,14	0,14	0,14	0,14	0,14	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,43	0,57	0,71	0,71	1	1	1	1	1	1	1	1	1	1
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,67	1	1	1	1	1	1	1	1	1	1	1
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,33	0,33	0,33	0,33	0,33	0,33	0,5	0,5	0,5	0,67	0,67	0,67	0,83	1	1	1	1	1	1	1	1	1	1
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
42	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabelle 8-8: Übergangswahrscheinlichkeitsmatrix des Elektrolichtbogenofens für die dritte Prozessphase

Übergangswahrscheinlichkeitsmatrix Pfannenofen

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	0,87	0,88	0,89	0,89	0,9	0,91	0,91	0,92	0,92	0,92	0,93	0,93	0,94	0,95	0,95	0,95	0,96	0,96	0,96	0,96	0,97	0,97	0,98	0,99	0,99	1	1	1	1	1
1	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,6	1	1	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,5	0,5	1	1	1	1
3	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	1	1	1	1	1	1	1	1
4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,8	1	1	1	1
5	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	1	1	1	1
6	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	1	1	1	1	1	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,33	1	1	1	1	1
8	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	1	1	1	1	1	1	1
9	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	1	1	1	1
10	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	1	1	1	1	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
12	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,5	0,5	1	1	1	1
13	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,67	0,83	1	1	1	1
14	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,83	0,83	0,83	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	1	1	1	1	1
17	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	1	1	1	1	1	1	1	1	1	1	1
18	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	1	1	1	1	1	1	1	1	1	1	1
19	0,2	0,2	0,2	0,2	0,4	0,4	0,4	0,4	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,8	1	1	1	1	1
20	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,57	0,71	0,86	1	1	1
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	0,38	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,46	0,62	0,62	0,62	0,62	0,62	0,62	0,62	0,85	0,85	0,85	0,85	0,85	1	1	1
23	0	0	0	0	0	0	0	0	0	0	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,18	0,73	0,82	0,91	1	1	1	1
24	0,02	0,02	0,02	0,02	0,02	0,03	0,03	0,03	0,03	0,05	0,05	0,05	0,05	0,05	0,08	0,08	0,08	0,08	0,08	0,08	0,11	0,11	0,13	0,13	0,85	1	1	1	1	1
25	0,01	0,02	0,02	0,03	0,03	0,03	0,03	0,03	0,03	0,04	0,07	0,07	0,07	0,08	0,08	0,08	0,08	0,09	0,1	0,1	0,11	0,12	0,15	0,15	0,2	0,94	1	1	1	1
26	0,02	0,02	0,02	0,03	0,03	0,05	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,1	0,1	0,1	0,11	0,13	0,13	0,13	0,13	0,13	0,14	0,14	0,16	0,27	0,94	1	1	1
27	0	0	0	0	0	0	0	0	0	0,05	0,05	0,05	0,09	0,09	0,14	0,14	0,14	0,14	0,18	0,18	0,18	0,18	0,18	0,18	0,18	0,18	0,32	1	1	1
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Tabelle 8-9: Übergangswahrscheinlichkeitsmatrix des Pfannenofens

Input

```
import GUI
import pandas as pd

# General
simulationTime = 60*int(GUI.simulationTime.get())
referenceTemperature = 273.15 + int(GUI.referenceTemperature.get())
plotOutput = int(GUI.plotOutput.get())
statOutput = int(GUI.statOutput.get())
fileOutput = int(GUI.fileOutput.get())
inputPathTPM = GUI.inputPathTPM.get()
path = inputPathTPM
outputPathResult = GUI.outputPathResult.get()

# EAF
EAFnumber = int(GUI.EAFnumber.get())
output = float(GUI.output.get())
tapToTapTimeReal= float(GUI.tapToTapTimeReal.get())
tapToTapTimeStatic = float(GUI.tapToTapTimeStatic.get())
tapToTapTimeSpecific = float(GUI.tapToTapTimeSpecific.get())
chargingWeightReal= float(GUI.chargingWeightReal.get())
chargingWeightStatic = float(GUI.chargingWeightStatic.get())
chargingWeightSpecific = float(GUI.chargingWeightSpecific.get())
EAFspecificOxygenConsumptionReal= float(GUI.EAFspecificOxygenConsumptionReal.get())
EAFspecificOxygenConsumptionStatic = float(GUI.EAFspecificOxygenConsumptionStatic.get())
EAFspecificOxygenConsumptionSpecific = float(GUI.EAFspecificOxygenConsumptionSpecific.get())
EAFpowerFactor = float(GUI.EAFpowerFactor.get())
EAFspecificWasteHeat = float(GUI.EAFspecificWasteHeat.get())
TPM1_Data = pd.read_excel(path, sheet_name="TPM1")
TPM1 = TPM1_Data.to_numpy()
TPM2_Data = pd.read_excel(path, sheet_name="TPM2")
TPM2 = TPM2_Data.to_numpy()
TPM3_Data = pd.read_excel(path, sheet_name="TPM3")
TPM3 = TPM3_Data.to_numpy()
TPM4_Data = pd.read_excel(path, sheet_name="TPM3")
TPM4 = TPM4_Data.to_numpy()
TPM5_Data = pd.read_excel(path, sheet_name="TPM5")
TPM5 = TPM5_Data.to_numpy()
TPM6_Data = pd.read_excel(path, sheet_name="TPM6")
TPM6 = TPM6_Data.to_numpy()

# Fans
EAFfanNominalPowerlowPrimReal = float(GUI.EAFfanNominalPowerlowPrimReal.get())
EAFfanNominalPowerlowPrimSpecific = float(GUI.EAFfanNominalPowerlowPrimSpecific.get())
EAFfanNominalPowerlowPrimStatic = float(GUI.EAFfanNominalPowerlowPrimStatic.get())
EAFfanPowerFactorlowPrim = float(GUI.EAFfanPowerFactorPrimlow.get())
EAFfanNominalPowerhighPrimReal = float(GUI.EAFfanNominalPowerhighPrimReal.get())
EAFfanNominalPowerhighPrimSpecific = float(GUI.EAFfanNominalPowerhighPrimSpecific.get())
EAFfanNominalPowerhighPrimStatic = float(GUI.EAFfanNominalPowerhighPrimStatic.get())
EAFfanPowerFactorhighPrim = float(GUI.EAFfanPowerFactorhighPrim.get())
EAFfanNominalPowerlowSecReal = float(GUI.EAFfanNominalPowerlowSecReal.get())
EAFfanNominalPowerlowSecSpecific = float(GUI.EAFfanNominalPowerlowSecSpecific.get())
EAFfanNominalPowerlowSecStatic = float(GUI.EAFfanNominalPowerlowSecStatic.get())
EAFfanPowerFactorlowSec = float(GUI.EAFfanPowerFactorSeclow.get())
EAFfanNominalPowerhighSecReal = float(GUI.EAFfanNominalPowerhighSecReal.get())
EAFfanNominalPowerhighSecSpecific = float(GUI.EAFfanNominalPowerhighSecSpecific.get())
EAFfanNominalPowerhighSecStatic = float(GUI.EAFfanNominalPowerhighSecStatic.get())
EAFfanPowerFactorhighSec = float(GUI.EAFfanPowerFactorhighSec.get())
ID9fanNominalPowerhighReal = float(GUI.ID9fanNominalPowerhighReal.get())
ID9fanNominalPowerhighSpecific = float(GUI.ID9fanNominalPowerhighSpecific.get())
ID9fanNominalPowerhighStatic = float(GUI.ID9fanNominalPowerhighStatic.get())
```

```
ID9fanPowerFactorhigh = float(GUI.ID9fanPowerFactorhigh.get())
ID9fanNominalPowerlowReal = float(GUI.ID9fanNominalPowerlowReal.get())
ID9fanNominalPowerlowSpecific = float(GUI.ID9fanNominalPowerlowSpecific.get())
ID9fanNominalPowerlowStatic = float(GUI.ID9fanNominalPowerlowStatic.get())
ID9fanPowerFactorlow = float(GUI.ID9fanPowerFactorlow.get())
ID1fanNominalPowerReal = float(GUI.ID1fanNominalPowerReal.get())
ID1fanNominalPowerSpecific = float(GUI.ID1fanNominalPowerSpecific.get())
ID1fanNominalPowerStatic = float(GUI.ID1fanNominalPowerStatic.get())
ID1fanPowerFactor = float(GUI.ID1fanPowerFactor.get())

# Secondary metallurgy & LF
LfProcess1TimeLow = float(GUI.LfProcess1TimeLow.get())
LfProcess1TimeHigh = float(GUI.LfProcess1TimeHigh.get())
LfDegasTimeLow = float(GUI.LfDegasTimeLow.get())
LfDegasTimeHigh = float(GUI.LfDegasTimeHigh.get())
LfProcess2TimeLow = float(GUI.LfProcess2TimeLow.get())
LfProcess2TimeHigh = float(GUI.LfProcess2TimeHigh.get())
LfwaitingTime1Low = float(GUI.LfwaitingTime1Low.get())
LfwaitingTime1High = float(GUI.LfwaitingTime1High.get())
LfwaitingTime2Low = float(GUI.LfwaitingTime2Low.get())
LfwaitingTime2High = float(GUI.LfwaitingTime2High.get())
LfwaitingTime3Low = float(GUI.LfwaitingTime3Low.get())
LfwaitingTime3High = float(GUI.LfwaitingTime3High.get())
LFnumber = int(GUI.LFnumber.get())
totalProcessingTime = int(GUI.totalProcessingTime.get())
VDprocessingTime = int(GUI.VDprocessingTime.get())
VDsteamFlow = int(GUI.VDsteamFlow.get())/60
steamFeedSpecificEnthalpy = 2852
steamFeedTemperature = 210+273.15
VODprocessingTime = int(GUI.VODprocessingTime.get())
VODsteamFlow = int(GUI.VODsteamFlow.get())/60
VODspecificOxygenConsumption = int(GUI.VODspecificOxygenConsumption.get())
VODproportion = int(GUI.VODproportion.get())
steamBoilerEfficiency = int(GUI.steamBoilerEfficiency.get())/100
maxPowerLF1 = float(GUI.LF1Power.get())
LF1PowerFactor = float(GUI.LF1PowerFactor.get())
maxPowerLF2 = float(GUI.LF2Power.get())
LF2PowerFactor = float(GUI.LF2PowerFactor.get())
maxPowerLF3 = float(GUI.LF3Power.get())
LF3PowerFactor = float(GUI.LF3PowerFactor.get())
maxPowerLF4 = float(GUI.LF4Power.get())
LF4PowerFactor = float(GUI.LF3PowerFactor.get())

# Base load
thermalBaseLoad = float(GUI.thermalBaseLoad.get())
wasteHeatBaseLoad = 0
oxygenBaseLoad = 0
BaseNominalPower = int(GUI.BasePower.get())
BasePowerFactor = float(GUI.BasePowerFactor.get())

# Difference Load
TPM7_Data = pd.read_excel(path, sheet_name="TPM7")  # Last unter der Woche
TPM7 = TPM7_Data.to_numpy()
DiffPowerReal = float(GUI.DiffPowerReal.get())
DiffPowerSpecific = float(GUI.DiffPowerSpecific.get())
DiffPowerStatic = float(GUI.DiffPowerStatic.get())
DiffPowerFactor = float(GUI.DiffPowerFactor.get())

# Ladle logistics
LHnumber = int(GUI.LHnumber.get())
ladleCount = int(GUI.ladleCount.get())
liningTime = int(GUI.liningTime.get())
```

```
123 dryingTime = int(GUI.dryingTime.get())
124
125 # Analysis
126 NGemissionIntensity = float(GUI.NGemissionIntensity.get())
127 EEemissionIntensity = float(GUI.EEemissionIntensity.get())
128 NGprice = float(GUI.NGprice.get())
129 EEprice = float(GUI.EEprice.get())
130 emissionPrice = float(GUI.emissionPrice.get())
131
132 # Plotting
133 outputTime1 = int(GUI.outputTime1.get())
134 outputTime15 = int(GUI.outputTime15.get())
135 outputTime30 = int(GUI.outputTime30.get())
136 outputTime60 = int(GUI.outputTime60.get())
137 plotStack = int(GUI.plotStack.get())
138 plotTotalP = int(GUI.plotTotalP.get())
139 plotEafP = int(GUI.plotEafP.get())
140 plotLfP = int(GUI.plotLfP.get())
141 plotFanP = int(GUI.plotFanP.get())
142 plotBaseP = int(GUI.plotBaseP.get())
143 plotDiffP = int(GUI.plotDiffP.get())
144 plotTotalQ = int(GUI.plotTotalQ.get())
145 plotEafQ = int(GUI.plotEafQ.get())
146 plotLfQ = int(GUI.plotLfQ.get())
147 plotFanQ = int(GUI.plotFanQ.get())
148 plotBaseQ = int(GUI.plotBaseP.get())
149 plotDiffQ = int(GUI.plotDiffQ.get())
150 plotTotalS = int(GUI.plotTotalS.get())
151 plotEafS = int(GUI.plotEafS.get())
152 plotLfS = int(GUI.plotLfS.get())
153 plotFanS = int(GUI.plotFanS.get())
154 plotBaseS = int(GUI.plotBaseP.get())
155 plotDiffS = int(GUI.plotDiffS.get())
```

Main

```
1  import LadleHeater, BaseLoad, DiffLoad, ElectricArcFurnace, FanEafPrim, FanEafSec, FanId1, FanId9, Results, Plotting,
   Data, VacuumDegassing, VacuumOxygenDecarburization, SteamBoiler, LadleFurnace, SecondaryMetallurgy, Days
2  import Input
3  from LoadProfiles import LoadProfiles
4
5  # Initialization
6  loadProfiles = LoadProfiles(Input)  # load profiles
7  EAF = []
8  for n in range(0, Input.EAFnumber):
9      EAF.append(ElectricArcFurnace.EAF(n, Input))  # electric arc furnace(s)
10 LF = []
11 for n in range(0, Input.LFnumber):
12     LF.append(LadleFurnace.LF(n, Input))  # ladle furnace(s)
13 fanEafprim = FanEafPrim.FANEAFPRIM(Input)  # ID fans
14 fanEafsec = FanEafSec.FANEAFSEC(Input)
15 fan1 = FanId1.FAN1(Input)
16 fan9 = FanId9.FAN9(Input)
17 day = Days.DAY(Input)
18
19 lh = LadleHeater.LH(Input)  # ladle heater
20 vd = VacuumDegassing.VD(Input)  # vacuum degassing
21 vod = VacuumOxygenDecarburization.VOD(Input)    # vacuum decarburization
22 steam = SteamBoiler.SteamBoiler(Input)  # steam boiler
23 sm = SecondaryMetallurgy.SM(Input)
```

```
24  bl = BaseLoad.BL(Input)  # base load
25  dl = DiffLoad.DL(Input)  # difference load
26
27  # Load profile construction
28  for t in range(0, Input.simulationTime):
29      for eaf in EAF:
30          eaf.load(t, loadProfiles)  # electric arc furnace(s)
31      fanEafprim.load(t, loadProfiles, EAF)  # ID fans
32      fanEafsec.load(t, loadProfiles, EAF)
33      fan1.load(t, loadProfiles)
34      fan9.load(t, loadProfiles)
35      sm.load(t, loadProfiles)  # secondary metallurgy
36      for lf in LF:
37          lf.load(t, loadProfiles)  # ladle furnace(s)
38      lh.load(t, loadProfiles, EAF)  # ladle heater
39      vd.load(t, loadProfiles)  # vacuum degassing
40      vod.load(t, loadProfiles)  # vacuum decarburization
41      steam.load(t, loadProfiles)  # steam boiler
42      bl.load(t, loadProfiles)  # base load
43      dl.load(t, loadProfiles)  # difference load
44      day.info(t, loadProfiles)  # day
45
46  # Output
47  if Input.plotOutput == 1:
48      Plotting.profiles(Input, loadProfiles)  # plots
49  if Input.statOutput == 1:
50      Results.statistics(Input, loadProfiles)  # results
51  if Input.fileOutput == 1:
52      Data.dataexport(loadProfiles)  # data export to excel
```

LoadProfiles

```
1   import numpy as np
2   import Input
3
4   class LoadProfiles(object):
5       x = 200
6       # Time scale
7       time = np.arange(0, Input.simulationTime+x)/60
8       day = []
9       eafProduction = [] # eafProduction und FanProduction: in Days werden für diese die "Einsatzzeiten" festgelegt
10      fanProduction = []
11      processing = []
12      ladlefurnace = []
13      EAFtapToTapTime = []
14      EAFchargingWeight = []
15      EAFenergyConsumption = []
16      EAFoxygenConsumption = []
17      EAFwasteHeat = []
18
19      # Production KPIs
20      batchCount = 0
21      producedSteel = 0
22      ladleCount = 0
23
24      # Load profiles
25      totalElectricLoad = np.zeros((Input.simulationTime+x, 3))
26      totalNaturalGasFlow = np.zeros((Input.simulationTime+x, 1))
27      totalThermalLoad = np.zeros((Input.simulationTime+x, 1))
28      totalOxygenFlow = np.zeros((Input.simulationTime+x, 1))
```

```
29  totalSteamFlow = np.zeros((Input.simulationTime+x, 1))
30  totalWasteHeatFlow = np.zeros((Input.simulationTime+x, 2))
31
32  # Cumulative consumption
33  totalElectricEnergyConsumption = 0
34  totalNaturalGasConsumption = 0
35  totalThermalEnergyDemand = 0
36  totalOxygenDemand = 0
37  totalSteamDemand = 0
38  totalWasteHeatPotential = 0
39
40  # EAF
41  EAFelectricLoad = np.zeros((Input.simulationTime+x, 3))
42  EAFelectricEnergyConsumption = 0
43  EAFwasteHeatFlow = np.zeros((Input.simulationTime+x, 2))
44  EAFwasteHeatPotential = 0
45  EAFoxygenFlow = np.zeros((Input.simulationTime+x, 1))
46  EAFoxygenDemand = 0
47
48  # LF
49  LFelectricLoad = np.zeros((Input.simulationTime+x, 3))
50  LFelectricEnergyConsumption = 0
51
52  # LH
53  LHlogistics = np.zeros((Input.simulationTime+x, 16))
54  LHthermalLoad = np.zeros((Input.simulationTime+x, 2))
55  LHthermalEnergyDemand = 0
56  LHnaturalGasFlow = np.zeros((Input.simulationTime+x, 1))
57  LHnaturalGasConsumption = 0
58  LHoxygenFlow = np.zeros((Input.simulationTime+x, 1))
59  LHoxygenDemand = 0
60  LHwasteHeatFlow = np.zeros((Input.simulationTime+x, 2))
61  LHwasteHeatPotential = 0
62
63  #Fans
64  fanElectricLoad1 = np.zeros((Input.simulationTime+x, 3))
65  fanElectricEnergyConsumption1 = 0
66  fanElectricLoad9 = np.zeros((Input.simulationTime+x, 3))
67  fanElectricEnergyConsumption9 = 0
68  fanElectricLoad9high = np.zeros((Input.simulationTime+x, 3))
69  fanElectricEnergyConsumption9high = 0
70  fanElectricLoadEAFprim = np.zeros((Input.simulationTime + x, 3))
71  fanElectricLoadEAFsec= np.zeros((Input.simulationTime + x, 3))
72  fanElectricEnergyConsumptionEAF = 0
73
74  # Steam
75  steamThermalLoad = np.zeros((Input.simulationTime+x, 2))
76  steamThermalEnergyDemand = 0
77  steamNaturalGasFlow = np.zeros((Input.simulationTime+x, 1))
78  steamNaturalGasConsumption = 0
79  steamWasteHeatFlow = np.zeros((Input.simulationTime+x, 2))
80  steamWasteHeatPotential = 0
81  VODoxygenFlow = np.zeros((Input.simulationTime+x, 1))
82  VODoxygenDemand = 0
83  VDsteamFlow = np.zeros((Input.simulationTime+x, 1))
84  VODsteamFlow = np.zeros((Input.simulationTime+x, 1))
85
86  # Base load
87  BLelectricLoad = np.zeros((Input.simulationTime+x, 3))
88  BLelectricEnergyConsumption = 0
89  BLthermalLoad = np.zeros((Input.simulationTime+x, 2))
90  BLthermalEnergyDemand = 0
```

```
    BLnaturalGasFlow = np.zeros((Input.simulationTime+x, 1))
    BLnaturalGasConsumption = 0

    # Difference load
    DLelectricLoad = np.zeros((Input.simulationTime + x, 3))
    DLelectricEnergyConsumption = 0
    DLthermalLoad = np.zeros((Input.simulationTime + x, 2))
    DLthermalEnergyDemand = 0
    DLnaturalGasFlow = np.zeros((Input.simulationTime + x, 1))
    DLnaturalGasConsumption = 0

    # Energy demand
    EAFenergyDemand = 0
    LFenergyDemand = 0
    LHenergyDemand = 0
    fanEnergyDemand = 0
    steamEnergyDemand = 0
    BLenergyDemand = 0

    def __init__(self, t):
        self.t = t
```

Days

```
class DAY(object):
    def __init__(self, input):
        self.Input = input
        self.weekCount = 0

    def info(self, t, lp):
        factor = 60
        if t == (0 + self.weekCount) * factor:
            lp.day = 'sunday'
            lp.eafProduction = True
            lp.fanProduction = True
        elif t == (24 + self.weekCount) * factor:
            lp.day = 'monday'
            lp.eafProduction = True
            lp.fanProduction = True
        elif t == (48 + self.weekCount) * factor:
            lp.day = 'tuesday'
            lp.eafProduction = True
            lp.fanProduction = True
        elif t == (72 + self.weekCount) * factor:
            lp.day = 'wednesday'
            lp.eafProduction = True
            lp.fanProduction = True
        elif t == (96 + self.weekCount) * factor:
            lp.day = 'thursday'
            lp.eafProduction = True
            lp.fanProduction = True
        elif t == (110 + self.weekCount) * factor:
            lp.eafProduction = False
        elif t == (120 + self.weekCount) * factor:
            lp.day = 'friday'
            lp.fanProduction = False
        elif t == (144 + self.weekCount) * factor:
            lp.day = 'saturday'
            lp.eafProduction = False
            lp.fanProduction = False
```

```
37        elif t == (162 + self.weekCount) * factor:
38            lp.fanProduction = True
39        elif t == (167 + self.weekCount) * factor:
40            self.weekCount += 168
```

Secondary Metallurgy

```
1  import random
2  import Input
3
4
5  def t_onphase1(input):
6      return random.randint(Input.LfProcess1TimeLow, Input.LfProcess1TimeHigh)
7
8
9  def t_offdegas(input):
10     return random.randint(Input.LfDegasTimeLow, Input.LfDegasTimeHigh)
11
12
13 def t_onphase2(input):
14     return random.randint(Input.LfProcess2TimeLow, Input.LfProcess2TimeHigh)
15
16
17 def LfwaitingTime1(input):
18     return random.randint(Input.LfwaitingTime1Low, Input.LfwaitingTime1High)
19
20
21 def LfwaitingTime2(input):
22     return random.randint(Input.LfwaitingTime2Low, Input.LfwaitingTime2High)
23
24
25 class SM(object):
26     def __init__(self, input):
27         self.Input = input
28         self.count = 0
29         self.T_onPhase1 = t_onphase1(self.Input)
30         self.T_offDegas = t_offdegas(self.Input)
31         self.T_onPhase2 = t_onphase2(self.Input)
32         self.LfWaitingTime1 = LfwaitingTime1(self.Input)
33         self.LfWaitingTime2 = LfwaitingTime2(self.Input)
34         self.furnace_triples = []  # (Furnace, Time, Status, Chargen)
35
36     def init_furnace_triples(self, furnace_list):
37         for i in range(len(self.furnace_triples), len(furnace_list)):
38             self.furnace_triples.append([furnace_list[i], 0, True, 0])
39
40     def get_next_furnace(self, t):
41         for next in self.furnace_triples:
42             if t > next[1] and next[2] is True:
43                 next[1] = t + Input.LfProcess1TimeHigh + Input.LfDegasTimeHigh + Input.LfProcess2TimeHigh
44                 next[3] += 1
45                 return next[0]
46         return None
47
48     def load(self, t, lp):
49         self.init_furnace_triples(lp.furnace)
50         lp.ladleProductionJam.append(lp.batchCount) if lp.batchCount not in lp.ladleProductionJam else
   lp.ladleProductionJam
51         for ladle in lp.processing:
52             if ladle.refractory == 'standard':
```

```
53              if t == ladle.operationTime + self.LfWaitingTime1:
54                  next_furnace = self.get_next_furnace(t)
55                  if next_furnace is None:
56                      ladle.operationTime = t
57                      return
58                  if len(lp.ladleProductionJam) > 0:
59                      lp.ladleProductionJam.popleft()
60                  lp.furnaceNumber = next_furnace
61                  ladle.treatment = 'process1'
62              elif t == ladle.operationTime + self.LfWaitingTime1 + self.T_onPhase1:
63                  ladle.treatment = 'vd'
64              elif t == ladle.operationTime + self.LfWaitingTime1 + self.T_onPhase1 + self.T_offDegas:
65                  ladle.treatment = 'process2'
66              elif t == ladle.operationTime + self.LfWaitingTime1 + self.T_onPhase1 + self.T_offDegas + self.T_onPhase2:
67                  ladle.treatment = 'lh'
68                  ladle.casting = True
69              elif t == ladle.operationTime + self.LfWaitingTime1 + self.T_onPhase1 + self.T_offDegas + self.T_onPhase2 +
    1:
70                  ladle.casting = False
71                  # set new times
72                  self.LfWaitingTime1 = LfwaitingTime1(input)
73                  self.LfWaitingTime2 = LfwaitingTime2(input)
74                  self.T_onPhase1 = t_onphase1(input)
75                  self.T_onPhase2 = t_onphase2(input)
76                  self.t_offdegas = t_offdegas(input)
77
78          elif ladle.refractory == 'VOD':
79              if t == ladle.operationTime + self.LfWaitingTime1:
80                  ladle.treatment = 'vod'
81              elif t == ladle.operationTime + self.LfWaitingTime2:
82                  next_furnace = self.get_next_furnace(t)
83                  if next_furnace is None:
84                      ladle.operationTime = t
85                      return
86                  if len(lp.ladleProductionJam) > 0:
87                      lp.ladleProductionJam.popleft()
88                  lp.furnaceNumber = next_furnace
89                  ladle.treatment = 'process1'
90              elif t == ladle.operationTime + self.LfWaitingTime2 + self.T_onPhase1:
91                  ladle.treatment = 'vd'
92              elif t == ladle.operationTime + self.LfWaitingTime2 + self.T_onPhase1 + self.T_offDegas:
93                  ladle.treatment = 'process2'
94              elif t == ladle.operationTime + self.LfWaitingTime2 + self.T_onPhase1 + self.T_offDegas + self.T_onPhase2:
95                  ladle.treatment = 'lh'
96                  ladle.casting = True
97              elif t == ladle.operationTime + self.LfWaitingTime2 + self.T_onPhase1 + self.T_offDegas + self.T_onPhase2 +
    1:
98                  ladle.casting = False
99                  # set new times
100                 self.LfWaitingTime1 = LfwaitingTime1(input)
101                 self.LfWaitingTime2 = LfwaitingTime2(input)
102                 self.T_onPhase1 = t_onphase1(input)
103                 self.T_onPhase2 = t_onphase2(input)
104                 self.t_offdegas = t_offdegas(input)
```

Electric Arc Furnace

```
1  import numpy as np
2  import random
3  import math
```

```
import Input
np.seterr(divide='ignore', invalid='ignore')

# EAF stochastic properties
def taptotaptime(input):
    values = (93, 118, 143, 167, 192, 217, 242, 266, 291, 316)
    distr = (0.016, 0.296, 0.409, 0.194, 0.048, 0.016, 0.005, 0.006, 0.005, 0.005)
    return int(np.random.choice(values, p=distr))
def chargingweight(input):
    values = (49, 52, 55, 58, 61, 64, 67, 71, 74, 77)
    distr = (0.045, 0.038, 0.076, 0.023, 0.091, 0.114, 0.227, 0.167, 0.189, 0.0300)
    return int(np.random.choice(values, p=distr))
def specificoxygenconsumption(input):
    values = (11.3, 14.9, 18.4, 22.0, 25.5, 29.1, 32.6, 36.2, 39.7, 43.3)
    distr = (0.205, 0.462, 0.167, 0.053, 0.030, 0.023, 0.023, 0.008, 0.015, 0.014)
    return int(np.random.choice(values, p=distr))

# EAF class definition
class EAF(object):
    def __init__(self, number, input):
        self.Input = input
        self.tapping = False
        self.powerOn = False
        self.u = 1
        self.v = 0
        self.MCpower = np.zeros((self.Input.simulationTime, 1))
        self.MCpower = self.MCpower.astype(int)
        self.MCtemp = np.zeros((self.Input.simulationTime, 1))
        self.MCtemp = self.MCtemp.astype(int)
        self.x1 = 0.4
        self.x2 = 0.4
        self.x3 = 0.2
        self.tapToTapTime = 0
        self.chargingWeight = 0
        self.specificOxygenConsumption = 0
        self.powerOnTime = 0
        self.refiningTime = 30
        self.delayFactor = [3, 2, 1.5]

    def load(self, t, lp):
        self.tapping = False
        self.powerOn = False
        if (t == self.u) & ((self.u + 500) < self.Input.simulationTime):
            if Input.tapToTapTimeReal == 1:
                self.tapToTapTime = taptotaptime(self.Input)
            if Input.tapToTapTimeSpecific > 0:
                self.tapToTapTime = int(random.gauss(Input.tapToTapTimeSpecific, 32.23))
            if Input.tapToTapTimeStatic > 0:
                self.tapToTapTime = int(Input.tapToTapTimeStatic)
            if (Input.tapToTapTimeReal == 0) & (Input.tapToTapTimeSpecific == 0) & (Input.tapToTapTimeStatic == 0):
                self.tapToTapTime = int(random.uniform(taptotaptime(self.Input), taptotaptime(self.Input)))
            if Input.chargingWeightReal == 1:
                self.chargingWeight = chargingweight(self.Input)
            if Input.chargingWeightSpecific > 0:
                self.chargingWeight = int(random.gauss(Input.chargingWeightSpecific, 7.74))
            if Input.chargingWeightStatic > 0:
                self.chargingWeight = int(Input.chargingWeightStatic)
            if (Input.chargingWeightReal == 0) & (Input.chargingWeightSpecific == 0) & (
                    Input.chargingWeightStatic == 0):
                self.chargingWeight = int(chargingweight(self.Input))
            if Input.EAFspecificOxygenConsumptionReal == 1:
                self.specificOxygenConsumption = specificoxygenconsumption(self.Input)
```

```
66          if Input.EAFspecificOxygenConsumptionSpecific > 0:
67              self.specificOxygenConsumption = int(random.gauss(Input.EAFspecificOxygenConsumptionSpecific,
    6.81))
68          if Input.EAFspecificOxygenConsumptionStatic > 0:
69              self.specificOxygenConsumption = int(Input.EAFspecificOxygenConsumptionStatic)
70          if (Input.EAFspecificOxygenConsumptionReal == 0) & (Input.EAFspecificOxygenConsumptionSpecific == 0)
    & (
71                  Input.EAFspecificOxygenConsumptionStatic == 0):
72              self.specificOxygenConsumption = specificoxygenconsumption(self.Input)
73          if Input.LFnumber == 1:
74              if len(lp.ladleProductionJam) > Input.LFnumber:
75                  self.tapToTapTime = self.tapToTapTime * self.delayFactor[0]
76          if Input.LFnumber == 2:
77              if len(lp.ladleProductionJam) > Input.LFnumber:
78                  self.tapToTapTime = self.tapToTapTime * self.delayFactor[1]
79          if Input.LFnumber == 3:
80              if len(lp.ladleProductionJam) > Input.LFnumber:
81                  self.tapToTapTime = self.tapToTapTime * self.delayFactor[2]
82          self.powerOnTime = int(self.tapToTapTime * 0.9)
83          self.v = self.u
84          self.u = t + self.tapToTapTime
85          e_tot = self.chargingWeight * 383.7 + 5490
86          e_phase = (self.x1 * e_tot, (self.x1 + self.x2) * e_tot, (self.x1 + self.x2 + self.x3) * e_tot)
87          if lp.eafProduction is True:
88              while True:
89                  e = 0
90                  b = 0
91                  w = 0
92                  self.MCpower[0, 0] = 1
93                  for s in range(t, t + self.powerOnTime):
94                      lp.EAFwasteHeatFlow[s, 0] = 0
95                      lp.EAFwasteHeatFlow[s, 1] = 0
96                  for s in range(t, t + self.powerOnTime):
97                      if b < 2:
98                          r = random.random()
99                          self.MCpower[s, 0] = (np.sum(r > self.Input.TPM1[self.MCpower[s - 1, 0], :]))
100                         lp.EAFelectricLoad[s, 0] = self.MCpower[s, 0] * 1e3
101                         e += lp.EAFelectricLoad[s, 0] / 60
102                         if lp.EAFelectricLoad[s, 0] > 0:
103                             self.MCtemp[s, 0] = (np.sum(r > self.Input.TPM3[self.MCtemp[s - 1, 0], :]))
104                             temperature = self.MCtemp[s, 0] * 25 + 273.15
105                             lp.EAFwasteHeatFlow[s, 0] = 10 * (935.15 + 0.2197 * temperature) * (temperature
106                                                                 - self.Input.referenceTemperature) / 1e3
107                             lp.EAFwasteHeatFlow[s, 1] = temperature
108                         w += lp.EAFwasteHeatFlow[s, 0] / 60
109                         if e > e_phase[b]:
110                             lp.EAFelectricLoad[s, 0] = 0
111                             b += 1
112                     else:
113                         r = random.random()
114                         self.MCpower[s, 0] = (np.sum(r > self.Input.TPM2[self.MCpower[s - 1, 0], :]))
115                         lp.EAFelectricLoad[s, 0] = (self.MCpower[s, 0] - 1) * 1e3
116                         e += lp.EAFelectricLoad[s, 0] / 60
117                         if lp.EAFelectricLoad[s, 0] > 0:
118                             self.MCtemp[s, 0] = (np.sum(r > self.Input.TPM3[self.MCtemp[s - 1, 0], :]))
119                             temperature = self.MCtemp[s, 0] * 25 + 273.15
120                             lp.EAFwasteHeatFlow[s, 0] = 10 * (935.15 + 0.2197 * temperature) * (temperature
121                                                                 - self.Input.referenceTemperature) / 1e3
122                             lp.EAFwasteHeatFlow[s, 1] = temperature
123                         w += lp.EAFwasteHeatFlow[s, 0] / 60
124                         if e >= e_phase[-1]:
125                             break
```

```
126                 if self.Input.EAFpowerFactor > 0:
127                     lp.EAFelectricLoad[s, 1] = -lp.EAFelectricLoad[s, 0] * math.tan(
128                         math.acos(self.Input.EAFpowerFactor))
129                     lp.EAFelectricLoad[s, 2] = lp.EAFelectricLoad[s, 0] / self.Input.EAFpowerFactor
130                 else:
131                     lp.EAFelectricLoad[s, 1] = -lp.EAFelectricLoad[s, 0] * 0.6736 + 239 / 10e3
132                     lp.EAFelectricLoad[s, 2] = lp.EAFelectricLoad[s, 0] * 1.256 + 532 / 10e3
133             for r in range(t + 30, t + 30 + self.refiningTime):
134                 lp.EAFoxygenFlow[
135                     r, 0] = self.specificOxygenConsumption * self.chargingWeight / self.refiningTime
136             if (e < e_tot * (1 + 0.02)) & (e > e_tot * (1 - 0.02)):
137                 break
138         # Production KPIs
139         lp.producedSteel += self.chargingWeight * self.Input.output / 100
140         lp.batchCount += 1
141         lp.ladleCount.append(lp.batchCount)
142         lp.EAFchargingWeight.append(self.chargingWeight)
143         lp.EAFtapToTapTime.append(self.tapToTapTime)
144         lp.EAFenergyConsumption.append(e)
145         lp.EAFwasteHeat.append(w)
146         lp.EAFoxygenConsumption.append(self.specificOxygenConsumption * self.chargingWeight)
147         self.tapping = True
148     if (t >= self.v) & (t < self.v + self.powerOnTime):
149         self.powerOn = True
150     # electric energy
151     lp.totalElectricLoad[t, 0] += lp.EAFelectricLoad[t, 0]
152     lp.totalElectricLoad[t, 1] += lp.EAFelectricLoad[t, 1]
153     lp.totalElectricLoad[t, 2] += lp.EAFelectricLoad[t, 2]
154     lp.EAFelectricEnergyConsumption += lp.EAFelectricLoad[t, 0] / 60
155     lp.totalElectricEnergyConsumption += lp.EAFelectricLoad[t, 0] / 60
156     # oxygen
156     lp.EAFOxygenDemand = lp.EAFoxygenFlow[t, 0]
156     lp.totalOxygenFlow[t, 0] += lp.EAFoxygenFlow[t, 0]
156     lp.totalOxygenDemand += lp.EAFoxygenFlow[t, 0]
156     # waste heat
156     lp.EAFwasteHeatPotential += lp.EAFwasteHeatFlow[t, 0] / 60
156     lp.totalWasteHeatFlow[t, 0] += lp.EAFwasteHeatFlow[t, 0]
156     lp.totalWasteHeatPotential += lp.EAFwasteHeatFlow[t, 1] / 60
156     # energy demand
156     lp.EAFenergyDemand += lp.EAFelectricLoad[t, 0] / 60
156     return lp
```

Ladle Furnace

```
1  import Input
2  import numpy as np
3  import random
4  import math
5  np.seterr(divide='ignore', invalid='ignore')
6
7  class LF(object):
8      def __init__(self, number, input):
9          self.Input = input
10         self.chain = np.zeros((self.Input.simulationTime + 1440, 1))
11         self.chain = self.chain.astype(int)
12         self.maxValue = 0
13         self.powerFactor = 0
14         self.count = 0
15         self.number = number
16
```

```
17      def load(self, t, lp):
18          lp.furnace.append(self.number) if self.number not in lp.furnace else lp.furnace
19          if Input.PowerLfStatic > 0:
20              self.maxValue = self.Input.PowerLfStatic
21              self.powerFactor = self.Input.PowerFactorLfStatic
22          else:
23              if self.number == 1:
24                  randFurnace = random.randint(1, 2)
25                  if randFurnace == 1:
26                      self.maxValue = self.Input.maxPowerLF1
27                      self.powerFactor = self.Input.LF1PowerFactor
28                  if randFurnace == 2:
29                      self.maxValue = self.Input.maxPowerLF2
30                      self.powerFactor = self.Input.LF2PowerFactor
31              if self.number == 2:
32                  randFurnace = random.randint(3, 4)
33                  if randFurnace == 3:
34                      self.maxValue = self.Input.maxPowerLF3
35                      self.powerFactor = self.Input.LF3PowerFactor
36                  if randFurnace == 4:
37                      self.maxValue = self.Input.maxPowerLF4
38                      self.powerFactor = self.Input.LF4PowerFactor
39          for ladle in lp.processing:
40              if (len(lp.ladleCount) % 2) == 0 or Input.LFnumber == 1:
41                  if self.number == 1:
42                      if ladle.treatment == 'process1' or ladle.treatment == 'process2':
43                          r = random.random()
44                          if Input.PowerLfStatic > 0:
45                              self.chain[t, 0] = self.maxValue
46                              lp.LFelectricLoad[t, 0] = self.chain[t, 0]
47                              lp.LFelectricLoad[t, 1] = -lp.LFelectricLoad[t, 0] * math.tan(math.acos(self.powerFactor))
48                              lp.LFelectricLoad[t, 2] = lp.LFelectricLoad[t, 0] / self.powerFactor
49                              lp.LFelectricEnergyConsumption += lp.LFelectricLoad[t, 0] / 60
50                          else:
51                              self.chain[t, 0] = (np.sum(r > self.Input.TPM5[self.chain[t - 1, 0], :])) - 1
52                              lp.LFelectricLoad[t, 0] = (self.chain[t, 0]) * self.maxValue / 30
53                              if lp.LFelectricLoad[t, 0] <= self.maxValue * 0.8 or lp.LFelectricLoad[t, 0] >= self.maxValue *
    Input.LFnumber:
54                                  lp.LFelectricLoad[t, 0] = 0
55                              lp.LFelectricLoad[t, 1] = -lp.LFelectricLoad[t, 0] * math.tan(math.acos(self.powerFactor))
56                              lp.LFelectricLoad[t, 2] = lp.LFelectricLoad[t, 0] / self.powerFactor
57                              # electric load
58                              lp.totalElectricLoad[t, 0] += lp.LFelectricLoad[t, 0]
59                              lp.totalElectricLoad[t, 1] += lp.LFelectricLoad[t, 1]
60                              lp.totalElectricLoad[t, 2] += lp.LFelectricLoad[t, 2]
61                              lp.totalElectricEnergyConsumption += lp.LFelectricLoad[t, 0] / 60
62                          # energy demand
63                          lp.LFenergyDemand += lp.LFelectricLoad[t, 0] / 60
64                  else:
65                      if self.number == 2:
66                          if ladle.treatment == 'process1' or ladle.treatment == 'process2':
67                              r = random.random()
68                              if Input.PowerLfStatic > 0:
69                                  self.chain[t, 0] = self.maxValue
70                                  lp.LFelectricLoad[t, 0] = self.chain[t, 0]
71                                  lp.LFelectricLoad[t, 1] = -lp.LFelectricLoad[t, 0] * math.tan(math.acos(self.powerFactor))
72                                  lp.LFelectricLoad[t, 2] = lp.LFelectricLoad[t, 0] / self.powerFactor
73                                  lp.LFelectricEnergyConsumption += lp.LFelectricLoad[t, 0] / 60
74                              else:
75                                  self.chain[t, 0] = (np.sum(r > self.Input.TPM5[self.chain[t - 1, 0], :])) - 1
76                                  lp.LFelectricLoad[t, 0] = (self.chain[t, 0]) * self.maxValue / 30
77                                  if lp.LFelectricLoad[t, 0] <= self.maxValue * 0.8 or lp.LFelectricLoad[t, 0] >= self.maxValue *
```

```
    Input.LFnumber:
78                      lp.LFelectricLoad[t, 0] = 0
79                  lp.LFelectricLoad[t, 1] = -lp.LFelectricLoad[t, 0] * math.tan(math.acos(self.powerFactor))
80                  lp.LFelectricLoad[t, 2] = lp.LFelectricLoad[t, 0] / self.powerFactor
81                  # electric load
82                  lp.totalElectricLoad[t, 0] += lp.LFelectricLoad[t, 0]
83                  lp.totalElectricLoad[t, 1] += lp.LFelectricLoad[t, 1]
84                  lp.totalElectricLoad[t, 2] += lp.LFelectricLoad[t, 2]
85                  lp.totalElectricEnergyConsumption += lp.LFelectricLoad[t, 0] / 60
86              # energy demand
87              lp.LFenergyDemand += lp.LFelectricLoad[t, 0] / 60
88      return lp
```

EAF Primary Dedusting Fan

```
1  import numpy as np
2  import math
3  import random
4  import Input
5
6  # Fans stochastic properties
7  def eafFanPowerPrimaryhigh(input):
8     values = (163, 166, 169, 172, 175, 178, 181, 184, 187, 190, 193,196,199)
9     distr = (0.00500, 0.01832, 0.06580, 0.20627, 0.35397, 0.24431, 0.08245, 0.01444, 0.00444, 0.00083, 0.00111, 0.00167,
   0.00139)
10    return int(np.random.choice(values, p=distr))
11 def eafFanPowerPrimarylow(input):
12    values = (116, 120, 124, 128, 132, 136, 140, 144, 148, 152, 156,160)
13    distr = (0.00117, 0.04021, 0.00699, 0.00175, 0.06643, 0.26748, 0.31876, 0.12471, 0.10315, 0.01923, 0.03963, 0.01049)
14    return int(np.random.choice(values, p=distr))
15
16 class FANEAFPRIM(object):
17    def __init__(self, input):
18       self.Input = input
19       self.EafFanNominalPowerPrim = 0
20    def load(self, t, lp, eaf):
21       if lp.eafProduction is True:
22          for EAF in eaf:
23             if (EAF.powerOn is True) & (lp.EAFelectricLoad[t, 0] > 0):
24                if Input.EAFfanNominalPowerhighPrimReal == 1:
25                   self.EafFanNominalPowerPrim = int(random.uniform(eafFanPowerPrimaryhigh(self.Input) - 1,
   eafFanPowerPrimaryhigh(self.Input) + 1))
26                if Input.EAFfanNominalPowerhighPrimSpecific > 0:
27                   self.EafFanNominalPowerPrim = int(random.gauss(Input.EAFfanNominalPowerhighPrimSpecific, 4.09))
28                if Input.EAFfanNominalPowerhighPrimStatic > 0:
29                   self.EafFanNominalPowerPrim = int(Input.EAFfanNominalPowerhighPrimStatic)
30                if (Input.EAFfanNominalPowerhighPrimReal == 0) & (Input.EAFfanNominalPowerhighPrimSpecific == 0) &
   (Input.EAFfanNominalPowerhighPrimStatic == 0):
31                   self.EafFanNominalPowerPrim = int(random.uniform(eafFanPowerPrimaryhigh(self.Input) - 1,
   eafFanPowerPrimaryhigh(self.Input) + 1))
32                lp.fanElectricLoadEAFprim[t, 0] = self.EafFanNominalPowerPrim
33                lp.fanElectricLoadEAFprim[t, 2] = lp.fanElectricLoadEAFprim[t, 0] / self.Input.EAFfanPowerFactorhighPrim
34                lp.fanElectricLoadEAFprim[t, 1] = lp.fanElectricLoadEAFprim[t, 0] *
   math.tan(math.acos(self.Input.EAFfanPowerFactorhighPrim))
35             else:
36                if Input.EAFfanNominalPowerlowPrimReal == 1:
37                   self.EafFanNominalPowerPrim = int(random.uniform(eafFanPowerPrimarylow(self.Input) - 2,
   eafFanPowerPrimarylow(self.Input) + 2))
38                if Input.EAFfanNominalPowerlowPrimSpecific > 0:
39                   self.EafFanNominalPowerPrim = int(random.gauss(Input.EAFfanNominalPowerlowPrimSpecific, 7.60))
```

```
            if Input.EAFfanNominalPowerlowPrimStatic > 0:
                self.EafFanNominalPowerPrim = int(Input.EAFfanNominalPowerlowPrimStatic)
            if (Input.EAFfanNominalPowerlowPrimReal == 0) & (Input.EAFfanNominalPowerlowPrimSpecific == 0) &
(Input.EAFfanNominalPowerlowPrimStatic == 0):
                self.EafFanNominalPowerPrim = int(random.uniform(eafFanPowerPrimarylow(self.Input) - 2,
eafFanPowerPrimarylow(self.Input) + 2))
            lp.fanElectricLoadEAFprim[t, 0] = self.EafFanNominalPowerPrim
            lp.fanElectricLoadEAFprim[t, 2] = lp.fanElectricLoadEAFprim[t, 0] / self.Input.EAFfanPowerFactorlowPrim
            lp.fanElectricLoadEAFprim[t, 1] = lp.fanElectricLoadEAFprim[t, 0] *
math.tan(math.acos(self.Input.EAFfanPowerFactorlowPrim))
        # electric load
        lp.totalElectricLoad[t, 0] += lp.fanElectricLoadEAFprim[t, 0]
        lp.totalElectricLoad[t, 1] += lp.fanElectricLoadEAFprim[t, 1]
        lp.totalElectricLoad[t, 2] += lp.fanElectricLoadEAFprim[t, 2]
        lp.totalElectricEnergyConsumption += lp.fanElectricLoadEAFprim[t, 0] / 60
        # energy demand
        lp.fanEnergyDemand += lp.fanElectricLoadEAFprim[t, 0] / 60
```

EAF Secondary Dedusting Fan

```
import numpy as np
import math
import random
import Input

# Fan stochastic properties
def eafFanPowerSecondaryhigh(input):
    values = (185, 190, 195, 200, 205, 210, 215, 220, 225, 230, 235,240,245)
    distr = (0.00425, 0.00150, 0.002, 0.002, 0.003, 0.00275, 0.00175, 0.01899, 0.09295, 0.35757, 0.4018, 0.11119, 0.00025)
    return int(np.random.choice(values, p=distr))
def eafFanPowerSecondarylow(input):
    values = (50, 70, 90, 110, 130, 150, 170, 190)
    distr = (0.00109, 0.00109, 0, 0.08188, 0.47817, 0.31441, 0.03493, 0.08843)
    return int(np.random.choice(values, p=distr))

class FANEAFSEC(object):
    def __init__(self, input):
        self.Input = input
        self.EafFanNominalPowerSec = 0

    def load(self, t, lp, eaf):
        if lp.eafProduction is True:
            for EAF in eaf:
                if (EAF.powerOn is True) & (lp.EAFelectricLoad[t, 0] > 0):
                    if Input.EAFfanNominalPowerhighSecReal == 1:
                        self.EafFanNominalPowerSec = int(random.uniform(eafFanPowerSecondaryhigh(self.Input) - 1,
eafFanPowerSecondaryhigh(self.Input) + 1))
                    if Input.EAFfanNominalPowerhighSecSpecific > 0:
                        self.EafFanNominalPowerSec = int(random.gauss(Input.EAFfanNominalPowerhighSecSpecific, 6.61))
                    if Input.EAFfanNominalPowerhighSecStatic > 0:
                        self.EafFanNominalPowerSec = int(Input.EAFfanNominalPowerhighSecStatic)
                    if (Input.EAFfanNominalPowerhighSecReal == 0) & (Input.EAFfanNominalPowerhighSecSpecific == 0) &
(Input.EAFfanNominalPowerhighSecStatic == 0):
                        self.EafFanNominalPowerSec = int(random.uniform(eafFanPowerSecondaryhigh(self.Input) - 1,
eafFanPowerSecondaryhigh(self.Input) + 1))
                    lp.fanElectricLoadEAFsec[t, 0] = self.EafFanNominalPowerSec
                    lp.fanElectricLoadEAFsec[t, 2] = lp.fanElectricLoadEAFsec[t, 0] / self.Input.EAFfanPowerFactorhighSec
                    lp.fanElectricLoadEAFsec[t, 1] = lp.fanElectricLoadEAFsec[t, 0] *
math.tan(math.acos(self.Input.EAFfanPowerFactorhighSec))
                else:
```

```
37                if Input.EAFfanNominalPowerlowSecReal == 1:
38                    self.EafFanNominalPowerSec = int(random.uniform(eafFanPowerSecondarylow(self.Input) - 2,
    eafFanPowerSecondarylow(self.Input) + 2))
39                if Input.EAFfanNominalPowerlowSecSpecific > 0:
40                    self.EafFanNominalPowerSec = int(random.gauss(Input.EAFfanNominalPowerlowSecSpecific, 21.71))
41                if Input.EAFfanNominalPowerlowSecStatic > 0:
42                    self.EafFanNominalPowerSec = int(Input.EAFfanNominalPowerlowSecStatic)
43                if (Input.EAFfanNominalPowerlowSecReal == 0) & (Input.EAFfanNominalPowerlowSecSpecific == 0) &
    (Input.EAFfanNominalPowerlowSecStatic == 0):
44                    self.EafFanNominalPowerSec = int(random.uniform(eafFanPowerSecondarylow(self.Input) - 2,
    eafFanPowerSecondarylow(self.Input) + 2))
45                lp.fanElectricLoadEAFsec[t, 0] = self.EafFanNominalPowerSec
46                lp.fanElectricLoadEAFsec[t, 2] = lp.fanElectricLoadEAFsec[t, 0] / self.Input.EAFfanPowerFactorlowSec
47                lp.fanElectricLoadEAFsec[t, 1] = lp.fanElectricLoadEAFsec[t, 0] *
    math.tan(math.acos(self.Input.EAFfanPowerFactorlowSec))
48            # electric load
49            lp.totalElectricLoad[t, 0] += lp.fanElectricLoadEAFsec[t, 0]
50            lp.totalElectricLoad[t, 1] += lp.fanElectricLoadEAFsec[t, 1]
51            lp.totalElectricLoad[t, 2] += lp.fanElectricLoadEAFsec[t, 2]
52            lp.totalElectricEnergyConsumption += lp.fanElectricLoadEAFsec[t, 0] / 60
53            # energy demand
54            lp.fanEnergyDemand += lp.fanElectricLoadEAFsec[t, 0] / 60
```

Dedusting Fan Hall 9

```
1  import numpy as np
2  import math
3  import random
4  import Input
5
6  # Fan stochastic properties
7  def castingfanpower(input): #Halle 9 Volllast Leistungsverteilung
8      values = (215, 230, 245, 260, 275, 290, 305, 320, 335, 350)
9      distr = (0.00559, 0.00571, 0.00571, 0.00535, 0.00559, 0.00595, 0.06161, 0.52898, 0.37477, 0.00074)
10     return int(np.random.choice(values, p=distr))
11 def castingfantime(input): #Halle 9 Volllast Zeitverteilung
12     values = (12, 20, 28, 36, 44, 52, 60, 68, 76, 84)
13     distr = (0.01961, 0.01961, 0.51634, 0.07843, 0.07190, 0.15686, 0.09804, 0.01307, 0.01307, 0.01307)
14     return int(np.random.choice(values, p=distr))
15 def id9fanpower(input): #Halle 9 Teillast Leistungsverteilung
16     values = (300, 310, 320, 330, 340, 350, 360, 370, 380, 390)
17     distr = (0.00081, 0.00511, 0.03353, 0.24439, 0.41149, 0.27605, 0.02543, 0.00112, 0.00050, 0.00157)
18     return int(np.random.choice(values, p=distr))
19 def LfwaitingTime3(input):
20     return random.randint(Input.LfwaitingTime3Low, Input.LfwaitingTime3High)
21
22 class FAN9(object):
23     def __init__(self, input):
24         self.Input = input
25         self.castingFanPower = 0
26         self.castingFanTime = 0
27         self.ID9FanPower = 0
28         self.LfwaitingTime3 = LfwaitingTime3(self.Input)
29
30     def load(self, t, lp):
31         # Partial Load
32         if lp.fanProduction is True:
33             if Input.ID9fanNominalPowerlowReal == 1:
34                 self.ID9FanPower = int(random.uniform(id9fanpower(self.Input) - 4, id9fanpower(self.Input) + 4))
35             if Input.ID9fanNominalPowerlowSpecific > 0:
```

```
36            self.ID9FanPower = int(random.gauss(Input.ID9fanNominalPowerlowSpecific, 7.97))
37          if Input.ID9fanNominalPowerlowStatic > 0:
38            self.ID9FanPower = int(Input.ID9fanNominalPowerlowStatic)
39          if (Input.ID9fanNominalPowerlowReal == 0) & (Input.ID9fanNominalPowerlowSpecific == 0) & (
40                Input.ID9fanNominalPowerlowStatic == 0):
41            self.ID9FanPower = int(random.uniform(id9fanpower(self.Input) - 4, id9fanpower(self.Input) + 4))
42          lp.fanElectricLoad9[t, 0] = self.ID9FanPower
43          lp.fanElectricLoad9[t, 2] = lp.fanElectricLoad9[t, 0] / self.Input.ID9fanPowerFactorlow
44          lp.fanElectricLoad9[t, 1] = lp.fanElectricLoad9[t, 0] * math.tan(math.acos(self.Input.ID9fanPowerFactorlow))
45        else:
46          lp.fanElectricLoad9[t, 0] = 0
47          lp.fanElectricLoad9[t, 2] = 0
48          lp.fanElectricLoad9[t, 1] = 0
49        # electric load
50        lp.totalElectricLoad[t, 0] += lp.fanElectricLoad9[t, 0]
51        lp.totalElectricLoad[t, 1] += lp.fanElectricLoad9[t, 1]
52        lp.totalElectricLoad[t, 2] += lp.fanElectricLoad9[t, 2]
53        lp.totalElectricEnergyConsumption += lp.fanElectricLoad9[t, 0] / 60
54        # energy demand
55        lp.fanEnergyDemand += lp.fanElectricLoad9[t, 0] / 60
56
57        # Full Load while casting
58        for ladle in lp.processing:
59          if ladle.casting == True:
60            self.castingFanTime = int(random.uniform(castingfantime(self.Input) - 4, castingfantime(self.Input) + 4))
61            for t in range(t+self.LfwaitingTime3, t + self.castingFanTime + self.LfwaitingTime3): # Zeitdauer des Gusses
62              if Input.ID9fanNominalPowerhighReal == 1:
63                self.castingFanPower = int(random.uniform(castingfanpower(self.Input), castingfanpower(self.Input)))
64              if Input.ID9fanNominalPowerhighSpecific > 0:
65                self.castingFanPower = int(random.gauss(Input.ID9fanNominalPowerhighSpecific, 32.29))
66              if Input.ID9fanNominalPowerhighStatic > 0:
67                self.castingFanPower = int(Input.ID9fanNominalPowerhighStatic)
68              if (Input.ID9fanNominalPowerhighReal == 0) & (Input.ID9fanNominalPowerhighSpecific == 0) &
(Input.ID9fanNominalPowerhighStatic == 0):
69                self.castingFanPower = int(random.uniform(castingfanpower(self.Input) , castingfanpower(self.Input)))
70              lp.fanElectricLoad9high[t, 0] = self.castingFanPower
71              lp.fanElectricLoad9high[t, 2] = lp.fanElectricLoad9high[t , 0] / self.Input.ID9fanPowerFactorhigh
72              lp.fanElectricLoad9high[t, 1] = lp.fanElectricLoad9high[t , 0] *
math.tan(math.acos(self.Input.ID9fanPowerFactorhigh))
73          # electric load
74          lp.totalElectricLoad[t, 0] += lp.fanElectricLoad9high[t, 0]
75          lp.totalElectricLoad[t, 1] += lp.fanElectricLoad9high[t, 1]
76          lp.totalElectricLoad[t, 2] += lp.fanElectricLoad9high[t, 2]
77          lp.totalElectricEnergyConsumption += lp.fanElectricLoad9high[t, 0] / 60
78          # energy demand
79          lp.fanEnergyDemand += lp.fanElectricLoad9high[t, 0] / 60
```

Dedusting Fan Hall 1

```
1  import numpy as np
2  import math
3  import random
4  import Input
5
6  # Fan stochastic properties
7  def id1fanpower(input): #Halle 1 Leistungsverteilung
8    values = (910, 928, 946, 964, 982, 1000, 1018, 1036, 1054, 1072)
9    distr = (0.00122, 0.03932, 0.12337, 0.22025, 0.36950, 0.17095, 0.04493, 0.02781, 0.00224, 0.00041)
10   return int(np.random.choice(values, p=distr))
11
```

```
class FAN1(object):
    def __init__(self, input):
        self.Input = input
        self.ID1FanPower = 0

    def load(self, t, lp):
        if lp.fanProduction is True:
            if Input.ID1fanNominalPowerReal == 1:
                self.ID1FanPower = int(random.uniform(id1fanpower(self.Input) - 9, id1fanpower(self.Input) + 9))
            if Input.ID1fanNominalPowerSpecific > 0:
                self.ID1FanPower = int(random.gauss(Input.ID1fanNominalPowerSpecific, 23.39))
            if Input.ID1fanNominalPowerStatic > 0:
                self.ID1FanPower = int(Input.ID1fanNominalPowerStatic)
            if (Input.ID1fanNominalPowerReal == 0) & (Input.ID1fanNominalPowerSpecific == 0) & (
                    Input.ID1fanNominalPowerStatic == 0):
                self.ID1FanPower = int(random.uniform(id1fanpower(self.Input) - 9, id1fanpower(self.Input) + 9))
            lp.fanElectricLoad1[t, 0] = self.ID1FanPower
            lp.fanElectricLoad1[t, 2] = lp.fanElectricLoad1[t, 0] / self.Input.ID1fanPowerFactor
            lp.fanElectricLoad1[t, 1] = lp.fanElectricLoad1[t, 0] * math.tan(math.acos(self.Input.ID1fanPowerFactor))
        else:
            lp.fanElectricLoad1[t, 0] = 0
            lp.fanElectricLoad1[t, 2] = 0
            lp.fanElectricLoad1[t, 1] = 0
        # electric load
        lp.totalElectricLoad[t, 0] += lp.fanElectricLoad1[t, 0]
        lp.totalElectricLoad[t, 1] += lp.fanElectricLoad1[t, 1]
        lp.totalElectricLoad[t, 2] += lp.fanElectricLoad1[t, 2]
        lp.totalElectricEnergyConsumption += lp.fanElectricLoad1[t, 0] / 60
        # energy demand
        lp.fanEnergyDemand += lp.fanElectricLoad1[t, 0] / 60
```

Base-Load

```
import math

class BL(object):
    def __init__(self, input):
        self.Input = input

    def load(self, t, lp):
        # electric energy
        lp.BLelectricLoad[t, 0] = self.Input.BaseNominalPower   # niedrigster Wert im Betrachtungszeitraum
        lp.BLelectricLoad[t, 1] = lp.BLelectricLoad[t, 0]*math.tan(math.acos(self.Input.BasePowerFactor))
        lp.BLelectricLoad[t, 2] = lp.BLelectricLoad[t, 0]/self.Input.BasePowerFactor
        lp.BLelectricEnergyConsumption += lp.BLelectricLoad[t, 0]/60
        lp.totalElectricLoad[t, 0] += lp.BLelectricLoad[t, 0]
        lp.totalElectricLoad[t, 1] += lp.BLelectricLoad[t, 1]
        lp.totalElectricLoad[t, 2] += lp.BLelectricLoad[t, 2]
        lp.totalElectricEnergyConsumption += lp.BLelectricLoad[t, 0]/60
        # thermal load
        lp.BLthermalLoad[t, 0] = self.Input.thermalBaseLoad
        lp.BLthermalEnergyDemand += lp.BLthermalLoad[t, 0]/60
        lp.totalThermalLoad[t, 0] += lp.BLthermalLoad[t, 0]
        lp.totalThermalEnergyDemand += lp.BLthermalLoad[t, 0]/60
        # natural gas
        lp.BLnaturalGasFlow[t, 0] = lp.BLthermalLoad[t, 0]
        lp.BLnaturalGasConsumption += lp.BLthermalLoad[t, 0]/60
        lp.totalNaturalGasFlow[t, 0] += lp.BLthermalLoad[t, 0]
        lp.totalNaturalGasConsumption += lp.BLthermalLoad[t, 0]/60
        # energy demand
```

```
        lp.BLenergyDemand += (lp.BLelectricLoad[t, 0]+lp.BLnaturalGasFlow[t, 0])/60
        return lp
```

Difference- Load

```
import numpy as np
import math
import random
import Input
np.seterr(divide='ignore', invalid='ignore')

def diffloadweekend(input):  # Halle 1 Last
    values = (300, 600, 900, 1200, 1500, 1800, 2100, 2400, 2700, 3000)
    distr = (0.05143, 0.01714, 0.04572, 0.01714, 0.09714, 0.20000, 0.14857, 0.24571, 0.17143, 0.00572)
    return int(np.random.choice(values, p=distr))

class DL(object):
    def __init__(self, input):
        self.Input = input
        self.chain = np.zeros((self.Input.simulationTime + 1000, 1))
        self.chain = self.chain.astype(int)
        self.diffLoadWeekday = 0
        self.diffLoadWeekend = 0

    def load(self, t, lp):
        if lp.eafProduction == True:
            if Input.DiffPowerSpecific == 1:
                interval = 200
                r = random.random()
                self.chain[t, 0] = (np.sum(r > self.Input.TPM7[self.chain[t - 1, 0], :])) - 1
                self.diffLoadWeekday = (self.chain[t - 1, 0] - 1) * interval
            if Input.DiffPowerSpecific > 0:
                self.diffLoadWeekday = int(random.gauss(Input.DiffPowerSpecific, 7.60))
            if Input.DiffPowerStatic > 0:
                self.diffLoadWeekday = int(Input.DiffPowerStatic)
            if (Input.DiffPowerSpecific == 0) & (Input.DiffPowerSpecific == 0) & (Input.DiffPowerStatic == 0):
                interval = 200
                r = random.random()
                self.chain[t, 0] = (np.sum(r > self.Input.TPM7[self.chain[t - 1, 0], :])) - 1
                self.diffLoadWeekday = (self.chain[t - 1, 0] - 1) * interval
            lp.DLelectricLoad[t, 0] = self.diffLoadWeekday
            lp.DLelectricLoad[t, 1] = lp.DLelectricLoad[t, 0] * math.tan(math.acos(self.Input.DiffPowerFactor))
            lp.DLelectricLoad[t, 2] = lp.DLelectricLoad[t, 0] / self.Input.DiffPowerFactor
        else:
            self.diffLoadWeekend = int(random.uniform(diffloadweekend(self.Input) - 150, diffloadweekend(self.Input) + 150))
            lp.DLelectricLoad[t, 0] = self.diffLoadWeekend
            lp.DLelectricLoad[t, 1] = lp.DLelectricLoad[t, 0] * math.tan(math.acos(self.Input.DiffPowerFactor))
            lp.DLelectricLoad[t, 2] = lp.DLelectricLoad[t, 0] / self.Input.DiffPowerFactor
        lp.DLelectricEnergyConsumption += lp.DLelectricLoad[t, 0] / 60
        lp.totalElectricLoad[t, 0] += lp.DLelectricLoad[t, 0]
        lp.totalElectricLoad[t, 1] += lp.DLelectricLoad[t, 1]
        lp.totalElectricLoad[t, 2] += lp.DLelectricLoad[t, 2]
        lp.totalElectricEnergyConsumption += lp.DLelectricLoad[t, 0] / 60
        return lp
```

Ladle Heater

```
1  from collections import deque
2  import numpy as np
3  import random
4  import Input
5
6  def totalprocessingtime(input):
7      return int(np.random.normal(loc=input.totalProcessingTime, scale=20, size=1))
8
9  class Ladle(object):
10     def __init__(self, number, input):
11         self.number = number
12         self.treatment = 'lh'
13         self.casting = False
14         if random.random() >= (1-(Input.VODproportion/100)):
15             self.refractory = "VOD"
16         else:
17             self.refractory = "standard"
18         self.cycleCount = random.randint(0, 9)
19         self.operationTime = 10000
20         self.maintenanceTime = 0
21         self.heatingTime = 0
22         self.totalProcessingTime = totalprocessingtime(input)
23
24     def info(self):
25         print(self.number, self.cycleCount, self.operationTime,
26               self.heatingTime, self.maintenanceTime)
27
28     def refractory(self):
29         return self.refractory
30
31     def cycle(self):
32         return self.cycleCount
33
34 class LH(object):
35     def __init__(self, input):
36         self.Input = input
37         self.ladles = []
38         for n in range(1, self.Input.ladleCount + 1):
39             self.ladles.append(Ladle(n, input))
40         self.heating = deque([])
41         for ladle in self.ladles:
42             self.heating.append(ladle)
43         self.maintenance = []
44         self.operation = []
45         self.batchCount = 0
46
47     def load(self, t, lp, eaf):
48         lp.LHlogistics[t, 0] = t
49         for EAF in eaf:
50             if (EAF.tapping is True) & (t >= 60):
51                 self.operation.append(self.heating.popleft())
52                 self.operation[-1].cycleCount += 1
53                 self.operation[-1].operationTime = t
54         for ladle in self.operation:
55             if t == ladle.operationTime + Input.LfProcess1TimeHigh + Input.LfDegasTimeHigh +
   Input.LfProcess2TimeHigh:
56                 self.batchCount += 1
57                 if ladle.cycleCount == 10:
58                     self.maintenance.append(ladle)
59                     ladle.maintenanceTime = t
```

```
60                else:
61                    self.heating.append(ladle)
62                    ladle.heatingTime = t
63                    ladle.totalProcessingTime=totalprocessingtime(self.Input)
64                self.operation.remove(ladle)
65            for ladle in self.maintenance:
66                if t == ladle.maintenanceTime + self.Input.liningTime + self.Input.dryingTime:
67                    ladle.cycleCount = 0
68                    self.heating.append(ladle)
69                    ladle.heatingTime = t
70                    self.maintenance.remove(ladle)
71            for ladle in self.heating:
72                lp.LHlogistics[t, ladle.number] = 1
73            for ladle in self.operation:
74                lp.LHlogistics[t, ladle.number] = 2
75            for ladle in self.maintenance:
76                lp.LHlogistics[t, ladle.number] = 3
77            lp.processing = self.operation
78            # thermal energy
79            lp.LHthermalLoad[t, 0] = len(self.heating)*600
80            lp.LHthermalLoad[t, 1] = 1050 + 273.15
81            lp.LHthermalEnergyDemand += lp.LHthermalLoad[t, 0]/60
82            lp.totalThermalLoad[t, 0] += lp.LHthermalLoad[t, 0]
83            lp.totalThermalEnergyDemand += lp.LHthermalLoad[t, 0]/60
84            # natural gas
85            lp.LHnaturalGasFlow[t, 0] = lp.LHthermalLoad[t, 0]
86            lp.LHnaturalGasConsumption += lp.LHnaturalGasFlow[t, 0]/60
87            lp.totalNaturalGasFlow[t, 0] += lp.LHnaturalGasFlow[t, 0]
88            lp.totalNaturalGasConsumption += lp.LHnaturalGasFlow[t, 0]/60
89            # oxygen
90            lp.LHoxygenFlow[t, 0] = 0
91            lp.LHoxygenDemand += lp.LHoxygenFlow[t, 0]
92            lp.totalOxygenFlow[t, 0] += lp.LHoxygenFlow[t, 0]
93            lp.totalOxygenDemand += lp.LHoxygenFlow[t, 0]
94            # waste heat
95            lp.LHwasteHeatFlow[t, 0] = lp.LHthermalLoad[t, 0]*0.4
96            lp.LHwasteHeatFlow[t, 1] = 1050 + 273.15
97            lp.LHwasteHeatPotential += lp.LHwasteHeatFlow[t, 0]/60
98            lp.totalWasteHeatFlow[t, 0] += lp.LHwasteHeatFlow[t, 0]
99            lp.totalWasteHeatPotential += lp.LHwasteHeatFlow[t, 0]/60
100           # energy demand
101           lp.LHenergyDemand += lp.LHnaturalGasFlow[t, 0]/60
102           return lp
```

Vacuum Degassing

```
1   class VD(object):
2       def __init__(self, input):
3           self.Input = input
4
5       def load(self, t, lp):
6           for ladle in lp.processing:
7               if ladle.treatment == 'vd':
8                   lp.totalSteamFlow[t, 0] += self.Input.VDsteamFlow
9                   lp.VDsteamFlow[t, 0] += self.Input.VDsteamFlow
10          return lp
```

Vacuum Oxygen Decarburization

```
class VOD(object):
    def __init__(self, input):
        self.Input = input

    def load(self, t, lp):
        for ladle in lp.processing:
            if ladle.treatment == 'vod':
                # steam
                lp.totalSteamFlow[t, 0] += self.Input.VODsteamFlow
                lp.VODsteamFlow[t, 0] += self.Input.VODsteamFlow
                # oxygen
                lp.VODoxygenFlow[t, 0] += self.Input.VODspecificOxygenConsumption*self.Input.chargingWeightReal/
self.Input.VODprocessingTime
                lp.VODoxygenDemand += lp.VODoxygenFlow[t, 0]
                lp.totalOxygenFlow[t, 0] += lp.VODoxygenFlow[t, 0]
                lp.totalOxygenDemand += lp.VODoxygenFlow[t, 0]
        return lp
```

Steam Boiler

```
class SteamBoiler(object):
    def __init__(self, input):
        self.Input = input

    def load(self, t, lp):
        # steam
        lp.totalSteamDemand += lp.totalSteamFlow[t, 0]
        # thermal energy
        lp.steamThermalLoad[t, 0] = lp.totalSteamFlow[t, 0]*self.Input.steamFeedSpecificEnthalpy/60
        lp.steamThermalLoad[t, 1] = self.Input.steamFeedTemperature
        lp.steamThermalEnergyDemand += lp.totalSteamFlow[t, 0]*self.Input.steamFeedSpecificEnthalpy/3600
        lp.totalThermalLoad[t, 0] += lp.totalSteamFlow[t, 0]*self.Input.steamFeedSpecificEnthalpy/60
        lp.totalThermalEnergyDemand += lp.totalSteamFlow[t, 0]*self.Input.steamFeedSpecificEnthalpy/3600
        # natural gas
        lp.steamNaturalGasFlow[t, 0] = lp.totalSteamFlow[t,
0]*self.Input.steamFeedSpecificEnthalpy/self.Input.steamBoilerEfficiency/60
        lp.steamNaturalGasConsumption += lp.steamNaturalGasFlow[t, 0]/60
        lp.totalNaturalGasFlow[t, 0] += lp.totalSteamFlow[t,
0]*self.Input.steamFeedSpecificEnthalpy/self.Input.steamBoilerEfficiency/60
        lp.totalNaturalGasConsumption += lp.totalSteamFlow[t,
0]*self.Input.steamFeedSpecificEnthalpy/self.Input.steamBoilerEfficiency/3600
        # energy demand
        lp.steamEnergyDemand += lp.steamNaturalGasFlow[t, 0]/60
        return lp
```

Transition Probability Matrix

```
import numpy as np
import pandas as pd
import random
np.seterr(divide='ignore', invalid='ignore')

# Input
maxValue = int(4e7)
interval = int(1e6)
DATA_FILE = "Import_EAF.xlsx"
```

```
T_on = pd.read_excel(DATA_FILE, sheet_name="T_on")
T_off = pd.read_excel(DATA_FILE, sheet_name="T_off")
inputSequence1 = pd.read_excel(DATA_FILE,sheet_name="inputSequence1")
inputSequence2 = pd.read_excel(DATA_FILE,sheet_name="inputSequence2")

# 1. transition frequency matrix 1
def histc(inputSequence1, bins):
    map_to_bins = np.digitize(inputSequence1, bins)
    r = np.zeros(bins.shape)
    for i in map_to_bins:
        r[i-1] += 1
    return [r, map_to_bins]
if __name__=="__main__":
    inputSequence1 = np.array(inputSequence1)
    bins = np.arange(0, maxValue, interval)
    A, numbers1 = histc(inputSequence1, bins)
for steps in range(0, maxValue, interval):
    ticks = steps
zeros = (41,41)
frequencyMatrix = np.zeros(zeros)
probabilityMatrix = np.zeros(zeros)
oddNumbers = numbers1[0:-2]
evenNumbers = numbers1[1:-1]
for i in range(0, np.size(oddNumbers)): # 11x11 anstatt 10x10 Matrix da sonst axis out of bounds error?!
    frequencyMatrix[oddNumbers[i],evenNumbers[i]] = frequencyMatrix[oddNumbers[i],evenNumbers[i]] +1
frequencyMatrix = np.delete(frequencyMatrix, (0), axis=0)
frequencyMatrix = np.delete(frequencyMatrix, (0), axis=1)
for i in range(0, np.size(oddNumbers)):
    probabilityMatrix[oddNumbers[i], evenNumbers[i]] += 1
probabilityMatrix = np.delete(probabilityMatrix, (0), axis=0)
probabilityMatrix = np.delete(probabilityMatrix, (0), axis=1)
# 2. transition probability matrix 1
M = int(round(maxValue/interval))
frequencyMatrix_cum = frequencyMatrix
for j in range(1, M):
    frequencyMatrix_cum[:, j] = frequencyMatrix_cum[:, j-1] + frequencyMatrix[:, j]
M=M-1
for j in range(0, M):
    probabilityMatrix[:, j] = np.divide(probabilityMatrix[:, j], frequencyMatrix_cum[:, M])
# 3. cumulative transition probability matrix 1
probabilityMatrix_cum = probabilityMatrix
for j in range(1, M):
    probabilityMatrix_cum[:,j] = probabilityMatrix_cum[:,j-1] + probabilityMatrix[:,j]
TPM1 = probabilityMatrix_cum

# 2. transition frequency matrix 2
def histc(inputSequence2, bins):
    map_to_bins = np.digitize(inputSequence2,bins)
    r = np.zeros(bins.shape)
    for i in map_to_bins:
        r[i-1] += 1
    return [r, map_to_bins]
if __name__=="__main__":
    inputSequence2 = np.array(inputSequence2)
    bins = np.arange(0, maxValue, interval)
    [A,numbers2] = histc(inputSequence2, bins)
for steps in range(0, maxValue, interval):
    ticks = steps
zeros = (41,41)
frequencyMatrix = np.zeros(zeros)
probabilityMatrix = np.zeros(zeros)
oddNumbers = numbers2[0:-2]
```

```
evenNumbers = numbers2[1:-1]
for i in range(0, np.size(oddNumbers)):
    frequencyMatrix[oddNumbers[i],evenNumbers[i]] = frequencyMatrix[oddNumbers[i],evenNumbers[i]] + 1
frequencyMatrix = np.delete(frequencyMatrix, (0), axis=0)
frequencyMatrix = np.delete(frequencyMatrix, (0), axis=1)
for i in range(0, np.size(oddNumbers)):
    probabilityMatrix[oddNumbers[i], evenNumbers[i]] += 1
probabilityMatrix = np.delete(probabilityMatrix, (0), axis=0)
probabilityMatrix = np.delete(probabilityMatrix, (0), axis=1)
# 2. transition probability matrix 2
M = int(round(maxValue/interval))
frequencyMatrix_cum = frequencyMatrix
for j in range(1, M):
    frequencyMatrix_cum[:, j] = frequencyMatrix_cum[:, j-1] + frequencyMatrix[:, j]
M=M-1
for j in range(0, M):
    probabilityMatrix[:, j] = np.divide(probabilityMatrix[:, j], frequencyMatrix_cum[:, M])
# 3. cumulative transition probability matrix 2
probabilityMatrix_cum = probabilityMatrix
for j in range(1, M):
    probabilityMatrix_cum[:, j] = probabilityMatrix_cum[:, j-1] + probabilityMatrix[:, j]
TPM2 = probabilityMatrix_cum
TransitionProbabilityMatrix1 = pd.DataFrame(TPM1)
TransitionProbabilityMatrix2 = pd.DataFrame(TPM2)
writer = pd.ExcelWriter("TPM.xlsx")
TransitionProbabilityMatrix1.to_excel(writer, sheet_name="TPM1")
TransitionProbabilityMatrix2.to_excel(writer, sheet_name="TPM2")
writer.save()
```