BACHELORARBEIT

zur Erlangung des akademischen Grades

„Bachelor of Science in Engineering“ im Studiengang Mechatronik/Robotik

Neuentwicklung der Schlacke-auswurferkennung auf der Basis von MATLAB

Ausgeführt von: Andreas Kriegler
Personenkennzeichen: 1510330012

1. Begutachter: Dipl.-Chem. Dr. rer. nat. Olaf Nowitzki

Wien, 20.01.2018

FH University of Applied Sciences
TECHNIKUM WIEN

Eidesstattliche Erklärung

„Ich, als Autor / als Autorin und Urheber / Urheberin der vorliegenden Arbeit, bestätige mit meiner Unterschrift die Kenntnisnahme der einschlägigen urheber- und hochschulrechtlichen Bestimmungen (vgl. Urheberrechtsgesetz idgF sowie Satzungsteil Studienrechtliche Bestimmungen / Prüfungsordnung der FH Technikum Wien idgF).

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig angefertigt und Gedankengut jeglicher Art aus fremden sowie selbst verfassten Quellen zur Gänze zitiert habe. Ich bin mir bei Nachweis fehlender Eigen- und Selbstständigkeit sowie dem Nachweis eines Vorsatzes zur Erschleichung einer positiven Beurteilung dieser Arbeit der Konsequenzen bewusst, die von der Studiengangsleitung ausgesprochen werden können (vgl. Satzungsteil Studienrechtliche Bestimmungen / Prüfungsordnung der FH Technikum Wien idgF).

Weiters bestätige ich, dass ich die vorliegende Arbeit bis dato nicht veröffentlicht und weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt habe. Ich versichere, dass die abgegebene Version jener im Uploadtool entspricht.“

Ort, Datum	Unterschrift

Kurzfassung

Das weltweit am häufigsten verwendete Verfahren zur Stahlerzeugung findet mit Hilfe von sogenannten Linz-Donawitz Konvertern statt. In diesen Konvertern wird reiner Sauerstoff auf die Oberfläche von geschmolzenem Roheisen geblasen, um den Kohlenstoff und andere Begleitstoffe zu oxidieren und das Roheisen in Rohstahl umzuwandeln. Es ensteht Schlacke und Schaumschlacke, die aus dem Konverter austreten können. Damit dieser unerwünschte Schlackeauswurf verhindert werden kann, muss der/die BläserIn mit Ergebnissen eines Vorhersage- und Erkennungssystems versorgt werden, um Schritte setzen zu können, die einen Rückgang des Schlackepegels bewirken.

Dazu wurde vor mehreren Jahren in graphischer Programmierung (LabVIEW) ein optoakustisches System entwickelt und im Stahlwerk der Salzgitter AG installiert, welches jedoch sowohl hinsichtlich der Berechnung als auch der Visualisierung, als unzureichend erwiesen hat. Die Vorhersage und Erkennung eines Schlackeauswurfs in diesem System ist unzuverlässig und die Mensch-Maschine Schnittstelle (MMS) entspricht nicht den neuen, hausinternen Gestaltungsrichtlinien (GR). Darum findet in dieser Arbeit eine Neuentwicklung des Systems statt, um dem/der BläserIn eine aussagekräftigere Schnittstelle mit dem Prozess zu bieten, die mehrere Ansätze zur Vorhersage und Erkennung eines Schlackeauswurfs vereint.

Dafür wird auf der Basis eines hauseigenen, objektorientierten MATLAB-Programmiergerüsts (PG) eine komplette Mensch-Maschine Schnittstelle entwickelt, die nun auch den Gestaltungsrichtlinien entspricht. Zahlreiche Schnittstellen mit unterschiedlichen Komponenten des Systems, wie zwei Mikrofonen, einer Kamera, der Speicher-programmierbaren Steuerung und einer Datenbank, werden zuerst entworfen, dann entwickelt und abschließend simuliert und getestet. Darüber hinaus wird ein komplexer Algorithmus, der statische und dynamische Prozessdaten analysiert, um einen Schlackeauswurf vorherzusagen, in das System integriert. Ein Feldtest im Stahlwerk war nicht mehr im Rahmen dieser Arbeit durchführbar, dennoch haben sich die Offline-Simulationen als vielversprechend erwiesen. Die optoakustischen Funktionalitäten des alten Systems konnten erhalten bleiben und verbessert werden. Der Datenanalyse-Algorithmus konnte einen Schlackeauswurf in mehr als 85 von 100 analysierten Schmelzen korrekt voraussagen.

Schlagwörter: LD-Konverter, Schlackeauswurf, MATLAB-MMS, Datenanalyse, Audiosignalverarbeitung

Abstract

Basic Oxygen Steelmaking is the most popular method of steelmaking in the world. It uses Basic Oxygen Furnaces (BOFs for short) to blow pure oxygen at supersonic speed onto the surface of the molten pig iron to oxidize the carbon and create steel. During this process slag is being created, which fills the converter. When it cannot be contained in the vessel any longer it is forced out through the opening at the top. This event is called slopping and it pollutes the surrounding environment and reduces the amount of steel yielded by the process. To avoid slopping, the person in charge of the process, refered to as the blower, needs to be provided with accurate results of a slopping-forecast and slopping-detection system. They can then take actions to lower the level of slag within the vessel.

Such a system has been developed in graphical programming (LabVIEW) numerous years ago and it displays analysis data from optoacoustical measurement devices. Nonetheless, this system is nowhere near the current state-of-the-art regarding both the Frontend and Backend functionalities. The slopping-prediction is unreliable, and the human-machine interface (HMI) does not follow the new companywide styleguide. Therefore, a new system is being developed in this thesis, which provides the blower with an intuitive User Interface of the process. Furthermore, multiple techniques for slopping-prediction and detection are combined in this system.

An in-house MATLAB-framework is used for the basis of the program, in which modern object-oriented programming techniques are prevalent. Interfaces with various components of the system, such as two microphones, a camera, the Programmable Logic Controller (PLC) and a database, are being developed and simulated. Following the styleguide, a clearly arranged HMI is created and thoroughly tested. A complex machine-learning algorithm, which uses both static and dynamic process data, is integrated into the system. Even though the system could not be tested in real plant conditions, the simulations and results were promising. With this algorithm, slopping could be detected in more than 85 out of a 100 analyzed heats.

Keywords: BOF, Slopping Detection, MATLAB-HMI, Data Analysis, Acoustic Signal Processing

Danksagung

Ich möchte zuerst meinem Betreuer in der SMS group GmbH *Jörg Thomasberger* für die Möglichkeit danken, dass ich ein Teil dieses Projektes werden konnte und ich bin dankbar für die tatkräftige Unterstützung, die ich im Zuge der letzten 6 Monate von ihm erhielt. Weiterhin danke ich meinem Betreuer an der Fachhochschule FH Technikum Wien *Olaf Nowitzki* für die Orientierung und Zielsetzung, die maßgeblich zur Fertigstellung dieser Arbeit beigetragen haben. Ohne der intellektuellen Freiheit, die mir erlaubt wurde und der gleichzeitigen Forderung nach einer qualitativ hochwertigen Entwicklung und Bachelorarbeit, wäre die Fertigstellung in diesem Umfang nicht möglich gewesen.

In diesem Sinne möchte ich besonders *Felix Fanghänel* in der SMS group GmbH, dem Autor des verwendeten Programmiergerüsts, auf dem die gesamte Arbeit aufbaut, danken. Ohne seiner Entwicklungsarbeit und programmiertechnischen Unterstützung und Finesse wäre diese Arbeit niemals möglich gewesen.

Zuletzt möchte ich mich bei meiner Familie und meinen Freunden bedanken, die mich sowohl von der Ferne, als auch durch Besuche in Düsseldorf, unterstützten. Nur durch deren motivierende Worte und Zuneigung konnten sämtliche Hindernisse überwunden werden und ich bin ihnen dafür für immer dankbar.

Inhaltsverzeichnis

1 Einleitung

1.1 Stahlerzeugung und Schlackebildung

Das Linz-Donawitz-Verfahren (auch LD-Verfahren) ist das weltweit meistverwendete Verfahren zur Stahlerzeugung (Worldsteel, 2015). Beim LD-Verfahren wird geschmolzenes, kohlenstoffreiches Roheisen durch Aufblasen von reinem Sauerstoff, zu kohlenstoffarmem Rohstahl umgewandelt. Dieser Vorgang wird auch als Frischen bezeichnet und dauert 15 bis 20 Minuten, wobei zu Beginn des Vorgangs die Schlackebildung am heftigsten ist. Die Schlacke bildet sich an der Oberfläche des Roheisenbades und die entstehende Metall-Schlacke–Emulsion füllt häufig den gesamten Raum des Konverters, wodurch flüssige und/oder feste Schlacke aus dem Konverter ausgeworfen wird (Lange, 1982).

1.2 Alte LabVIEW – Mensch-Maschine Schnittstelle

Um diesen Schlackeauswurf verhindern zu können, wurde vor mehreren Jahren ein System in graphischer Programmiersprache (LabVIEW) entwickelt und im Stahlwerk Salzgitter installiert. Das System umfasst neben der Visualisierung zwei Kernkomponenten: Die Verarbeitung der Signale von zwei Mikrofonen, die nahe des Konverters und im Abgasschacht darüber angebracht sind, sowie die Einbindung eines Videoanalysesystems, das Bilder einer Kamera, die auf den Konvertermund und die äußere Konverterwand gerichtet ist, verarbeitet. Der Ansatz des akustischen Teils beruht darauf, dass durch das Ansteigen des Schlackepegels der emittierte Schall der Sauerstofflanze abgedämpft wird (Evestedt & Medvedev, 2009). Die Kamera wurde installiert, um aufschäumende, aus dem Konverter austretende Schlacke aufzeichnen zu können. Auch dieser Ansatz ist üblich, um einen Schlackeauswurf zu erkennen, wie z.B. in Kattenbelt et al. (2008).

1.3 Motivation zur Neuentwicklung

Die Praxis hat jedoch gezeigt, dass beide Komponenten des Systems Defizite aufweisen. Um den Konverterprozess überhaupt erst effektiv zu machen, wird eine große Menge an Kohlenstoffmonoxid bzw. Metall-Gas-Schlacke–Emulsion benötigt (Evestedt et al., 2007). Dazu ist es sinnvoll, den Stellring zu senken, um eine Abschottung des Konverterinnerens von der Umgebungsatmosphäre zu garantieren. Der Stellring wird aber oftmals auch angehoben, um einen Blick ins Innere des Konverters zu ermöglichen. Bei jeder Bewegung des Stellrings ändert sich aber die gesamte Geräuschkulisse. Außerdem erfasst die Kamera zu viel des Feuers, das aus dem Konverter austritt. Deswegen hat Ghosh (2017) einen weiteren Ansatz getroffen: Er hat ein Modell entwickelt, das statische und dynamische Prozessvariablen verwendet, um einen Schlackauswurf vorherzusagen. Dieser dritte Ansatz, ein komplexer Algorithmus, der auf neuronalen Netzen und dem Ansatz des tiefgehenden Lernens aufbaut, wurde somit entwickelt und benötigt nun eine Integration in eine neue, moderne MMS die außerdem den aktuellen, firmeninternen Gestaltungsrichtlinien entspricht.

1.4 Umfang und Ziele der Arbeit

Der Zweck dieser Arbeit ist es, eine Mensch-Maschine Schnittstelle zu entwickeln, die unterschiedliche Ansätze zur Vorhersage und Erkennung eines Schlackeauswurfs vereint. Die MMS soll hinsichtlich der verwendeten Programmiersprache, der Visualisierung, der Netzwerkschnittstellen, der Alarmgenerierung und Speicherung von Prozessdaten dem aktuellen Stand der Technik entsprechen. Somit werden konkret folgene Ziele gesetzt:

- Entwicklung einer MMS unter Verwendung des hauseigenen MATLAB-Programmiergerüsts. Diese MMS soll modernen, hauseigenen Richtlinien (X-Pact Vision 2015) zur Gestaltung einer MMS entsprechen.
- Integration einer Schnittstelle, die mittels OPC-DA Daten von der SPS empfängt.
- Einbindung der Mikrofone und Entwicklung einer Audiosignalverarbeitung, um aktuelle Informationen über die Frequenzspektren der Mikrofone zu erhalten.
- Entwicklung einer Schnittstelle mit dem Videoanalysesystem durch UDP-Telegramme.
- Entwicklung eines Konfigurationsmoduls zur Anpassung von Systemparametern.
- Entwicklung einer HTML-basierten minimalistischen MMS, um eine Einbindung der MMS in die InTouch-MMS des Stahlwerks zu ermöglichen.
- Entwicklung eines Alarmsystems, das interaktive Meldungen und Dialoge generiert.
- Entwurf und Entwicklung einer MySQL-Datenbank inklusive Verschlüsselung, in der Prozessdaten systematisch gespeichert und abgerufen werden können.
- Integration des Python-basierten Algorithmus zur Schlackeauswurfvorhersage via Transfertabellen und Visualisierung dessen Ergebnisse.

Da das System mehrere Komponenten zusammenführt, die nicht im Zuge dieser Arbeit entwickelt wurden, sind folgene Elemente nicht oder nur minimal Teil dieser Arbeit:

- Exakter Inhalt der urheberrechtlich geschützten Gestaltungsrichtlinien.
- Entwicklung des MATLAB-PG. Dieses wird nur projektspezifisch umgesetzt.
- „Flex-OPC-Server“ sowie „OPC-Copy“ werden ebenfalls nur verwendet.
- Das Videoanalysesystem wird lediglich durch Schnittstellen eingebunden.
- Die Entwicklung des Python-Algorithmus hat durch Ghosh (2017) stattgefunden.

1.5 Gliederung der Arbeit

In Abschnitt 2 werden Konverterprozesse erklärt und Ansätze zur Vermeidung eines Schlacke-auswurfs beschrieben. Standards zur objektorientierten Programmierung in MATLAB werden festgehalten. Es folgt ein Überblick über die hauseigenen Gestaltungsrichtlinien und das verwendete MATLAB-PG. In Abschnitt 3 wird ein Überblick über das System geliefert und das PG umgesetzt. In Abschnitt 4 wird die OPC-Kommunikation beschrieben. In Abschnitt 5 wird die Audiosignalverarbeitung und Einbindung des Videosystems beschrieben. In Abschnitt 6 wird ein Alarmsystem entwickelt. Abschnitt 7 beinhaltet die Erstellung der MySQL-Datenbank und Einbindung des Python-Algorithmus. Abschnitt 8 liefert einen Überblick über den zyklischen Programmablauf des Systems. In Abschnitt 9 und 10 werden die erzielten Ergebnisse diskutiert und mögliche Erweiterungen und Verbesserungen des Systems dargestellt.

2 Theorie und Stand der Technik

2.1 Konverterprozesse und Schlackedetektion

2.1.1 Schlackebildung

Laut Worldsteel (2015) wurden im Jahre 2015 1,2 Milliarden Tonnen, das sind mehr als 74% der gesamten Rohstahlerzeugung, in LD-Konvertern gefertigt. Vor allem in der Volksrepublik China, dem größten Stahlproduzenten der Welt, wird das LD-Verfahren dem Verfahren mittels Lichtbogenöfen weitgehend bevorzugt (Worldsteel 2015). Der LD-Konverter wird zu Beginn des Vorgangs mit geschmolzenem Eisen und Schrott verschiedenster Sorten gefüllt. Anschließend wird mit einer Sauerstofflanze mit Überschallgeschwindigkeit reiner Sauerstoff auf die Oberfläche des Roheisenbades geblasen. Beim Aufblasen des Sauerstoffs oxidiert der gebundene Kohlenstoff und es entsteht Kohlenmonoxid. Das Kohlenmonoxid steigt in Form von Blasen in der Schmelze auf und an der Oberfläche bildet sich flüssige Schlacke. Es wird gebrannter Kalk (ein Schlackebildungskatalysator) beigemengt, um die Bildung von Schlacke zu fördern - dadurch wird der Stahl von Unreinheiten befreit (Evestedt et al., 2007).

Damit sich Schaumschlacke bilden kann, muss erst genügend flüssige Schlacke vorhanden sein und die Entkohlung muss weit genug fortgeschritten sein; dies ist meist nach 5 bis 7 Minuten der Fall (Lange, 1982). Wie Lange (1982) beschrieben hat, stellt die enstehende Schaumschlacke eine „komplexe Mischung aus flüssigen Metalltropfen und Gasblasen, die in flüssiger Schlacke dispergiert sind“ dar. Zu diesem Zeitpunkt ist die Gefahr eines Schlackeauswurfs am größten. Dieser Vorgang ist in Abbildung 1 dargestellt. Es ist anzumerken, dass ein so eindeutiger Phasenübergang nicht der Realität entspricht (Lange, 1982). Außerdem wird häufig der gesamte Konverterraum mit Schlacke bzw. Schaumschlacke gefüllt, was den Auswurf verursacht.

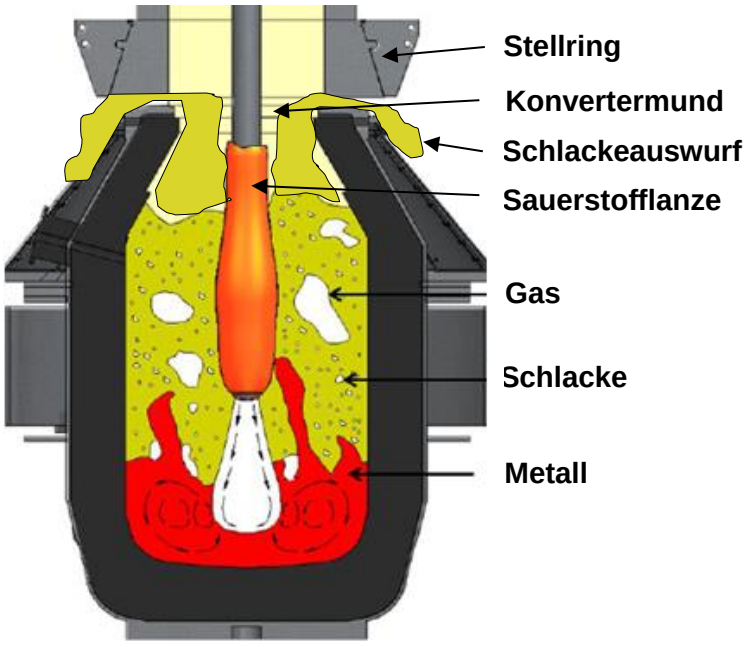


Abbildung 1: Schema des LD-Verfahrens mit Sauerstofflanze (Quelle: modifiziert übernommen aus (Ghosh, 2017), S.22).

2.1.2 Probleme der Schlackebildung

Der Schlackeauswurf ist trotz umfassender Forschung weiterhin ein Problem in der Stahlerzeugung mit LD-Konvertern. Damit das LD-Verfahren überhaupt erst effektiv wird, ist eine große Menge an Schlacke bzw. Schaumschlacke im Konverter notwendig, wobei die Menge durch die Größe des Konverters limitiert ist (Evestedt et al., 2007). Somit kommt auch ein psychologischer Faktor ins Spiel: Der/Die BläserIn will einen Auswurf der Schlacke vermeiden; zu vorsichtiges Verfahren während des Frischens mindert jedoch die Menge und Güte des produzierten Stahls. Außerdem erschweren es die rauen Verhältnisse eines Stahlwerks, ein verlässliches und akkurates System zu entwickeln, das dem/der BläserIn ermöglicht, adäquate Maßnahmen zur Vermeidung eines Auswurfs zu treffen.
Dass das Phänomen des Schlackeauswurfs ein hochdynamisches und –komplexes ist, zeigte auch Walker et al. (2005) mit einer Liste von relevanten Variablen. Ein Ausschnitt dieser Liste und eine mathematische Modellierung des Vorgangs durch Ghag et al. (1998), Birk et al. (2001) und Ito & Fruehan (1989), werden in Anhang A - Mathematische Hintergründe erörtert.

2.1.3 Verfahren zur Vorhersage und Detektion eines Auswurfs

Um diesem Problem entgegenzuwirken, wurden über die Jahrzehnte unterschiedliche Ansätze gesetzt. Diese können zu einer der folgenden Kategorien gezählt werden:

a.) Modelle, die Prozessvariablen verwenden und auf Algorithmen aufbauen, um einen Schlackeauswurf vorherzusagen.
b.) Sensorik und Erfassungsgeräte, die die Schlackebildung erkennen, wie Kameras, Schallmesser und Lanzenvibrationsmessgeräte.
c.) Methoden, die in Echtzeit gesetzt werden, um der Schlackebildung entgegenzuwirken.

Als Beispiele für Verfahren der Gruppe **a.)** sind Chukwulebe et al. (2004) und Shakirov et al. (2004) zu nennen. Chukwulebe et al. (2004) haben ein optimals Blasprofil aus der ursprünglichen chemischen Zusammensetzung berechnet und einen Echtzeitabgleich gemacht. Shakirov et al. (2004) haben ein System entwickelt, das für jede Schmelze eine Auswurfwahrscheinlichkeit berechnet.

Verfahren der Gruppe **b.)** finden am häufigsten Verwendung, diese können jedoch in ihrer Ausführung sehr unterschiedlich sein. Beispielsweise wurde ein Radargerät von Doe (1984) verwendet, um die Höhe des Schlackepegels relativ zum Konvertermund zu messen. Systeme, die durch Analyse der Lanzenvibration Rückschlüsse auf den Schlackepegel liefern, finden ebenfalls häufig Verwendung, wie zum Beispiel in Pak et al. (1996) und Walker et al. (2005) beschrieben wird.

Methoden der Gruppe **c.)** sind andererseits Schritte, die der/die AnlagenbedienerIn während des Frischens setzen kann, wenn ein Schlackeauswurf droht. Im Spektrum dieser Arbeit und beim Stahlwerk Salzgitter AG sind das konkret: Das Hochfahren des Stellrings, das Senken der Sauerstofflanze im Konverter und das Senken der Sauerstoffdurchflussrate durch die Lanze.

Systeme kombinieren jedoch oftmals verschiedene Ansätze, um einen Schlackeauswurf erkennen bzw. vermeiden zu können. Beispielsweise wurde in Kattenbelt et al. (2008) eine Bildverarbeitung implementiert, die Bilder einer Kamera, die auf den Konvertermund gerichtet ist, verarbeitet. Allerdings wurden in Kattenbelt et al. (2008) auch lineare, statistische Modelle, jedoch keine Maschinelles-Lernen Methode, verwendet.

Im IMPHOS-Projekt von Tata Steel Europe (CORDIS, 2009) wurden sowohl Schallmesser als auch Videokameras verwendet, aber Algorithmen, die Prozessdaten verwenden, wurden nicht implementiert.

In 2014 veröffentliche die Europäische Kommission (P. O. o. t. E., 2014) die Resultate eines weiteren Projekts zur Kontrolle der Schlackebildung in LD-Konvertern. Die Implementierung von akustischen und optischen Messmethoden fand in mehreren Stahlwerken von Tata Steel und Arcelor in Europa statt. Die gesammelten Daten wurden anschließend mit statistischen Methoden analysiert und in manchen Fällen resultierte dies zu einer erfolgreichen Vorhersage des Schlackeauswurfs.

Die meisten akustischen Verfahren verwenden zumindest einen Schallpegelmesser oder ein Mikrofon im Abgasschacht, um ein Audiosignal zu erfassen. Die Hauptschallquelle sind die Düsen der Sauerstofflanze. Es wird auch eine Messung der Abgasrate benötigt. Der beim Sauerstoffaufblasen entstehende Schaum absorbiert den Schall und reduziert somit den Schallpegel (Birk et al., 2001). Dadurch kann man über den Verlauf des Schallpegels Rückschlüsse auf den Pegel der Schaumschlacke ziehen. Ein Verfahren zur Berechnung des Schaumschlackepegels, unter der Verwendung von Frequenzanalyse mit Ergebnissen von Ingard (1994), wurde von Birk et al. (2001) entwickelt und ist in Anhang A - Mathematische Hintergründe beschrieben.

2.2 Objektorientierte (OO) Programmierung in MATLAB

In Bezug auf OO-Programmierung wurde mit der MATLAB Version 2008a ein großer Schritt gesetzt, um näher an OO-Standards von Java, Python und C++ zu liegen. (Murphy, 2009).

2.2.1 Objekt vs Struktur vs Klasse

Mit Klasse ist eine Sammlung von Code gemeint, die sich durch eine Reihe von ähnlichen Attributen und Methoden kategorisch vom Rest des Programms unterscheidet. Eine Instanz dieser Klasse wird als Objekt bezeichnet. Diese Objekte unterscheiden sich von einer gewöhnlichen Struktur durch folgende Merkmale (Murphy, 2009):

- Sowohl Strukturen als auch Objekte speichern Daten in benannten Feldern (Attribute), aber Objekte beinhalten ebenfalls Operationen, die auf diese Attribute angewendet werden können (Methoden).
- Feldernamen bei Strukturen sind frei wählbar, alle Objekte einer Klasse haben aber die selben Attribute. Der Wert eines Attributs kann sich zwischen Objekten unterscheiden.
- Strukturen werden mit *struct()* kreiert, Klassen verwenden hingegen Konstruktor-Methoden und eine eigene MATLAB – .m-Datei um Objekte zu erstellen.

2.2.2 Klassenkonstruktor

Das Schlüsselwort *classdef* gefolgt von <className> wird verwendet, um eine Klasse in einer .m-Datei mit dem Namen <className> zu definieren. Anschließend folgt eine Definition und gegebenfalls Initialisierung sämtlicher Attribute der Klasse. In den Methoden muss zumindest eine Konstruktor-Methode definiert werden, die den selben Namen wie <className> trägt und für die Erstellung von Objekten der Klasse verantwortlich ist. Dieser Konstruktor kann mehrere Argumente zur Initialisierung einzelner Attribute entgegennehmen und gibt ein Argument, das konstruierte Objekt, zurück.

2.2.3 Attribute

Um eine Trennung von Implementierung und Anwender-Schnittstelle erzielen zu können, ist eine Beschränkung der Klassen sinnvoll, die über Lese– und Schreibrechte, auf Attribute von bestimmten Klassen, verfügen (Murphy, 2009). In MATLAB sind dazu mehrere Stufen von Kontrolle implementiert, die sowohl für Lese- als auch Schreibrechte angewendet werden können. Diese umfassen **Private**, **Protected**, **Public** sowie **Constant** und **Hidden**.
Private Attribute sind Attribute, die lediglich von Methoden der selben Klasse manipuliert werden können.
Protected Attribute sind ähnlich den Private Attributen, mit dem Unterschied, dass sie auch von Subklassen der Klasse manipuliert werden können.
Public Attribute sind für alle Methoden und Funktionen am MATLAB-Pfad zugänglich.
Constant Attribute sind konstant und können nicht im Programmablauf verändert werden.
Hidden Attribute scheinen nicht auf, wenn das Objekt mit *properties()* betrachtet wird.

2.2.4 Methoden

Methoden einer Klasse sind Operationen, die Objekte dieser Klasse (oder auch anderer Klassen) manipulieren. Jede Methode, abgesehen von der Konstruktor-Methode, nimmt zumindest ein Eingangsargument, das Objekt der Klasse, in der sich die Methode befindet, entgegen. Dieses Argument muss beim Aufrufen der Methode nicht explizit übergeben werden. *Nargin(),* eine MATLAB-Funktion die die Anzahl der Eingangsargumente ermittelt, gibt somit in jedem Fall zumindest eins zurück. Beispielsweise sind die Aufrufe

```
obj.init();
```

und

```
init(obj);
```

funktionell ident. Altman (2014) erkannte jedoch, dass die zweite Variante wesentlich performanter ist: sie ist um ca. 33% schneller.

Methoden können genauso wie Attribute **Private, Public** oder **Protected** sein. Es ist üblich, eine Public-Methode in eine Reihe von Private-Methoden aufzuspalten, um den Überblick zu bewahren. Darüber hinaus gibt es **Static** Methoden. Diese Methoden sind mit einer Klasse, anstatt konkreten Objekten dieser Klasse, verbunden. Somit verwenden sie keine Objekte der Klassen und brauchen keine Eingangsargumente.

2.2.5 Vererbung und Hierarchie

Klassen können Attribute und Methoden verwenden, die in Wirklichkeit eine spezielle Ausführung von Attributen und Methoden anderer Klassen sind. Um eine Hierarchie in diese entstehende Struktur zu bringen, kann mit dem Ansatz der Vererbung gearbeitet werden.
Dazu können **Subklassen** erstellt werden, die sämtliche Attribute und Methoden einer **Superklasse** beinhalten und diese um spezielle Funktionalität erweitern. Methoden können überladen werden, in dem eine Methode, die in einer Superklasse deklariert ist, in der Subklasse unter dem selben Namen anders umgesetzt wird. Um eine solche Subklassen-Beziehung aufzubauen, muss die Definitionszeile der Klasse erweitert werden:

classdef <className> < <superclassName>

Im Allgemeinen werden in MATLAB Objekte per Wert, und nicht Referenz, weitergegeben. Das bedeutet, dass bei jedem Aufruf einer Methode eine komplette Kopie des Objekts übergeben und manipuliert wird - nicht das ursprüngliche Objekt. Um dies zu umgehen, ist in MATLAB eine Superklasse definiert, die das Übergeben von Objekten per Referenz anstatt des Wertes, ermöglicht. Diese Superklasse ist die **Handle-Klasse**. Durch Definition von Klassen als Subklassen der Handle-Klasse, z.B.:

classdef mydate < *handle*

beziehen sich nun sämtliche Aufrufe auf ein und dasselbe mydate-Objekt und es werden keine lokalen Kopien erstellt. Dadurch können in einer Klasse Handle-Attribute definiert werden, die Referenzen auf Objekte unterschiedlicher Klassen und somit sämtlicher Methoden und Attribute dieser Klassen, halten. Um nun Methoden dieser Objekte aufrufen zu können, wird lediglich das entsprechende Handle-Attribut benötigt. Dadurch kann ein komplexes Netz von Klassen mit zahlreichen Abhängigkeiten strukturiert aufgegliedert werden.

2.3 Standards zu MMS– und Datenbank-Design sowie dem MATLAB-Programmiergerüst

2.3.1 Firmeninterne Gestaltungsrichtlinien

Zur Erstellung der MMS werden hauseigene Gestaltungsrichtlinien (GR), der X-Pact Vision Styleguide, verwendet. Diese Richtlinien sind urheberrechtlich geschützt und folgen einer Geheimhaltungsverpflichtung laut [DIN ISO 16016]. Sämtliche Informationen in Bezug auf Platzierung von Elementen, Prozessnavigation, Schriftart und -größe, Farbschemata, UI-Elemente, Icons und Beispielbilder sind somit unter Verschluss. An Hand eines exemplarischen Entwurfs, der den Gestaltungsrichlinien vorausgeht, können die wichtigsten Elemente dennoch erläutert werden. Dieser Entwurf ist in Abbildung 2 auf der nächsten Seite zu sehen.

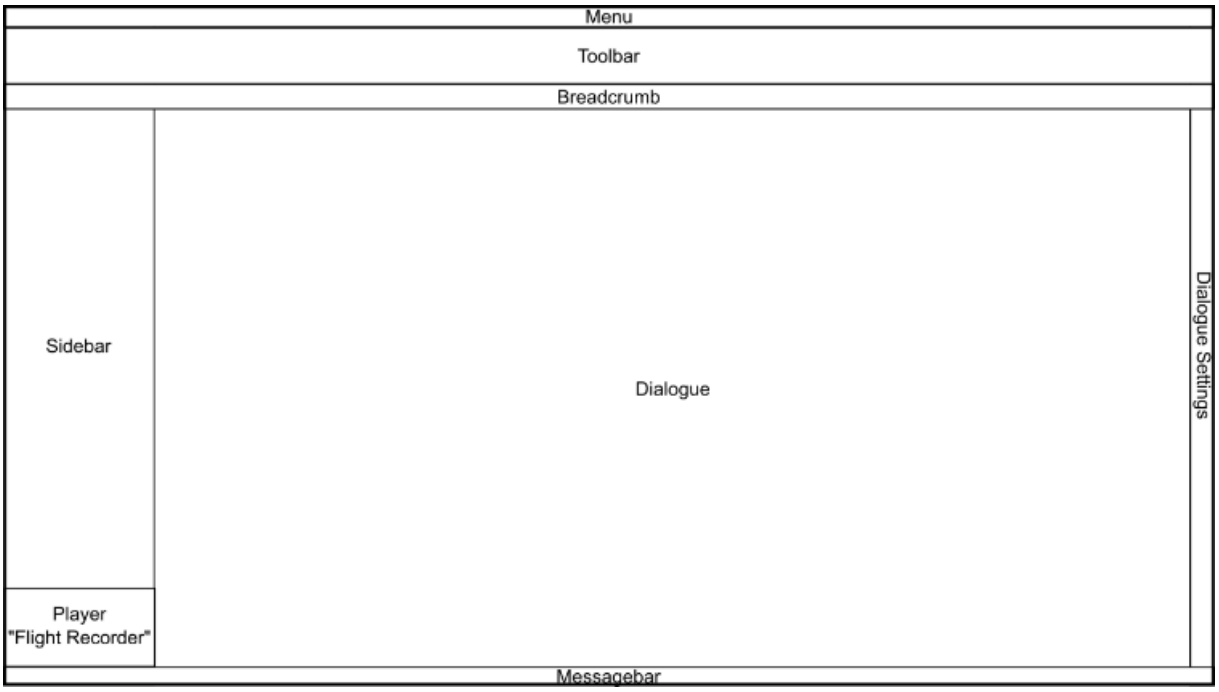


Abbildung 2: Layout einer MMS ähnlich den Gestaltungsrichtlinien.

Elemente der MMS

Die **Menüleiste** am oberen Rand umfasst auf linker Seite die Navigation zu einzelnen Prozessbildern und auf der rechten Seite verschiedene Einstellungen und Optionen.

Die **Symbolleiste** enthält das Logo der SMS group GmbH auf der linken Seite, in der Mitte eine kleine Tabelle, mit Daten zur aktuellen und nächsten Schmelze und auf der rechten Seite das Logo des Kunden sowie Datum und Uhrzeit.

Die **Brotkrumenleiste** bietet einen Überblick über die Navigation zwischen den einzelnen Prozessabbildern und beinhaltet Statusanzeigen für die wichtigsten Prozessvariablen.

Die **Seitenleiste** bietet Platz für sogenannte D1-Daten. Das sind die wichtigsten Daten des Systems, die zu jedem Augenblick sichtbar sein sollten.

Die **Nachrichtenleiste** enthält Statusmeldungen zum Prozessgeschehen, kann vergrößert werden, um Meldungen zu betrachten, und folgt einem Farbschema zur Identifikation der Meldungen.

Der Dialog oder **D2-Bereich** beinhaltet mehrere Prozessabbilder zur Visualisierung des Prozessgeschehens und beinhaltet Daten der Datenklasse D2. Diese werden zu einem Prozessabbild zusammengefasst und haben nicht die globale Bedeutung von Daten der Klasse D1. Dennoch ist die Visualisierung im Bereich D2 ausschlaggebend für das Prozessverständnis. Es wird immer nur ein Prozessabbild gezeigt und durch die Navigation kann dieses gewechselt werden. Der D2-Bereich ist der einzige Bereich, dessen Inhalt in Laufzeit verändert werden kann (abgesehen von den Meldungen in der Nachrichtenleiste).

Am rechten Rand befindet sich der **D3-Bereich** für Daten der Datenklasse D3. Das sind Daten, die nicht abhängig vom Prozessabbild sind, also eine allgemeine Bedeutung haben, deren Wichtigkeit jedoch geringer als die der D1-Daten ist. Eine Implementierung dieses Bereichs ist optional und kann beispielsweise als ausklappbares Fenster stattfinden.

2.3.2 MATLAB-Programmiergerüst

Die GR sind plattformunabhängig, sollen somit in WinCC, MATLAB, LabVIEW und anderen Programmen umgesetzt werden können. Zur Umsetzung in MATLAB wurde ein PG entwickelt, das als Sammlung von Handle-Klassen und supplementären Funktionen zu verstehen ist. Aufbauend auf diesem PG können projektspezifische Klassen und Funktionen erstellt werden. Durch Versionierung des Codes ist es möglich, Änderungen im PG problemlos zu übernehmen. Die Klassen des PG können in mehrere Kategorien unterteilt werden:

- UI-Klassen, die sich mit der Erstellung von MMS-Elementen befassen. Dabei sind Klassen für alle in Abschnitt 2.3.1 erwähnten Elemente vorhanden, mit Ausnahme des D2-Bereichs. Dieser ist projektspezifisch und muss daher entwickelt werden.
- Manager-Klassen, die zur Überwachung und Verarbeitung von ausgelösten Ereignissen während des Programmablaufs zuständig sind.
- Hilfsklassen, die klassenübergreifende Hilfsfunktionen beinhalten.

Das Programmiergerüst weicht jedoch etwas von den Gestaltungsrichtlinien ab:

- Die Navigation findet nicht per Menüleiste, sondern über Icons in der Symbolleiste statt.
- Die Menüleiste ist an das MATLAB-Fenster gebunden und ist somit nicht Teil der MMS.
- Die Pfeiltasten in der Brotkrumenleiste zur Prozessnavigation haben keine Funktion.
- Schriftarten und -größen stimmen nicht exakt überein.
- Es ist keine Fußzeile implementiert, die weitere Optionen ermöglicht.
- Bestimmte Farbübergänge und Trennlinien sind ebenfalls nicht exakt.

Abbildung 3 zeigt beispielsweise ein Prozessabbild einer MMS, die mit Hilfe des MATLAB-PG erstellt wurde. Abweichungen von den Konventionen aus den Gestaltungsrichtlinien müssen vermieden werden, um eine Konsistenz der MMS in allen Stahlwerken zu wahren.

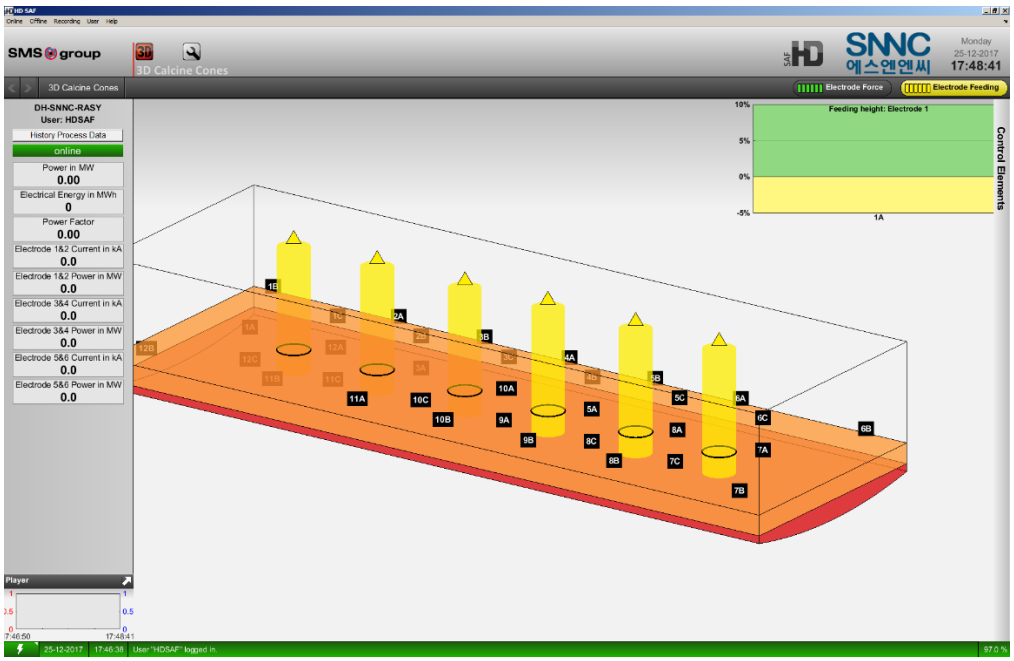


Abbildung 3: Prozessabbild einer MMS, die mit dem MATLAB-Programmiergerüst entwickelt wurde.

Das PG benötigt standardmäßig zumindest folgende Komponenten: einen Ordner mit Icons, mehrere .ini-Konfigurationsdateien für Grenzwerte und Serverinformationen, eine Sprachtabelle, die Zeichenfolgen in mehreren Sprachen enthält, eine Namensdatei, die Variablennamen am OPC-Server sowie im MATLAB-Arbeitsbereich definiert, ein hauseigener Flex OPC-Server und die OPC-Copy-Funktion, die Daten von einem OPC-Server zu einem anderen via OPC-DA zyklisch kopiert. Zur Einrichtung und Simulation des Servers wird außerdem der *MatrikonOPC Explorer* und die MATLAB-OPC-Toolbox benötigt.

2.3.3 Datenbank-Design

Für die Speicherung von Konfigurations- und Prozessdaten können verschiedene Datenbanken verwendet werden. Die Wahl der Datenbank ist von Projekt zu Projekt unterschiedlich. MySQL-Datenbanken sind weit verbreitet, die Dokumentation ist ausführlich und sie sind einfach zu entwerfen und zu entwickeln. Deswegen wird in dieser Arbeit eine MySQL-Datenbank verwendet. Es gibt wiederum firmeninterne Richtlinien, nach denen benötigte Datenbanken entworfen und entwickelt werden müssen. Diese Richtlinien umfassen die Verschlüsselung einer relationalen Datenbank, die Sortierung sämtlicher Variablen in passenden Tabellen, die Bezeichnungen von Variablen und Tabellen sowie die Einheit, in denen die Werte gespeichert werden.

2.3.3.1 Relationale Datenbanken: Primary und Foreign Keys

Datenbanken werden als relational bezeichnet, wenn sie nach dem relationalen Modell organisiert sind. Das relationale Modell wiederum baut auf der von E. Codd formulierten Prädikatenlogik auf. In Bezug auf Datenbanken bedeutet das, dass alle Daten in Spalten, Reihen und Tabellen sortiert werden. Jede Reihe beinhaltet dabei einen oder mehrere eindeutige Werte. Mit Hilfe dieser Werte können unterschiedliche Tabellen einfach verknüpft und Benutzeranfragen schnell bearbeitet werden. Dazu wird das Konzept von Primary und Foreign Keys (PK und FK) verwendet.

Als **Primary Key** wird eine Spalte oder Gruppe von Spalten bezeichnet, deren Eintrag in jeder Reihe die jeweilige Reihe eindeutig definiert. Dazu sind alle Einträge in einer PK-Spalte einzigartig, es können nicht zwei Reihen mit dem selben Eintrag in der PK-Spalte existieren.

Wenn dieselbe Spalte oder Gruppe von Spalten in einer anderen Tabelle vorkommt, wird sie dort als **Foreign Key** bezeichnet. Foreign Keys kann man sich wie Zeiger vorstellen, die auf den PK einer anderen Tabelle zeigen. Die Werte, die dabei in den Reihen von Foreign Key-Spalten stehen können, sind dabei durch die Werte im referenzierten Primary Key beschränkt.

Beispielsweise kann in einer Tabelle mit Materialdaten für Holzstühle der Primary Key die Produktnummer sein. In einer zweiten Tabelle mit monetären Daten für diese Holzstühle kann ein Foreign Key die selbe Spalte mit Produktnummern sein. Die Tabellen und somit etwaige weitere Spalten werden dadurch verknüpft.

Laut Richtlinien ist der wichtigste interne PK die Produktnummer oder PRODUCT_NO. Jede Schmelze hat eine eindeutige Schmelzen-ID, die vom Stahlwerksbetreiber definiert wird und bekommt dazu eine neue, eindeutige Produktnummer. Diese PRODUCT_NO wird in den

meisten Tabellen als FK verwendet. Somit können mittels Schmelzen-ID sämtliche Prozessdaten aus unterschiedlichen Tabellen abgerufen werden.

2.3.3.2 Tabellentypen und Prefixe

Es kann zwischen fünf unterschiedlichen Arten von Tabellen unterschieden werden:

- GC_XXX Tabellen für **Allgemeine Konfigurationen**: Daten in dieser Tabelle (z.B. Gassorten, Stahlwerksdaten) ändern sich nach der Inbetriebnahme nicht und können von der Level2-MMS nicht manipuliert werden.
- GT_XXX Tabellen für **Allgemeine Einstellungen**: Daten in dieser Tabelle (z.B. Materialkenndaten, Systemparameter) werden gelegentlich modifiziert und können von der Level2-MMs angepasst werden.
- PD_XXX Tabellen für sämtliche **Prozessdaten,** die in Referenz zu einer Schmelze stehen (z.B. gemessene Daten, Materialdaten, Berechnungen aus der Analyse).
- PP_XXX Tabellen für Daten zur **Produktplanung:** was wird wann und wo produziert.
- HD_XXX Tabellen, die **Kopien von GT_XXX** Tabellen darstellen: da sich GT_Tabellen verändern können, werden Kopien von GT_Tabellen gespeichert, um eine spätere Evaluierung zu ermöglichen.

Darüber hinaus kann das Prefix noch einen dritten Buchstaben enthalten, der den Prozess im Stahlwerk identifiziert. Beispielsweise wird „B“ verwendet für BOFs, also LD-Konverter. Werte werden im Allgemeinen in SI-Einheiten oder abgeleiteten SI-Einheiten gespeichert. Dies gilt auch für Werte, die in der MMS in anderen Einheiten abgebildet werden. Darüber hinaus muss jede Variable und Tabelle eine eindeutige Beschreibung haben. Das Datenformat, in dem die Werte gespeichert werden, ist an die Einheit gebunden.

3 Programmstruktur & Erstellung der MMS

3.1 Systemüberblick

Damit die MMS den in Abschnitt 1.4 erwähnten Anforderungen gerecht werden konnte, genügte es nicht, das alte LabVIEW-System lediglich in MATLAB umzuprogrammieren. Stattdessen wurde ein System entworfen, dass Funktionalitäten des alten Systems mit neuen Methoden umsetzt und um andere Komponenten erweitert. In Abbildung 4 ist ein Überblick über das neue System ersichtlich: Harware und Software-Komponenten, sowie die Schnittstellen und deren Protokolle, mit denen MATLAB mit den Komponenten kommuniziert, sind dargestellt.

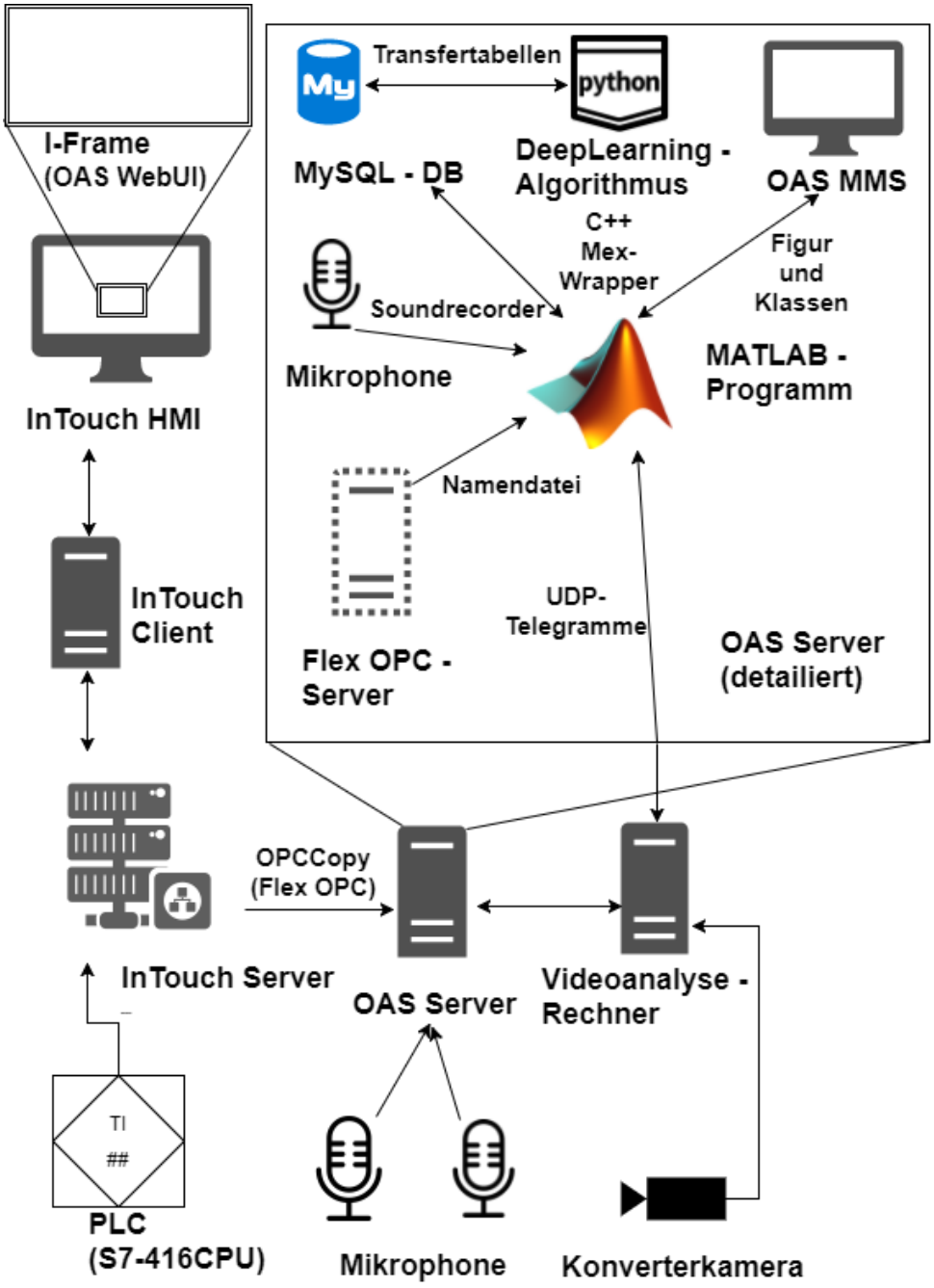


Abbildung 4: Systemüberblick und Methoden der Schnittstellen.

3.2 Klassenhierarchie

Sämtliche Klassen wurden als Handle-Klassen implementiert. Um eine Hierarchie, wie sie durch gewöhnliche Sub- und Superklassen entsteht, erreichen zu können, wurden Attribute von Klassen deklariert, die die vom Klassenkonstruktor einer anderen Klasse zurückgegebene Referenz, auf das entsprechende Objekt der anderen Klasse, halten. Dann kann wie bei einer gewöhnlichen Struktur mit Punkt-Notation auf Attribute und Methoden eines Objekts zugegriffen werden. Beispielsweise wird in Zeile 47 in *classOAS.m* ein Attribut *Statusbar* deklariert. Anschließend wird in Zeile 424 das Attribut mit dem Rückgabewert der Klasse *classHDStatusbar* initialisiert. Dieser Vorgang ist in den Abbildungen 5 und 6 dargestellt.

```
46          % handle to statusbar
47 -        Statusbar;
424 -           obj.Statusbar = classHDStatusbar(...
```

Abbildung 5: Deklaration und Initialisierung eines Referenzattributs (*classOAS*).

```
89          %% - Constructor
90          function obj = classHDStatusbar(Parent_,
```

Abbildung 6: Rückgabewert eines Klassenkonstruktors (*classHDSidebar*).

Der Klassenkonstruktor der Klasse *classHDSidebar* bildet dazu ein Objekt. Da *classHDSidebar* eine Subklasse der *Handle*-Klasse ist, wird eine Referenz auf dieses Objekt an das Attribut *Statusbar* in *classOAS* zurückgegeben. Eine Klasse kann somit höher in der Rangordnung gesehen werden, wenn sie zahlreiche dieser Attribute beinhaltet. Damit wird weiters eine einseitige Beziehung aufgebaut: Klassen höherer Ordnung haben *get()* und *set()* Möglichkeiten (also Lese- und Schreibfähigkeiten) auf Klassen niederer Ordnung, aber nicht umgekehrt. Abbildung 7 zeigt einen Ausschnitt aus der Klassenhierarchie des Systems.

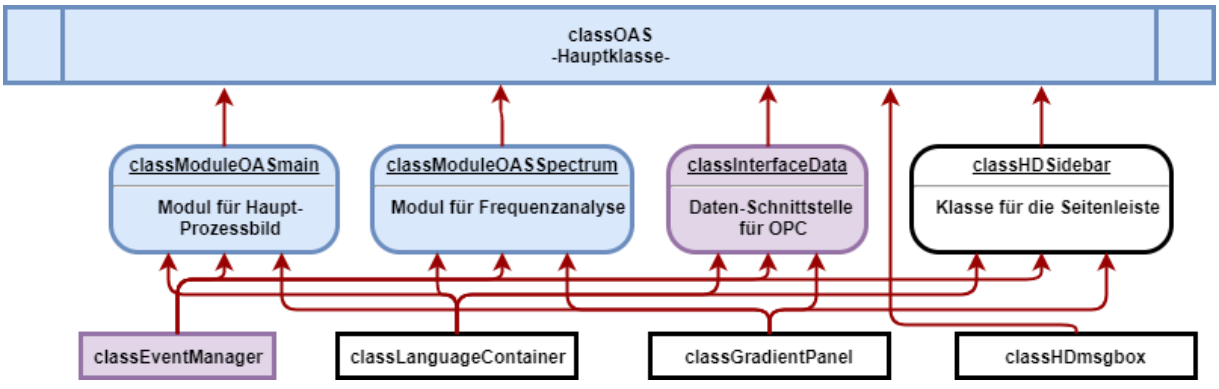


Abbildung 7: Klassenhierarchie wichtiger Klassen.

Drei Ordnungen von Klassen können unterschieden werden:

- Auf **Level 1** befinden sich Klassen und Funktionen, die Hilfstätigkeiten für andere Levels ausüben. Beispielsweise wird *classGradientPanel* immer dann ausgeführt, wenn ein Hintergrundobjekt mit bestimmtem Farbverlauf erstellt werden soll.

- Auf **Level 2** befinden sich Klassen, die Objekte erstellen, die den in Abschnitt 2.3.1 erwähnten Visualisierungselementen der MMS entsprechen. Es gibt auch Klassen mit dem Prefix *classModule*. Diese erstellen Prozessabbilder. Die modulare Umsetzung dieser Klassen durch die Aufteilung der Referenzattribute bedeutet, dass zusätzliche Prozessabbilder zum System leicht hinzugefügt werden können.
- Auf **Level 3** befindet sich nur *classOAS*. *classOAS* ist die Hauptklasse und hält Referenzen auf Objekte sämtlicher Level 2 (und mancher Level 1) Klassen. Sie beinhaltet sowohl die Initialisierung als auch den zyklischen Programmablauf.

Die Initialisierung des Programms findet in *classOAS* statt und ist einem Fließdiagramm in Anhang B - Schritte der Programminitialisierung dargestellt. Auf der linken Seite ist ein Überblick über die wichtigsten Attribute und Methoden der Hauptklasse *classOAS* zu sehen. Bei Aufrufen der Art *classXXX* < *handle* wird ein Attribut von *classOAS* mit einer Referenz auf das Objekt der Klasse XXX versehen. *XXX()* sind aufgerufene MATLAB oder *classOAS*-Methoden.

3.3 Visualisierungselemente

Anders als bei bisherigen Projekten, die auf dem MATLAB-PG aufbauen, leben sämtliche Visualisierungselemente in einer einzigen MATLAB-Figur (mit Ausnahme des *WebUI*s) anstatt in einer MATLAB-GUIDE-Umgebung. Dadurch ist es leichter, eine Referenz auf die Figur an Funktionen weiterzugeben. Für andere Elemente wirkt die Figur als (Groß)Eltern-Umgebung.

Für sämtliche Module, also Prozessabbilder für den D2-Bereich, wird die Figur als direkter Elternteil betrachtet. Eine Referenz auf sie wird bei der Initialisierung der Module übergeben, und mit Hilfe von *classGradientPanel* wird ein Hintergrundobjekt in der Figur erstellt, das bereits den gewünschten Farbverlauf enthält. Auf diesem Hintergrundobjekt werden dann entweder weitere Panele mit *uiPanel()* oder Achsen mit *axes()* erstellt.

Für *classModuleOASmain* beispielsweise wird ein Hintergrundobjekt in der Figur erstellt, mit zwei Achsen bestückt, die dann mit *yyaxis()* erweitert werden. In diesen Achsen werden anschließend mit *line()* Linienobjekte erstellt, wodurch der Verlauf von zwei Prozessgrößen über eine X-Achse je Graph betrachtet werden kann. Das sind Schallpegel und Sauerstoffrate im linken Graph respektive Auswurfwahrscheinlichkeit und Anzahl-heller-Pixel im rechten. Durch die Verwendung eines Hintergrundobjekts können Module einfach ein- bzw. ausgeblendet werden, in dem die Sichtbarkeit des Hintergrunds umgeschaltet wird.

Die MATLAB-Figur und die Module verfügen weiters über „Neu-Skalierungs“ Methoden. Dazu besitzt die Figur für die Eigenschaft ‚ResizeFcn‘ eine Methode, die ausgelöst wird, wenn die MATLAB-Figur vergrößert oder verkleinert wird. In dieser Methode werden Ränder neu berechnet und der *EventManager* bekommt eine Benachrichtigung, dass eine Neuskalierung stattfindet. Weiters wird das aktive Modul sowie sämtliche MMS-Elemente angepasst.

MMS-Elemente werden ähnlich erstellt, mit dem Unterschied, dass das Hintergrundobjekt nicht ausgeblendet werden kann und die Abmessungen und Abstände der Panele und Trennlinien durch die GR vorgeschrieben sind. Das bedeutet, dass bei einer Vergrößerung der Figur vornehmlich der D2-Bereich an Fläche gewinnt und die MMS-Elemente lediglich ihre Position ändern.

4 OPC – Kommunikation

4.1 OPCCopy, Flex-OPC Server und Variablennamen-Datei

Das alte LabVIEW-System empfing dynam. Prozessdaten (Daten, die sich während der Schmelze ändern) via TCP/IP-Telegrammen und verwendete einen Prosis-Server zusammen mit einer Access 2007 Datenbank, um Prozessdaten zu speichern. Der von Ghosh (2017) entwickelte Algorithmus wurde jedoch auf Excel-Tabellen eingelernt. Diese Tabellen wurden mit Daten mehrerer Monate aus dem Stahlwerk zusammengestellt und umfassen mehrere Hunderttausende Zeilen von statischen und dyn. Prozessdaten. Damit der Algorithmus auch mit statischen Prozessdaten, das sind Daten, die sich während der Schmelze nicht verändern, versorgt werden kann, musste eine neue Schnittstelle entworfen werden. Dabei wurde den PG-Standards gefolgt und eine Kombination aus OPC-Servern, OPC-DA Telegrammen und Variablentabelle verwendet. Demnach findet der Informationsfluss nun wie folgt statt:

- Sensorik und SPS liefern Prozessdaten an den InTouch-Server, der vom Stahlwerksbetreiber verwendet wird, um deren MMS zu generieren. Dieser Server stellt einen OPC-Server bereit, auf dem die aktuellen Prozessdaten liegen.
- Auf diesen Server wird OPC-Copy angewandt, eine hauseigene Funktion, die via OPC-DA-Telegrammen Daten von einem OPC-Server auf den nächsten kopiert.
- Dieser zweite Server ist der virtuelle Flex-OPC Server, der auf dem OAS-Server läuft. Dieser hat einen definierten Namensbereich für Variablennamen.
- Um die Daten in den Arbeitsbereich von MATLAB zu bekommen, werden zwei PG-Klassen *classInterfaceOPC* und *classInterfaceData* verwendet. Die IP-Adresse sowie die ID des Servers sind in einer Konfigurationsdatei definiert und werden zusammen mit der Variablentabelle von *classInterfaceOPC* benötigt, um eine Schnittstelle zu erstellen. In der Variablentabelle sind die Variablennamen im Namensbereich des Servers und im MATLAB-Arbeitsbereich, sowie die Einheit und Skalierung für jede Variable, definiert. Verwendete MATLAB-Methoden kommen aus der OPC-Toolbox und umfassen unter anderem *connect(), read()* und *setTagFile()*.

5 Integration der Mikrofone und Kamera

5.1 Audiosystem

In Anhang A - Mathematische Hintergründe wurde ein Verfahren präsentiert, in dem via Schallmessungen und unter Bestimmung von frequenzabhängigen Dämpfungskoeffizienten Algorithmen angewendet werden konnten, um auf die Höhe des Schlackeschaumlevels zu schließen. Die Bestimmung des Dämpfungskoeffizienten $\beta_F(\omega)$ ist jedoch sehr aufwendig und benötigt eine langfristige Evaluierung der Intensitäten verschiedener Frequenzbänder. Außerdem wird in Salzgitter oftmals der Stellring bewegt, wodurch sich die Geräuschkulisse ändert. Dennoch bedeutet ein größeres Volumen an Schlacke bzw. Schaumschlacke eine, auch für das menschliche Gehör wahrnehmbare, Verminderung des Schallpegels. Der Verlauf dieses Schallpegels und die Leistungen einzelner Frequenzbänder sind dabei Informationen, die in der MMS visualisiert werden sollen.

5.1.1 Einbindung der Mikrophone

Damit die Entwicklungs- und Instandhaltungskosten des Systems geringgehalten werden können, wird für die Einbindung der Mikrofone keine MATLAB-Toolbox verwendet. Lediglich zur Signalverarbeitung wird die SignalProcessing-Toolbox benutzt. In einer Konfigurationsdatei sind die WindowsIDs, Erneuerungsraten der Messungen (44.1 kHz), Bits pro Messung (8) und Anzahl der Kanäle (2) der Mikrofone definiert. Diese Informationen wurden aus der Analyse des LabVIEW-Systems gewonnen. Diese Daten werden von der MATLAB-Funktion *audiorecorder()* verwendet, um zwei Aufnahmeobjekte zu erstellen. Anschließend werden Signale der Mikrofone mit dem Befehl *record(obj.SoundRecorderX, 1)* für eine Sekunde aufgenommen. Die Aufnahme muss kurz gehalten werden, da *audiorecorder()* nicht für längere Aufnahmen ausgelegt ist. Es verwendet Arbeitsspeicher, anstatt einer Festplatte, um aktuelle Aufnahmen temporär zwischenzuspeichern.

5.1.2 Audiosignalverarbeitung und Frequenzspektrum

Zur Berechnung des Schallpegels wird das untere der beiden Mikrofone, nahe des Konverters, verwendet. Der Schallpegelverlauf dieses Mikrofons liegt näher an dem Verlauf, der von Stahlwerksmitarbeitern in der Umgebung des Konverters wahrgenommen wird. Für die Signalverarbeitung werden Daten aus den Aufnahmeobjekten mit der Funktion *getaudiodata()* im Datenformat ‚uint8' gelesen. Die Informationen im zweiten Kanal beider Mikrofone werden gelöscht, da sie ident, oder nahezu ident, mit dem anderen Kanal sind und somit drastisch Rechenzeit gespart werden kann.

Die zwei Vektoren mit jeweils 44100 Werten werden danach transponiert. Der Vektor des unteren Mikrofons wird um 0 gemittelt und der Absolutwert wird berechnet. Anschließend wird der Mittelwert sämtlicher Werte ermittelt und mit einem Faktor von 10 multipliziert. Dadurch wird ein Wert gebildet, der für den Schallpegel in dieser Sekunde steht. Dieser ist dimensionslos und kann durch den Skalierungsfaktor angepasst werden.

Zur Berechnung des Frequenzspektrums werden die aktuellen Aufnahmen beider Mikrofone, zusammen mit den Grenzwerten der Frequenzbänder und Erneuerungsraten der Mikrofone, in *updateEverything()* an das Modul *classModuleOASSpectrum* weitergegeben. Darin wird die MATLAB-Funktion *periodogram()* verwendet um einen PSD (Power-Spectral-Density) – Graph zu erstellen. Der PSD-Graph zeigt die Verteilung der Leistung eines Signals über die Frequenz. Er liefert also ein Frequenzspektrum. Beim Aufruf von *periodogram()* werden dazu vier Eingangsgrößen übergeben:

- Der Vektor mit der aktuellen Aufnahme des Signals vom Mikrofon.
- Ein leeres Fenster (leerer Vektor). Das ganze Signal soll auf einmal analysiert werden.
- Die Länge des Vektors (NFFT). Dies entspricht dann der Anzahl der Punkte, die in der Fast-Fourier-Transformation (FFT) verwendet werden.
- Die Erneuerungsrate des Mikrofons in Hz.

Dannach berechnet *periodogram()* die zeitdiskrete Fouriertransformierte des Signals mit Hilfe von FFTs bei einer finiten Anzahl von Frequenzpunkten. Der Algorithmus ist effizienter, wenn die Anzahl der verwendeten Punkte einer Zweier-Potenz entspricht, das Eingangssignal wird

dazu evt. mit Nullern aufgefüllt. Die Daten des Mikrofons sind reell, keine komplexe Information ist vorhanden. Demnach sind die Magnituden der Frequenzen symmetrisch um Frequenz 0 (Gleichstromanteil) und nur die ersten (1+NFFT/2) Punkte im Ergebnisvektor sind eindeutig. Dadurch wird das Frequenzband nach dem Nyquist-Shannon–Theorem halbiert: Es können nur Informationen über Frequenzen zwischen 0 Hz und 22050 Hz gewonnen werden.

Als Rückgabewert liefert *periodogram()* zwei Vektoren. Einer beinhaltet alle analysierten Frequenzen, enthält also alle ganzen Zahlen von 0 bis 22050. Der andere Vektor liefert Informationen über die Leistung, die im Signal enthalten ist, als Funktion der Frequenz. Diese Werte sind die skalierten Quadrate der Magnituden. Die Einheit ist W/Hz bei definierten Eingangssignalen mit bekannten Spannungen und Widerständen, ansonsten dB/Hz, wenn die Erneuerungsrate bekannt ist. Schwingungen mit sehr niedriger Frequenz (unter 45 Hz) werden manuell aus dem Signal herausgefiltert, um den Gleichstromanteil sowie Basisschwingungen zu eliminieren; sonst wäre die Leistungsspitze immer bei 0 Hz. Anschließend kann der Magnitudenvektor über dem Frequenzvektor in einem Graph dargestellt werden.

Um Informationen über die Leistung eines bestimmten Frequenzbandes zu gewinnen, wird die MATLAB-Funktion *bandpower()* verwendet. Sie nimmt den Magnituden- und Frequenzvektor von *periodogram()* entgegen und errechnet die totale Leistung, die in dem Signal enthalten ist. Alternativ nimmt sie auch den oberen und unteren Grenzwert eines Frequenzbandes entgegen, um die Leistung in diesem Band zu berechnen. Die Leistungen der einzelnen Bänder werden gegenüber der Gesamtleistung gewichtet. Aus dem Magnitudenvektor wird weiters der Index und Wert der maximalen Leistung ermittelt. Diese Berechnung findet für beide Mikrofone jede Sekunde mit neuen Werten statt.

5.2 Videosystem

Über das Videoanalysesystem kann zusammenfassend gesagt warden, dass Bilder der Kamera, die auf die Außenwand des Konverters gerichtet ist, verarbeitet werden, um helle Pixel (also Schlackepixel) zu zählen. Diese Anzahl-heller-Pixel und die Bedienoberfläche des Systems, sollen in die Mensch-Maschine Schnittstelle eingebunden werden.

5.2.1 UDP-Kommunikation

Das Videoanalysesystem kommunizierte mit LabVIEW über UDP-Telegramme. Um diese Kommunikation in dem neuen System mit MATLAB zu ermöglichen, wurde in C ein Programm geschrieben, das die Standard–Windowsbibliothek winsock2 verwendet.

Zuerst wird die Konfigurationsdatei *UDP-Parameter.txt* geöffnet und dessen Inhalt eingelesen. Parameter sind zum Beispiel *deployed, time_out* sowie lokale und externe Portnummern. *Deployed* gibt an, ob das System offline oder im Stahlwerk betrieben wird. *WSASocketA()* wird verwendet um einen UDP-Socket zu erstellen. Dieser Socket wird anschließend mit *bind()* and die lokale Adresse gebunden. Mit Hilfe von *recvfrom()* wird ein Telegramm vom Videosystem empfangen und in ein Array zwischengespeichert. Der Aufruf zu *recvfrom()* ist blockierend, deswegen wurde mit *select()* eine Zeitbeschränkung implementiert, damit das Programm nicht endlos blockiert wird. Das Videosystem auf der anderen Seite sendet alle 20 ms ein neues

Telegramm. Damit die Datei *Bright_Pixels_Values.txt* nicht zu groß wird, wird nur jedes 25te empfangene Telegramm formatiert und der Inhalt in die Datei geschrieben. Diese Datei wird wiederum von MATLAB eingelesen und zu Beginn einer neuen Schmelze gelöscht.

Im Gegenzug erwartet das Videosystem ein Telegramm vom OAS-Server. In diesem Telegramm wird unter anderem die ID der Schmelze und die Information, ob Frischen im Moment stattfindet, übermittelt. Nur wenn tatsächlich Sauerstoff geblasen wird, berechnet der Algorithmus eine Anzahl-heller-Pixel, ansonsten wird -2 übertragen. Dazu wird *Blowing_Active_UPD.txt* eingelesen, eine Datei, die von MATLAB geschrieben wird. Das Telegramm wird mittels *sendto()* an den Port des Videoanalysesystems übertragen.

5.3 Visualisierung und Speicherung

Die Visualisierung der Mikrofondaten findet in den Modulen *classModuleOASmain, classModuleOASSpectrum* sowie der Seitenleiste, statt. Der errechnete Schallpegel des unteren Mikrofons wird als *CurrentSoundLevel* zwischengespeichert und der Vektor *SoundLevelVector* wird um den aktuellen Wert erweitert. Im Prozessabbild von *classModuleOASmain* wird der Schallpegelvektor auf der linken Y-Achse des linken Graphen über dem Blaszeitvektor mit Hilfe von *line()* aufgetragen. Zur Visualisierung des Frequenzspektrums wird im Graph des Prozessabbilds von *classModuleOASSpectrum* der skalierte Magnitudenvektor über den Frequenzvektor aufgetragen. Schriftzüge im Graphen des Frequenspektrums geben Informationen über den Index und die Leistung des Spitzenwerts des gesamten Frequenzbandes sowie den prozentuellen Anteil der Leisungen, die in den einzelnen Frequenzbändern enthalten ist.

Die Visualisierung der Videodaten findet in *classModuleOASmain* und *classModuleVideoFeed* statt. In *classModuleOASmain* wird der Vektor mit der Anzahl heller Pixel auf der rechten Y-Achse des rechten Graphen über dem Blaszeitvektor aufgetragen. In *classModuleVideoFeed* wird ein simpler MATLAB-Webbrowser als Java-Objekt erstellt, der auf die IP-Adresse des Webservers des Videosystems hört. So können Kamerabilder betrachtet und Konfigurationen zum Videosystem von MATLAB aus vorgenommen werden.

6 Konfigurationsmodul, Alarmsystem und HTML-basierte MMS

6.1 Konfigurationsmodul

Ein Konfigurationsmodul wurde entwickelt, damit in Laufzeit verschiedene Parameter der Audiosignalverarbeitung sowie Generierung von Alarmen angepasst werden können. Diese Parameter sind dabei in folgene Kategorien unterteilt:

- Prozentuelle Grenzwerte der Bereiche, in Bezug auf Maximalwert, in denen sich Schallpegel, Anzahl Heller Pixel und Auswurfwahrscheinlickeit (S/A/A) befinden können. Bei einem maximalen Schallpegel von z.B. 200, bedeutet ein Grünlimit von 50%, dass der Schallpegel als „grün“ eingestuft wird, solange er *größer* als 100 ist.

- Verzögerungszeiten für die Generierung von Dialogen bei „rotem“ (kritischem) Zustand der Prozessvariablen S/A/A.
- Grenzwerte der drei Frequenzbereiche beider Mikrofone für das Frequenzspektrum und eine Mikrofonauswahl, wessen Frequenzbereich dargestellt werden soll.

Für jede der drei Kategorien wurde auf dem Hintergrundpanel ein weiteres Panel mit *uipanel()* erstellt. Anschließend wurden auf diesem Panel mit *uicontrol(‚Style‘, ‚slider‘,...)* Schieberegler erstellt, mit denen die Parameter angepasst werden können. Zu jedem Schieberegler ist außerdem eine Rückruffunktion implementiert, die den Wert des Reglers speichert und das Label der einzelnen Regler entsprechend dem neuen Wert anpasst.

6.2 Statusvariablen, Bargraphen und Ampelsystem

In *calculateLevels()* wird ermittelt, in welchem der drei Bereiche (grün, gelb oder rot) sich die aktuellen Werte von S/A/A befinden. Falls das Frischen im Moment nicht stattfindet, werden die Statusvariablen „blau“ gesetzt, anstatt grün/gelb/rot.

In *updateBarGraphs()* werden die kleinen Bargraphen, die sich in der Seitenleiste unter den numerischen Anzeigen befinden, erneuert, in dem die X-Werte der Oberfläche und die Farbe angepasst wird. Die Breite entspricht dem aktuellen Wert und die Farbe entspricht der Statusvariablen der jeweiligen Prozessgröße. Dadurch ist auf einen Blick erkennbar, was ein Schallpegel von z.B. 105, in Bezug auf den Maximalwert und die drei Bereiche, bedeutet.

In *updatePlotsBG()* werden die Hintergründe der zwei Graphen von *classModuleOASmain* entsprechend den aktuellen Grenzwerten aus dem Konfigurationsmodul angepasst. Zur Erstellung des farblichen Hintergrundverlaufs wurde eine dritte Achse in die beiden Graphen hinter die Achsen, die die *line()*-Objekte halten, gelegt. Die vorderen beiden Achsen werden transparent gesetzt und die Hintergrundachse erhält eine - mit *surface()* erstellte - Oberfläche, die durch das Setzen des Attributs ‚*CData*‘ einen ebenen Farbverlauf trägt. Falls die Statusvariablen „blau“ sind, trägt der Hintergrund einen den GR entnommenen grauen Farbton. Ansonsten folgt der Hintergrund dem Ampelsystem, wobei der Verlauf im linken Graph von unten nach oben Rot-Gelb-Grün lautet, da der Schallpegel hoch sein soll, und im rechten Graph Grün-Gelb-Rot ist, da eine geringe Auswurfwahrscheinlichkeit erwünscht ist.

In *updateBreadcrumb()* werden boolsche Anzeigen mit den aktuellen OPC-Daten erneuert und drei Warnungsfelder werden in *updateTrafficlight()* angepasst. Dazu wird, entsprechend den Statusvariablen, 3, 2 oder -1 an *updateTrafficlight()* übergeben, worauf das Feld angepasst wird, um eine Warnung oder einen Alarm im Sinne der GR symbolisieren zu können.

6.3 Meldungen, Setzbare Aktionen und Dialoge

PrintStatusBar() wird verwendet, um Meldungen zur Nachrichtenleiste hinzuzufügen. Statusmeldungen werden generiert, falls sich S/A/A im gelben oder roten Bereich befinden. Die Statusmeldungen werden dabei in einer Art Nachrichtendatenbank gespeichert. Durch diese Datenbank wird zuerst iteriert, um zu überprüfen, ob eine Statusmeldung, deren Inhalt aus der Sprachdatei entnommen wird, bereits vorliegt. Dazu wird *strcmp()* verwendet, damit

keine Statusmeldung doppelt ausgegeben wird. Anschließend werden die Statusvariablen überprüft und mit *Statusbar.add()* werden Warnungen oder Alarme hinzugefügt. Die Statusleiste zeigt verkleinert die oberste Nachricht (die zuletzt hinzugefügte) an und kann vergrößert werden, um sämtliche Nachrichten inklusive Details anzuzeigen. Diese Details sind die Uhrzeit/Datum, zu der die Meldung generiert wurde, sowie eine exakte Beschreibung der Fehlermeldung. Durch Betätigung eines Bedienfeldes können diese quittiert werden. Die oberste Nachricht verschwindet nach einer bestimmten Dauer von Sekunden, die in einer Konfigurationsdatei festgelegt ist, automatisch.

In *classModulesControlPanel()* wird der D3-Bereich initialisiert, der neben Variablenanzeigen auch drei Felder enthält, die von dem/der BläserIn betätigt werden können. Diese Felder sind als *uicontrol(‚Style‘, ‚pushbutton‘,...)* realisiert, tragen Labels, die in der Sprachdatei festgelegt sind und verfügen über Rückruffunktionen, in denen eine Betätigung gespeichert wird. Sie symbolisieren die drei Aktionen, die im Stahlwerk Salzgitter AG üblich sind, um einen Schlackeauswurf zu vermeiden (siehe Abschnitt 2.1.3).

In *PrintDialogue()* wird zuerst *isempty()* und *isvalid()* auf das Objekt von *classHDmsgbox()* angewandt und überprüft, ob ein Dialogfenster bereits gezeigt wird. Dann werden Zeichenfolgen mit *strcat()* und TeX-Syntax erstellt, die entweder die Schriftfarbe schwarz oder grau tragen, abhängig davon, ob eine Aktion getätigt wurde oder nicht. Anschließend wird ein Zähler inkrementiert und nur wenn eine Prozessvariable für die eingestellte Zeitdauer im „roten“ Bereich ist, wird ein Dialog mit *classHDmsgbox()* erstellt. In diesem Dialog sind Informationen über den kritischen Prozesszustand sowie die Zeichenfolgen ersichtlich.

In der Seitenleiste wird ein zusätzliches Bedienfeld mit *uicontrol(‚Style‘, ‚pushbutton‘,...)* erstellt, mit dem der/die BläserIn einen Schlackeauswurf vermerken kann. Dazu wird ein Vektor mit der Länge des Blaszeitvektors erstellt, der überall mit Null initalisiert wird. Falls nun ein Schlackeauswurf auftritt und der/die BläserIn den Auswurf speichern will, wird bei Betätigung des Feldes im Vektor an dem Index der aktuellen Blaszeit eine Eins gespeichert.

6.4 HTML-basierte MMS

Um den Anforderungen des Stahlwerks an eine minimalistische, webbasierte MMS gerecht zu werden, wurde eine kleine MMS mit der MATLAB-Funktion *uifigure()*, anstatt *figure()*, erstellt. MATLAB verwendet beim Aufruf zu *figure()* Java-Methoden, um ein Objekt zu erstellen. *Uifigure()* und andere Funktionen der *ui*-Familie wie *uiaxes()*, konstruieren hingegen Objekte in HTML-Syntax. In *classOASWebUI* wurden *uifigure(), uiaxes()* und *surface()* verwendet, um eine Anzeige für die Schlackeauswurfwahrscheinlichkeit mit Bargraph und Label zu erstellen, die den Elementen der Seitenleiste ähnelt. In der Initialisierung wird durch die Verwendung der externen Funktionen der *mlapptools* von Iliya & Altman (2017), speziell *mlapptools.getHTML()*, der gesamte HTML-Code der *uifigure()* als eine Zeichenfolge extrahiert und in einem MATLAB-Webbrowser mit *web()* dargestellt. Um Veränderungen der Auswurfwahrscheinlichkeit darzustellen, müssen in *updateEverything()* die Attribute der Figur geändert werden. Daraufhin kann wieder der gesamte HTML-Code extrahiert und der MATLAB-Browser via *web()* erneuert werden.

7 Entwurf der MySQL-Datenbank und Integration des Vorhersage-Algorithmus

7.1 MySQL-Datenbank

7.1.1 Sammlung von Prozessvariablen

Für die Entwicklung der Datenbank (DB) wurden zuerst sämtlich Variablen, die darin gespeichert werden sollen, gesammelt. Diese Variablen lassen sich in mehrere Kategorien unterteilen:

- **Prozessvariablen** sind Variablen, die via OPC empfangen werden. Diese lassen sich weiter unterteilen in statische und dynamische Prozessvariablen (siehe Abschnitt 4.1).
- **Allgemeine Konfigurationen** sind Konfigurationsdaten, die nach der Inbetriebnahme nicht mehr verändert werden (z.B. Name des Stahlwerks, Konverternummer).
- **Systemkonfigurationen** sind sämtliche Variablen, deren Werte in *classModuleconfig* verändert werden können.
- **Analyseergebnisse** sind sechs Vektoren, die maßgeblich für eine Schmelze im Sinne der Schlackeprävention sind: Schlackeauswurfwahrscheinlichkeit, Anzahl-heller-Pixel, Schallpegel, Sauerstoffrate, Schlackeauswurf-Aufgetreten und Blaszeit.

7.1.2 Erstellen einzelner Tabellen

Für die Modellierung der Datenbank wurde die *MySQL-Workbench* sowie die Kommandozeile des Servers verwendet. Die Datenbankbezeichnung lautet „l2smc“ und mehrere Benutzer wurden erstellt, die über verschiedene Privilegien zur Manipulation der Datenbank verfügen. Anschließend wurden Tabellen erstellt, die den in 2.3.3 erwähnten Nameskonventionen entsprechen. Jede Tabelle, sowie jede Variable darin, wurde mit einer Beschreibung versehen, die eindeutig kennzeichnet, wozu die Tabelle bzw. Variable verwendet wird. Zur Speicherung wurde überwiegend das MySQL-Datenformat *DECIMAL(M,N)* für Zahlenwerte, und *VARCHAR(N)* für Zeichenfolgen verwendet. *M* und *N* definieren die Anzahl der Ziffern (M) und Dezimalstellen (N), mit denen die Werte gespeichert werden. Somit kann eine Variable (*M-N*) Vorkommastellen besitzen. *N* in *VARCHAR(N)* definiert die Anzahl der Charaktere. Viele Variablen sind als NN (Not Nullable) umgesetzt: sie müssen also immer einen Wert haben. Indices wurden für die wichtigsten Variablen angelegt, damit das Durchsuchen der Tabellen schneller geschieht. Primary Keys wurden entsprechend den Firmenkonventionen erstellt und Foreign Keys auf diese, in anderen Tabellen, angelegt. Der wichtigste PK ist dabei PRODUCT-_NO, eine mit jeder Reihe automatisch inkrementierende Variable in PD_HEAT_PLANT_REF, die die Verbindung zwischen HEAT_ID vom Stahlwerk und firmeninterner Produktnummer herstellt. Das in der *Workbench* erstellte EER-Diagramm mit sämtlichen Tabellen und deren Abhängigkeiten, ist in Anhang C - EER-Diagramm des Datenbankschemas ersichtlich.

7.1.3 Verbindung von MATLAB mit der Datenbank

Damit MATLAB mit der MySQL-Datenbank kommunizieren kann, wurde der C++ MeX-Wrapper von Almgren (2005) verwendet. Darin ist eine Funktion *mysql()* implementiert, die in MATLAB aufgerufen werden kann. Jede Zeichenfolge, die dabei an *mysql()* übergeben wird,

wird in der selben Syntax auf dem Server der Datenbank ausgeführt. Dadurch können mit *sprintf()* Anweisungen formatiert und als Zeichenfolge an *mysql()* übergeben werden. Das Schreiben und Lesen von Reihen mehrerer Tabellen der Datenbank findet in *updateSQL()* bzw. *updateSQLResults()* statt. Zu Beginn einer neuen Schmelze wird eine neue Zeile in PD_HEAT_PLANT_REF geschrieben und PRODUCT_NO somit um eins inkrementiert. Bei Konfigurationsänderungen wird GTB_HMICONFIG mit einer neuen Reihe erweitert. GC_PLANT und GC_PLANT_DEF müssen manuell bei der Inbetriebnahme mit Konfigurationsdaten beschrieben werden. Weiters werden in *updateSQLResults()* zu Beginn einer neuen Schmelze die sechs in Abschnitt 7.1.1 erwähnten Vektoren im JSON-Format encodiert und in PDB_RESULTS gespeichert, um eine spätere Analyse der Schmelze zu ermöglichen.

7.2 Algorithmus zur Vorhersage eines Schlackeauswurfs

Von Ghosh (2017) wurden drei Modelle zur Datenanalyse untersucht, um eine Vorhersage der Wahrscheinlichkeit eines Schlackeauswurfs errechnen zu können. Diese Modelle sind:

- **Decision Trees** Ein Modell, das den Wert einer Variablen voraussagt, in dem es einfache Entscheidungsregeln abhängig von Merkmalen der Daten lernt (Ghosh, 2017).
- **Random Forest** Hierbei wird eine Anzahl von Merkmalen auf mehrere Unterkategorien eines Datensets angewandt und es verwendet Standardisierung, um die Genauigkeit der Vorhersage sowie eine Überanpassung kontrollieren zu können (Ghosh, 2017).
- **Deep Learning mit H_2O** kombiniert verschiedene nichtlineare Transformationen, um eine komplexe Abstraktion der Daten modellieren zu können (Ghosh, 2017).

Das Deep-Learning Modell, unter Anwendung des H_2O-Moduls für Python, lieferte dabei die besten Ergebnisse: Die Genauigkeit ist am höchsten, der Standardfehler sehr gering (Ghosh, 2017). In den Algorithmus fließt neben den statischen und dynamischen Prozessdaten auch die Anzahl-heller-Pixel sowie Schallintensitäten einzelner Frequenzbänder ein, dadurch ist das Modell von Analysedaten der Mikrofone und der Kamera abhängig.

Die Integration des Algorithmus erfolgte über zwei Transfertabellen in der MySQL-Datenbank: PDB_SLOPCHANCEDATA und PDB_SLOPCHANCERESULTS. Dabei wird jede Sekunde in *updateSQL()* eine neue Reihe in PDB_SLOPCHANCEDATA mit den aktuellen Werten von statischen, dynamischen und errechneten Prozessdaten geschrieben und in Python eingelesen. Der Algortihmus berechnet daraufhin eine Auswurfwahrscheinlichkeit und trägt diese in PDB_SLOPCHANCERESULTS ein. Durch das Argument *ORDER BY REV_TIME DESC LIMIT 1* wird die Auswurfwahrscheinlichkeit in der untersten Zeile der Tabelle eingelesen.

Die Visualisierung findet dabei, analog zu der Visualisierung der Anzahl-helle-Pixel und dem Schallpegel, in *classModuleOASmain* statt. Jede Sekunde wird ein Vektor um die aktuelle Auswurfwahrscheinlichkeit erweitert und gespeichert. Auf der linken Y-Achse im rechten der beiden Graphen wird mit *line()* ein Linienobjekt erstellt. Anschließend wird jede Sekunde in *updatePlots()* der Verlauf der Auswurfwahrscheinlichkeit über der Blaszeit erneuert, in dem die X- und Y- Werte des Linienobjekts verändert werden. Die Auswurfwahrscheinlichkeit wird weiters in der Seitenleiste und der minimalistischen MMS abgebildet, sowohl numerisch, als auch mit einem kleinen Bargraphen.

8 Zyklischer Programmablauf

Um einen zyklischen Programmablauf gewährleisten zu können, wurden Zeitschaltuhren verwendet, die mit *timer()* erstellt werden. Die Periodendauer ist in einer Konfigurationsdatei festgelegt und beträgt eine Sekunde. *Timer()* besitzt mehrere Eigenschaften, die angepasst werden können. Es wurde ‚BusyMode' mit ‚drop' und ‚ExecutionMode' mit ‚FixedRate' initialisiert. Das bedeutet, dass nach Ablauf von einer Sekunde mit der Exekution der Rückruffunktion von ‚TimerFcn' begonnen wird, solange die Exekution des vorherigen Zyklus abgeschlossen ist. Als Rückruffunktion wurde *OnlineTimerFcn()* deklariert, die für den kompletten Ablauf der zyklischen Programmbearbeitung zuständig ist.

Um zu sehen, wie performant das Programm ist, wurden MATLAB-Funktionen *tic()* und *toc()* verwendet: *tic()* startet eine Stoppuhr und *toc()* stoppt diese. In der rechten unteren Ecke der MMS, als Teil der Nachrichtenleiste, findet sich eine Prozentzahl, die equivalent zu dieser Messung ist. Bei ~70% werden somit 0,7 Sekunden für einen Aufruf on *OnlineTimerFcn()* benötigt, ab diesem Grenzwert wird die MMS merklich langsamer und sie braucht länger, um Benutzereingaben zu verarbeiten. Um Rechenzeit zu sparen, werden Aufrufe zu tiefen Strukturen vermieden, in dem Kopien eines Zweiges erstellt werden. Dies findet vor allem in *updateEverything()* und *updatePlots()* statt. Ansonsten müsste für jeden einzelnen Knotenpunkt auf dem Zweig die Privilegien der aufrufenden Methode überprüft werden. Auch das implizite Übergeben der Referenz auf das Objekt der aktuellen Klasse spart Rechenzeit.

In Anhang D - Fließdiagramm des zyklischen Programmablaufs sind die wichtigsten Schritte von *OnlineTimerFcn()* und *updateEverything()* beschrieben. Darin ist kursiver, blau unterlegter Text Syntax aus dem Programm. In schwarzer Schrift darunter werden die Vorgänge in dem Prozess kurz beschrieben. Die *Try/Catch*-Struktur der *OnlineTimerFcn()* ist so zu verstehen, dass der komplette zyklische Programmablauf im *Try*-Zweig stattfindet: Zuerst *updateSound()* dann *updateData()* und anschließend *updateEverything().* Wenn MATLAB nun irgendwo in der Exekution auf einen Fehler stößt, stürzt das Programm nicht ab, sondern lediglich diese eine Iteration. Der *Catch*-Zweig kommt dann zum Tragen, wobei *ex* ein MATLAB-Objekt vom Typ *MException* ist. In dieser Struktur ist eine detailierte Fehlermeldung enthalten.

9 Ergebnisse und Diskussion

9.1 Verwendung des Programmiergerüsts und Umsetzung der Gestaltungsrichtlinien

Das neu entwickelte System verfügt über eine Mensch-Maschine Schnittstelle, die nun den X-Pact Vision 2015-Gestaltungsrichtlinien entspricht und es kann somit ins Produktportfolio der SMS group GmbH wiederaufgenommen werden, ohne dass die Markenpolitik negativ beeinflusst wird. Nach einem Feldtest, der bedauerlicherweise im Rahmen dieser Arbeit nicht mehr stattfinden konnte, kann das System im Stahlwerk Salzgitter und anderen Stahlwerken, die ein Schlackeauswurferkennungssystem benötigen, wie in Tula Sheremet'yevo in Russland, installiert werden. Die Nähe zu den Gestaltungsrichtlinien wurde laufend durch

Vorgesetzte und die R&D-Abteilung der SMS group GmbH überprüft und bestätigt. Das Programmiergerüst wurde sogar verbessert, um es noch näher an X-Pact Vision zu bringen: Die als Symbole ausgeführten Bedienfelder in der Symbolleiste (siehe Abbildung 3) wurden entfernt. Die Navigation zwischen den Prozessabbildern wurde stattdessen in einem Reiter der Menüleiste implementiert. In Abbildung 8 ist eine Gegenüberstellung der alten LabVIEW-Mensch-Maschine Schnittstelle mit dem Hauptprozessabbild der neuen MATLAB-Mensch-Maschine Schnittstelle ersichtlich.

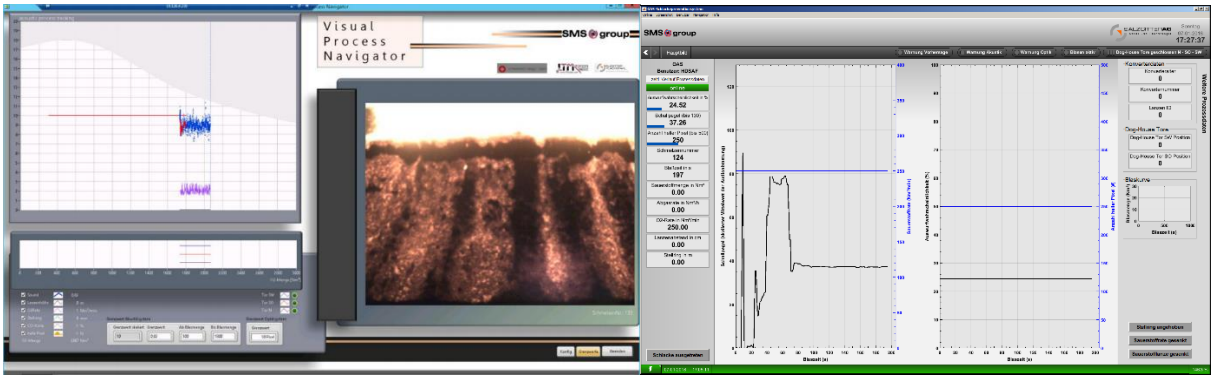


Abbildung 8: Gegenüberstellung der LabVIEW-MMS (links) mit der neuen MATLAB-MMS (rechts).

Während die LabVIEW-MMS nur über dieses eine Prozessabbild verfügt, besitzt die MATLAB-MMS vier modulare Prozessabbilder, die im D2-Bereich (Zentrum einer Ansicht) ausgeführt sind. Diese umfassen das Hauptprozessabbild, mit einer graphischen Darstellung des zeitlichen Verlaufs der wichtigsten Prozessgrößen (von links nach rechts: Schallpegel, Sauerstoffrate, Auswurfwahrscheinlichkeit und Anzahl-heller-Pixel), ein Frequenzspektrum zu einem wählbaren Mikrofonsignal, einen MATLAB-Webbrowser, zur Betrachtung der Kamerabilder und ein Konfigurationsmodul, in dem der/die BläserIn Systemparameter anpassen kann. Der D3-Bereich (das ausklappbare Fenster am rechten Rand) wurde grundlegend überarbeitet, um weitere Prozessdaten, sowie Bedienfelder zur Speicherung von gesetzten Aktionen zur Schlackeminderung, abzubilden.

Die einzelnen Komponenten des Systems können als Paare mit Back-End und Front-End Funktionalitäten beschrieben werden: eine Schnittstelle oder eine Berechnung, die Daten oder ein Eingangssignal entgegennimmt und die dazugehörende Visualisierung dieser (Analyse-) Daten in der MMS. Durch die Simulation der Eingangsgrößen mit definierten Werten/Signalen und Betrachtung der Visualisierung konnte somit die Funktionstüchtigkeit von beiden Aspekten aller Komponenten des Systems überprüft werden.

9.2 Schnittstellen mit der Speicherprogrammierbaren Steuerung, den Mikrofonen und der Kamera

Eine Schnittstelle für Prozessdaten von der SPS konnte nach Firmenkonventionen mit einem OPC-Flex-Server, OPC-DataAccess Telegrammen (OPC-Copy), einer Variablentabelle und Klassen aus dem Programmiergerüst entwickelt werden. Die Visualisierung findet in der Brotkrumenleiste, Seitenleiste, im D3-Bereich und dem Hauptprozessabbild statt. Die

Funktionstüchtigkeit wurde fortlaufend mit dem *MatrikonOPC Explorer* überprüft: Bei der Simulation der Variablen auf dem Flex-Server passten sich die Visualisierungselemente ständig den Änderungen an. Besonders wichtig waren hierbei die Variablen „Schmelzen-ID“, „Blaszeit“ und „Blasen-Aktiv“. Immer wenn sich die Schmelzen-ID verändert, liegt quasi eine neue Schmelze vor. Diverse, in Anhang D - Fließdiagramm des zyklischen Programmablaufs näher beschriebene Aktionen, treten dann in Kraft. Die Blaszeit wurde als Rampenfunktion von 0 bis 1200 Sekunden simuliert. Vor allem die zwei Graphen im Hauptprozessabbild sind hiervon betroffen. Da die X-Grenzwerte (Blaszeit) der Graphen variabel ausgeführt sind, „wachsen“ die Graphen ständig in X-Richtung. Durch Setzen der boolschen Variablen Blasen-Aktiv auf Eins wird das Alarmsystem eingeschaltet und das UDP-Telegramm angepasst.

Die Mikrofone, die bei der Simulation verwendet wurden, verwenden eine andere Bitrate pro Messung als die Mikrofone im Stahlwerk; die Abtastrate ist jedoch die selbe. Diese zwei Mikrofone konnten via MATLAB-Methoden und einer Konfigurationsdatei eingebunden werden. Simulierte Signale des „unteren“ Mikrofons wurden bearbeitet, um einen Schallpegel nahe des Konverters ermitteln zu können. Die Signale beider Mikrofone wurden mit FFT (Fast Fourier transform) analysiert, um ein Frequenzspektrum, sowie die Leistungen der Frequenzbänder, zu erhalten. Die Funktionstüchtigkeit wurde mit Online-Frequenzgeneratoren ausgiebig überprüft. Töne mit bestimmten Frequenzen wurden abgespielt und von einem Mikrofon aufgenommen. In Abbildung 9 ist das Frequenzspektrum von einem solchen definierten Audiosignal in dem zugehörigen Prozessabbild dargestellt.

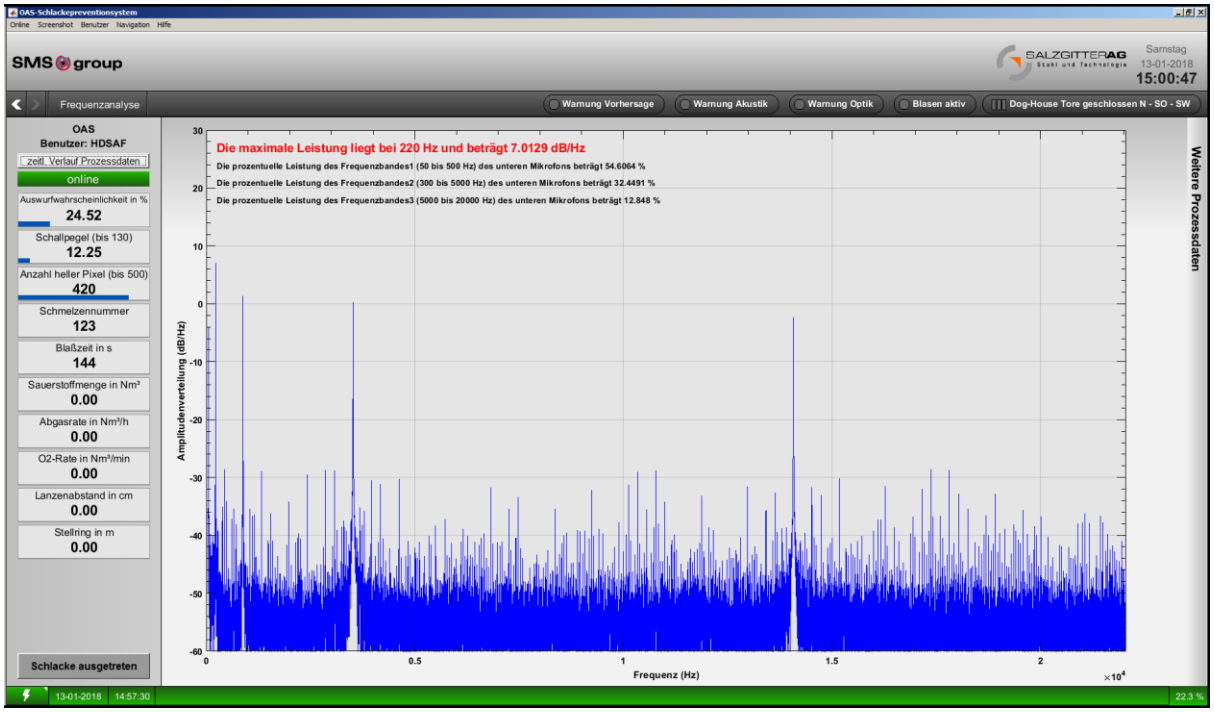


Abbildung 9: Darstellung des Frequenzspektrums eines simulierten Signals bestehend aus fünf Sinusschwingungen (52, 220, 880, 3520 und 14080 Hz).

Eine Transformation dieser Informationen zu einem Pegel der Schaumschlacke im Konverter ist jedoch nicht erfolgt. Die Entwicklung eines solchen Algorithmus ist aus Machbarkeits-

gründen nicht möglich gewesen. Die Auswurfwahrscheinlichkeit des Datenanalyse-Algorithmus wird stattdessen als ausschlaggebendes Merkmal zur Vorhersage eines Schlackeauswurfs verwendet.

Die Kommunikation mit dem Videoanalysesystem wurde in einem C-Programm, das UDP-Telegramme empfängt und sendet, entwickelt. Die Funktionalität dieses UDP-Clients wurde durch die Software *PacketSender* simuliert: In *PacketSender* wurden Telegramme, die eine Anzahl-heller-Pixel beinhalten, an die IP-Adresse und den Port des Systems gesendet. Daraufhin konnte eine Veränderung der Anzahl-heller-Pixel in der MMS betrachtet werden (siehe Abbildung 12). Diese Kommunikation wurde in zwei Varianten getestet: Mit IP-Adressen und Portnummern wie sie im Stahlwerk vorliegen und mit der lokaler-Host Variante (IP 127.0.0.1 und Portnummer. 8888). Ein Prozessabbild konnte außerdem entwickelt werden, das die Bedienoberfläche des Videoanalysesystems mitsamt der Kamerabilder in einem Webbrowser darstellt. Die Funktionstüchtigkeit dieses Webbrowsers konnte jedoch nicht getestet werden, da dazu der Webserver des Videoanalysesystems benötigt worden wäre.

9.3 Alarmsystem, MySQL-Datenbank und Integration des Vorhersage-Algorithmus

Ein Konfigurationsmodul wurde entwickelt, um Systemparameter in Echtzeit anzupassen. Die Funktionstüchtigkeit wurde überprüft, in dem die Schieberegler betätigt wurden, und beispielsweise das Frequenzspektrum die Änderung des selektierten Mikrofons sowie der Bändergrenzwerte widerspiegelt. Das Konfigurationsmodul folgt jedoch nicht komplett X-Pact Vision: die Schieberegler sind gewöhnliche MATLAB-Schieberegler und die Schriftart und -größe sind etwas anders. In Abbildung 10 ist ein Ausschnitt dieses Moduls dargestellt.

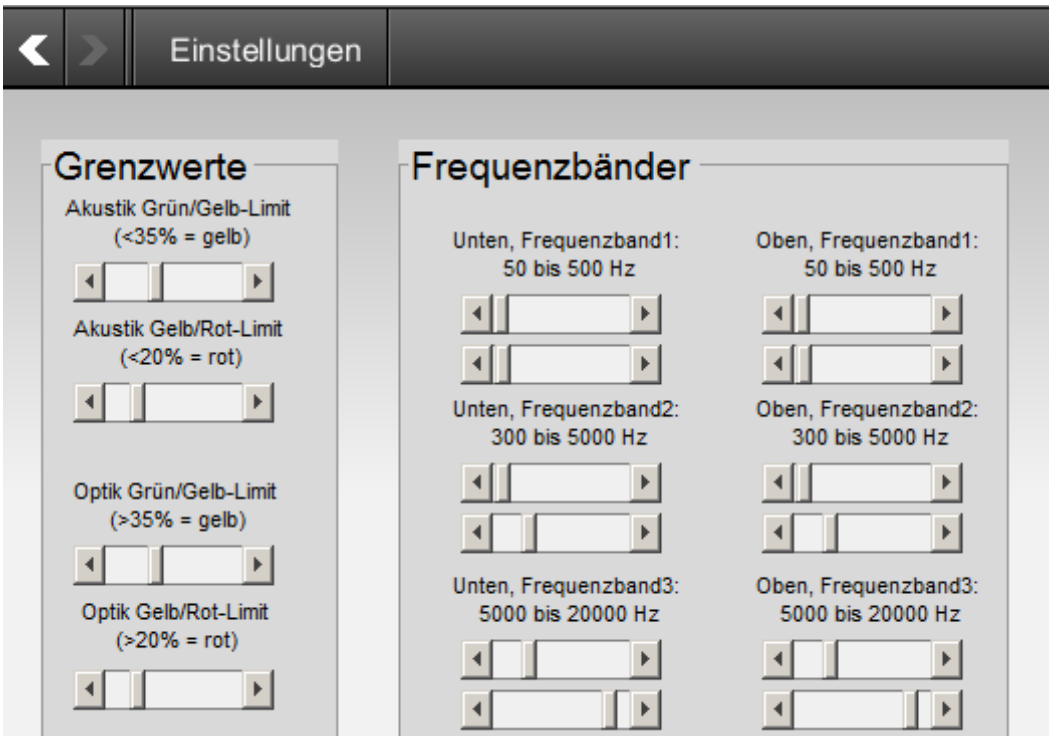


Abbildung 10: Ausschnitt aus dem Konfigurationsmodul.

Eine minimalistische MMS konnte als HTML-Seite erstellt werden. MATLAB stellt jedoch noch keinen Webserver bereit, auf dem diese HTML abgerufen werden kann. Für die Integration in die bestehende InTouch-MMS im Stahlwerk muss dieser Ansatz somit noch vervollständigt werden. In Abbildung 11 ist diese kleine MMS in einem MATLAB-Webbrowser ersichtlich.

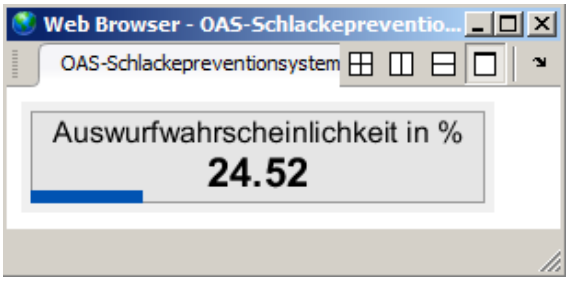


Abbildung 11: Minimalistische, HTML-basierte MMS zur Darstellung der Auswurfswahrscheinlichkeit.

Ein intuitives Ampelsystem wurde implementiert, das an Systemen in Stahlwerken von Wettbewerbern angelehnt ist. Statusmeldungen, Warnungen und Dialoge wurden generiert, in denen Aktionen vorgeschlagen werden, um einen Schlackeauswurf zu vermeiden. In Abbildung 12 ist das Hauptprozessabbild bei aktiviertem Alarmsystem zu sehen.

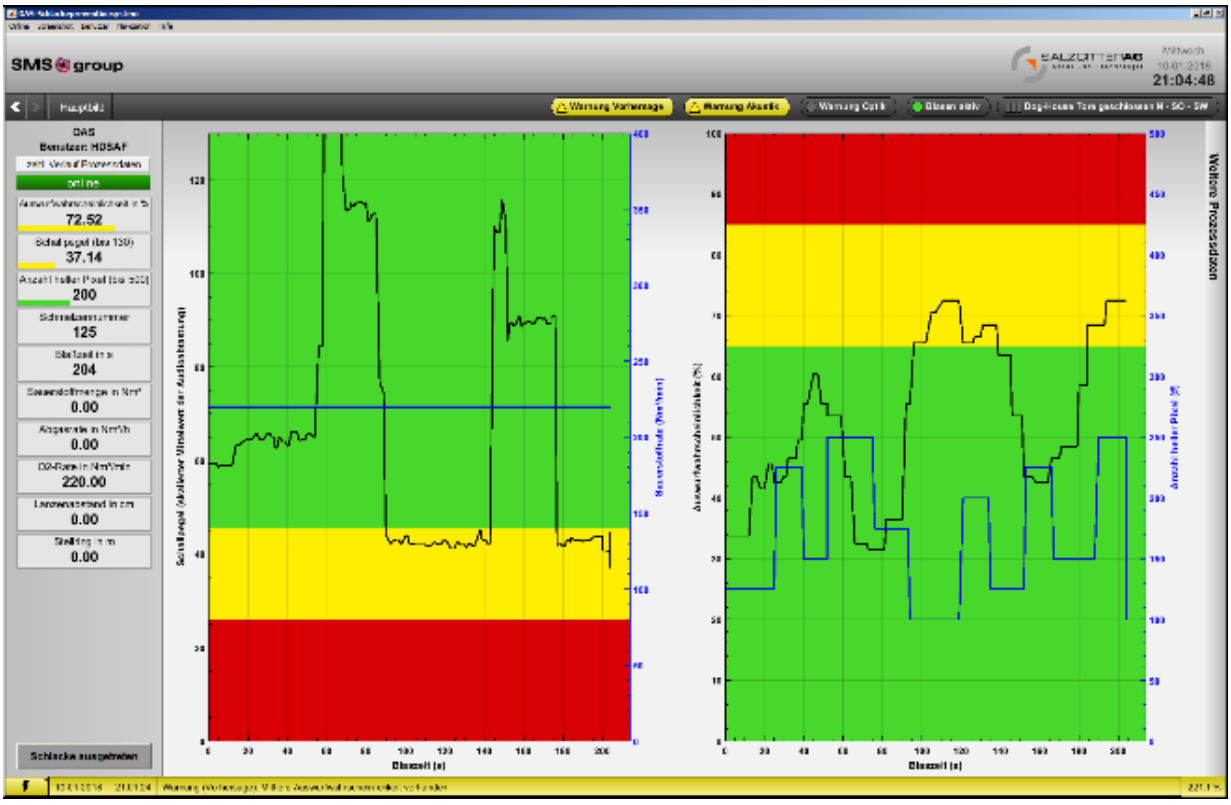


Abbildung 12: Hauptprozessabbild bei aktiviertem Alarmsystem.

Der farbige Hintergrundverlauf bezieht sich auf die Grenzwertebereiche der Prozesskenngrößen, die auf den linken Y-Achsen dargestellt sind. Es könnte dadurch beim rechten Graph zu Verwirrung kommen: Die Anzahl-heller-Pixel kann in dem gelben oder roten Grenzwertbereich liegen, während der Verlauf jedoch auf grünem Hintergrund der Auswurfwahr-

scheinlichkeit liegt. Ein Blick auf die Seitenleiste sorgt dann für Klarheit, da dort die Bargraphen den jeweiligen Status jedes Wertes anzeigen. In Abbildung 13 ist ein Dialog ersichtlich, der generiert wurde, weil der Schallpegel für zumindest die konfigurierte Zeitdauer im roten Bereich lag. Die zweite Aktion „Sauerstoffrate senken“ ist in dem Dialog ausgegraut, da durch das Bedienfeld im D3-Bereich gespeichert wurde, dass diese Aktion bereits erfolgt ist.

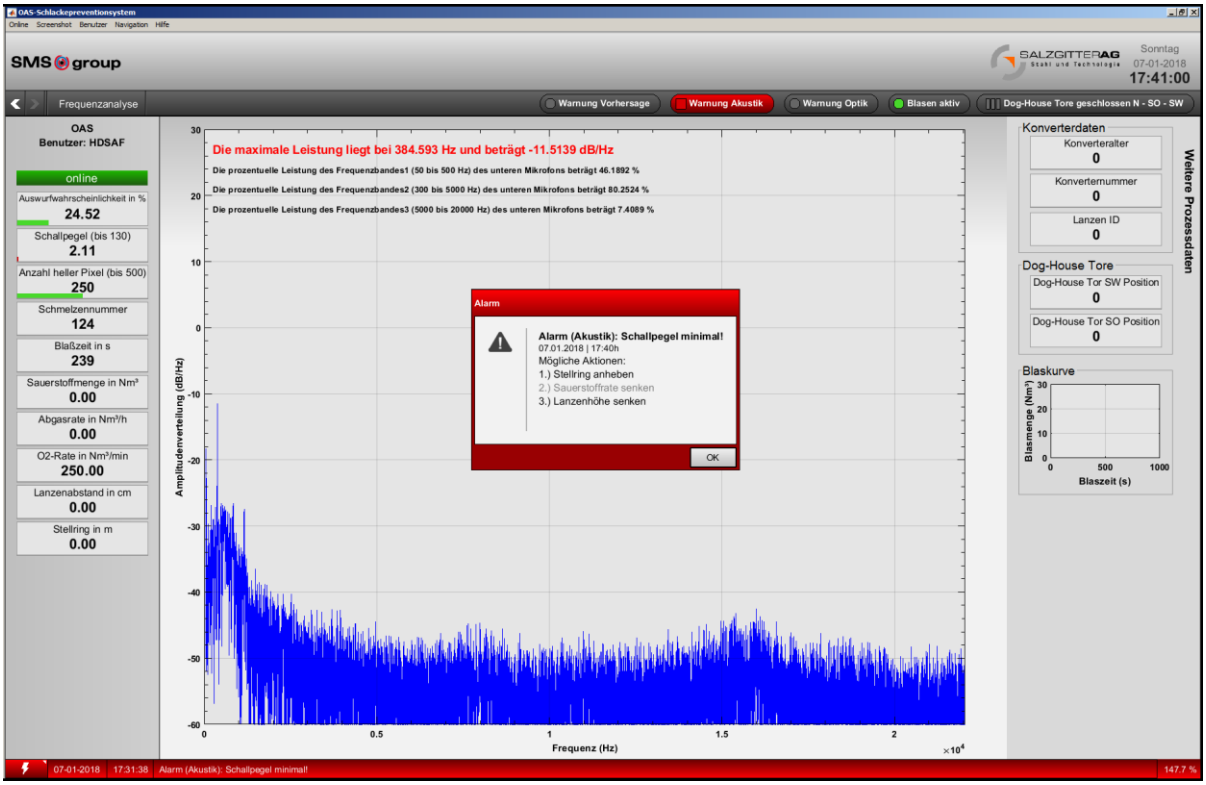


Abbildung 13: Dialog bei kritischem Prozesszustand (hier Akustik) mit setzbaren Aktionen.

Eine MySQL-Datenbank konnte in der *MySQL-Workbench* entworfen und entwickelt werden. Diese folgt den firmeninternen Richtlinien in Bezug auf Tabellen- und Variablennamen und der Verschlüsselung via Primary Keys (PK) und Foreign Keys (FK). Die Kommunikation mit dem Vorhersagealgorithmus findet durch Transfertabellen in der Datenbank statt. Diese Kommunikation konnte wie folgt getestet werden: Historische Daten von den Tabellen, mit denen der Algorithmus eingelernt wurde, wurden an dem Flex-Server angelegt. Die Audiowerte des alten Systems (sechs Schallintensitäten) wurden dabei fest programmiert übernommen. Die errechneten und abgebildeten Auswurfwahrscheinlichkeiten stimmten mit jenen überein, die nur von Python errechnet wurden. Mit diesen historischen Daten konnte Python in mehr als 85 von 100 analysierten Schmelzen einen Schlackeauswurf vorhersagen. Ohne der Audiowerte kam es zu einer Abweichung der errechneten Auswurfwahrscheinlichkeiten von durchschnittlich 15-20 %. Diese Abweichung konnte nicht ausgiebig überprüft werden, da es keine Möglichkeit gibt, mit Hilfe des *MatrikonOPC Explorers* große Datensätze zu simulieren und der Algorithmus mehr als 40 unterschiedliche Werte entgegennimmt. Die Audiowerte im Algorithmus müssen jedoch bei Inbetriebnahme vorläufig vernachlässigt werden, bis genügend Daten mit dem neuen System gesammelt wurden und damit der Algorithmus neu eingelernt werden kann.

Ein zyklischer Programmablauf konnte durch Zeitschaltuhren erreicht werden. Stoppuhren wurden integriert, um zu überprüfen, wie performant das System ist. Das Ergebnis dieser Messung kann auch in der MMS betrachtet werden und liegt im Durchschnitt bei 15 bis 25 Prozent einer Periode, also 150 bis 250 Millisekunden. Wenn auf Fehler gestoßen wird, bricht lediglich eine Iteration des zyklischen Ablaufs ab und detailierte Fehlermeldungen werden in der Kommandozeile und der MMS als Statusmeldung und in einer Logdatei wiedergegeben.

10 Zusammenfassung und Ausblick

Auf Basis eines bestehenden objektorientierten MATLAB-Programmiergerüsts konnte ein System zur Prävention und Erkennung eines Schlackeauswurfs für einen klassischen LD-Konverter mit Sauerstofflanze neu entwickelt werden. Auch wenn das System nicht unter echten Stahlwerksbedingungen getestet werden konnte, stellten sich die Simulationen aller Funktionalitäten des Systems als erfolgreich heraus. Sobald eine Schnittstelle für die Prozessdaten von Seiten des Stahlwerks vorhanden ist, kann das System in Betrieb genommen und getestet werden. Drei verschiedene Ansätze zur Vorhersage und Detektion eines Schlackeauswurfs wurden vereint:

- Der Schallpegelverlauf und das Frequenz- bzw. Leistungsspektrum von Audiosignalen der Sauerstofflanze, deren Schall durch die Schaumschlacke gedämpft wird.
- Die Bildverarbeitung einer Kamera, die auf den Konvertermund und die Konverterwand gerichtet ist und austretende Schlacke aufnimmt.
- Ein komplexer Datenanalysealgorithmus, der statische und dynamische Prozessdaten verwendet, um eine Vorhersage eines Schlackeauswurfs zu liefern.

Im Vergleich mit bestehenden Systemen, die sich auch auf dem Stand der Technik befinden, ist es schwierig, ein System zu finden, dass alle der oben genannten Ansätze vereint. In Tabelle 1 ist eine Gegenüberstellung des OAS-Systems mit anderen Systemen ersichtlich. Das experimentelle Verfahren der Vibrationsanalyse der Sauerstofflanze wurde dabei außer Acht gelassen.

Tabelle 1: Gegenüberstellung der entwickelten Systeme mit den existierenden Ansätzen (Quelle: modifiziert übernommen aus (Ghosh, 2017), S.67).

Projekt	Bildverar-beitung	Audioverar-beitung	Prozess-variablen	Maschinelles-Lernen Algorithmus	Ziel-setzung
Kattenbelt et al. (2008)	X	-	X	-	Schlacke-auswurf
IMPHOS (CORDIS, 2009)	X	X	-	-	Phospor-messung
Tata Steel Scunthorpe (P. O. o. t. E., 2014)	-	X	X	X	Schlacke-auswurf
Arcelor Mittal Gent (P. O. o. t. E., 2014)	-	-	X	X	Schlacke-auswurf
Arcelor Mittal Spain (P. O. o. t. E., 2014)	-	X	X	X	Schlacke-auswurf
OAS Optoakustisches-System	X	X	X	X	Schlacke-auswurf

Auch nach der Inbetriebnahme gibt es noch mehrere Tätigkeiten, die gesetzt werden können, um die Qualität des Systems zu steigern:

- Entwicklung eines Moduls, mit dem Schmelzen nochmals betrachtet werden können. Dazu können die in der Datenbank gespeicherten Ergebnisvektoren verwendet werden.
- Eine Visualisierung des zeitlichen Verlaufs der Bandleistungen. Dazu können die Ergebnisse zwischengespeichert und in drei neuen Graphen abgebildet werden.
- Das MATLAB-Programmiergerüst und die Icons der Mensch-Maschine Schnittstelle können vermehrt in Einklang mit den Gestaltungsrichtlinien gebracht werden.
- Die Ergebnisse des Videoanalysesystems können verbessert werden, indem die Kamera neu positioniert wird, so dass der Flammensprung weniger erfasst wird.
- Die Integration der minimalistischen HTML-basierten Mensch-Maschine Schnittstelle in die bestehende InTouch–Mensch-Maschine Schnittelle muss noch fortgesetzt werden. Dazu könnte MATLAB mit externen Methoden einen Webserver bereitstellen, auf dem die generierte HTML-Seite zur Verfügung steht.
- Ein neues Anliegen des Stahlwerkbetreibers ist die Ermittlung von vier Auswurfwahrscheinlichkeiten für Auswürfe von verschiedenen Intensitäten. Die Visualisierung und die MySQL-Datenbank müssten dementsprechend angepasst werden.
- Die Ermittlung eines Schaumschlackepegels via Schallintensitäten bestimmter Frequenzen kann entwickelt werden, falls der Python-Algorithmus wider Erwarten unbrauchbare Ergebnisse unter Stahlwerksbedingungen liefert.

Literaturverzeichnis

Almgren, R., 2005. *MySQL Database Connector.* [Online] Verfügbar unter: <https://de.mathworks.com/matlabcentral/fileexchange/8663-mysql-database-connector> [Zugang am 06.01.2018]

Altman, Y.M., 2014. *Accelerating MATLAB Performance: 1001 tips to speed up MATLAB programs.* Boca Raton: CRC Press.

Iliya, R. & Altman, Y.M., 2017. *mlapptools.* [Online] Verfügbar unter: <https://github.com/StackOverflowMATLABchat/mlapptools> [Zugang am 19.01.2018]

Birk, W., Arvanitidis, I., Jönsson, P. & Medvedev, A. 2001. Physical modeling and control of dynamic foaming in an LD-Converter process. *IEEE Transactions on industry applications.* 37(4), S.1067-1073.

CORDIS, European Commission, 2009. *Improving phosphorus refining.* [Online] Verfügbar unter: <http://cordis.europa.eu/project/rcn/80409_en.html> [Zugang am 13.12.2017]

Chukwulebe, B.O., Balajee, S.R., Robertson, K.J., Grattan, J.G. & Green, M.J., 2004. Computer optimization of oxygen blowing practices to control BOF slopping. In: *Association for Iron & Steel Technology Conference Proceedings (AISTech 2004),* 15.-17. September 2005, Nashville. Nashville, Tennesse, USA: Association for Iron & Steel Technology, S.751.

Doe, J. (1984). Prediction and control of slag slopping in BOF using microwave gauge. *Trans ISIJ*. 24(6), S.502.

Evestedt, M., Medvedev, A., Thoren, M. & Birk, W., 2007. Slopping warning system for the LD converter Process – an extended evaluation study. *12th IFAC Symposium on Automation in Mining, Mineral and Metal Processing.* 40(11), S.267-272.

Evestedt, M., & Medvedev, A., 2009. Model-based slopping warning in the LD steel converter process. *Journal of Process Control,* 19(6), S.1000-1010.

Ghag, S., Hayes, P. & Lee, H.-G., 1998. Model development of slag foaming, *ISIJ Int.* 38(11), S.1208-1215.

Ghosh, B., 2017. *Opto-Acoustic Slopping Prediction System in BOF (Basic Oxygen Steelmaking) Converters.* [Masterarbeit] Stockholm: KTH Royal Institue of Technology in Stockholm, Masterstudiengang Embedded Systems.

Ingard, K.U., 1994. *Notes on sound absorption technology.* Poughkeepsie, NY: Noise Control Foundation.

Ito, K. & Fruehan, R.J., 1989. Study on the foaming of $CaO - SiO_2 - FeO$ slags: Part II. Dimensional analysis and foaming in iron and steelmaking process. *Metallurgical Transactions B*. 20(4), S.515-521.

Kattenbelt, C., Spelbos, E., Mink, P., Roffel, B., 2008. Detection of Slopping in Basic Oxygen Steelmaking using a Camera Viewing the Converter Mouth. *Steel research international.* 79(11), S.821-825.

Lange, K., 1982. Physikalische und chemische Einflüsse beim Stoffübergang im Sauerstoffaufblasverfahren. *Forschungsberichte des Landes Nordrhein-Westfalen Nr. 3088.* [Forschungsbericht] Opladen: Westdeutscher Verlag

Murphy, K., 2009. *Object Oriented Programming in Matlab: basics.* [Online] Verfügbar unter: <https://www.cs.ubc.ca/~murphyk/Software/matlabTutorial/html/objectOriented.
html> [Zugang am 17.12.2017]

P. O. o. t. E. Union, 2014. *Control of slag and refining conditions in the BOF (Bathfoam).* [Online] Verfügbar unter: <https://publications.europa.eu/en/publication-detail/-/publication/9c8f4509-b410-4177-a51e-ce0e240ec16f> [Zugang am 13.01.2018]

Pak, J.J., Min, D.J. & You, B.D. 1996. Slag foaming phenomena and its suppression techniques in BOF steelmaking process. In: *Proceedings of the 79th steelmaking and 55th ironmaking conference (No. CONF-960317),* 24.-27. März 1996, Pittsburgh. Pittsburgh, PA, USA: Ironmaking conference proceedings, 55, S.807.

Shakirov, M., Boutchenkov, A., Galperine, G. & Schrade, B., 2004. Prediction and prevention of slopping in a BOF. *Iron & steel technology.* 1(1), S.38-44.

Walker, D. I., Kemeny, F.L. & Jones, J.A.T., 2005. Vessel slopping detection. In: *Association for Iron & Steel Technology Conference Proceedings (AISTech 2005),* 9.-12. Mai 2005, Charlotte. Charlotte, North Carolina, USA: Association for Iron & Steel Technology, S.711.

World Steel Association (worldsteel), 2015. *Steel Statistical Yearbook 2015*. [Online] Verfügbar unter: <https://www.worldsteel.org/en/dam/jcr:3e501c1b-6bf1-4b31-8503-a2e52431e0bf/Steel+Statistical+Yearbook+2015+r3.pdf> [Zugang am 14.12.2017]

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AG	Aktiengesellschaft
BOF	Basic Oxygen Furnace
CONFIG	Configuration
CPU	Central Processing Unit
D1/2/3	Datenklasse 1/2/3
DB	Datenbank
DESC	Descending
DIN	Deutsches Institut für Normung
EER	Enhanced Entity-Relationship
FCN	Function
FFT	Fast Fourier Transform
FK	Foreign Key
GC	General Configuration
GmbH	Gesellschaft mit begrenzter Haftung
GR	Gestaltungsrichtlinien
GT	General Table
HD	Historical/Heat-related Data
HMI	Human-Machine Interface
HTML	Hypertext Markup Language
ID	Identifier
INT	Integer
IP	Internet Protocol
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LD	Linz-Donawitz
MEX	MATLAB Executable
MIN	Minimal

MISC	Miscellaneous
MMS	Mensch-Maschine Schnittstelle
MySQL	My Structured Query Language
NFFT	Non-Uniform Fast Fourier Tranform (für FFT von MATLAB bezeichnet NFFT die Länge des Signals, von dem die FFT berechnet wird)
NN	Not nullable
OAS	Optoakustische Schlackenerkennung (firmeninterne Projektbezeichnung)
OO	Objektorientiert
OPC	Object linking and embedding for Process Control
OPC – DA	OPC – Data Access
PD	Production Data
PG	Programmiergerüst
PK	Primary Key
PLC	Programmable Logic Controller
PP	Production Planning
PSD	Power Spectral Density
R&D	Research and Development
REV	Renovation
S/A/A	Schallpegel, Anzahl-heller-Pixel und Auswurfwahrscheinlichkeit
SI	Système International d'unités
SISO	Single Input Single Output
SPS	Speicherprogrammierbare Steuerung
SSD	Solid State Drive
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
UINT	Unsigned Integer
WinCC	Windows Control Center

Anhang A - Mathematische Hintergründe

Konverterprozesse

Walker et al. (2005) präsentierten eine umfangreiche Liste von Prozessvariablen, die Einfluss auf die Entstehung der Schlacke und der Schaumschlacke haben. Dies bestätigte weiter die allgemeine Meinung von Forschern und Metallurgen, dass der Prozess der Schlackebildung ein hochkomplexer ist und als *chaotisch* bezeichnet werden kann. Im Folgenden ein kurzer Ausschnitt aus dieser Liste von Variablen.

- Schlackeviskosität
- Schlackeoberflächenspannung
- Dichte der Schlacke
- Größe der Gasbläschen, die im Zuge der Entkohlung entstehen
- Volumen und Form des Konverters
- Lanzenabstand über dem Eisenbad
- Sauerstoffdurchflußrate durch die Lanze
- Chemische Zusammensetzung des Eisens und des Schrotts
- Geschwindigkeit, mit der die Entkohlung im Roheisen fortschreitet

Bereits mehrere Jahre davor leiteten Ghag et al. (1998), unter Verwendung von mehrdimensionaler Analysis, folgende Formel ab, in der viele dieser Variablen vorkommen,

$$\frac{\Sigma \Delta\sigma}{\mu d_b} = k\left\{\frac{\Delta\sigma}{\rho g d_b^2}\right\}^{\delta} \tag{1}$$

wobei Σ der Schaumschlackeindex in der Dimension der Zeit ist. Er beschreibt die durchschnittliche Dauer, die die Abgase aus dem Konverter benötigen, um durch die Schicht der Schaumschlacke zu dringen (Birk et al., 2001). Der Schaumschlackeindex ist somit abhängig von der Dichte der flüssigen Phase ρ, der Gravitationskonstante g, dem Durchmesser der Gasbläschen d_b, der Schlackeviskosität μ, und der Differenz der Oberflächenspannungen $\Delta\sigma$ zwischen reiner Flüssigkeit und Lösung. Σ wurde nach Ito & Fruehan (1989) definiert als das Verhältnis aus Höhe des Schaumlevels h und Gasleerrohrgeschwindigkeit u_g.

$$\Sigma = h/u_g \tag{2}$$

Die Gasleerrohrgeschwindigkeit kann bei bekanntem Gasanteil annähernd aus der Abgasrate und dem Querschnitt des Konverters berrechnet werden. Die Höhe des Schaumlevels nimmt nur bis zu einem gewissen Grenzwert der Gasgeschwindikeit zu. Danach treten Turbulenzen auf und die Schaumschlacke beginnt zu kollabieren (Birk et al., 2001). Da mit steigender Viskosität der Schaumschlackeindex zunimmt, nimmt auch der Pegel des Schaumlevels zu.

Akustische Verfahren zur Bestimmung des Schlackepegels

Wie Ingard (1994) erkannte, nimmt der Schallpegel exponentiell mit dem Volumen und somit der Höhe der Schaumschlacke im Konverter ab.

$$I(h(t),\, \omega) = I_0 e^{-\beta_F(\omega) h(t)} \tag{3}$$

Unter Anwendung des Logarithmus kann diese Formel umgeschrieben werden zu

$$h(t) = \frac{\ln(I_0) - \ln(I(h(t),\, \omega))}{\beta_F(\omega)}. \tag{4}$$

I_0 beschreibt hierbei den Schallpegel zu Beginn des Frischens ohne Schlackevorkommen und $\beta_F(\omega)$ ist ein frequenzabhängiger Schalldämpfungskoeffizient. Demnach ist die Höhe der Schaumschlacke linear proportional zum Logartihmus des Schallpegels bei bestimmten Frequenzen (Birkt et al., 2001). Über den Schalldämpfungskoeffizienten $\beta_F(\omega)$ müssen dabei Schätzungen getroffen werden. Diese Schätzung kann auch für ein Frequenzband gelten. Wenn diese Schätzung mit $\hat{\beta}_F(\omega)$ bezeichnet wird, erhält man aus (4) folgende Formel.

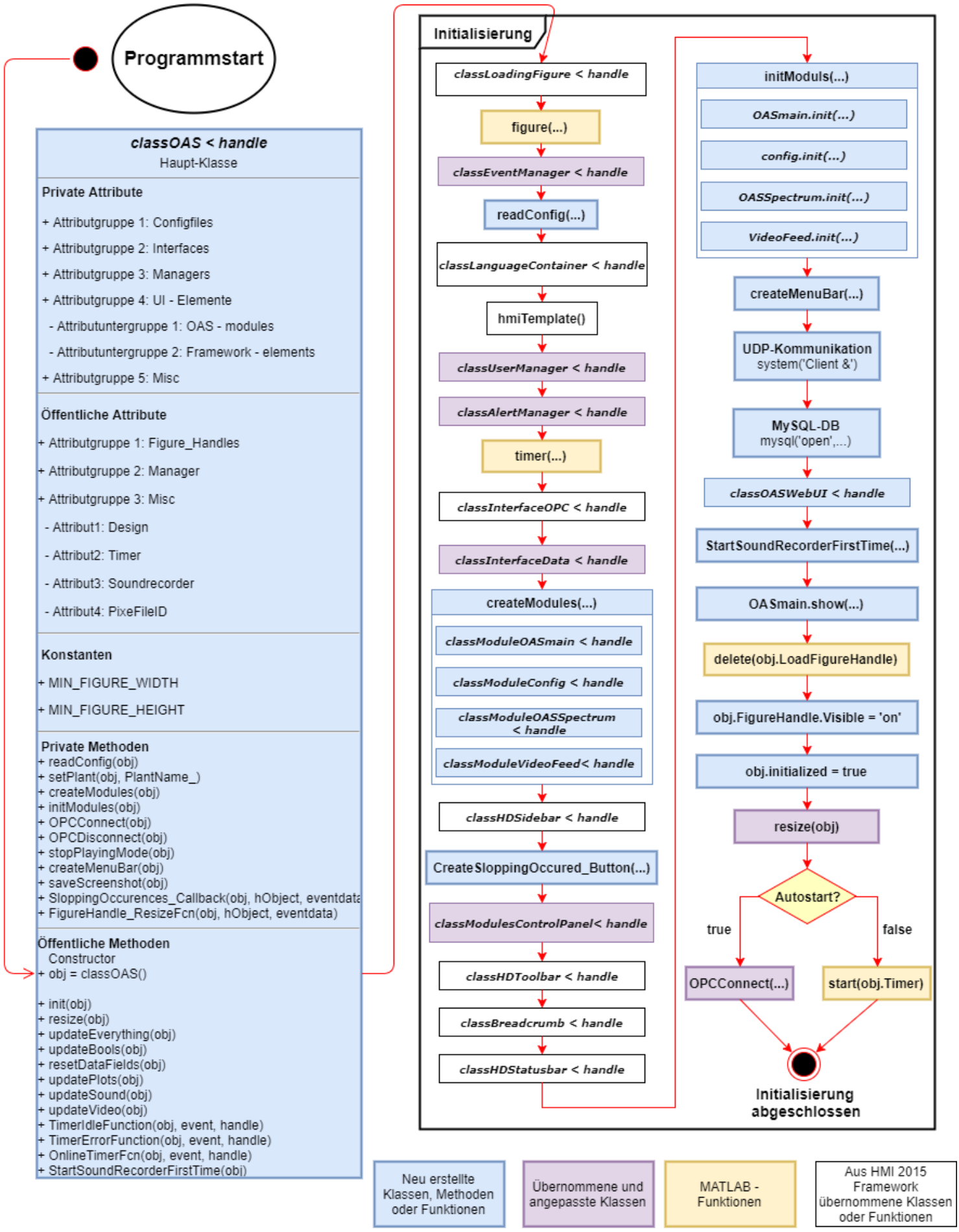
$$\hat{h}(t,\, \omega) = \frac{\ln(I_0) - \ln(I(h(t),\, \omega))}{\hat{\beta}_F(\omega)} \tag{5}$$

Damit ist nun die geschätzte Höhe der Schaumschlacke $\hat{h}(t, \omega)$ auch von der Frequenz ω abhängig. Um diese Abhängigkeit aufzulösen kann die Methode der Kleinsten-Quadrate über eine Reihe von Schätzungen je Frequenz (oder Frequenzband) ω angewendet werden (Birk et al., 2001). Dabei sollten Frequenzen, die keine Dämpfung des Schallpegels, sowie Frequenzen, die hauptsächlich Rauschen enthalten, verworfen oder mit Null gewichtet werden. Schätzungen über $\hat{\beta}_F(\omega)$ können lediglich empirisch getroffen werden. Dabei kann das SISO-System „Schaumschlacke“ über ein Eingangssignal, die Sauerstofflanze als Schallquelle und ein Ausgangssignal, der mit dem Mikrofon aufgezeichnete Schallpegel, identifiziert werden. Da das Eingangssignal nicht messbar ist, kann das System mit Randbedingungen identifiziert werden (Birkt et al., 2001).

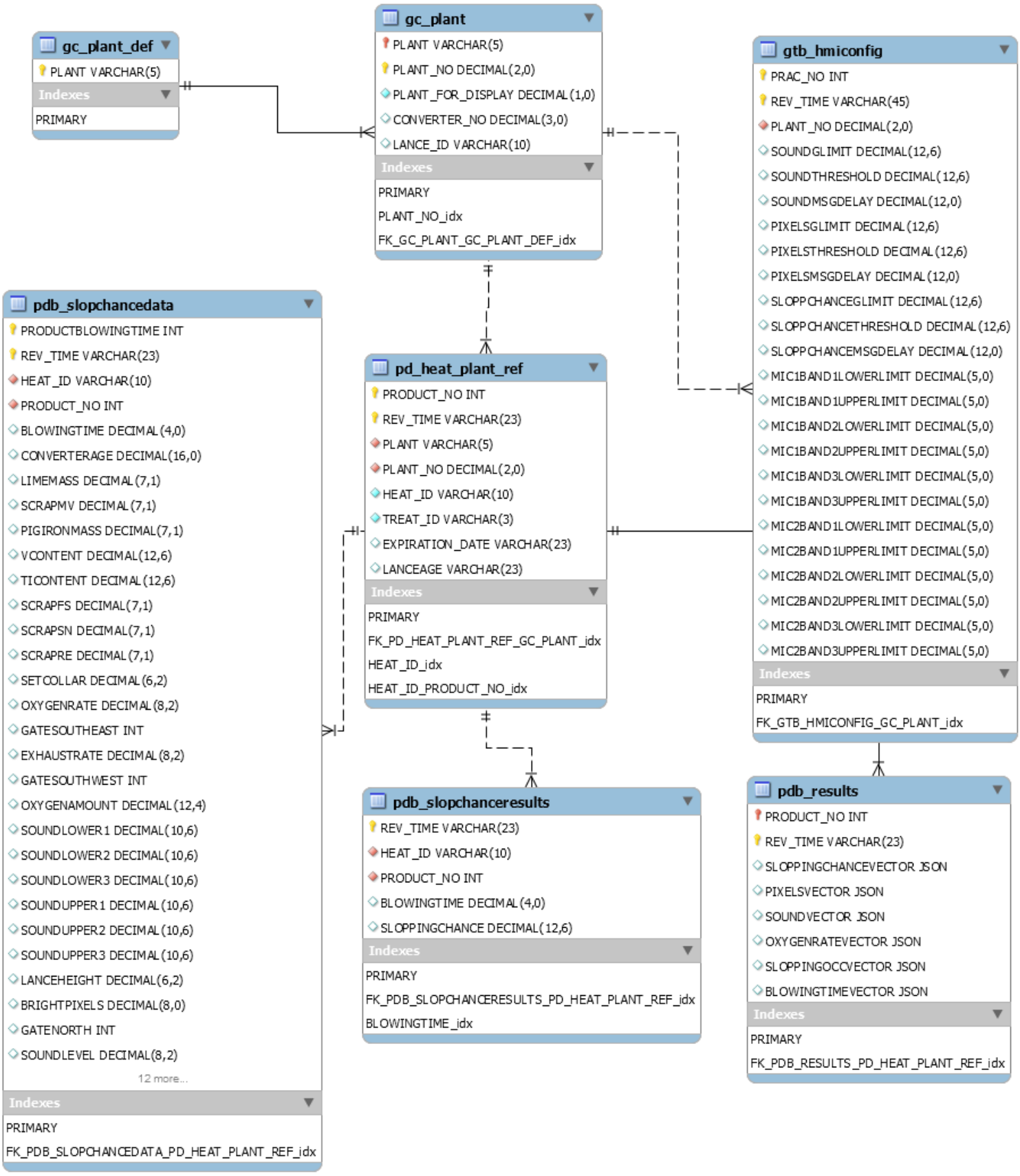
- Zu Beginn des Frischens, wenn der Sauerstofffluss gerade aktiviert wurde, beträgt die Höhe der Schaumschlacke null. Zu diesem Zeitpunkt wird I_0 aufgenommen.
- In dem Moment, wo die Schaumschlacke gerade anfängt, die Düse der Sauerstofflanze zu umgeben, ist die Höhe der Schaumschlacke gleich der Höhe der Sauerstofflanze über dem flüssigen Eisenbad. $I(h_L)$ wird hier aufgenommen.

Voraussetzung hierfür ist, dass beide Momente im Verlauf des Schallpegels identifizierbar sind und das die Sauerstoffrate durch die Lanze, sowie die Höhe der Sauerstofflanze über dem Eisenbad, konstant ist. Die Geräuschkulisse muss ebenfalls konstant bleiben.

Anhang B - Schritte der Programminitialisierung



Anhang C - EER-Diagramm des Datenbankschemas



Anhang D - Fließdiagramm des zyklischen Programmablaufs