Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Simon Navarro

Digitale Weiterverarbeitung und Analyse von Monitoringdaten im Anlagenbau mittels Generierung von Heatmap-Diagrammen

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Simon Navarro

Digitale Weiterverarbeitung und Analyse von Monitoringdaten im Anlagenbau mittels Generierung von Heatmap-Diagrammen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Maschinenbau / Energie- und Anlagensysteme
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
BZE Ökoplan
Hummelsbütteler Weg 36
22339 Hamburg

Erstprüfer/in: Prof. Heike Frischgesell
Zweitprüfer/in : Dipl. Ing. Bernd Schwarzfeld

Abgabedatum: 28.02.2017

Zusammenfassung

Simon Navarro

Thema der Bachelorthesis

Digitale Weiterverarbeitung und Analyse von Monitoringdaten im Anlagenbau mittels Generierung von Heatmap-Diagrammen

Stichworte

Technisches Monitoring, Anlagenmonitoring, Monitoringdaten, Visualisierung, Heatmap, Carpet-Plot, digitale Weiterverarbeitung, MATLAB

Kurzzusammenfassung

Im Rahmen dieser Arbeit wurde eine Anwendungssoftware zur Visualisierung und Weiterverarbeitung von Monitoringdaten in Form von Heat Maps (Carpet Plots) in MATLAB programmiert. Anschließend wird die Anwendungssoftware auf Monitoringdaten, die von zwei Teilsystemen einer Nahwärmeversorgungsanlage stammen, angewandt und die Ergebnisse werden analysiert.

Simon Navarro

Title of the paper

Digital processing and analysis of monitoring data in plant engineering by means of creation of heat map diagrams

Keywords

Plant monitoring, monitoring data, visualization, heat map, carpet plot, digital processing, MATLAB

Abstract

Within the framework of this thesis, an application software was written to visualize and continue processing the monitoring data in MATLAB in the form of heat maps (carpet plots).
Subsequently, the application software is applied to monitoring data originating from two subsystems of a local heat supply system and the results are analysed.

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Science

Aufgabenstellung

für die Bachelorthesis

von Herrn **Simon Navarro**
Matrikel-Nummer: 1961224

Thema:
Digitale Weiterverarbeitung und Analyse von Monitoringdaten im Anlagenbau mittels Generierung von Heatmap-Diagrammen

Schwerpunkte:
Das Anlagen-Monitoring ist ein Überbegriff für alle Arten der unmittelbaren systematischen Erfassung (Protokollierung), Messung, Beobachtung oder Überwachung von Anlagen[1]. Über die durch das Monitoring protokollierten Daten kann die Anlage auf verschiedene Gesichtspunkte hin ausgewertet werden, z.B. ob die Regelung richtig funktioniert oder wie das Laufverhalten der Anlage ist. Zur Unterstützung der Auswertung werden die Daten in der Regel grafisch dargestellt, z.B. durch Linien- oder Balkendiagramme. Diese Bachelorarbeit beschäftigt sich insbesondere mit einer weiteren Form der Darstellung, der sogenannten Heatmap.
Es gibt verschiedene Arten und Anwendungsformen von Heatmaps. Die für diese Bachelorarbeit verwendete Form der Heatmap ist so angeordnet, dass auf der Abzisse die Tage des Datensatzes aufgetragen werden und auf der Ordinate die dazugehörigen Minuten des Tages, im Schnittpunkt deren werden dann der qualitativen Werte der Messungen von Temperatur, Leistung, etc. als Farbe repräsentiert.

Im Rahmen dieser Bachelorthesis sollen daher die folgenden Fragestellungen im besonderen bearbeitet werden:

1. der Erstellung und Dokumentierung des Quellcodes in Matlab zur Generierung und Weiterverarbeitung der Heatmaps;

2. dem Vergleich der Darstellungsform Heatmap mit alternativen Darstellungsformen. Die Vorteile, Nachteile und Grenzen dieser Art Heatmap werden evaluiert, um eine Empfehlung darüber abzugeben, im welchem Bereich diese Darstellungsform sinnvoll eingesetzt werden kann;

3. der Anwendung der Software auf zwei Anlagen, in denen über einen Zeitraum von mehreren Jahren Monitoring-Daten aufgezeichnet wurden. Konkret handelt es sich um die Anlage des GALAB in Hamburg-Bergedorf und die Anlage des Eisenbahnbauvereins Harburg eG in Hamburg-Harburg.

Am Ende der Arbeit sind die Ergebnisse kritisch zu bewerten. Die Ergebnisse der Arbeit sind in entsprechender Schriftform darzustellen und zu dokumentieren. Der Fortgang der Arbeit ist in regelmäßigen Abständen mit den Betreuern der Arbeit zu diskutieren.

17.11.16 — Datum

Frischgesell — Prof. Dr.-Ing. Heike Frischgesell

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Abkürzung:	**Erläuterung**
GUIDE	graphical user interface development environment (Grafische Benutzeroberflächenentwicklungsumgebung)
TM	Technisches Monitoring
GA	Gebäudeautomation
WP	Wärmepumpe
kWh	Kilowattstunde

1. Einleitung

Durch das Einbinden von regenerativen Energiesystemen in Gebäuden werden die Energieversorgung und damit auch die Anlagentechnik von Gebäuden zunehmend komplexer. So werden aus monovalenten Systemen, bi- oder multivalente Systeme, in denen das klassische Anlagensystem, z.B. ein Heizkessel, nur noch als Backupsystem oder zur Spitzenlastabdeckung für die regenerativen Systeme dient. Um ein optimales Zusammenspiel der einzelnen Systeme sicherzustellen, ist eine Überwachung der Anlage besonders in den ersten Betriebsjahren zwingend erforderlich. Für die Überwachung spielen zwei Elemente eine zentrale Rolle: erstens die Fernüberwachung mittels Anlagenvisualisierung und zweitens die Analyse von historischen Messwerten und Zuständen der Anlage aus dem Technischen Monitoring.

Über die durch das Monitoring protokollierten Daten kann die Anlage auf verschiedene Gesichtspunkte hin ausgewertet werden, z.B. ob die Regelung richtig funktioniert oder wie das Laufverhalten der Anlage ist. Im Zuge des Monitorings werden Berichte der jeweiligen Anlage erstellt, um nachzuweisen, dass die Anlage wie geplant läuft und um die Anlage ggf. zu optimieren und/oder zu parametrieren. Zur Unterstützung der Auswertung werden die Daten grafisch dargestellt, z.B. durch Linien- oder Balkendiagramme.

Die Fragestellung dieser Arbeit ist nun, welche Erkenntnisse durch die andere visuelle Aufbereitung der Daten in Form der Heatmaps gewonnen werden können. Dies erfolgt vor der Zielsetzung, eine Empfehlung darüber abzugeben, in welchem Bereich diese Darstellungsform sinnvoll eingesetzt werden kann. Hierfür wurde im Rahmen dieser Bachelor-Thesis in MATLAB ein Software-Tool geschrieben, das die Rohdaten aus dem Anlagen-Monitoring in Heatmaps umrechnet. Die durch das Tool erzeugten Heatmaps können dann in demselben Tool noch weiterverarbeitet, analysiert oder boolesche Logiken generiert werden.

Die grafische Darstellungsform Heatmap wird in der Literatur auch als Carpet-Plot bezeichnet. [1][10] In dieser Arbeit wird jedoch der Begriff Heatmap verwendet. Um die Definition der Aufgabenstellung beizubehalten.

2. Technisches Monitoring als Teil der Gebäudeautomation

Nach der VDI 6041 (Facility-Management Technisches Monitoring von Gebäuden und gebäudetechnischen Anlagen) umfasst das Technische Monitoring (TM) drei Bereiche. Das Anlagenmonitoring (AM), das Energiemonitoring (EM) und das Gebäude- und Behaglichkeitsmonitoring (GBM). Des Weiteren wird das Technische Monitoring noch zeitlich in das Einregulierungsmonitoring (ERM) und in das Langzeitmonitoring (LZM) ausdifferenziert. Eine genaue Beschreibung zum Technischen Monitoring findet sich im Punkt 2.2 Technisches Monitoring. Um das TM in seiner Struktur besser verordnen zu können, wird als erstes der Aufbau der Gebäudeautomation in seinen drei Ebenen beschrieben. [1]

2.1 Die drei Ebenen der Gebäudeautomation

In Anlehnung an die Automatisierungspyramide (Abbildung 1) aus der Industrieautomatisierung mit ihren sechs Ebenen besteht die Automatisierungspyramide der Gebäudeautomation (GA) aus drei Ebenen (Abbildung 2). Hierfür wird alles oberhalb der (Prozess-)Leitebene (Level 2,3,4) ebenfalls der Leitebene zugeschrieben, bzw. entfällt ganz. Ebenso verhält es sich mit der Prozessebne (Level 0) da ein Gebäude in seinen Prozessen statischer ist und nicht wie Produktionsprozesse dynamisch, also nicht ständig abgeändert/verändert werden muss. Deswegen entfallen wie beschrieben auch die dafür notwendigen Prozess-, Betriebsleit- und Unternehmensebenen, die für die dynamischen Produktionsprozesse notwendig sind. Das Planen von den höheren Ebenen herunter in die niedrigeren Ebenen kann durch den Begriff Parametrieren[1] ersetzt werden:

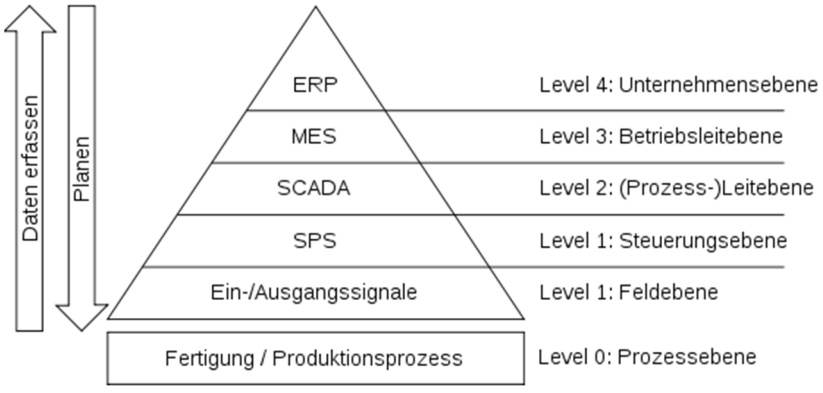


Abbildung 1: Automatisierungspyramide. [2]

[1] „Parametrieren: Festlegen der anlagenspezifischen Einstellwerte, um das gewünschte Verhalten von Anlage und System zu bewirken. Parameter können sein: Sollwerte, Grenzwerte, Kennlinien, Zeiten.“ [3]

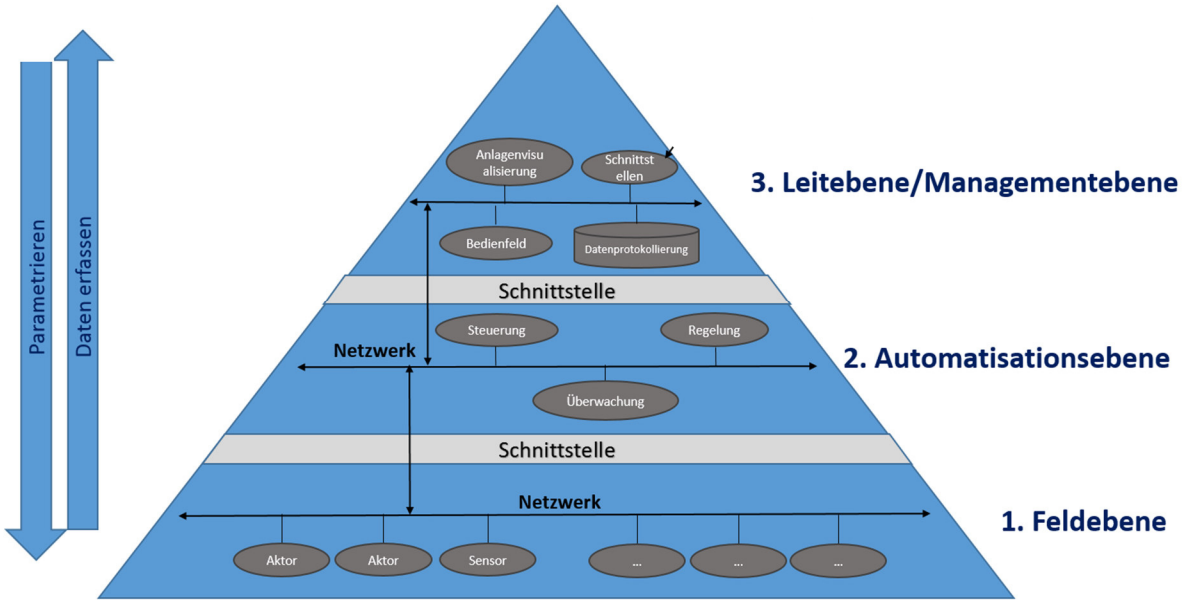


Abbildung 2: GA Pyramide [an 2 und 3]

Die Abbildung 2. zeigt eine vereinfachte Darstellung der Automatisierungspyramide der Gebäudeautomatisation. Die Grafik ist angelehnt an die Grafik aus Abbildung 1 und der detaillierten Beziehung zwischen den Komponenten und Ebenen in der Gebäudeautomatisation aus Abbildung 3

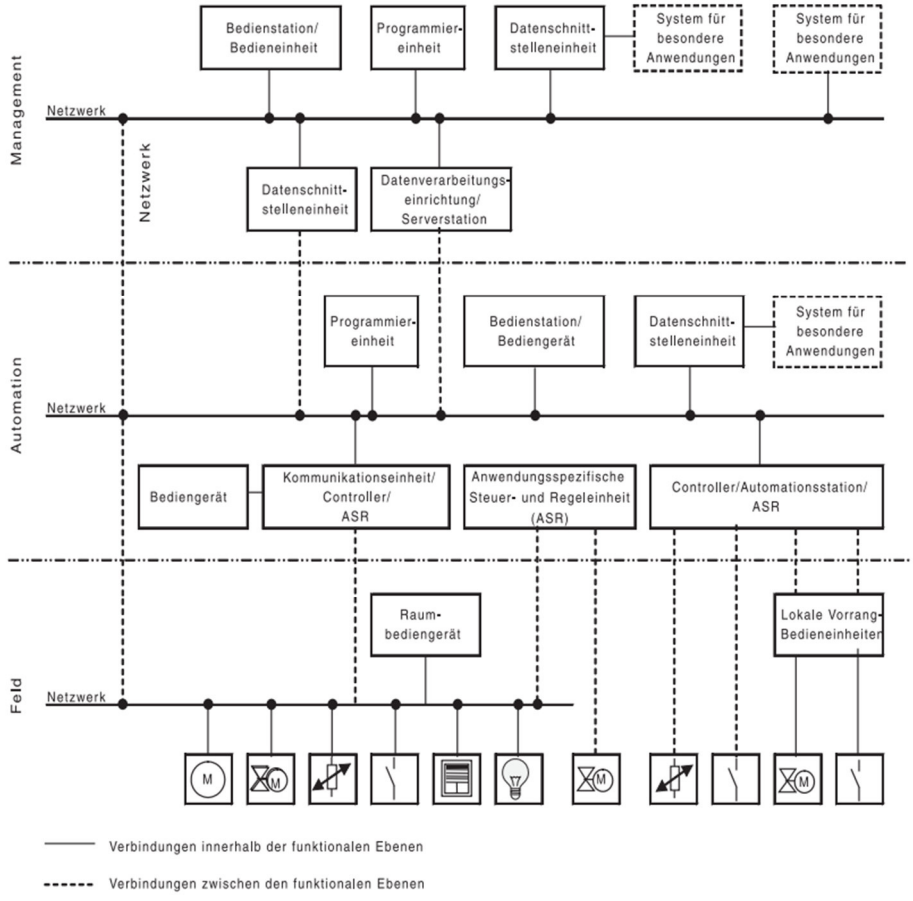


Abbildung 3: Ebenen der GA im Detail [3]

Feldebene:
Die Feldebene besteht aus allen Systemkomponenten, Sensoren und Aktoren, die über ein Bussystem[2], dem sogenannten Feldbus, miteinander kommunizieren können. Die von den Sensoren gemessenen physikalischen Größen oder Zustände werden so umgewandelt, dass sie der höheren Ebene (Automatisierungsebene) über den Feldbus zur Verfügung gestellt werden können, wo diese dann verarbeitet werden.[4] Umgekehrt können die höheren Ebenen (Automatisierungsebene und Leit-/ Managementebene) ebenfalls Signale über den Feldbus an die Elemente in der Feldebene senden.

Automatisierungsebene:
In der Automatisierungsebene befinden sich die Komponenten, die sich mit der Sammlung, der anschließenden Aufbereitung und Verarbeitung der Daten aus der Feldebene beschäftigen. Des Weiteren sind hier all die Komponenten zu finden, die zur Überwachung, Steuerung und Regelung benötigt werden. Ebenso gehört die Weiterleitung ausgewählter Daten und das Senden angefragter Daten von und an die höher gelegene Leit-/Managementebene dazu. [4]

Leit-/Managementebene:
Die Leit-/Managementebene[3] befasst sich unter anderem mit der Ausgabe und der Archivierung der von der Automatisierungsebene verarbeiteten Daten, z.B. in einer Anlagenvisualisierung und einem Datenbankserver. Außerdem gehört die Eingabe von Daten in die unteren Ebenen dazu, um z.B. eine Pumpe abzustellen oder eine Regelung mit neuen Parametern zu versorgen.[4] In dieser Ebene können auch noch komplexere Regelungen aufgebaut werden, wie z.B. Profile mit Parametern für verschiedene Jahreszeiten. Ebenso zählen zu dieser Ebene weitere Datenschnittstellen für die interne Kommunikation und Schnittstellen nach Außerhalb, um z.B. über das Internet eine Verbindung zur Anlagenvisualisierung oder zum Datenbankserver aufzubauen zu können.

Für das Technische Monitoring spielen somit alle Ebenen eine wichtige Rolle, angefangen mit der Feldebene, in der alle Komponenten für ein Messkonzept untergebracht werden müssen, der Automatisierungsebene, in der die Messwerte umgewandelt und an die Leit-/ Managementebene weiter geleitet werden, wo sie schließlich protokolliert und ausgegeben werden.

[2] „Bus: Kommunikationsmedium und -methode zwischen zwei oder mehreren Einrichtungen mit Schnittstellen für die Datenübertragung“ [5]
[3] Mit Begriff Management ist hier das Gebäudemanagement gemeint

2.2 Technisches Monitoring

> Technisches Monitoring steht als Oberbegriff für alle Arten der unmittelbaren systematischen Erfassung (Protokollierung), Beobachtung oder Überwachung eines Vorgangs oder Prozesses mittels technischer Hilfsmittel oder anderer Beobachtungssysteme mit sich wiederholender Durchführung als zentralem Element der jeweiligen Untersuchungsprogramme. Ziel des Technischen Monitoring ist es, anhand von Ergebnisvergleichen Schlussfolgerungen ziehen zu können. [1]

Das Technische Monitoring kann als Teil der Gebäudeautomation (GA) verstanden werden, da die Daten aus der GA, die in der Leit-/Managementebene anfallen, von einem Datenbank- oder Fileserver protokolliert werden. Die protokollierten Daten werden dann außerhalb der GA verarbeitet und ausgewertet.

Dies hat in der Regel drei Gründe:

1. Die Datenverarbeitung ist sehr rechenintensiv und es bietet sich an, dies lokal auf externen Systemen mit der Endanwendung zu tun. So wird keine Rechenkapazität der Systeme der GA genutzt.
2. Die Dateigrößen für die visuelle Darstellung der ausgewerteten Daten können um ein Vielfaches größer als die Rohdaten sein. Diese Daten müssen dann übertragen werden und je nach Übertragungsrate der Datenanbindung kann dies dann mehr Zeit in Anspruch nehmen, als nur die Rohdaten zu übertragen.
3. Es gilt, Sicherheitsaspekte zu beachten. Wenn die Daten voll automatisiert ausgespeist werden, kann der Zugriff von Außerhalb auf das System unterbunden werden.

Wie in Kapitel 2 schon beschrieben, umfasst das Technische Monitoring nach der VDI 6041 drei Bereiche: das Anlagenmonitoring (AM), das Energiemonitoring (EM) und das Gebäude- und Behaglichkeitsmonitoring (GBM). Zudem werden das Einregulierungsmonitoring (ERM) und das Langzeitmonitoring unterschieden. Aus den nachfolgenden zwei Abbildungen wird ersichtlich, dass sich die drei Bereiche AM, EM und GBM überschneiden. Das ERM und das LMZ sind hingegen zeitliche Einordnungen der drei Bereiche. Unmittelbar nach der Inbetriebnahme des Gebäudes oder von Anlagenteilen startet das ERM, welches nach dem Prozess der Einregulierung in das LMZ übergeht. Im Folgenden werden diese Begriffe noch weiter erläutert. [1]

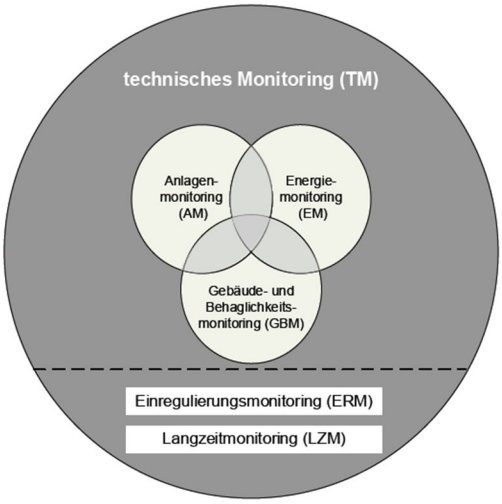


Abbildung 4: TM [1]

Aus der Abbildung 5 geht hervor, dass der Prozess des technischen Monitorings schon in der Entwicklungs-/Planungsphase des Gebäudes bzw. der Anlage beginnt. In dieser Phase muss entschieden werden, welche Bereiche abgedeckt und wie und in welcher Intensität die Daten erfasst werden sollen. Dabei wird die Intensität in drei Kategorien eingeteilt: niedrig, mittel und hoch. Bei der niedrigen Kategorie kommen auch mobile Messungen zum Einsatz und das Erfassungsintervall der Daten kann Wochen- oder Monatsweise geschehen. Die mittlere Kategorie hat eine Auflösung von Tagen oder einer Woche. Die höchste Kategorie erfasst die Daten in Minuten, bei Bedarf auch in Sekunden. Eine ausführliche Einteilung der Kategorien ist dem Anhang A1 zu entnehmen. Die Kategorien hängen dabei maßgeblich mit dem Grad der GA zusammen, da ein höherer Automatisierungsgrad auch immer mehr Zustandssignale in einer hohen zeitlichen Auflösung benötigt. Das operative Monitoring findet dann erst in der Nutzungsphase statt, somit ist die Qualität des Monitorings maßgeblich von dem Grad der GA und von der gewählten Kategorie des TM abhängig.[1]

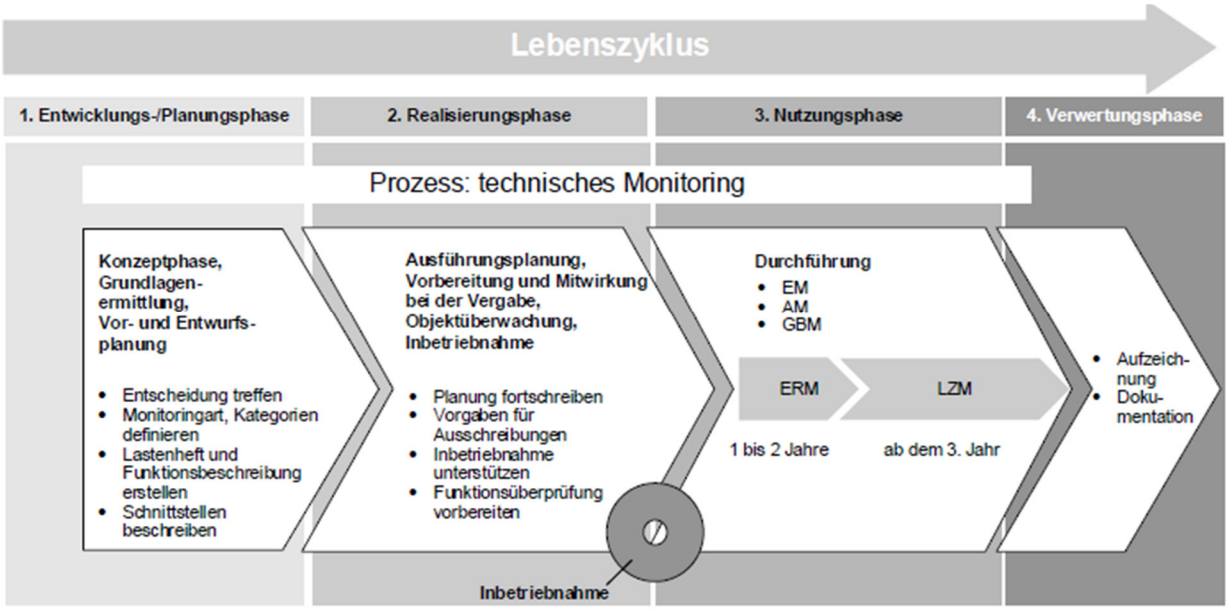


Abbildung 5: Prozess Technisches Monitoring [1]

Energiemonitoring (EM)

Das Energiemonitoring (EM) beinhaltet die Erfassung und Auswertung von Energie- und Medienverbräuchen, Leistungen und Volumenströmen bzw. Massenströmen. Die Auswertung umfasst das gesamte Objekt, im Gegensatz zum Anlagenmonitoring, welches einzelne Komponenten und Analgenteile betrachtet**.** Die Messausstattung enthält Zählwerke für Energieträger (Flüssig oder Gasförmig), Wärme/Kälte, Strom, Wasser und Dampf. Für Energieträger, die in fester Form vorliegen wie Biomasse[4], kann der Verbrauch über die Masse oder das Volumen erfasst werden.[1] Dabei kommt es zwangsweise zu Überschneidungen mit der Messtechnik des Analgenmonitorings. Das Erfassungsintervall der Daten für die Verbräuche muss hierbei nicht hoch sein, um aussagekräftige Ergebnisse für eine Bilanzierung zu erhalten, so reicht es z.B. aus, die geleistete Arbeit in einem Viertelstundentakt zu erheben. Wenn nur Verbrauchwerte in der GA erfasst werden, sollte geprüft werden ob die Zählwerke eine feinere Auflösung zu lassen, um daraus auf Leistungs- oder Durchfluss Werte zu schließen. Für Leistungen und Volumenströme/Massenströme hingegen ist ein erhöhtes Erfassungsintervall für das Anlagenmonitoring sinnvoll. Nach welchen Verfahren dies geschehen kann, ist Punkt 2.4 zu entnehmen.

Anlagenmonitoring (AM)

> „Ziel des Anlagenmonitorings (AM) ist die Erfassung und Analyse der Betriebszustände technischer Anlagen. Relevante Größen sind…“ Zustands- und Prozessgrößen, „Schaltzustände, Betriebszeiten, Störmeldungen und die Betriebsweise. Anders als im EM werden dazu einzelne Komponenten und Anlagenteile betrachtet. Dadurch werden eine Funktionskontrolle, bedarfsgerechte Wartung und Betriebsoptimierung ermöglicht. Die im AM erfassten Daten liefern eine detaillierte Grundlage zur Analyse von Fehlzuständen, welche sich auf den Energie- und Medienverbrauch (EM) auswirken.“ [1]

Gebäude- und Behaglichkeitsmonitoring (GBM)

> „Das GBM fasst die Parameter der Raumkonditionierung wie Temperatur, Helligkeit, Feuchte und Kohlenstoffdioxid-Konzentration sowie das Nutzerverhalten mit gewähltem Offset der Raumtemperatur, das Lüftungsverhalten und die Belegungsdaten zusammen. Abgesehen von der messtechnischen Erfassung der Größen sind Nutzerbefragungen ein Werkzeug, um die Raumkonditionierung und das Nutzerverhalten zu erfassen.“ [1]

Einregulierungsmonitoring (ERM)

> „Mit dem ERM beginnt die kontinuierliche Überwachung der technischen Anlagen im Gebäudebetrieb sowie deren Optimierung. Das ERM beginnt mit der Nutzung des Gebäudes und erstreckt sich über zwei Jahre…“ (Abbildung 5). „Während dieser Zeit werden detaillierte Analysen zum Energieverbrauch sowie Funktion und Leistung der

[4] Biomasse bezeichnet „den biologisch abbaubaren Teil von Erzeugnissen, Abfällen und Reststoffen der Landwirtschaft mit biologischem Ursprung (einschließlich pflanzlicher und tierischer Stoffe), der Forstwirtschaft und damit verbundener Wirtschaftszweige einschließlich der Fischerei und der Aquakultur sowie den biologisch abbaubaren Teil von Abfällen aus Industrie und Haushalten“. [6]

Anlagentechnik erstellt. Über das reale Nutzungsprofil können sich Optimierungspotenziale für den Betrieb der technischen Anlagen ergeben. Die Ergebnisse des ERM dienen der endgültigen Einstellung von technischen Anlagen und der Festlegung von Kenn- und Sollwerten, die als Basis für das nachfolgende LZM dienen.“[1]

Langzeitmonitoring (LZM)

„Das LZM beginnt nach dem ERM und erstreckt sich über die gesamte Nutzungsphase des Gebäudes…“ (Abbildung 5). „Ziel des LZM ist die Erhaltung des Betriebsoptimums, ggf. durch die Veranlassung entsprechender Maßnahmen (Nachregulierung, Instandhaltung, Behebung von Fehlern und Ausfällen, Verfolgung von Zielvorgaben). Um den im ERM ermittelten energetischen Zustand zu erhalten, werden während des LZM laufend Soll-Ist-Vergleiche durchgeführt. Durch die laufende Kontrolle der Betriebsdaten werden Veränderungen der Kennwerte sofort ersichtlich, sodass zeitnah eingegriffen und der Sollzustand wiederhergestellt werden kann. Fehler, Ausfälle und höhere Energieverbräuche an technischen Anlagen können über das LZM festgestellt werden. Bei signifikanten Änderungen der Anlagentechnik oder Nutzung kann erneut ein ERM erforderlich sein.“ [1]

In dieser Arbeit wird sich mit dem Anlagen- und Energiemonitoring beschäftigt, da die Gebäudeautomatisation in den zu untersuchenden Objekten nur für diese Bereiche installiert wurde und somit auch nur Daten zu diesen Bereichen vorliegen. Es sollten aber der Vollständigkeit wegen alle Bereiche des TM vorgestellt werden.

2.3 Datenbeschaffung und Datenaufbau

Die Datenbeschaffung findet über den Datenbank-Server auf der Leit-/Managementebene statt, konkret für die in dieser Arbeit betrachteten Anlagen über die Software „AVISO - Chart Viewer“ von der Firma „IMAS Falkenberg Meßtechnik GmbH“. Die Software baut eine Datenbankverbindung auf, in der alle integrierten Datenpunkte in eigens erstellten Chart-Vorlagen eingebunden und angezeigt werden können. Eine automatische Visualisierung der Daten in Form von Liniendiagrammen findet dann lokal auf dem Rechner mit der Endanwendung „Chart Viewer“ statt. Hierfür werden nur die Roh- und Konfigurationsdaten der ausgewählten Datenpunkte vom Server an die Endanwendung übertragen. Der Chart-Viewer besitzt zusätzlich noch eine Exportfunktion, um die Daten als Grafik-File (jpg, png etc.) oder Datenfile (SVG, txt, xml etc.) zu exportieren. Diese exportierten Datenfiles werden für die Weiterverarbeitung zu Heatmaps genutzt. Die Tabelle 1 (nächste Seite) zeigt den Datenaufbau, wie er den exportierten Datenfiles entnommen werden kann. Es handelt sich hierbei um Daten, die aus einer Namenskennzeichnung in der ersten Zeile bestehen, in der dann ein Zeitstempel sowie die Messwerte fortlaufend protokolliert werden. Die Anzahl der Datenpunkte wird dabei nur von dem Format der exportierten Dateistruktur beschränkt.

Tabelle 1: Datenstruktur

TT.MM.JJJJ HH:MM:ss	Messwerte
01.01.2016 00:00:01	100
01.01.2016 00:30:00	160
01.01.2016 01:00:55	120
01.01.2016 02:00:15	100
…	…
…	…
…	…
31.12.2016 23:30:05	0
31.12.2016 23:59:44	110

Je nachdem, was erfasst werden soll, können die Datenpunkte in der GA in verschiedene Typen aufgeteilt werden. In der Tabelle 2 ist aufgelistet, welche Daten typischerweise protokolliert werden. Die kursiven Datenpunkttypen enthalten anstelle eines Messwertes Informationen als Text darüber, welches Ereignis vorgefallen ist. Diese Daten werden zwar nicht weiterverarbeitet, aber ihre Information ist für die Auswertung ebenfalls relevant, da dort schnell deutlich wird, welche Parameter, manuelle Eingriffe, Störungen etc. vorlagen.

Tabelle 2: Datenpunkttypen

Datenpunkttypen	Beschreibung
IST-Werte (Messwerte)	Zustands- und Prozessgrößen
SOLL-Werte	Zustands- und Prozessgrößen.
Virtuelle Datenpunkte	z.B. aus zwei oder mehr IST-Werten
Statusmeldungen	*An/Aus, Auf/Zu, Automatik/Manuell, Wartung etc.*
Regelwerte	*Nachlaufzeit, Zeitschaltprogramme, Offset etc.*
Störmeldungen	*Ventil klemmt, Motor kaputt etc.*
Warnmeldung	*z.B. Sollwert überschritten*

„Lassen sich physikalische Parameter nicht direkt messen, so sind geeignete Methoden zu beschreiben, wie und mit welcher Genauigkeit die benötigten Daten aus anderen Größen berechnet werden können …“. Diese abgeleiteten Größen werden als virtuelle Datenpunkte bezeichnet. „Sie entstehen durch Verknüpfung von Messwerten sowie durch Anwendung von Rechenvorschriften und/oder statistischen Methoden.“[1] Ein Beispiel für einen virtuellen Datenpunkt ist die Leistungszahl von Wärmepumpen oder von Kältemaschinen. Für virtuelle Datenpunkte sollte immer im Vorfeld überprüft werden, ob die zeitlichen Erfassungsintervalle der Quelldaten für die Berechnungen aufeinander abgestimmt sind. „Bei einer kurzen Berechnungsdauer können die virtuellen Datenpunkte wie reguläre Messwerte gespeichert werden oder aber, wenn eine Datenverdichtung bewirkt werden soll, kann z.B. eine stündliche Abspeicherung eines akkumulierten Werts anstelle von hochfrequenten Rohdaten erfolgen.„ [1]

2.4 Verfahren zum Protokollieren der Daten

Neben den in dem vorherigen Punkt erwähnten Rohdaten spielt das Erfassungsintervall eine wichtige Rolle. Die Tabelle 3 zeigt exemplarisch, aus wie vielen Datensätzen nur ein einziger Datenpunkt für einen Tag oder ein Jahr mit 365 Tagen bei verschieden gewählten Erfassungsintervallen bestehen muss.

Tabelle 3: Menge an Daten pro Datenpunkt

Erfassungsintervall in Minuten	**10**	**5**	**3**	**2**	**1**
Erfasste Datenpunkte am Tag	144	288	480	720	1.440
Erfasste Datenpunkte im Jahr	52.560	105.120	175.200	262.800	525.600

Je nach Größe der Anlage besteht diese aus mehreren Dutzend bis mehreren Hundert Datenpunkten, die erfasst werden können. Um die Daten zu reduzieren, wird nicht wahllos jeder Datenpunkt zu jeder Minute oder in einer noch kleineren Zeiteinheit protokolliert, sondern verschiedene Verfahren genutzt, um die Anzahl der Datensätze zu reduzieren. Im Folgenden werden drei wichtige Verfahren vorgestellt.

Voreinstellung des Erfassungsintervalls (samplebasiert):

Bei dieser Variante wird ein festes Erfassungsintervall (sample) definiert und die Daten als äquidistante Zeitreihe gespeichert. [1] Die Vorteile sind eine einfache Realisierung und, dass diese Messwerte als Referenzwerte dafür dienen können, ob die Protokollierung oder das zu protokollierende System ausgefallen ist. Als Nachteil kann aufgeführt werden, dass Änderungen zwischen den Intervallen nicht erfasst werden.

Erfassen durch Änderung des Messwertes (Change of Value):

Eine weitere Option ist die Speicherung der Werte nach der Abhängigkeit der Änderung der Messwerte, auch „Change of Value“ genannt [1]. So wird der Wert erst gespeichert, wenn sich dieser vom letzten gespeicherten Wert um eine vordefinierte Größe/Steigung ändert. Die Vorteile sind, dass ein konstanter Wert über einen längeren Zeitraum nur einmal erfasst wird und das eine gute Rekonstruktion der Daten möglich ist. Der Nachteil ist, dass bei über längerer Zeit gleichbleibenden Werten nur ein Mal ein Wert gespeichert wird, und somit keine Information vorliegt, ob das System eventuell ausgefallen ist.

Analyse der Daten mit anschließender Speicherung:

Hierbei werden die Daten über einen Zeitraum hinweg vorgehalten und analysiert. Es werden dann nur Daten-Punkte gespeichert wie z.B. Minima, Maxima oder der Beginn und das Ende einer konstanten Steigung. Der Vorteil ist, dass die Daten auf das Nötigste reduziert und exakt rekonstruiert werden können. Der Nachteil ist, dass das Vorhalten von Messwerten und deren Auswertung über einen Zeitraum mehr Speicher- und Rechenkapazität benötigt.

Welches Verfahren genutzt werden sollte, hängt vom Datenpunkt ab, so ist z.B. eine samplebasierte Protokollierung mit einem Viertelstundenzyklus von Verbrauchwerten, wie es im Energiemonitoring vorkommt, durchaus geeignet um genaue Auswertungen für Bilanzierungen vornehmen zu können. Wenn nur Verbrauchwerte in der GA erfasst werden, sollte geprüft werden ob die Zählwerke eine feinere Auflösung zu lassen, um daraus auf Leistungs- oder Durchfluss Werte zu schließen. Für Datenpunkte wie z.B. Leistung, Durchfluss oder Temperaturen empfehlt sich eine Kombination aus samplebasierten Verfahren und Change of Value, um die jeweiligen Nachteile zu kompensieren. Das letzte Verfahren wird in der Regel nur dann verwendet, wenn man die zu übertragenden Daten aufgrund von sehr niedrigen Übertragungsraten aufs Nötigste reduzieren muss.

2.5 Beurteilung der Qualität der protokollierten Messwerte

Die Qualität der Messwerte kann in zwei Kategorien eingeteilt werden, erstens in die Qualität der Erfassung und zweitens in Mess- und Umwandlungsfehler. Die Beurteilung der Qualität Erfassung der Daten ist anhand der protokollierten Daten nur schwer möglich. Es müsste das Verfahren, das für die Protokollierung zuständig ist, angeschaut werden. Anhand der Datendichte und Erfahrungswerte kann nur eine grobe Einschätzung abgegeben werden, wie sich ein bestimmter Messwert verhalten sollte, und damit indirekt eine Aussage über die Datenqualität. Aus diesem Grund ist es wichtig, schon in der Planungsphase exakt festzulegen, wie die Daten erfasst werden sollen. Mess- und Umwandlungsfehler können nur erkannt werden, indem im Anlagensystem zusammenhängende Datenpunkte betrachtet werden, beispielsweise indem nach Abweichungen in Volumen-/Massenströmen gesucht wird, die Rücklauftemperatur von System A zur Vorlauftemperatur zu System B verglichen wird oder Leistungsdaten von Anlagen mit den technischen Spezifikationen der Hersteller verglichen werden. Eine Überprüfung Mess- und Umwandlungsfehler sollte daher zu Beginn des Einregulierungsmonitoring vorgenommen werden, um für die späteren Auswertungen im technischen Monitoring eine aussagekräftige Datengrundlage zu besitzen.

3. Vergleich der Darstellungsarten Liniendiagramm und Heatmaps

Es gibt eine Vielzahl von Diagrammtypen für diverse Anwendungen, die auch im Technischen Monitoring ihre Anwendung finden. Da das Liniendiagramm die Form ist, in der die Rohdaten üblicherweise dargestellt werden, wird in diesem Kapitel das Liniendiagramm der Heatmap gegenübergestellt. Bei den beiden Diagrammen handelt es sich um zweidimensionale Diagramme mit einer Abszisse, Ordinate und einem kartesischen Koordinatensystem. Es wird davon ausgegangen, dass die Informationen in einem bestimmten Format vorliegen, so besteht ein Informationssatz immer aus einem Zeitwert (Datum/Uhrzeit) und dem dazugehörigen Messwert, wie in 2.3 Datenbeschaffung und Datenaufbau dargelegt wurde. Nachfolgend werden die beiden Diagrammtypen zunächst erläutert, um dann direkt miteinander verglichen zu werden. Des Weiteren wird in dem Kapitel 5 erörtert, wie die Heatmap sich für das Technische Monitoring eignet.

3.1 Liniendiagramme

Liniendiagramme sind Diagramme, in denen die abgetragenen Messwerte durch Linien verbunden werden. Es handelt sich in der einfachsten Form im Grunde genommen um eine grafische Interpolation zwischen den Datenpunkten, durch deren Verbindung mit Geraden. Für die Interpolation können verschiedene mathematische Verfahren genutzt werden. Das üblichste Verfahren ist die lineare Interpolation, in der die Steigung zwischen den beiden Punkten berechnet wird und somit jeder Wert zwischen den Punkten ermittelt werden kann. Es können aber auch Verfahren genutzt werden, in denen die Punkte z.B. durch Kurven verbunden werden, dazu werden mehrere benachbarte Punkte in die Berechnung mit einbezogen.

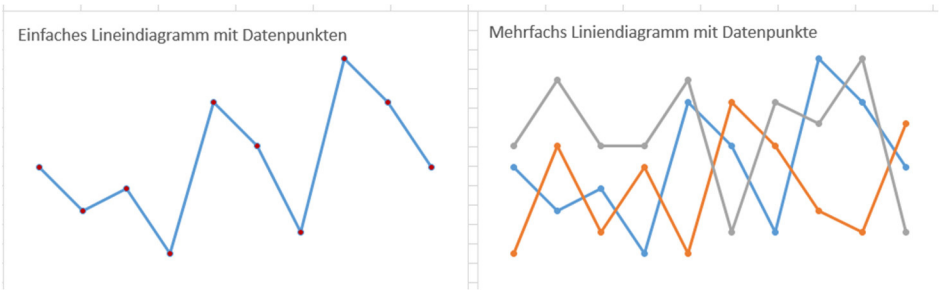


Abbildung 6: Liniendiagramme

3.1.1 Vor- und Nachteile des Liniendiagramms

Im Folgenden werden die allgemeinen Vor- und Nachteile des Liniendiagramms aufgelistet, das Verhalten zur Visualisierung von TM-Daten wird hier noch nicht betrachtet.

Vorteile:

Die Vorteile der Liniendarstellung liegen darin:

1. dass problemlos mehrere Datenreihen in einem Diagramm dargestellt werden können. Nur das Kriterium der Übersichtlichkeit beschränkt die Anzahl der abgebildeten Datensätze.
2. Darüber hinaus können auch mehrere Achsen in einem Diagramm verwendet werden, wodurch unterschiedliche Datenreihen den Achsen zugeordnet und so besser dargestellt werden können.

Nachteile:

Die Nachteile gestalten sich darin:

1. dass die Interpolation dazu führt, dass das Liniendiagramm suggeriert, zwischen den Datenpunkten bestünde eine lineare Verbindung, die aber eventuell nicht existiert. Um das eventuelle Problem der Linearität zu beheben, bieten sich verschiedene mathematische Verfahren an, aber auch hier sollte man sich sicher sein, ob das Verfahren zur Darstellung der Daten geeignet ist, oder ob man mit der Darstellung aus Gründen der Vereinfachung gut arbeiten kann.
2. Des Weiteren kann das Liniendiagramm suggerieren, dass mehr Datenpunkte existieren als tatsächlich vorhanden sind, da an jeder Stelle auf der Linie der zugehörige Wert auf der Abszisse und Ordinate abgelesen werden kann. Wenn die Daten so erfasst wurden, dass die Interpolation eine gute Rekonstruktion erlaubt, spielt dieser Punkt keine große Rolle und es können problemlos auch interpolierte Werte benutzt werden.
3. Bereiche hoher Frequenzen, in Bezug auf das Diagramm, lassen sich nicht mehr gut erkennen. Unter hoher Frequenz ist in einem Diagramm zu verstehen, dass sich der Wert in Bezug auf seine angezeigte Achse (Zeitraum) schnell ändert. Dabei spielt es keine Rolle welchen Zeitraum das Diagramm anzeigt. In der Abbildung 7 sieht man eine Sinuskurve deren Frequenz sich stückweise erhöht. Bei ganz hohen Frequenzen sehen die Werte dann nur noch wie ein durchgehendes Band aus und es kann nicht mehr erkannt werden, welche Werte vorliegen.

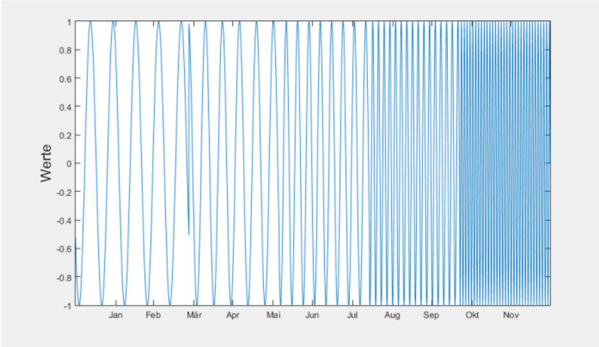


Abbildung 7: Frequenz im Diagramm

3.2 Heatmaps

„Eine Heatmap (englisch heat = ‚Hitze', ‚Wärme'; map = ‚Karte') ist ein Diagramm zur Visualisierung von Daten, deren abhängige Werte einer zweidimensionalen Definitionsmenge als Farben repräsentiert werden. Sie dient dazu, in einer großen Datenmenge intuitiv und schnell besonders markante Werte zu erfassen." [7]

Die für diese Bachelorarbeit verwendete Form der Heatmap ist so angeordnet, dass auf der Abszisse die Tage des Datensatzes aufgetragen werden und auf der Ordinate die dazugehörigen Minuten des Tages. In deren Schnittpunkt werden dann die qualitativen Werte

der Messungen von Temperatur, Leistung, etc. als Farbe repräsentiert. Bei dieser Form von Heatmap spricht man auch von einem Rasterdiagramm oder Carpet-Plot, dennoch wird die Bezeichnung Heatmap in dieser Arbeit weiter verwendet, da der Titel der dieser Bacherlor-Thesis nicht mehr geändert werden konnte.

Abbildung 8 zeigt den strukturellen Aufbau der oben beschriebenen Heatmap. In den Zellen, in denen es harte Schnittpunkte gibt, sind Zufallszahlen von 1 bis 10 eingetragen und der Zellenhintergrund ist in Abhängigkeit des Zahlenwertes eingefärbt. Dabei geht die Farbskala von 0=grün über gelb bis 10=rot.

1440	4	0	5	...	...	...	2	8	9
1439	6	8	7	...	...	...	5	8	3
1438	10	6	4	...	...	...	0	6	1
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
3	1	9	9	...	...	...	3	4	9
2	5	2	4	...	...	...	0	2	8
1	2	3	10	...	...	...	4	9	5
Minuten / Tage	1	2	3	...	...	...	364	365	(366)

Abbildung 8. Heatmap Struktur

Aus dieser Struktur ergeben sich dann folgende Heatmaps (Abbildung unten), in denen nur noch die repräsentativen Farben dargestellt werden. Zur Generierung der Diagramme wird MATLAB benutzt, was in Kapitel 5 weiter ausgeführt wird. Da nicht zu jedem Zeitpunkt – in diesem Fall in Minuten – Messwerte protokolliert werden, muss zwischen den einzelnen Messwerten interpoliert werden. Dies geschieht in dem programmierten Tool durch eine lineare Interpolation (siehe 4.2.2.3). Den Unterschied zwischen den originalen und den interpolierten Daten ist in Abbildung 9 dargestellt.

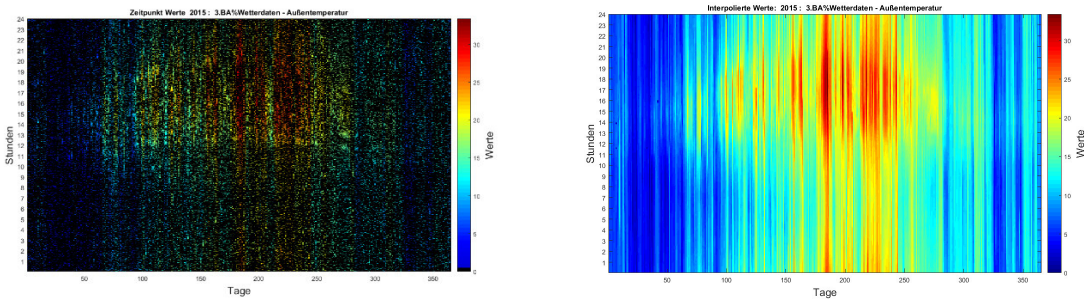


Abbildung 9: Heatmap links originale Daten, rechts interpolierte Daten

3.2.1 Vor- und Nachteile von Heatmaps

Im Folgenden werden die allgemeinen Vor- und Nachteile von Heatmaps aufgelistet, das Verhalten zur Visualisierung von TM Daten wird hier noch nicht betrachtet.

Vorteile:

Der große Vorteil der Heatmap besteht darin, dass die Werte einfach zeitlich verordnet werden können und Muster schnell zu erkennen sind. So bekommt man einen Überblick darüber, wie sich die jeweilige Messung über die 1440 Minuten bzw. 24 Stunden verhält, was anhand des Liniendiagramms in der Abbildung 10 schwerer abzuschätzen ist. Des Weiteren lassen sich Werte gut erkennen, die für ein Liniendiagramm schon zu hochfrequent wären (Siehe 3.1.1 Nachteile).

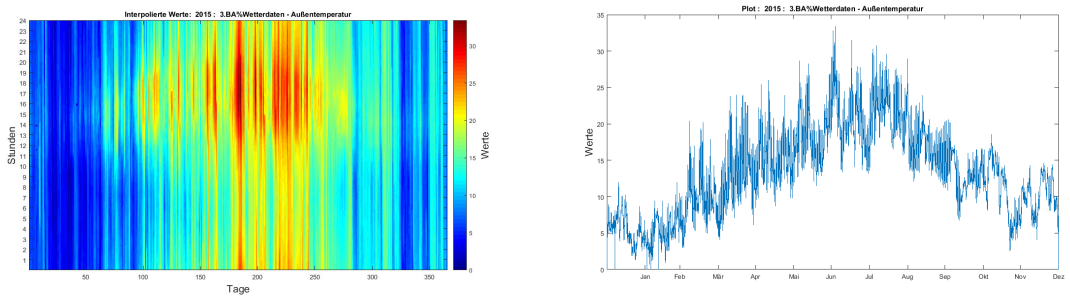


Abbildung .10: Heatmap und Liniendiagramm derselben Daten

Nachteile:

Die Nachteile werden der Übersichtlichkeit wegen in kurzen Stichpunkten aufgeführt. Bei den Punkten zwei und drei sind die Nachteile deckungsgleich mit jenen der Liniendiagramme und sie können der Datenerfassung sowie Datenaufbereitung zuordnen. Da sie sich aber unmittelbar auf die Visualisierung auswirken, sind diese Punkte jeweils auch in den Nachteilen mit ausgelistet.

1. Bei der Heatmap kann pro Diagramm nur ein Datensatz angezeigt werden, bei den Liniendiagrammen hingegen ist diese Einschränkung nicht gegeben.
2. Bei der Interpolation zwischen den Datenpunkten wird vorausgesetzt, dass sich die Werte zwischen zwei Datenpunkten linear zu diesen verhalten, dies muss aber nicht zwangsläufig der Fall sein.
3. Ebenso wirkt sich die Interpolation negativ auf eine klare Aussage über die Datengrundlage aus, da es erst einmal eine hohe Datendichte suggeriert.
4. Die Farbdarstellung der additiven und subtraktiven Farbmischung kann zu Problemen führen. Bei der additiven Farbmischung wird verschiedenfarbiges Licht überlagert, um eine gewünschte Farbe zu erhalten, diese kommt z.B. bei Farbdisplays zum Einsatz. Die

subtraktive Farbmischung hingegen basiert auf der Absorption von Farbanteilen des Lichts, dies wird z.B. beim Drucken verwendet, indem mehrere Farblagen übereinander gedruckt werden und jede Lage einen bestimmten Farbanteil absorbiert. Das Problem ist, dass bei jeder Umwandlung von der additiven in die subtraktive Farbmischung, z.B. beim Drucken aus dem PC, diese umgerechnet werden muss. Dabei kommt es zwangsläufig zu Verfälschungen. Hinzu kommt noch, dass bei Displays die Helligkeit, der Kontrast und andere Farbwerte eingestellt werden können und so dieselbe Grafik auf einem abweichend eingestellten Display oder einem Ausdruck komplett anders wirken kann. Da in den Heatmaps die Werte über die Farbe dargestellt werden, sollte dies immer mitberücksichtigt werden. Ebenso sollte man erwägen, dass Personen mit Farbfehlwahrnehmungen mit dieser Auswertungsform Probleme haben dürften.

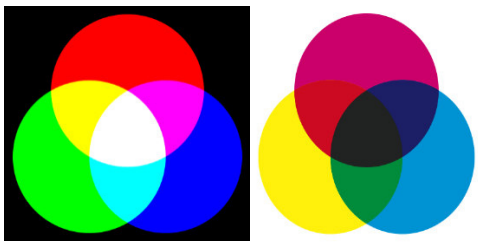


Abbildung 11: Additive und subtraktive Farbmischung

3.3 Direkter Vergleich: Liniendiagramm & Heatmap

Für den direkten Vergleich des Liniendiagramms mit der Heatmap werden die Messwerte für die Leistung eines Kessels für das Jahr 2015 betrachtet. Wie eine ausführliche Analyse der Darstellungsart der Heatmap im technischen Monitoring genutzt werden kann, wird im Kapitel 5 beschrieben. Die Abbildung 12 zeigt einen Jahresverlauf der thermischen Leistung eines Kessels in kW. Aufgrund des scheinbar breiten schwarzen Bandes, das sich durch das Diagramm mit Unterbrechungen zieht, kann man auf eine hohe Datendichte und eine hohe Änderungsrate (Frequenz) zurückschließen. Dies macht es aber unmöglich zu erkennen, welche Werte in diesen Segmenten abgebildet sind.

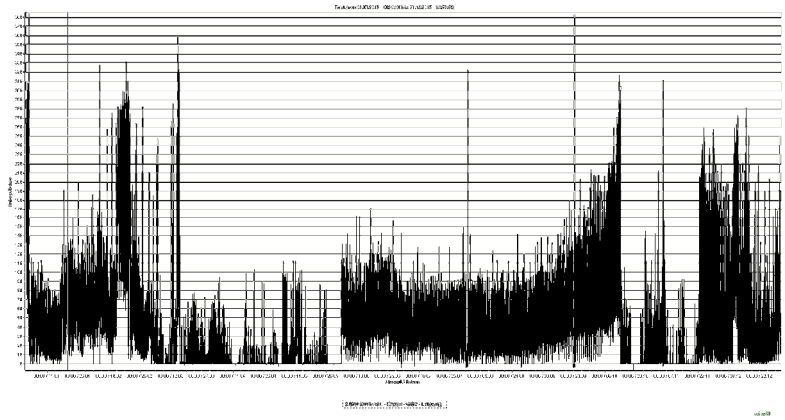


Abbildung 12: Kesselleistung in kW 2015 Liniendiagramm

Hier wird schnell deutlich, welchen Vorteil die Heatmap gegenüber dem Liniendiagramm hat, so kann problemlos eingeordnet werden, welche Werte vorliegen. Es ist aber auch zu sehen, dass eine exakte Zuordnung des Zahlenwertes aufgrund der Farbübergänge, in denen die Werte dargestellt werden, nicht mehr möglich ist.

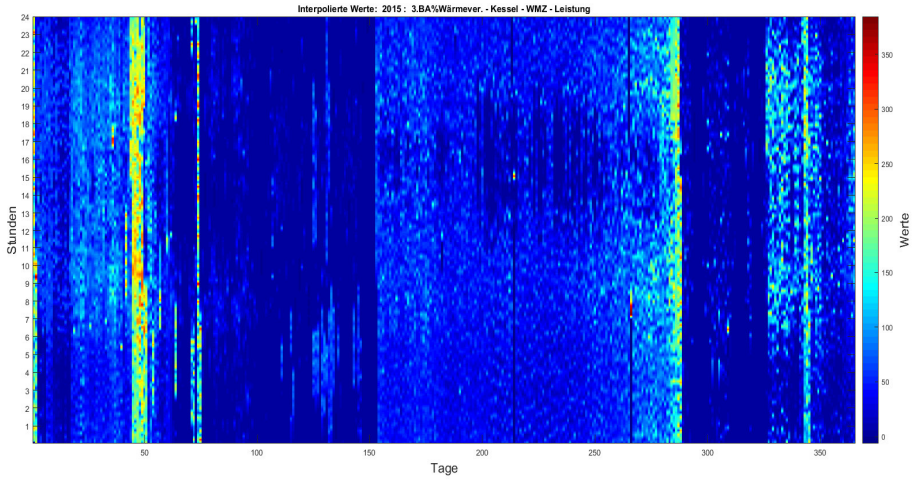


Abbildung 13: Kesselleistung 2015 Heatmap

In der Abbildung 14 ist die Heatmap über das Liniendiagramm gelegt worden, um dies noch ein Mal gegenüberzustellen.

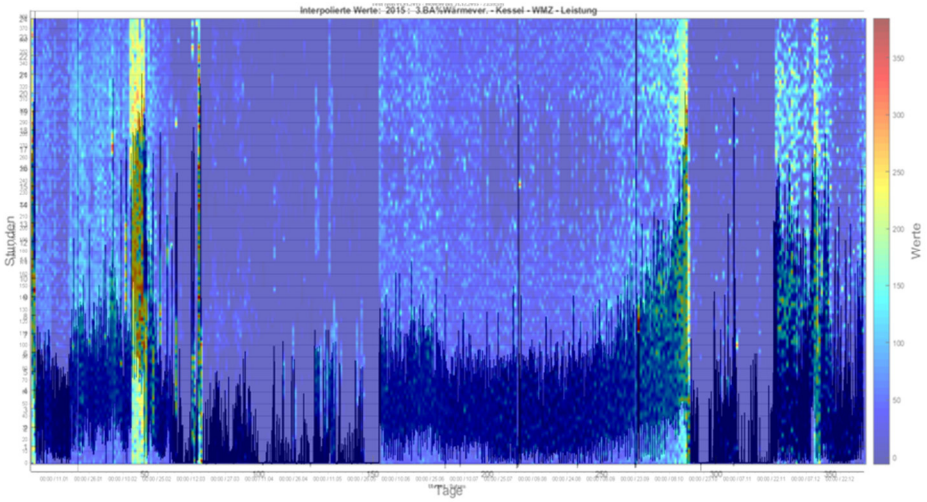


Abbildung 14: Überlagerung Liniendiagramm und Heatmap Kesselleistung 2015

Zusammenfassend kann festgehalten werden, dass die Heatmap ihre Stärke in der schnellen Erfassung von großen Datenmengen über einen großen Zeitraum hat, die auch noch bei einer schnellen zeitlichen Änderung der Messwerte noch gut erfasst werden kann.

4. Beschreibung der Software

In diesem Kapitel wird beschrieben, welche Programmier-Software und welche Dateiformate (File-Typen) für die Anwendungssoftware genutzt wurden. Zudem wird die Struktur des geschriebenen Programmcodes mit Struktogrammen dargestellt und im Fließtext beschrieben, was der Programmcode in den einzelnen Funktionen berechnet. Der Programmcode ist mit Kommentaren als MATLAB-Files und HTML-Dokument und auf dem digitalen Datenträger im Anhang zu finden.

Der Programmcode für die Anwendungssoftware wurde mit der Software MATLAB geschrieben. MATLAB ist eine Abkürzung für MATrix LABoratoty und eine kommerzielle Software des Unternehmens MathWorks zur Lösung komplexer mathematischer und technischer Probleme sowie zur grafischen Darstellung der Ergebnisse.

Für das Programmieren der Anwendungssoftware wurde MATLAB gewählt, da das Unternehmen BZE einerseits eine Lizenz besitzt und andererseits, da MATLAB über viele offene Standardfunktionen verfügt, die die Weiterverarbeitung und die grafische Darstellung vereinfacht. So gibt es z.B. eine integrierte Umgebung, um eine grafische Benutzeroberfläche zu erstellen, die sogenannte GUIDE[5]. Ein weiteres Argument ist der volle Zugriff auf den Quellcode der MATLAB-Funktionen, womit eine weitere Adaption der Funktionen/Software möglich ist. Des Weiteren ist es möglich, den MATLAB-Programmiercode in andere Programmiersprachen, wie z.B. JAVA oder C++, zu implementieren.

4.1 Verwendete MATLAB-Files

Für den erstellten Programmcode wurden drei Typen von MATLAB-Files genutzt: einmal das .m-File, welches den MATLAB Quellcode enthält, dann das .mat-File das gespeicherte Daten wie Variablen, Matrizen und Arrays enthalten kann und der dritte Typ ist das .fig-File, welches die Informationen, wie die grafischen Benutzeroberflächen-Elemente angeordnet sind, enthält.

Tabelle 4: Verwendete MATLAB-File-Typen

.m File (Quellcode)	.mat File (Daten)	.fig File
Enthält die Anweisungen, die ausgeführt werden sollen	Enthält Variablen, Matrizen und Arrays	Informationen zu den einzelnen grafischen Elementen der erstellten Benutzeroberflächen

[5] graphical user interface development environment, wörtlich aus dem Englischen übersetzt = Grafische Benutzeroberflächenentwicklungsumgebung

4.2 Aufbau des erstellten MATLAB Codes (Heatmap-Toolbox)

Dieser Abschnitt dient dazu, sich im Quellcode der einzelnen Dateien zurecht zu finden. Es wird beschrieben, wie die selbst gewählte Syntax zur Kommentierung und Differenzierung von dem mit MATLAB generierten Code und dem selbst geschriebenen Code aufgebaut ist.

Alle Funktionen, die mit dem GUIDE erstellt wurden, sind daran zu erkennen, dass zu dem .m-File noch ein gleichbenanntes fig-File vorliegt. In diesen Files steht der von MATLAB generierte Code ohne einen Tab-Sprung in dem File, alles andere mit mehr als einem TAB-Sprung wurde gecodet. Alle anderen Files wurden bis auf eine Ausnahme vollständig selbst gecodet und sind der Struktur mit dem Tab-Sprung nicht unterworfen. Als externer Funktionsbaustein wurde der Funktionssatz „setfigdocked“[6] aus der Mathworks Community des MATLAB File-Exchange genutzt. Dieser Funktionsbaustein ermöglicht es, Plots Gruppen zuzuweisen und alle Plots derselben Gruppe in einem Grafik-Container darzustellen. Bei längeren Code-Abschnitten, die zusammen gehören, wird der Abschnitt mit „**%$“** gestartet und mit **„%/“** geschlossen, um eine schnellere Orientierung zu gewährleisten. Bei kürzeren Abschnitten ist der Code durch eine Leerzeile getrennt. Ein Funktionsaufruf einer selbst geschriebenen Funktion wird mit „**%§§§§§§§§§§§§§§§§ %FUNCTION**“ gekennzeichnet. In der Tabelle 5 ist die Syntax noch einmal aufgelistet.

Tabelle 5: Syntax Kommentierung

Längerer Abschnitt	**Kürzerer Abschnitt**
%S Kurzbeschreibung Codezeile 1 … … … Codezeile n %/ Kurzbeschreibung	% Kurzbeschreibung Codezeile 1 Codezeile 2 Codezeile 3 (Leerzeile) Codezeile 4 (Neuer Abschnitt)
Funktionsaufruf	
%§§§§§§§§§§§§§§§§ %FUNCTION Funktionsname	

4.3 Anwendungssoftware „Heatmap Toolbox“

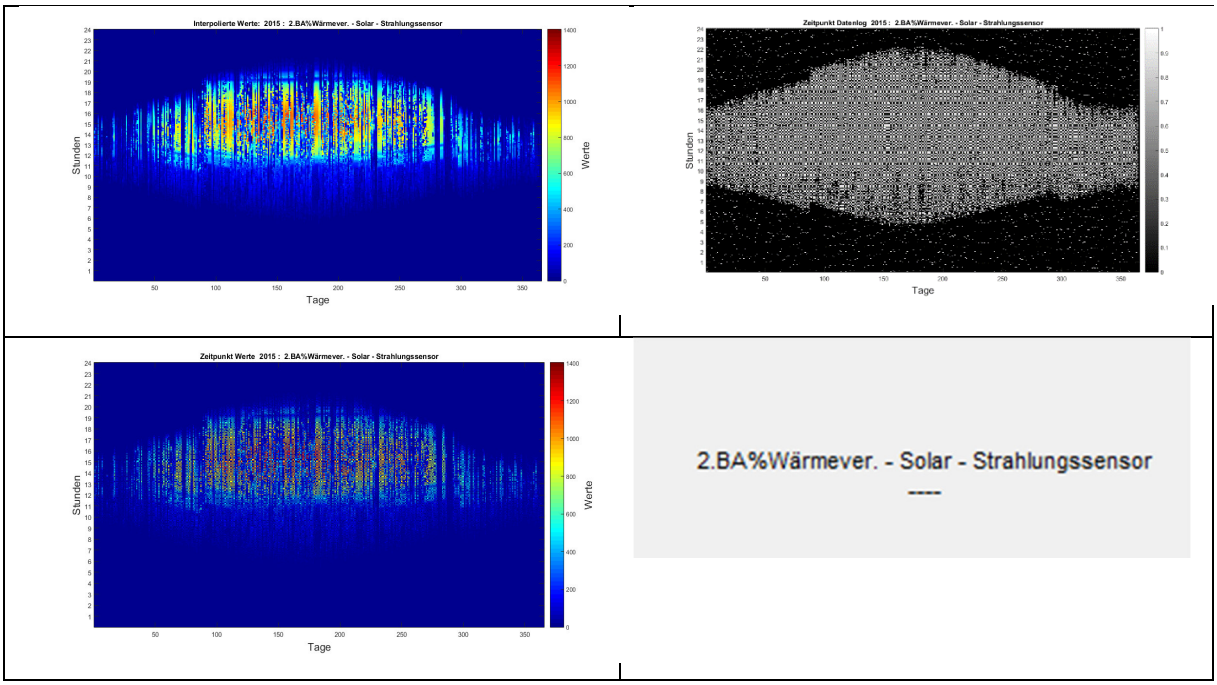
Die Anwendungssoftware „Heatmap Toolbox“ stellt als Basisfunktion die Aufbereitung der Rohdaten aus dem Technischen Monitoring bereit (Kapitel 2), in dem Rohdatensätze eingelesen, weiterverarbeitet und in die normierte Struktur von Matrizen geschrieben werden. Die Struktur der Matrizen besteht aus den Minuten eines Tages (1440), die den Zeilenindex darstellen und der Anzahl der Tage im Jahr (365 oder 366) die den Spaltenindex repräsentieren.

[6] Eine detaillierte Beschreibung zu diesem Baustein findet man unter [https://de.mathworks.com/matlabcentral/fileexchange/18106-manage-and-dock-figures-into-group].Abgerufen am 20.12.2016

Es werden hierbei folgende Matrizen generiert: die linear interpolierten Werte aus den Rohdaten, eine Matrix, die mit einem Wahrheitswert festhält, zu welchem Zeitpunkt ein Messwert protokolliert wurde, und eine Matrix, die in diesem Zeitpunkt den Messwert schreibt. Zudem wird noch in einem Array festgehalten, um welchen Datensatz es sich handelt, und in einer Variable, zu welchem Jahr dieser gehört. Alle diese Daten werden in einer Datei abgelegt und gespeichert, um diese dann weiter verarbeiten zu können.

Die Matrizen und das Array werden dann über eine grafische Ausgabe visualisiert. Zur Veranschaulichung sind die 3 Matrizen und das Array unten in der Tabelle 6 dargestellt, wie sie in der Ausgabe über eine Registerkarte aussehen. Links oben sind die interpolierten Werte zu sehen, rechts oben die Wahrheitswerte zu den Erfassungszeitpunkten, links unten die Messwerte zu den Erfassungszeitpunkten und rechts unten die Ausgabe der Information zum Datensatz.

Tabelle 6: Ausgabe der Daten



Die interpolierten Daten können noch analysiert oder mit anderen Daten verarbeitet werden. Die Analyse erfolgt über die Standard-Tools von MATLAB oder über ein Analyse-Tool das für die Toolbox geschrieben wurde, was in Unterkapitel 4.4.3.1 näher ausgeführt wird. Die Weiterverarbeitung der generierten Datensätze erlaubt es, die interpolierten Daten miteinander zu verarbeiten, um beispielsweise zwei Datensätze miteinander zu addieren oder eine Logik (Boolsche Wahrheitstablle) aus den interpolierten Daten zu generieren. Diese Logiken können dann auch wiederrum mit anderen Datensätzen verarbeitet werden.

4.4 Beschreibung der „Heatmap-Toolbox“

In diesem Unterpunkt werden die einzelnen Funktionen der Heatmap-Toolbox in Form eines Fließtextes beschrieben. Dabei werden zusammenhängende Funktionen mit demselben Index gekennzeichnet. Die Funktion „HeatmapToolbox.m“ ist ein grafisches Auswahlmenu, in dem auf die verschiedenen Funktionen der Toolbox zugegriffen werden kann.

4.4.1 HeatmapToolbox.m (.fig)

Die Funktion HeatmapToolbox.m stellt folgende sechs Funktionen zur Verfügung:

1. neue Rohdatensätze auswählen und das anschließende Generieren von Heatmap-Datensätzen
2. vorhandenen Heatmap-Datensätze analysieren
3. vorhandene Heatmap-Datensätze bearbeiten. Daten normieren, abschneiden oder einfache mathematische Berechnung durchführen
4. zwei Heatmaps mit einander kombinieren/manipulieren.
5. Logik-Aufbau mit Heatmap-Datensätzen
6. Colormap Editor (Funktion von MATLAB)

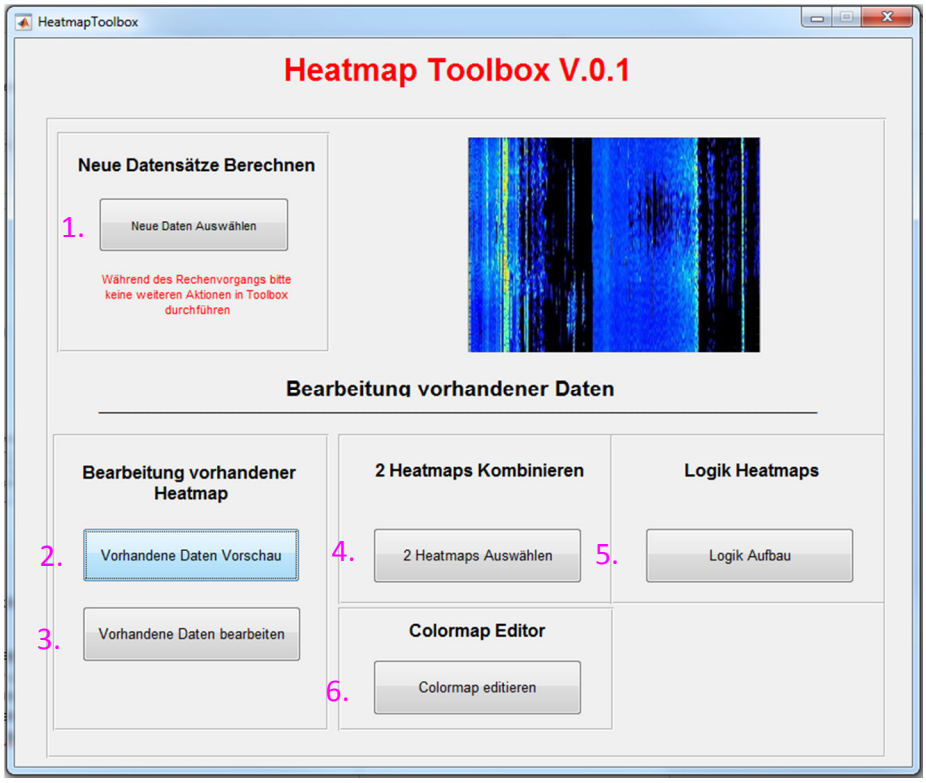


Abbildung 15: Heatmap Toolbox

In dem Programmablaufplan sind die Zusammenhänge der einzelnen Funktionen nochmal dargestellt. Eine größere Darstellung des Programmablaufplans ist dem Anhang A3 zu entnehmen.

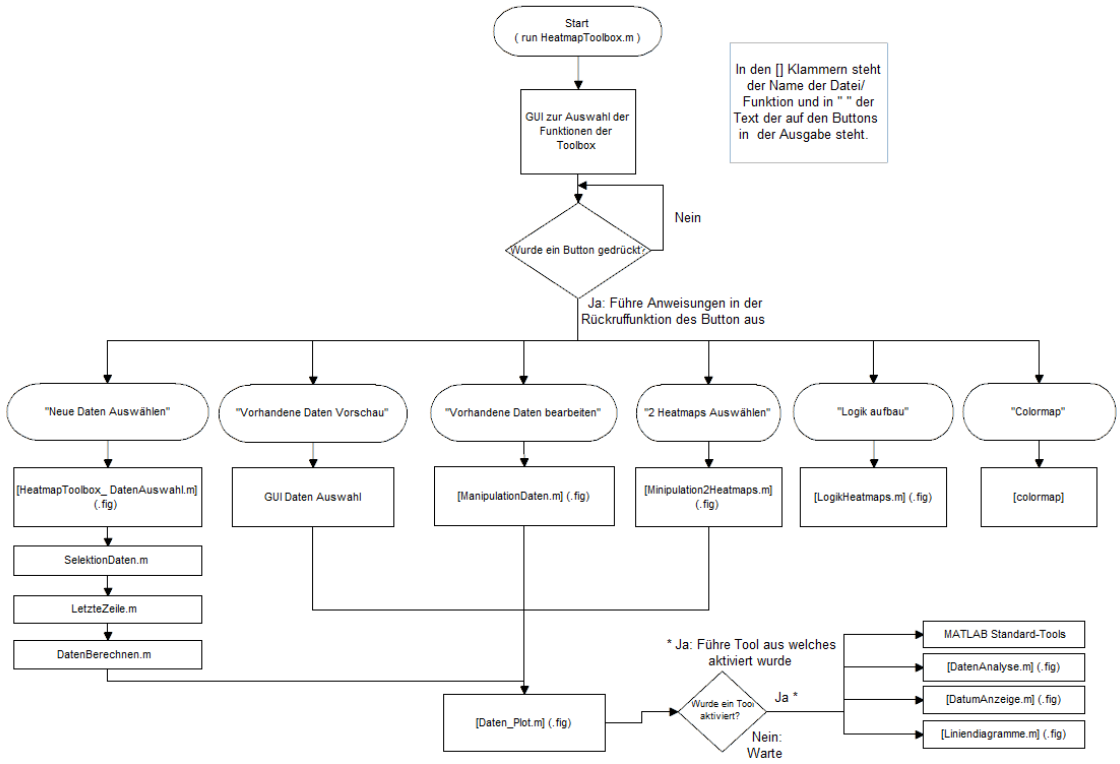


Abbildung 16: Programmablaufplan

4.4.2. HeatmapToolbox_DatenAuswahl.m (.fig)

Hier kann über eine Benutzeroberfläche die Funktion „SelektionDaten.m“ und die Funktion „DatenBerechnen.m“ gestartet werden. Die von der Funktion „SelektionDaten.m“ wiedergegebene Matrix aus Rohdaten und die Metadaten[7] werden beim Ausführen der Funktion „DatenBerechnen.m“ an diese übergeben. Wenn nicht alle Datensätze berechnet werden sollen, kann der Benutzer in einem Menü die gewünschten Datensätze anhand des Metadaten-Namens auswählen, der Index der Auswahl wird beim Ausführen der Funktion „DatenBerechnen.m“ ebenfalls zur Referenzierung übergeben.

4.4.2.1 SelektionDaten.m

Hier kann über eine Benutzeroberfläche ein Excel-File mit Rohdaten ausgewählt werden. Die Rohdatensätze werden aus dem Excel-File geholt und in eine Matrix gespeichert. Die Metadaten, wie der Namen der Datensätze, die Anzahl der Datensätze, die Anzahl der protokollierten Messwerte (LetzteZeile.m) und das Jahr, werden in einem Array abgelegt und an die Funktion „HeatmapToolbox_DatenAuswahl.m“ mit der Rohdaten-Matrix wiedergegeben.

[7] Die Metadaten bestehen aus Informationen von den Rohdaten. Hierzu gehören Name der Datensätze, Anzahl der Datensätze, das Jahr der Datensätze, die Anzahl der protokollierten Messwerte (LetzteZeile.m) und der Index zur Referenzierung der Datensätze

4.4.2.2 LetzteZeile.m

Die Funktion ist notwendig, wenn mehrere Rohdatensätze berechnet werden sollen, da die Rohdatensätze in der Regel unterschiedlich lang sind und die Matrix die aus dem Excel generiert wird sich an der höchsten Anzahl orientiert. Die Funktion soll für jeden Rohdatensatz die letzte Zeile suchen und an „SelektionDaten.m“ zurückgeben.

4.2.2.3 DatenBerechnen.m

Bei dieser Funktion handelt es sich um die Kernfunktionen der Toolbox, da hier die Rohdaten aus der Matrix ausgelesen und in die benötigte Form für die Heatmaps umgeschrieben werden. Es werden folgend vier Matrizen und ein Array pro Datensatz angelegt und im Anschluss daran mit den berechneten Werten beschrieben.

1. Rohdaten 01 (Matrix)
2. Rohdaten Werte (Matrix)
3. Interpolierte Werte (Matrix)
4. Historie der Werte (Array)

Nach der Berechnung werden diese Daten an die Funktion „DatenPlot.m“ übergeben, grafisch dargestellt und in einem .m File in einem vom Benutzer vorher definierten Ordner abgelegt. Der Name wird automatisch aus dem Jahr und dem Namen des Datenpunkts generiert. Wenn dieser Datensatz schon existiert, wird er mit dem neuem Datensatz überschrieben. Die Matrizen werden zu Beginn mit dem Wert „0“ deklariert und anhand der Tage im Jahr (365 bzw. 366) und Minuten pro Tag (1440) dimensioniert. Jeder Datensatz, der aus „SelektionDaten.m“ ausgewählt wurde, wird nacheinander, Zeile für Zeile ausgelesen und der Zeitstempel in Datum und Uhrzeit gesplittet. Das Datum wird anschließend in den Gesamttag des Jahres umgerechnet, also in den Wert 1-365(366) und die Uhrzeit in die Gesamtminuten des Tages, also in den Wert 1-1440. Danach werden für jede ausgelesene Zeile die nachfolgenden Programmabschnitte durchlaufen.

1. Rohdaten 01

Anhand der beiden Indizes Gesamt-Tag und Gesamt-Minute wird in die Matrix-Rohdaten 01 der Wert „1“ geschrieben. Um zu verdeutlichen, wie dies zu interpretieren ist, wird der Plot aus der Matrix in Abbildung 17 (nächste Seite) dargestellt. Die weißen Punkte repräsentieren den Wert „1“ und die schwarzen den Wert „0“. An jeder Stelle mit dem Wert „1“ wurde also ein Messwert aufgenommen.

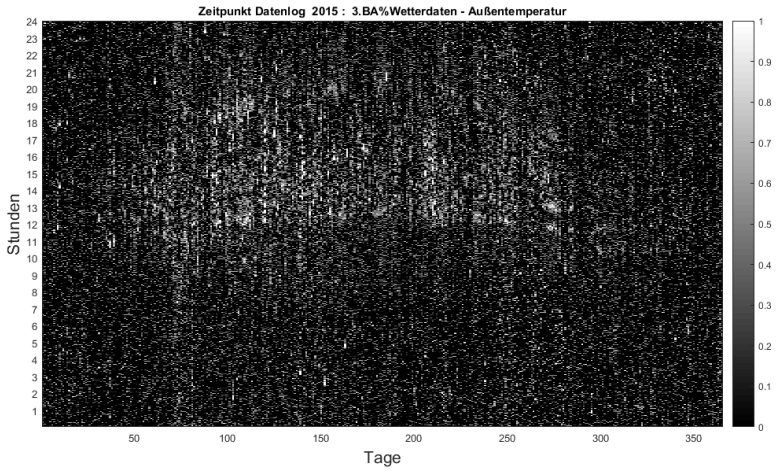


Abbildung 17: Rohdaten 01

2. Rohdaten-Wert

Mit denselben Indizes wie von „Rohdaten 01“ wird nun der zugehörige Messwert verordnet und die Matrix „RohdatenWerte“ geschrieben. Um dies grafisch zu veranschaulichen wird nachfolgend der Plot der Matrix dargestellt. Die weißen Punkte aus „Rohdaten 01“ sind jetzt anhand ihres Zahlenwertes des Messwertes einer Farbskala, die MATLAB automatisch aus den Minimal- und Maximalwerten erstellt, zugeordnet.

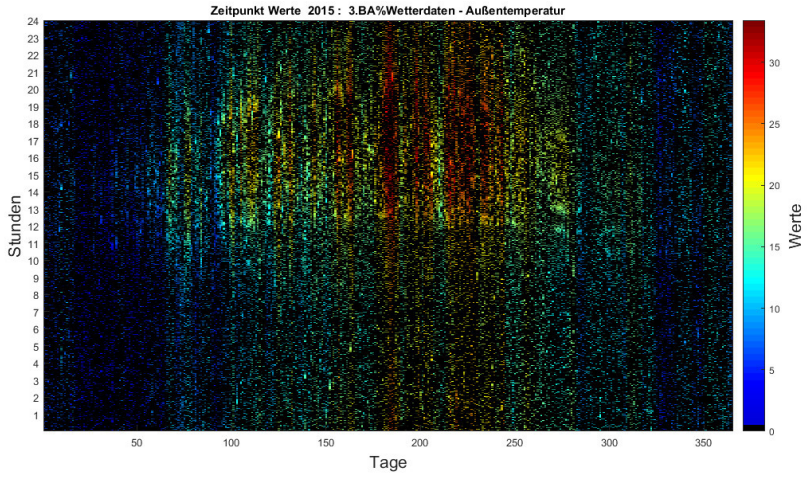


Abbildung 18: Rohdaten-Werte

3. Interpolation von Daten

Da bei den Monitoring-Daten aus Effizienzgründen, siehe 2.4 nicht jede Minute an einem Datenpunkt protokolliert wird, müssen die restlichen Daten zwischen den protokollierten Zeitpunkten interpoliert werden. Dies geschieht in diesem Fall linear. Hierfür wird geprüft, ob der nächste erfasste Messwert noch im selben Tag liegt und, ob dieser mehr als eine Minute entfernt ist. Wenn dies der Fall ist, wird zwischen den beiden Punkten mit einer Schrittweite von einer Minute interpoliert und in jedem Schritt der Wert in die Matrix „InterpolierteDaten“ geschrieben. Wenn der nächste Punkt in einem neuen Tag liegt, wird dieses Verfahren ebenfalls genutzt, es müssen dann aber noch die Minuten umgerechnet werden, da nach 1440

wieder die (Minute) 1 in der nächsten Spalte der Matrix folgt. Wenn die Messpunkte mehr als einen Tag auseinander liegen, berücksichtigt der Algorithmus dies nicht und die Werte bleiben zwischen beiden Punkten bei dem zu Beginn deklarierten Wert 0. So könnte man die falsche Annahme treffen, dass in diesen Abschnitten real der Wert 0 gemessen wurde. Um das zu verhindern, werden immer noch die zwei weiteren oben beschriebenen Heatmaps generiert, die anzeigen wo es Lücken in der Datenerfassung gibt.

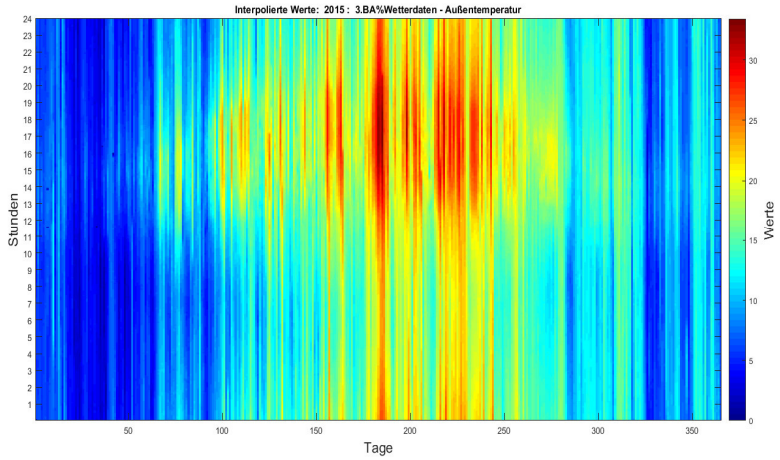


Abbildung 19: Interpolierte Werte

4. Historie

Bevor die Daten ausgegeben und gespeichert werden, wird noch ein Text-Array „Historie“ mit dem Namen des Datenpunktes aus den Metadaten angelegt. Dieses Text-Array kann bei späterer Bearbeitung der Daten mit weiteren Informationen beschrieben werden und dient der Nachverfolgung der Daten.

4.4.3. Daten_Plot.m

Die aus den Funktionen übergebenen Daten werden in dieser Funktion mit der von MATLAB mitgelieferten Funktion „figure“ grafisch dargestellt (Abb. 20 nächste Seite). Die MATLAB-Funktion liefert umfassende Werkzeuge und Optionen, die bei Bedarf noch manipuliert oder hinzugefügt werden können. Die Daten werden jeweils in einer Registerkarte ausgegeben, jede Registerkarte hat ihre eigene Werkzeug- und Optionspalette, die bei Auswahl der Registerkarte aktiviert wird.

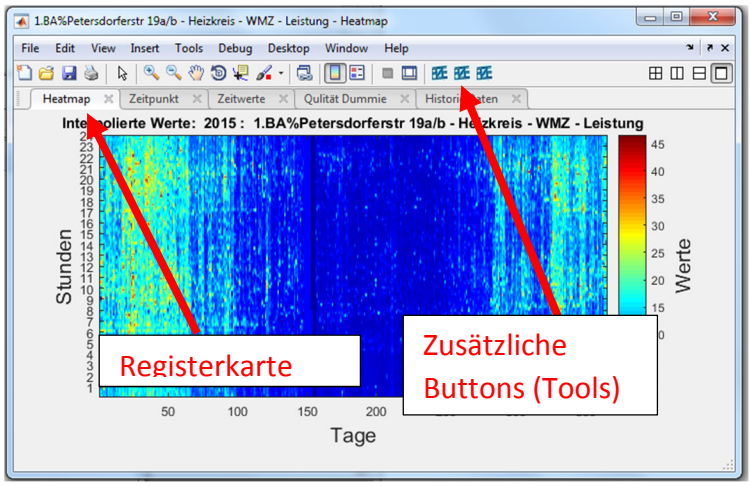


Abbildung 20: Figure mit allen Registerkarten und zusätzlichen Buttons

Für die Registerkarte „Heatmap“ wurden noch drei weitere Buttons in der Werkzeugpalette hinzugefügt, die jeweils eine Funktion aufrufen. Die Beschreibung der Funktionen ist in den jeweiligen Unterpunkten zu finden.

1. Datenanalyse (Datenanalyse.m)
2. Datum anzeigen (DatumAnzeige.m)
3. Liniendiagramme (Liniendiagramme.m)

Die grafische Darstellung der Matrizen erfolgt mit der MATLAB Funktion „imagesc”.

> „imagesc” zeigt die Daten als Bild an, das die gesamten Farben einer vordefinierten Farbpalette (Colormap) verwendet. Jedes Element der Matrix gibt die Farbe für ein Pixel des Bildes an. Das resultierende Bild ist eine m-von-n-Matrix von Pixeln, wobei m die Anzahl der Spalten und n die Anzahl der Zeilen ist. Die Zeilen- und Spaltenindizes der Elemente bestimmen die Zentren der entsprechenden Pixel. [8]

4.4.3.1 Datenanalyse.m

Beim Anklicken des Buttons „DatenAnalyse“ aus der Werkzeugpalette der Heatmap, wird die Funktion „Datenanalyse.m“ gestartet. Der Benutzer wird aufgefordert, zwei Punkte in der Heatmap mit dem „DataCursor[8]“ von MATLAB auszuwählen und nacheinander mit der Eingabe-Taste zu bestätigen. Der „DataCursor“ gibt die aktuelle Position und den Wert des Pixels aus, hierbei wird nur die Position weiterverwendet. Anhand der Position der beiden Punkte, in der Abbildung 21 (nächste Seite) als rote Kreise dargestellt, werden folgende Daten berechnet:

1. Die Daten innerhalb des Rechtecks
2. Die Daten außerhalb des Rechtecks (2 minus 1)
3. Die Daten direkt unterhalb des Rechtecks
4. Die Daten direkt oberhalb des Rechtecks
5. Die Daten innerhalb der Tagesgrenze (1, 3 und 4)

[8] Mit dem DataCurser kann der Index der Spalte und Zeile sowie der Zahlenwert angezeigt werden.

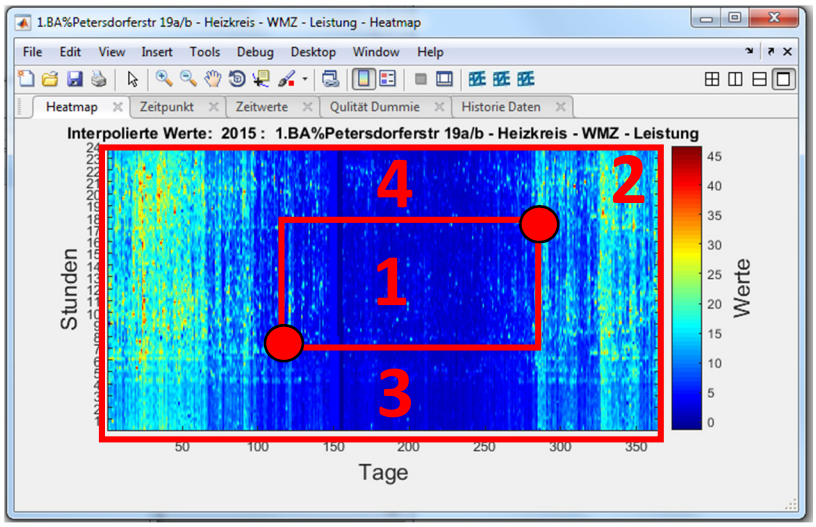
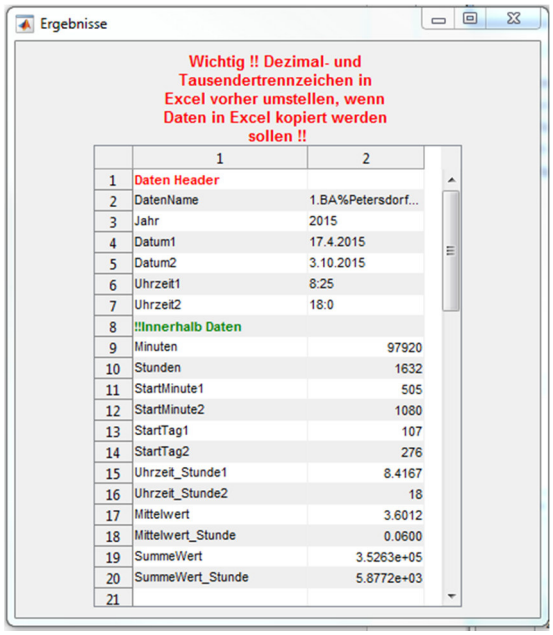


Abbildung 21: Bereiche der Auswertung

Die berechneten Daten werden in einer Tabelle in einer Benutzeroberfläche ausgegeben. Ein Teil der Tabelle ist in der Abbildung 22 zu sehen, exemplarisch sieht man dort den Tabellen-Kopf mit den Informationen der ausgewählten Daten wie Datensatz-Namen, Jahr, und das Datum mit Uhrzeit der beiden ausgewählten Punkte. Für Datum und Uhrzeit werden die Funktionen „DatumUmrechnen.m“ und „UhrzeitUmrechnen.m“ genutzt. Des Weiteren werden die Anzahl der Minuten, Anzahl der Stunden, Werte der Zeitstempel der beiden Punkte sowie verschiedene Mittelwerte und akkumulierte Werte (pro Tag bzw. Stunde) in der Funktion berechnet. Die Tabelle kann markiert und per Copy&Paste in andere Anwendungen übertragen werden, es ist dabei aber zu beachten, dass MATLAB das Komma als Dezimaltrennzeichen verwendet.



	1	2
1	Daten Header	
2	DatenName	1.BA%Petersdorf...
3	Jahr	2015
4	Datum1	17.4.2015
5	Datum2	3.10.2015
6	Uhrzeit1	8:25
7	Uhrzeit2	18:0
8	!!Innerhalb Daten	
9	Minuten	97920
10	Stunden	1632
11	StartMinute1	505
12	StartMinute2	1080
13	StartTag1	107
14	StartTag2	276
15	Uhrzeit_Stunde1	8.4167
16	Uhrzeit_Stunde2	18
17	Mittelwert	3.6012
18	Mittelwert_Stunde	0.0600
19	SummeWert	3.5263e+05
20	SummeWert_Stunde	5.8772e+03
21		

Abbildung 22: Tabelle mit berechneten Daten

4.4.3.2 DatumAnzeige.m

Beim Anklicken des Buttons „DatumAnzeige“ aus der Werkzeugpalette der Heatmap, wird die Funktion „DatumAnzeige.m“ gestartet. Der aktuelle Punkt des „Data Curser“ wird ausgelesen und in Datum und Uhrzeit umgerechnet und in einem Nachrichtenfenster ausgegeben.

4.4.3.3 Liniendiagramme.m

Beim Anklicken des Buttons „Liniendiagramme“ aus der Werkzeugpalette der Heatmap wird die Funktion „Liniendiagramme.m“ gestartet. Die Daten der Heatmap Matrix (IntpolWerte) werden so aufbereitet, dass sie in Liniendiagrammen dargestellt werden können. Dafür wird die Matrix in ein eindimensionales Array umgeschrieben, die Schrittweite des Arrays beträgt eine Minute. Das Array wird dann mit der Schrittweite von einer Minute bzw. einem Tag (Mittelwerte und akkumulierte Werte) in mehreren Registerkarten geplottet. (Anmerkung: Diese Funktion findet in dieser Bachelor-Thesis keine Anwendung)

4.4.4. ManipulationDaten.m

Diese Funktion kann aus dem grafischen Auswahlmenu von der Funktion „HeatmapToolbox.m“ aufgerufen werden und fordert den Benutzer über eine Benutzeroberfläche (Abb. 23 nächste Seite) auf, einen schon berechneten Datensatz auszuwählen. Der vorhandene Datensatz kann hier auf drei Arten manipuliert werden.

1. Eine Normierung kann hier vorgenommen werden. Der Benutzer bekommt den Minimal- und Maximalwert der Heatmap angezeigt und kann diesen neu definieren. Dazu wird für den Minimalwert das erste Element der Matrix mit den neuen Minimalwert beschrieben und für den Maximalwert das letzte Element der Matrix.
2. Die Daten können abgeschnitten und es kann ein Minimal- und Maximalwert definiert werden. Alle Werte die kleiner bzw. größer als diese sind, werden abgeschnitten und somit auf die definierten Werte begrenzt.
3. Es kann eine einfache mathematische Operation durchgeführt werden, so kann jedes Element der Daten mit einem Wert addiert, subtrahiert, multipliziert, dividiert oder der Betrag gebildet werden.

Der Benutzer muss zur Bestätigung der Manipulation auf einen Vorschau-Button klicken, woraufhin das Ergebnis angezeigt wird. Das Ergebnis kann dann gespeichert werden, wobei der originale Datensatz überschrieben wird. Wenn das Ergebnis nicht gefällt, kann der Benutzer die Funktion schließen und die originalen Daten werden nicht überschrieben.

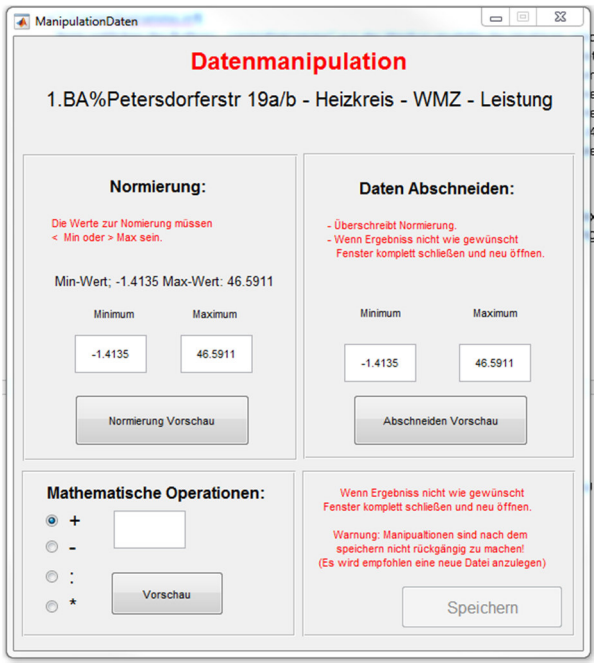


Abbildung 23: Benutzeroberfläche ManipulationDaten

4.4.5. Manipulation2Heatmaps.m

Diese Funktion kann aus dem grafischen Auswahlmenu von der Funktion „HeatmapToolbox.m“ aufgerufen werden. Der Benutzer kann hier zwei Datensätze (Heatmap oder Logik) über eine Benutzeroberfläche wählen und diese miteinader addieren, subtrahieren, dividieren und multiplizieren. Dafür wird nur die Matrix „IntpolWerte“ aus den Datensätzen verwendet, die anderen werden nicht berücksichtigt. Die Berechnung kann über den Button „Vorschau“ gestartet und angezeigt werden. Über den Button „Speichern“ kann dann ein Name für die Datei und ein Ordner, in dem die Datei gespeichert werden soll, gewählt werden. Die anderen Matrizen „Rohdaten01“ und „RohdatenWerte“ bleiben hierbei leer, nur in der Historie wird vermerkt, welche Datensätze mit welcher Operation durchgeführt wurden.

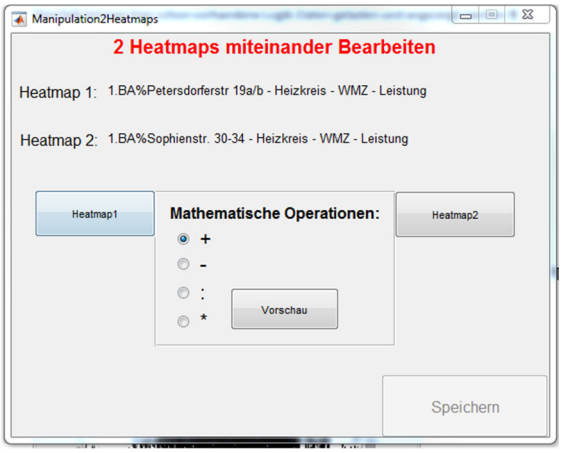


Abbildung 24: Benutzeroberfläche zum Kombinieren von Heatmaps

4.4.6 LogikHeatmaps.m (.fig)

Diese Funktion kann aus dem grafischen Auswahlmenu von der Funktion „HeatmapToolbox.m“ aufgerufen werden. Der Benutzer kann hier mit beliebig vielen (Heatmap) Datensätzen eine Logik aufbauen. Zu diesem Zweck kann über eine Benutzeroberfläche ein Datensatz geladen werden und Bedingungen wie =, <, >, >=, <=, ≠ sowie ein Wert und eine Verknüpfung wie UND, ODER, XOR ausgewählt werden. Wenn der Benutzer mit den Einstellungen fertig ist, werden die Einstellungen und die Heatmap (IntpolWerte) in neuen Variablen abgelegt und in den zwei rechten Listboxen angezeigt. Der Benutzer kann den Vorgang beliebig oft wiederholen und die Einstellungen bei Bedarf in der Anzeige noch ändern. Der Button „Berechnen“ führt dann die zusammengesetzte Logik aus und der Benutzer kann sie sich in einem „figure“ anschauen. Der Wert 1 entspricht hierbei der Aussage „wahr“ und der Wert 0 der Aussage „nicht wahr“. Mit dem Button „Speichern“ können die berechneten Werte gespeichert und ein Datei-Namen gewählt werden. Der gespeicherte Datensatz enthält die berechnete Logik-Matrix und die dazu vorgenommenen Einstellungen als Text-Array. Ebenfalls können hier schon vorhandene Logik-Daten geladen und angezeigt werden.

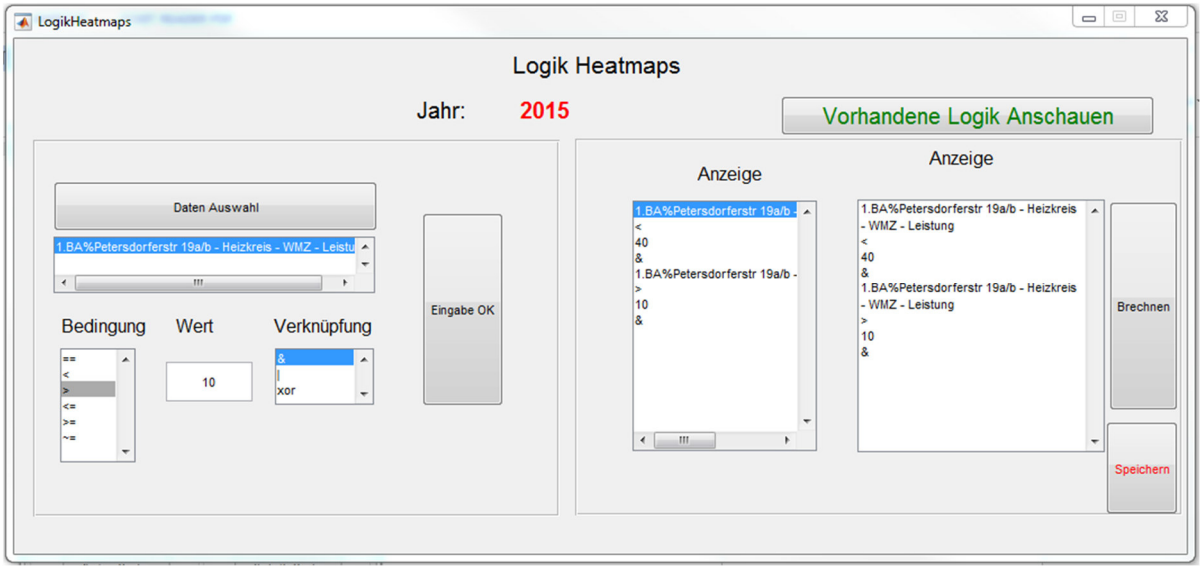


Abbildung 25: Benutzeroberfläche Logik

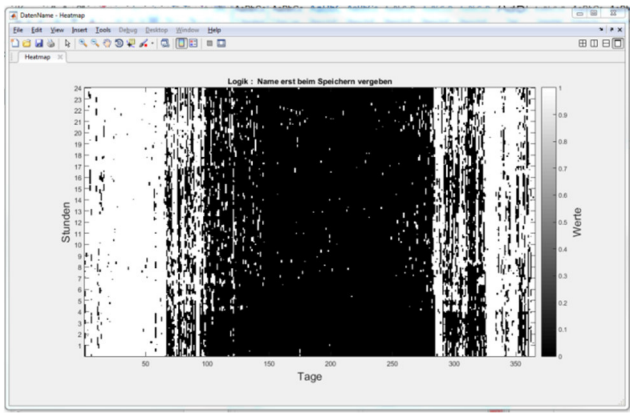


Abbildung 26: Ergebnis Logik

4.4.7 DatumUmrechnen.m

Die Funktion rechnet einen übergebenen Gesamt-Tag des Jahres, also den Index der Abszisse von der Heatmap, in ein Datum um und gibt den Wert als String wieder zurück.

4.4.8 UhrzeitUmrechnen.m

Die Funktion bekommt die Minute (Index der Zeile 1-1440) übergeben und rechnet diese in eine Uhrzeit HH:MM um und gibt den Wert als String zurück.

4.7 Hinweise

1. Die Dauer der Berechnung und Visualisierung kann strak Variieren und hängt von der Leistung des Computer-Systems ab, so konnte beobachtet werden, dass die Darstellung von schon erzeugten Datensätzen auf einem leistungsschwachen Computer-System mehrere Minuten gedauert hat.
2. In der Version „0.1 Date: 2017/01/20“ sollte während der Generierung neuer Datensätze keine weitere Funktion ausgeführt werden, da in der Version noch globale[9] Variablen verwendet werden und es beim Ausführen von anderen Funktionen zu Konflikten kommen könnten.

[9] Globale Variablen sind Variablen die funktionsübergreifend gelesen und beschrieben werden können

5. Anwendung der Software auf eine Anlage zur Nahwärmeversorgung in Harburg

In diesem Kapitel wird die Software auf zwei Teilsysteme (5.2, 5.3) einer Anlage zur Nahwärmeversorgung einer Wohnungsbaugenossenschaft angewandt. Anhand der untersuchten Teilsysteme wird gezeigt, wie man das Software-Tool mit seinen verschiedenen Funktionen für eine Untersuchung nutzen kann. Dabei wird noch ergänzend mit der Software „Chart-Viewer“ gearbeitet, da diese eine komfortable Darstellung der Rohdaten in Form von Liniendiagrammen gewährleisten kann. Für eine Einordnung der Teilsysteme in die Gesamtanlage wird im folgenden Punkt ein Überblick über diese gegeben.

5.1 Anlagenbeschreibung

Die in dieser Arbeit untersuchte Anlage dient zur Wärmebereitstellung für die Raumbeheizung und Brauchwassererwärmung von 23 Gebäuden bzw. 483 Wohneinheiten einer Hamburger Wohnungsbaugenossenschaft im Stadtteil Hamburg-Wilstorf (siehe Abbildung 27). Sie ist in drei Bauabschnitte (BA) gegliedert und von Oktober 2012 bis Oktober 2014 schrittweise in Betrieb genommen worden.[9] Bei der Wärmebereitstellung handelt es sich um eine niedertemperierte Nahwärmeversorgung mit einer Vorlauftemperatur von 50°C und einer Rücklauftemperatur von 40°C. Die Warmwasserbereitung erfolgt hierbei mit dezentralen Frischwarmwasserstationen.

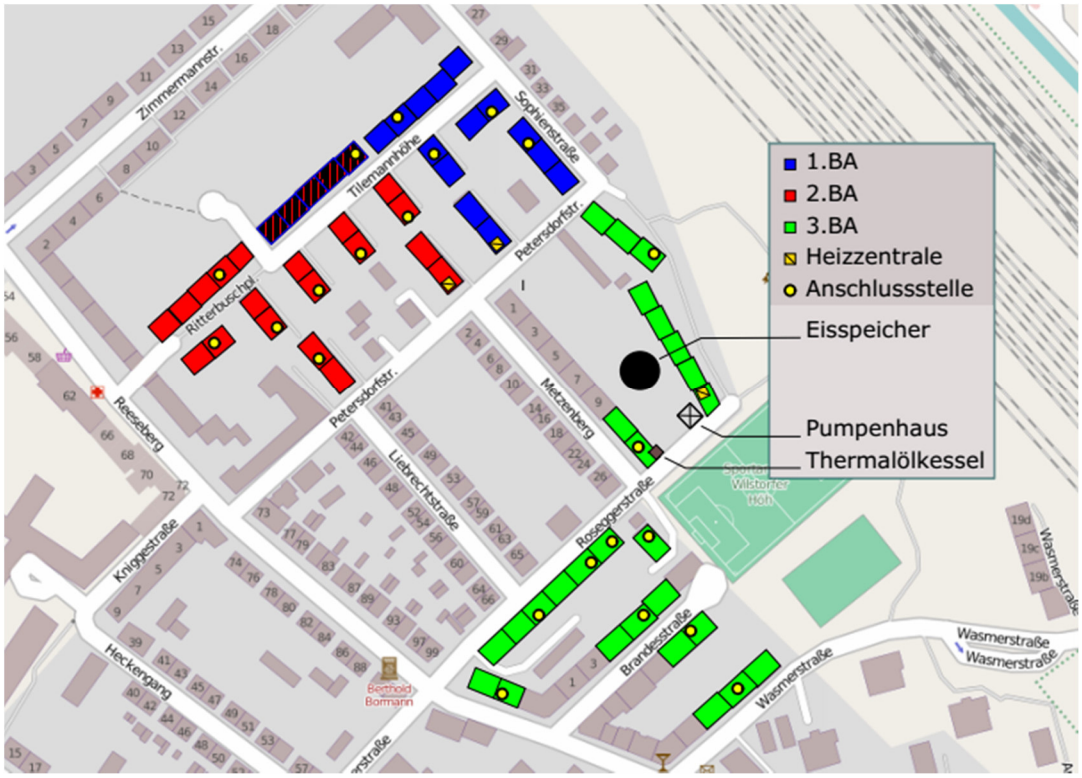


Abbildung 27:Plan [9]

Die Wärmeversorgung des Gebietes erfolgt, wie in dem Schema in Abbildung 28 auf der folgenden Seite dargestellt, semizentral. Die Energie wird durch drei unterschiedliche Wärmeerzeuger in ein Nahwärmenetz eingespeist. Jeder Bauabschnitt hat eine eigene solarthermische Anlage, die auf die Größe des Bauabschnittes angepasst ist, und einen nach der jeweiligen Heizlast der Gebäude in dem Bauabschnitt ausgelegten Gasbrennwertkessel. Jeder Bauabschnitt verfügt über eine eigene Heizzentrale, von der aus die Wärmeversorgung über das Nahwärmenetz des Bauabschnittes in die Gebäude organisiert wird. In jedem Gebäude befindet sich ein Heizungspufferspeicher, aus dem die Wärme in das Gebäude verteilt wird. Insgesamt sind 25 Heizungspufferspeicher (Abbildung 27 gelbe Kreise, vorherige Seite) an das Nahwärmenetz angebunden.[9]

Die zentrale Gas-Absorptionswärmepumpe, die übergeordnet in das System eingebunden ist, versorgt alle drei Bauabschnitte mit Wärme. Als Quelle dienen der Wärmepumpe sowohl der Eisspeicher als auch die Solar-Luft-Kollektoren. Die von der Wärmepumpe bereitgestellte Energie wird in die Heizzentralen der drei Bauabschnitte geleifert und von den Heizungspufferspeichern in die Gebäude verteilt. Ist die bereitgestellte Wärme nicht ausreichend, dann wird die zusätzlich erforderliche Wärme durch den Gas-Brennwertkessel bereitgestellt. Es entsteht daher ein kombinierter Betrieb aus Solarthermie, Gas-Absorptionswärmepumpe und Gas-Brennwertkessel. Zur Überwachung und Steuerung ist das gesamte System mit einer Gebäudeautomation ausgestattet.

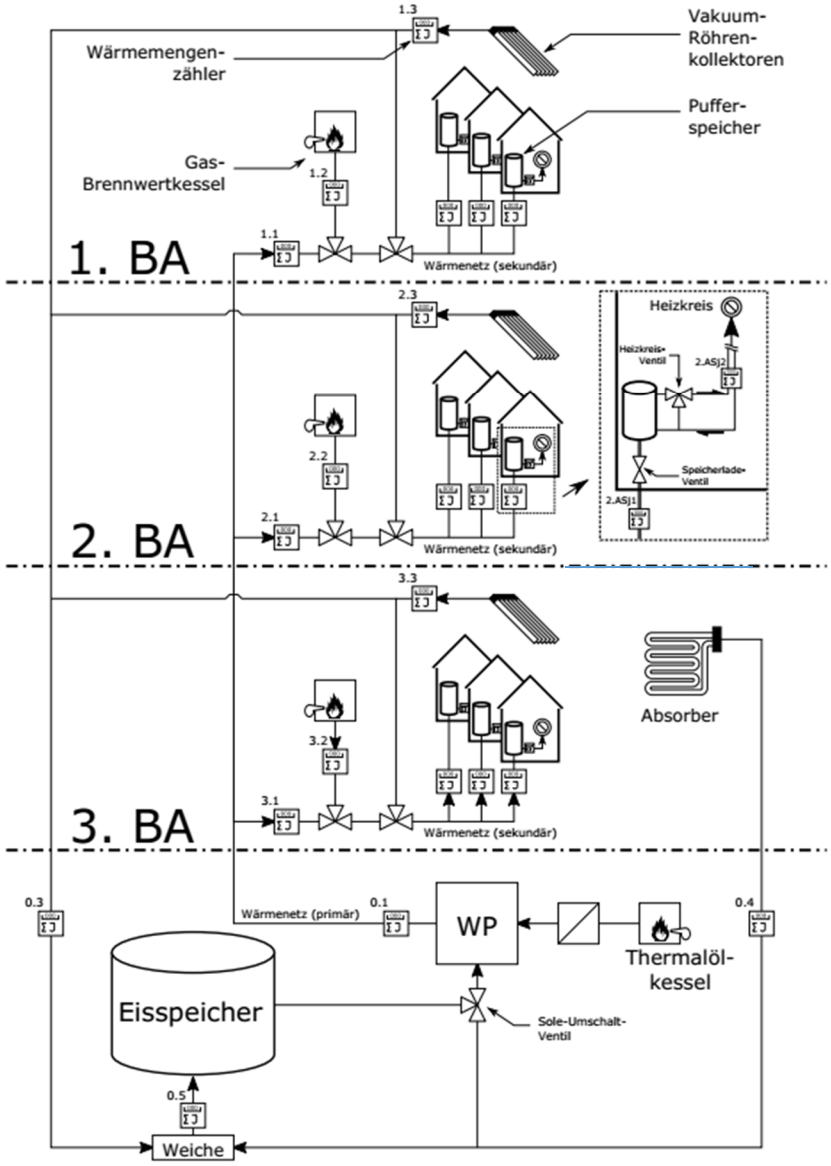


Abbildung 28: Anlagen Schema [9]

5.2 Betrachtung der thermischen Gas-Brennwertkessel-Leistung

In den folgenden Unterkapiteln werden die Heatmap-Datensätze der Kesselleistung der drei BAs visuell analysiert. Dazu werden die generierten Heatmap-Datensätze der thermischen Kesselleistung der drei Kessel in den einzelnen BAs jeweils für das Jahr 2015 und 2016 addiert. Zu sehen sind die Heatmaps in Abbildung 29 (S.35) und 31 (S.36). Die Kessel dienen als unterstützende Systeme zu der Absorptionswärmepumpe und den Solarkollektoren, für den Fall, dass die benötigte thermische Leistung diejenige der Wärmepumpe oder der Solarkollektoren übersteigt. Außerhalb der Heizperiode ist die Wärmepumpe komplett außer Betrieb, so dass nur noch ein bivalenter Betrieb mit Kesseln und Solarkollektoren vorliegt. Hierbei sollte aufgezeigt werden, dass man anhand verschiedener Datenpunkte Annahmen bestätigen kann.

5.2.1 Ein- und Abschaltzeitpunkt der Wärmepumpe

Den Einschaltzeitpunkt (2) der Wärmepumpe kann man auch ohne die Information des planmäßigen Abschaltzeitraumes entnehmen, da die Leistung der Kessel deutlich abfällt und von einem gelb-roten Band[10] in ein blaues Band übergeht. Der Abschaltzeitpunkt der Wärmepumpe (1) hingegen ist nur im Jahr 2015 deutlich zu erkennen, in dem das Muster nach dem Abschaltzeitpunkt (1) homogener ist. Dies liegt daran, dass die Kessel nach dem Abschalten der Wärmepumpe konstant Wärme liefern müssen und nicht mehr nur noch zur Spitzenlast-Abdeckung. Zur Überprüfung der Vermutungen werden externe Daten der planmäßigen Abschaltung der Wärmepumpe zur Hilfe herangezogen. So konnte der GA entnommen werden, dass im Jahr 2015 die Wärmepumpe vom 3. Juni (Tag 154) bis zum 14. Oktober (Tag 287) außer Betrieb war. Im Jahr 2016 (Abb. 30 nächste Seite) waren es der 6. Mai (Tag 127) und der 16. November (Tag 321). Diese Zeitpunkte sind in den beiden Heatmaps jeweils mit einem lila waagerechten Strich markiert. Somit hat sich die oben beschriebene Vermutung bestätigt. Ein weiteres Vorgehen, die Annahme anhand eines anderen Datenpunkts zu bestätigen, wird im nächsten Unterkapitel beschrieben und dient gleichzeitig als eine Einführung in die Funktion „LogikHetamaps.m“ der Tool-Box.

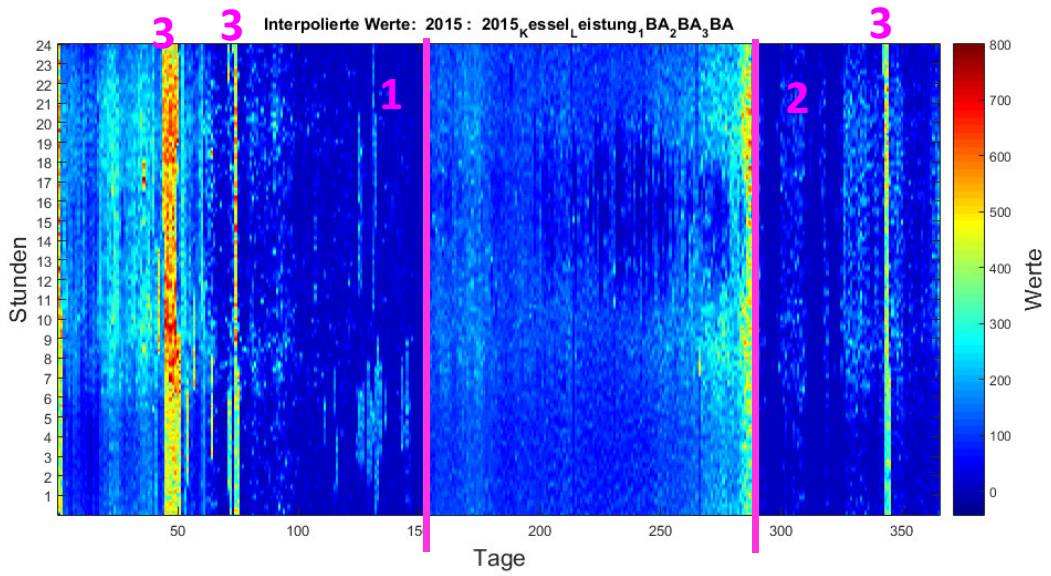


Abbildung 29: Kessel-Leistung aller BAs in [kW] 2015

[10] Mit Band ist ein Bereich gemeint, in dem sich über mehrere Tage hinweg dasselbe Muster an Werten bzw. Farben abzeichnet.

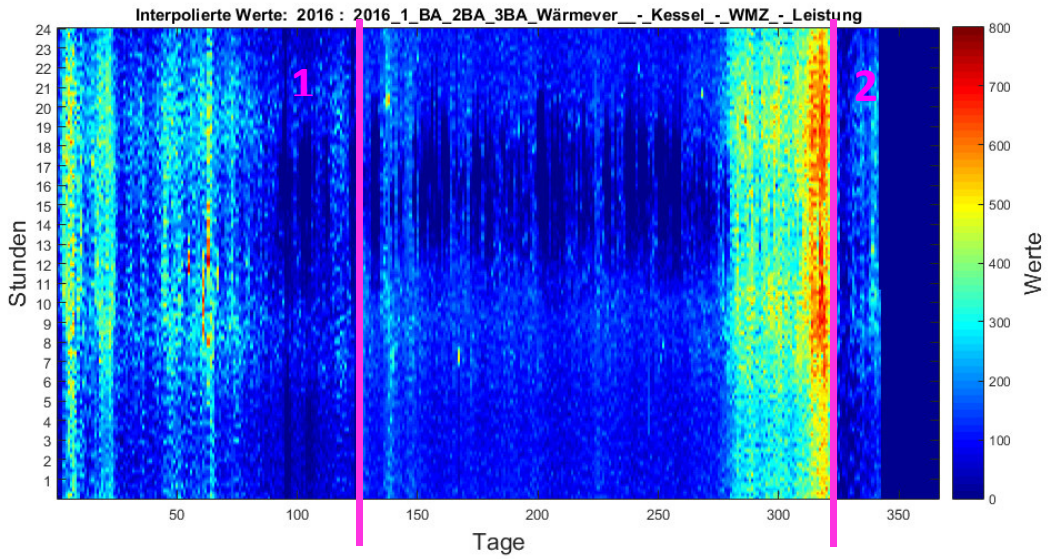
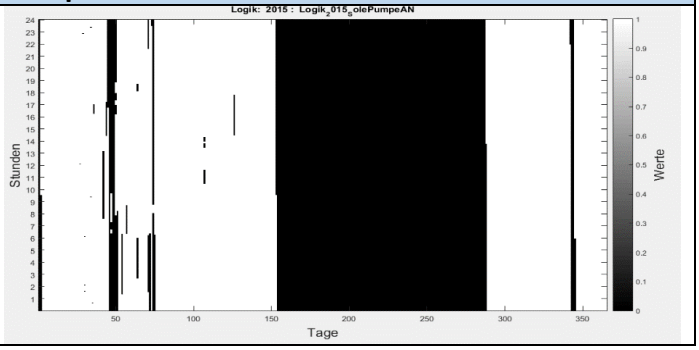


Abbildung 30: Kessel-Leistung aller BAs in [kW] 2016

5.2.2 Auffälligkeiten im Kesselbetrieb

Eine Auffälligkeit ist, dass es im Jahr 2015 (Abb. 29) innerhalb der Bereiche, in denen die Wärmepumpe angeschaltet war, Phasen gab, in denen die Kessel-Leistung stark angestiegen ist (3). Dabei muss es sich um Abschnitte handeln, in denen die Wärmepumpe außerplanmäßig nicht lief. Diese Annahme kann mit einer weiteren Heatmap überprüft werden, wozu die Solepumpe an der Verdampferseite der Wärmepumpe zur Hilfe genommen werden kann. Dafür wurde die Heatmap der Drehzahl der Solepumpe verwendet, die den WP-Verdampfer speist. Wenn die Pumpe außer Betreib ist, ist somit auch die Wärmepumpe ausgeschaltet. Wenn die Drehzahl der Pumpe den Schwellenwert von 2400 1/min erreicht, was der minimalen Drehzahl der Pumpe entspricht, wird der Wert 1 angegeben. Daraus resultiert dann die Logik in Tabelle 7, wobei weiß „wahr“ bedeutet, also die Pumpe läuft, und schwarz „nicht wahr“, also die Pumpe läuft nicht. Wenn man die Logik der Tabelle 7 und die Abbildung 29 gegenüberstellt, wird deutlich, dass die Bereiche korrelieren, was bestätigt, dass zu diesen Zeitpunkten die Wärmepumpe nicht lief.

Tabelle 7: Logik Solekreispumpe AN

Logik Aufbau	**Logik Ergebnis**
Pumpe Solekreis An	**Pumpe Solekreis An**
Pumpenhaus%Solekreis - Pumpe - Drehzahl > 2400	

Des Weiteren fällt auf, dass im Jahr 2016 (Abb. 30 S.36) der Einschaltzeitpunkt (2) der Wärmepumpe weiter hinten liegt und die Kesselleistung massiv ansteigt. Dies liegt daran, dass es einen Schaden an einer der Pumpen gab, die die Wärmepumpe antreibt. Diese Information konnte vom Betreiber der Anlage eingeholt werden und war nicht aus anderen Datenpunkten zu entnehmen. Außerdem ist zum Ende des Jahres 2016, um den Tag 350, nur noch der Wert „0“ angegeben. Der Konstante Wert über einen längeren Zeitraum hinweg deutet auf eine Lücke in der Erfassung der Messwerte hin. Dies konnte schnell bestätigt werden, indem die zu jeder angelegten Heatmap gehörende Matrix „Werte“ angeschaut wurde, die den Zeitpunkt der Erfassung des Messwerts festhält (siehe 4.2.2.3 DatenBerechnen.m).

5.2.3 Nutzerverhalten

In der Abbildung 31 kann man zu Beginn der Jahre auch gut das Nutzerverhalten bzw. den Wärmebedarf für die Brauchwassererwärmung der Nutzer erkennen. Neben der konstant benötigten Heizleistung zum Heizen der Wohnungen wird zusätzlich noch eine Heizleistung für die Warmwasserbereitung benötigt. So steigt gegen 6 Uhr morgens der Wärmebedarf merklich an und sinkt gegen 23 Uhr wieder. Um dies zu bestätigen, wurden die Daten aus den einzelnen Heizkreisen (Abb 32, nächste Seite) des ersten BA für das Jahr 2015 addiert. Auch hier ist dasselbe Muster zu erkennen, nachdem der Wärmebedarf ab 6 Uhr morgens steigt und ab 23 Uhr sinkt.

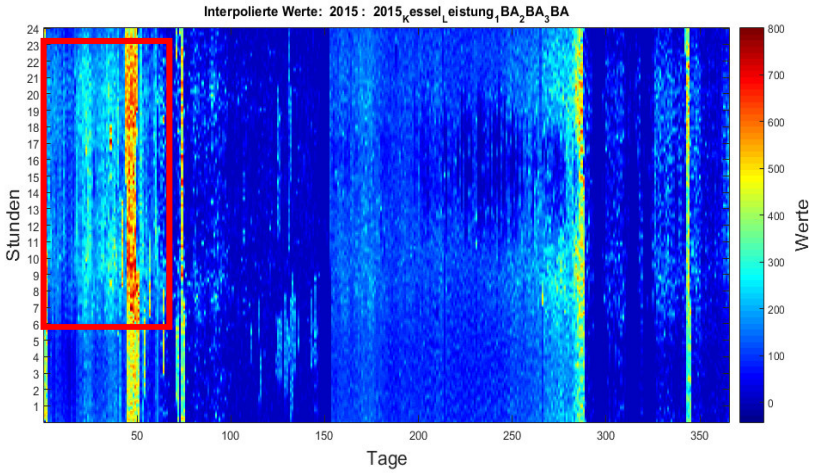


Abbildung 31:Nutzerverhalten Kesselleistung 2015

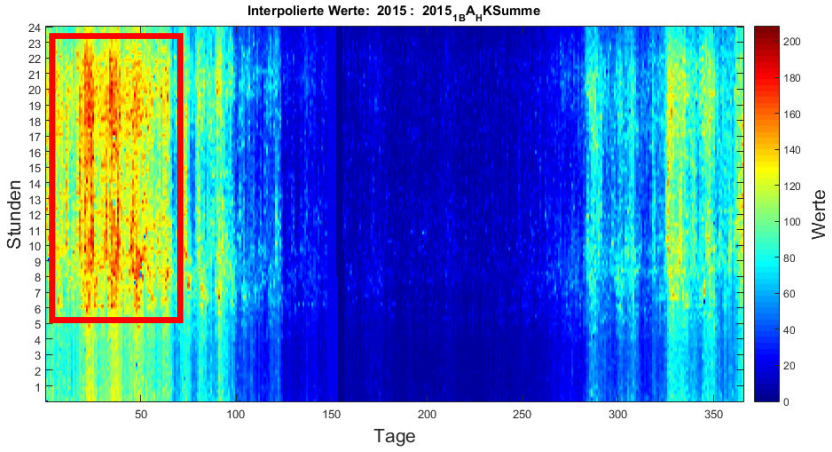


Abbildung 32: Heizleistung aller Heizkreise des 1BA

5.2.4 Einspeisung der Vakuumröhrenkollektoren

Das Betriebsverhalten der Vakuumröhrenkollektoren kann anhand der Kesselleistungen ebenfalls untersucht werden. Die Kollektoren können über ein 3-Wege-Ventil „Ladeheizkreis-Ventil“ bei Bedarf in den Ladeheizkreis integriert werden. Sobald die Kollektoren eine ausreichend hohe VL-Temperatur liefern können, wird der Weg zum Kessel und der Wärmepumpe hin geschlossen, sodass der Weg zu den Kollektoren hin vollständig geöffnet ist. Dadurch entsteht ein geschlossener Kreislauf zwischen Ladeheizkreis und den Kollektoren und die Wärmeversorgung findet nur noch monovalent über die Kollektoren satt.

Um ein aussagekräftigeres Ergebnis darüber zu bekommen, wie das Betriebsverhalten der Kollektoren funktioniert, wird der Abschnitt zwischen (1) und (2) näher betrachtet, in dem die Wärmeversorgung nur noch monovalent über die Kessel und die Kollektoren sattfindet.

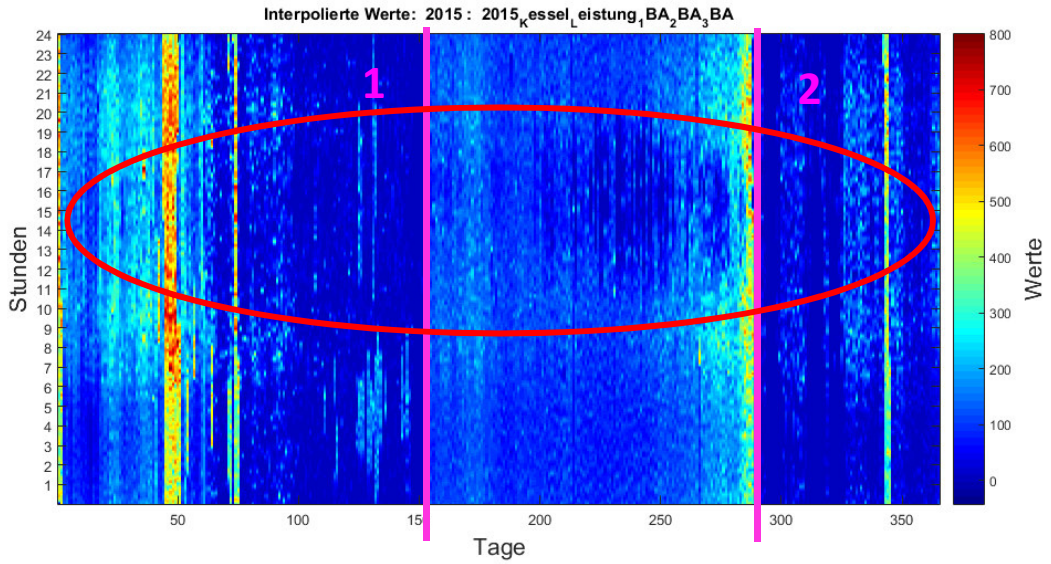


Abbildung 33: Kessel-Leistung aller BAs in [kW] 2015

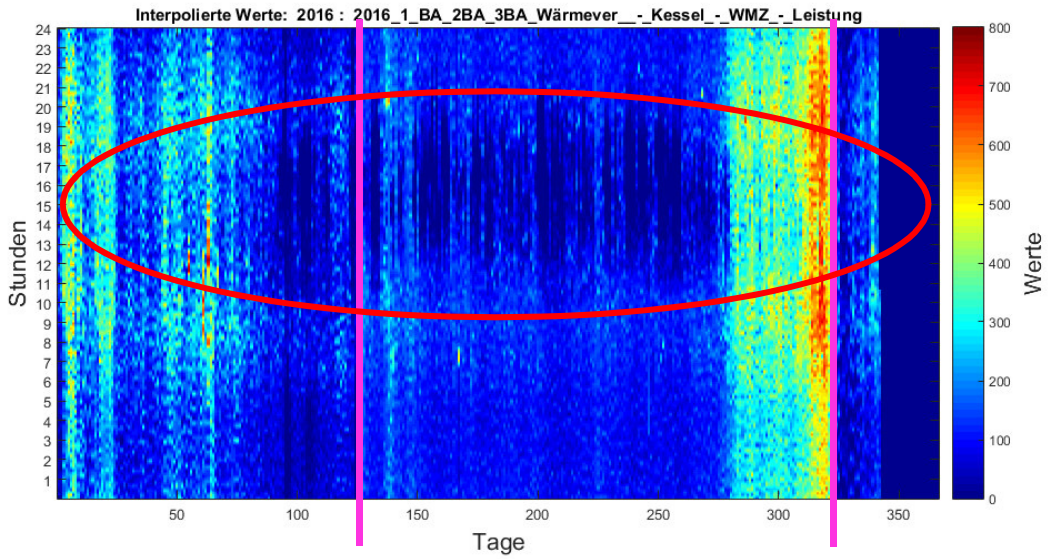


Abbildung 34: Kessel-Leistung aller BAs in [kW] 2016

Zum Vergleich werden die Daten zusätzlich als Liniendiagramm dargestellt, dazu wurden die Rohdaten der Kesselleistung des ersten BA für das Jahr 2015 verwendet. Es können auch hier viele Informationen entnommen werden, wie z.B. der Ein- und Abschaltzeitpunkt (2 und 1) der Wärmepumpe, der Ausfall der Wärmepumpe (3) ist hingegen schwerer auszumachen. Das Verhalten der Verdrängung der Kessel-Leistung durch die Solar-Kollektoren kann dem Diagramm allerdings nicht entnommen werden.

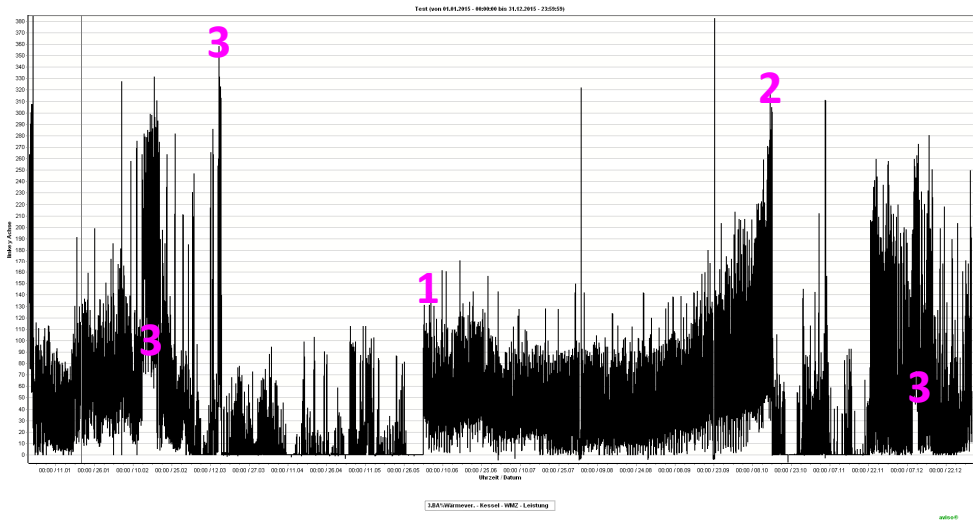


Abbildung 35: Liniendiagramm Kesseleistung 2015

Als Referenz, wann mit solaren Erträgen zu rechnen ist, wurde die Heatmap des Strahlungssensors des zweiten BAs für das Jahr 2015 angeschaut (Abb 36) und mit einer roten Ellipse in der Abbildung umrandet. Die Ellipse wurde auf die beiden Abbildungen 33 und 34 übertragen. Ersichtlich ist, dass die Leistung der Kessel im Jahr 2016 (Abb. 34), in dem Zeitbereich, in dem Solare-Erträge zu erwarten sind, wesentlich geringer war. Dies spricht für eine Verdrängung des Kessels durch die Solar-Kollektoren. Im Jahr 2015 (Abb. 33 S.38) ist

dieses nur zum Ende hin ausgeprägt, was darauf zurückzuführen ist, dass die Solar-Kollektoren zwischen dem 25. Mai (Tag 145) und dem 25. Juli (Tag 210) nicht ordnungsgemäß in Betrieb waren. Dies wird in dem folgenden Unterkapitel 6.2.5 weiter ausgeführt.

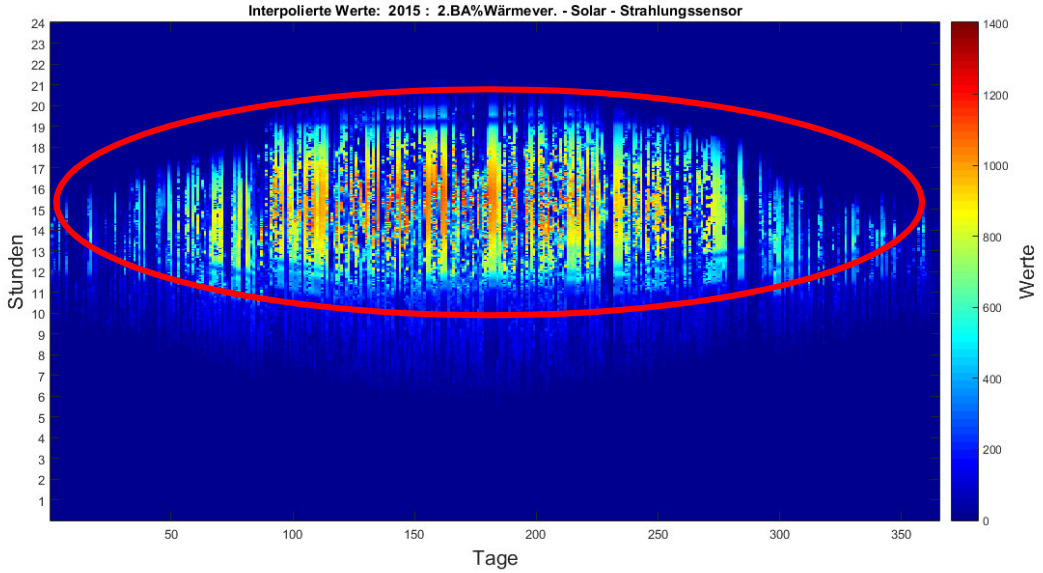


Abbildung 36: Heatmap Strahlungssensor 2BA 2015

Um sicher gehen zu können, dass es sich bei der verringerten Kessel-Leistung um einen Solar-Ertrag handelt, müssen noch weitere Referenzen hinzugezogen werden. So können zum Beispiel die addierte Heizleistung (Abb. 37) der Anschlussstellen aller Heizkreise des ersten BAs für das Jahr 2015 oder die Ventil-Stellung des Ladeheizkreises (Abb. 38 S.41 und 39 S.42) als zusätzliche Referenzen genutzt werden. Es handelt sich hierbei um ein 3-Wege-Ventil, dass bei 0% den Weg zu den Kollektoren vollständig öffnet und bei 100% komplett schließt.

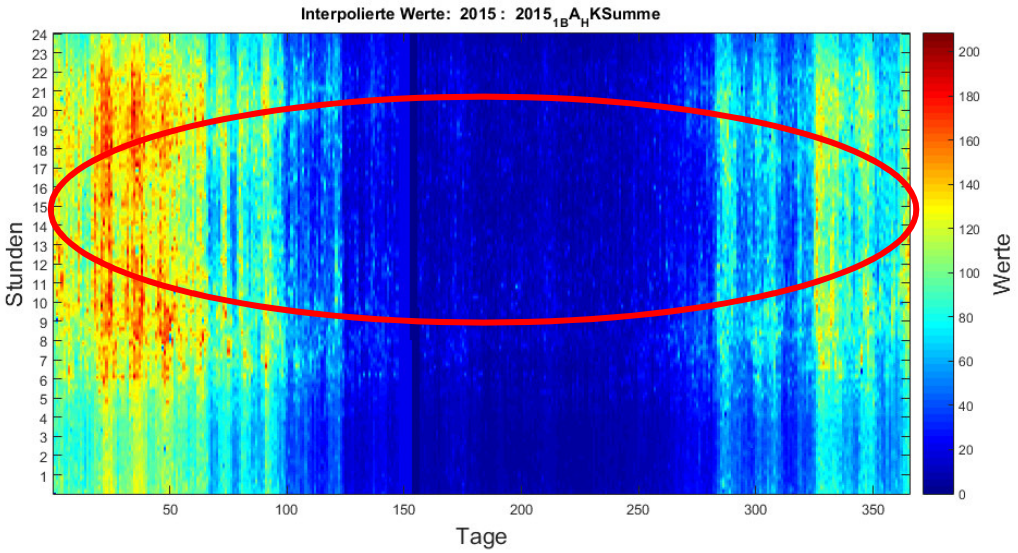


Abbildung 37: Addierte Heizkreis-Leistung aller Heizkreise des 1BAs [kW] 2015

In der Abbildung 37 ist die summierte Heizleistung aller Heizkreise der Anschlussstellen des ersten BAs für das Jahr 2105 zu sehen. Man erkennt, dass die benötigte Heizleistung in dem Bereich der roten Elipse nicht abnimmt, sondern konstant bleibt, was dafür spricht, dass es sich bei der Reduzierung der Kesselleistung (Abb. 34 S.38 und 35 S.39) um einen Solaren-Eintrag handelt. In den Abbildungen 38 und 39 wird die Annahme nochmal bestätigt, da dort zu sehen ist, wie die Ladeheizkreis-Ventilstellung geschaltet war. Hierfür wurde nur die Ventilstellung des zweiten BAs zu Grunde gelegt. Auch der Ausfall der Solar-Kollektoren zwischen dem 25. Mai (Tag 145) und dem 25. Juli (Tag 210) im Jahr 2015 ist der Ventilstellung zu entnehmen.

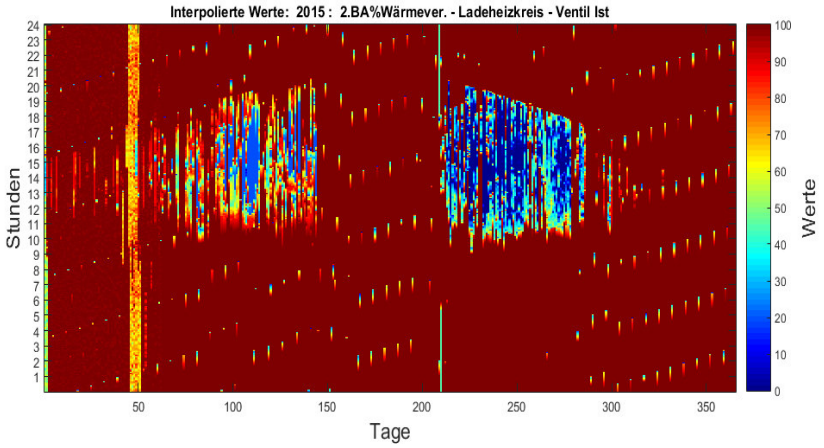


Abbildung 38: Ventilstellung Ladeheizkreis 2015

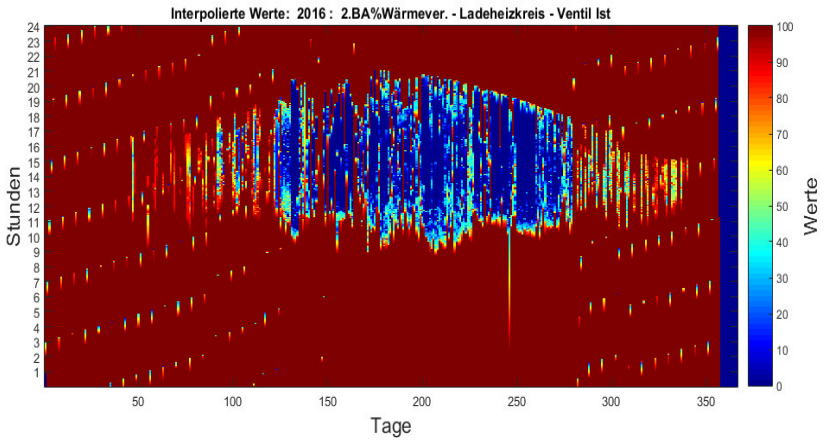


Abbildung 39: Ventilstellung Ladeheizkreis 2016

5.2.5 Ergebnis der Betrachtung

Die Auswahl des unterstützenden System Gas-Brennwertkessel hat sich als ein guter Ansatz erweisen, aus dem viele Informationen entnommen werden können, wie die Abschalt- und Ausfallszeiten des Hauptsystems Gas-Absorptionswärmepumpe, das Nutzerverhalten und das Verhalten der Vakuumröhrenkollektoren.

So geht aus den visualisierten Daten hervor, dass der Betrieb der Solar-Röhren-Kollektoren direkt stattfindet. Dies wird an der Gegenüberstellung der Ladeheizkreis-Ventilstellung (Abb. 38 und 39) und der Leistungsabnahme der Kessel (Abb. 29 S.35 und Abb. 40 S.36) deutlich. Sobald die Kollektoren keine Leistung mehr liefern, liefert der Kessel sofort wieder Wärme.

Dabei ist kein dämpfender Effekt zu erkennen, der zum Beispiel auf ein durchladen der Pufferspeicher in den Anschlussstellen zurückzuführen wäre. Es wird aber auch deutlich, dass die Solarthermie den Wärmebedarf für Warmwasser vollständig decken kann.

5.3. Absorber-Schaltung

Bei den Absorbern handelt es sich um einfache unverglaste Solar-Luft-Kollektoren in Röhrenbauart aus Kunststoff. Der Absorber hat zwei Funktionen: erstens soll die Verdampferseite der Wärmepumpe mit Umweltwärme versorgt werden, zweitens ist das beladen des Eisspeichers mit Umweltwärme zu nennen. In diesem Kapitel wird über die Schaltzustände der Pumpen rekonstruiert, wie die Umweltwärme aufgeteilt wird. Zunächst findet eine Aufbereitung der Daten mit „Heatmap-Toolbox“ statt, um im Anschluss zu klären, wie die zeitliche Aufteilung der Betriebszeit aussieht und darüber die Aufteilung der geleisteten Arbeit zu rekonstruieren.

5.3.1 Auffälligkeiten aus der Betrachtung der Daten

Ausfall der Aufzeichnung:

Bei der Betrachtung der Absorber-Leistung ist aufgefallen, dass die Aufzeichnung zu Beginn des Jahres ausgefallen ist. Der Ausfall ist in den beiden folgenden Abbildungen rot markiert. In der Abbildung 40 sieht es so aus also ob die Erfassung der Daten überwiegend Stundenweise geschieht, beim Vergrößern eines Ausschnitts ging aber hervor, dass die Erfassung ca. alle 15 Minuten geschah. Es wird für die folgenden Unterkapitel zur Auswertung erstmal nicht mit berücksichtig bei der Gegenüberstellung in Unterkapitel 5.3.4 wird es aber wieder aufgegriffen.

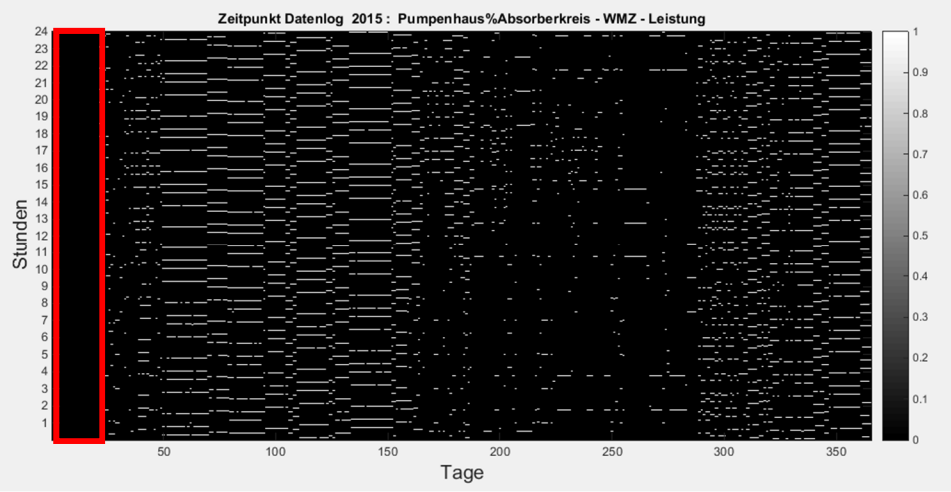


Abbildung 40: Ausfall Aufzeichnung der Leistung

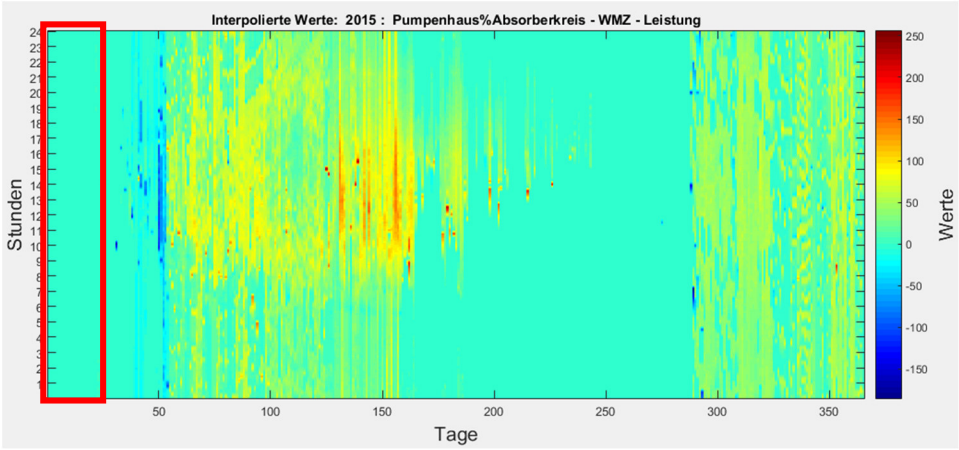


Abbildung 41: Darstellung

Negative Leistung:

Das Phänomen der negativen Leistung des Absorberkreises, zu sehen in der Abbildung 41 als blau dargestellte Werte, konnte in der Anlagenvisualisierung (Abb. 42) beobachtet werden. Die von den Solarröhrenkollektoren über die Weiche eingespeiste Sole (1) durchmischt sich in der Weiche mit der kalten Sole aus dem Eispeicher (2) bzw. wärmt die Weiche durch. Aus der „kalten" Seite der Weiche (3) speist sich der Absorberkreislauf. Die Vorlauftemperatur (4) des Absorbers ist jetzt so hoch, dass Wärme an die Umgebung abgegeben und die Rücklauftemperatur (5) somit geringer wird und es zu der negativen Leistung in der Messung kommt.

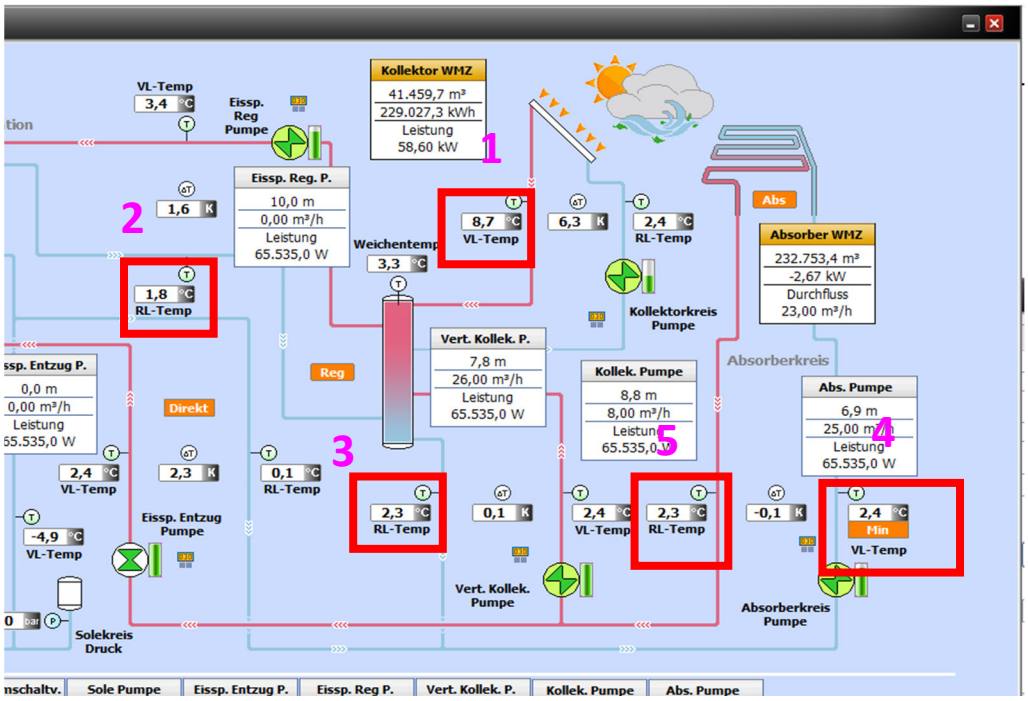


Abbildung 42: Temperaturen, negativ Leistung des Absorbers

5.3.2 Aufbereitung der Daten

Zunächst muss geklärt werden, wie die Anlage geschaltet ist bzw. konkret, wie die Sole in der Anlage gefahren wurde. Hierfür wurden Heatmap-Daten der Drehzahlen der Pumpen verwendet. In der Tabelle auf der nächsten Seite wird aufgelistet, mit welchen Drehzahlen sich die Pumpen bewegen, um somit einen Schwellenwert zu ermitteln, bei dem die Pumpen eindeutig in Betrieb waren. Die Werte wurden aus den generierten Datensätzen aus der Toolbox abgelesen.

Tabelle 8: Pumpendrehzahl

Pumpe	Drehzahl [1/min]	Schwellenwert [1/min]
Absorberkreis (1)	800-1500	800
Kollektorkreis (2)	1400-2800	1400
Solekreis (WP-Verdampfer) (3)	2400-2900	2400
Eisspeicher-Entzug [11] (4)	780-1300	780
Eisspeicher-Regenration (5)	800-1500	800
Verteiler-Kollektor (6)	Keine Daten protokolliert	

In der Abbildung 43 sind alle relevanten Pumpen[12] aus der Anlagenvisualisierung der GA zu sehen (nächste Seite), die für die weitere Untersuchung wichtig sind. Eine Vergrößerte Abbildung ist dem Anhang A.2 zu entnehmen.

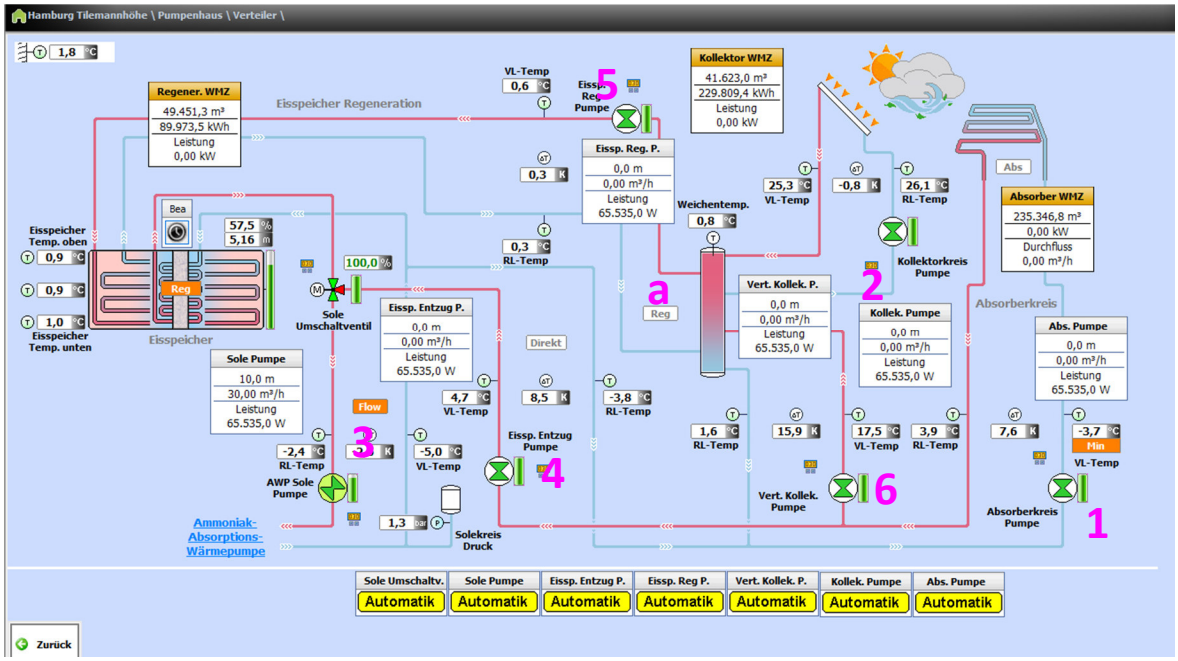


Abbildung 43: Pumpenhaus aus GA-Software

Als Schwellenwert wird die untere Drehzahl der Pumpen aus der Tabelle 8 (Seite 44) für die folgende Analyse genutzt. Es wird untersucht, wann der Absorber die Sole über den WP-

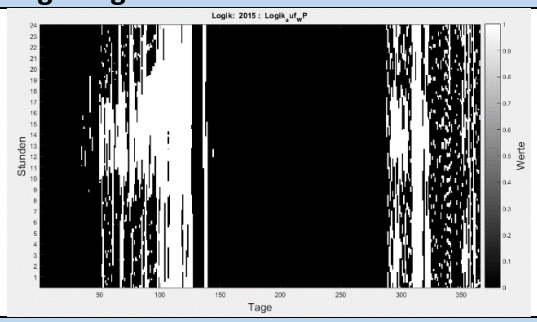
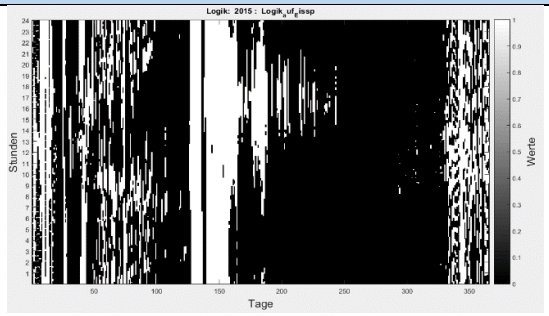
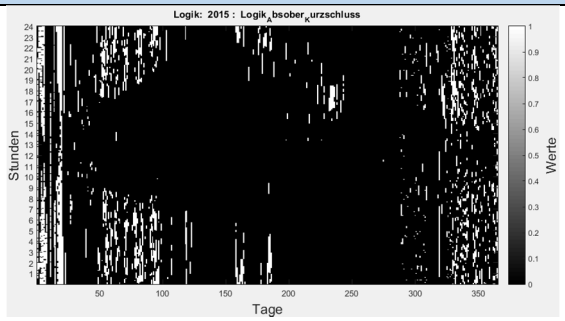
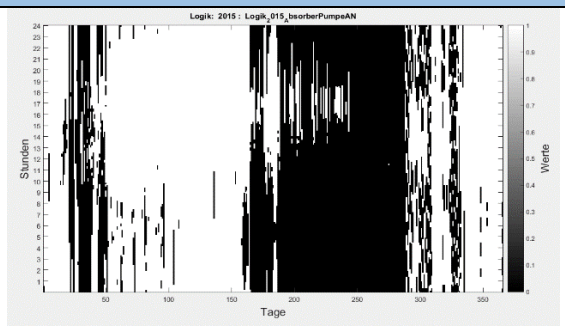
[11] Der Name ist irreführend, da die Pumpe zu dem WP-Verdampfer führt. Es wird sich aber hier an die Bezeichnung gehalten, die in der Anlagenvisualisierung zu finden ist.

[12] Die Pumpen werden als grünes Objekt in einem Kreis dargestellt.

Verdampfer gefahren hat, wann diese über den Eisspeicher gefahren wurde und ob es noch einen Zustand gibt, in den weder auf den WP-Verdampfer oder den Eispeicher gefahren wurde. Der letzte Zustand wird hier als „Absorber auf Kollektor“ (siehe Tabelle 9) bezeichnet, dabei wird die Sole über den Verteiler-Kollektor (Abb. 41 a) gefahren und zirkuliert zwischen der Verteiler-Kollektor und dem Absorber wenn der Verteiler-Kollektor durchgeladen ist wird die Sole aus dem Verteiler-Kollektor zum beladen des Eisspeichers genutzt. Da dieser bei der Beobachtung der Anlagenvisualisierung der Gebäudeautomation beobachtet werden konnte, sollte er weiter untersucht werden.

Der auf der nächsten Seite folgenden Tabelle 9 (folgende Seite) kann man den Aufbau der Logik entnehmen. In den Ergebnissen der Logik entspricht ein weißer Pixel dem Wert 1 „wahr“ und schwarz dem Wert 0 „nicht wahr“. Wenn der Wert wahr angenommen wird, bedeutet dies, dass die Pumpen laufen. Zur Erzeugung der Logik können auch andere Werte der Pumpe herangezogen werden, wie z.B. die Durchflussmenge oder die elektrische Leistungsaufnahme, wichtig ist nur, dass es sich um ein Kriterium handelt, an dem eindeutig feststellbar ist, dass das System Pumpe in Betrieb war.

Tabelle 9: Logik Aufbau

Logik Aufbau	
Absorber auf WP	**Logik Ergebnis**
Pumpenhaus%Absorberkreis - Pumpe – Drehzahl > 800 & Pumpenhaus%Eisspeicher Entzug - Pumpe - Drehzahl > 780	
Absorber aus Eisspeicher	
Pumpenhaus%Absorberkreis - Pumpe - Drehzahl > 800 & Pumpenhaus%Eisspeicher Entzug - Pumpe - Drehzahl < 780 & Pumpenhaus%Eisspeicher Regeneration - Pumpe - Drehzahl > 800	
Absorber auf Kollektor	
Pumpenhaus%Absorberkreis - Pumpe - Drehzahl > 800 & Pumpenhaus%Eisspeicher Entzug - Pumpe – Drehzahl < 780 & Pumpenhaus%Eisspeicher Regeneration - Pumpe – Drehzahl < 800	
Absorber Pumpe An	
Pumpenhaus%Absorberkreis - Pumpe - Drehzahl > 800	

5.3.3 Auswertung der anteiligen Betriebszeit

Mit der Funktion Datenanalyse.m kann, in den figures der einzelnen Logik-Daten (Tabelle 9 S.46), ausgewertet werden, wie der Absorber zeitlich anteilig die Sole gefahren hat. Dazu wird die Summe aller Minuten, die den Wert 1 besitzen, aus der ausgegebenen Tabelle von der Funktion Datenanalyse.m genutzt. In der Tabelle 9 sind die entnommenen Werte zu sehen. Die in der Tabelle 10 stehenden Spalten (5) und (6) wurden zur Kontrolle mit aufgenommen, eine große negative Differenz der Werte (6) würde noch auf einen weiteren Schaltzustand der Pumpen hindeuten. Die Abweichung von -3 Minuten kann vernachlässigt werden, da davon ausgegangen werden kann, dass es sich dabei nicht um einen weiteren Schaltzustand handelt, sondern auf die Datenaufbereitung zurückzuführen ist.

Zur Berechnung der zeitlichen Anteile in Prozent wird sich immer auf die 306131 Minuten bezogen, in denen die Absorber-Pumpe in Betrieb war. Exemplarisch ist die Rechnung einmal durchgeführt worden.

$$Zeitlicher\ Anteil\ der\ Sole\ auf\ den\ Eisspeicher\ in\ \% = \frac{139387\ Min}{306131\ Min} * 100\% \approx 45{,}53\%$$

Tabelle 10: Betriebszeit Absorber

	Absorber auf Eisspeicher (1)	**Absorber auf WP-Verdampfer (2)**	**Absorber auf Kollektor (3)**	**Absorber Pumpe AN (4)**	**Summe (5 =1 +2+3)**	**Differenz (6 = 5 zu 4)**
Min.	139387	121899	44842	306131	306128	-3
Stunden	2323,1	2031,7	747,4	5102,2	5102,2	0
% Stunden auf (4) bezogen	45,53%	39,82%	14,65%	100%	99,999%13	-0,001%-

Die Untersuchung der zeitlichen Anteile wurde erstmals als Grundlage genutzt, um zu klären, welche Schaltzustände vorliegen und, ob eventuell ein Zustand übersehen wurde. Die Auswertung zeigt, dass die vorher angenommenen drei Zustände auch vorliegen. Für eine weitere Untersuchung wird im Anschluss die Wärmemenge angeschaut.

5.3.4 Auswertung der anteiligen Wärmemenge

Für die weitere Betrachtung wird zur Vereinfachung angenommen, dass die Leistung des Absorbers mit dem Umschalten der Pumpen auch unverzüglich auf das Zielsystem Wärmepumpe, Eispeicher oder den Verteiler-Kollektor gelangt. Wenn man diese Vereinfachung nicht trifft, müssten noch Totzeiten und Verluste berücksichtigt werden, bis die bilanzierte Sole das jeweilige System erreicht. Die Totzeit entspricht der Zeit, die ein Sole-

[13] Gerundet 99,999 % im Vergleich zu dem Wert aus (4) 306131 Minuten.

Partikel durch die Rohre von der Bilanzgrenze des System A zu der Bilanzgrenze des System B braucht, was von dem Massestrom, der Länge und dem Querschnitt der Leitung abhängt.

Um die anteilige Wärmemenge zu bestimmen, die zu dem Eisspeicher bzw. dem WP-Verdampfer geleitet wurde, wird die Heatmap der Leistung des Absorberkreis (Abb. 44) mit den Logiken aus Tabelle 8 kombiniert. Dazu wurde die Heatmap der Leistung des Absorberkreises mit der Funktion „Datenanalyse.m“ bearbeitet und nur die positiven Werte ausgegeben, da in der Heatmap auch negative Leistung zu verzeichnen ist.

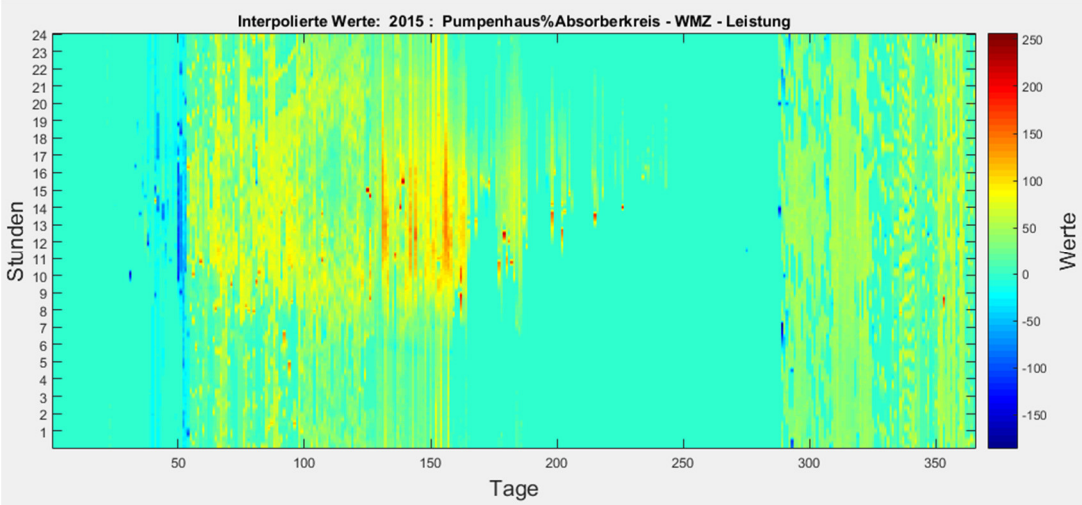


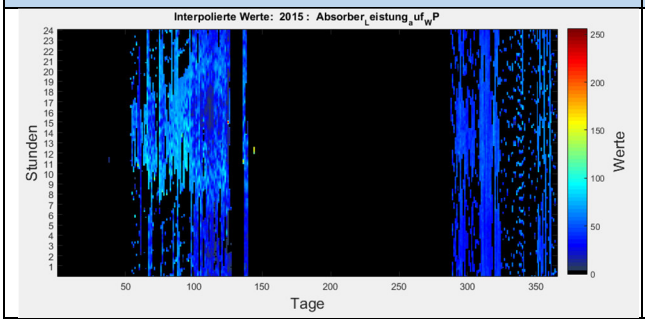
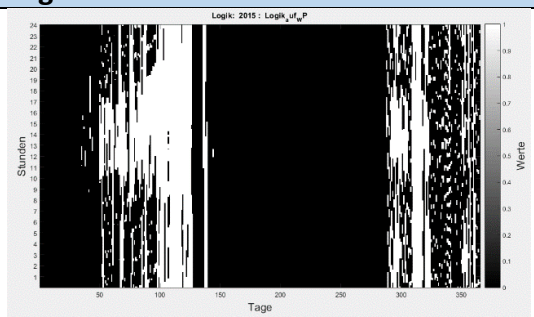
Abbildung 44: Absorber Leistung 2015 in [kW]

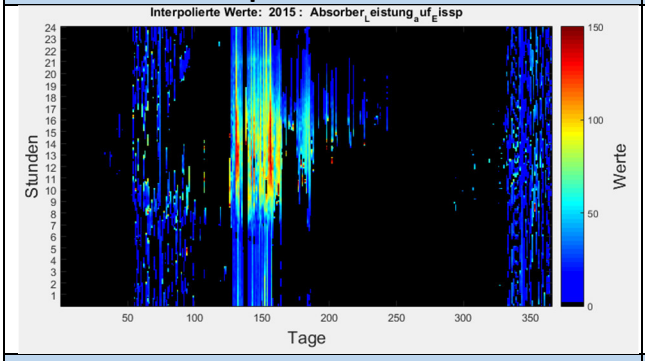
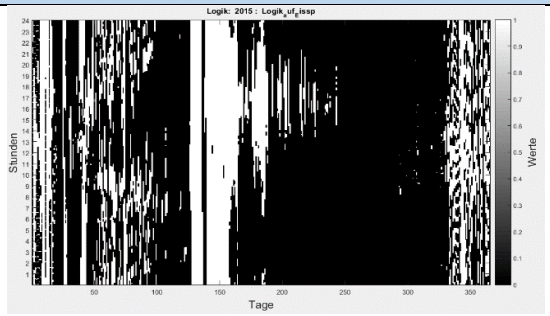
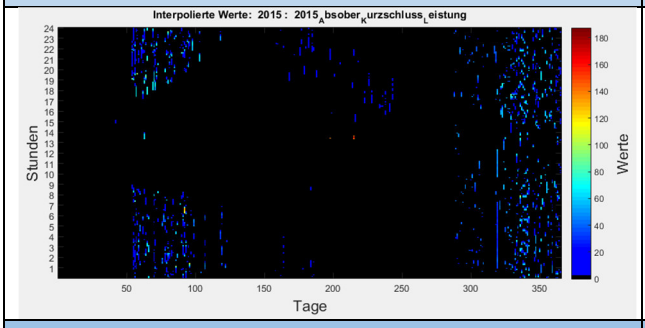
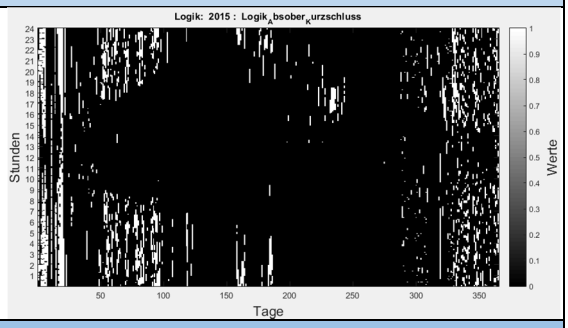
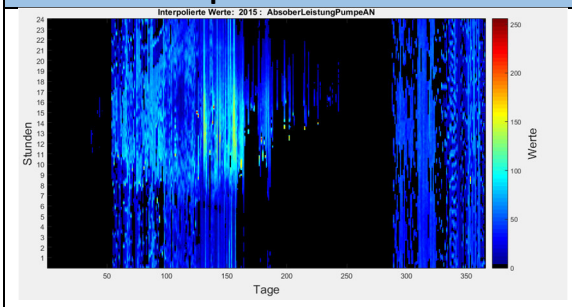
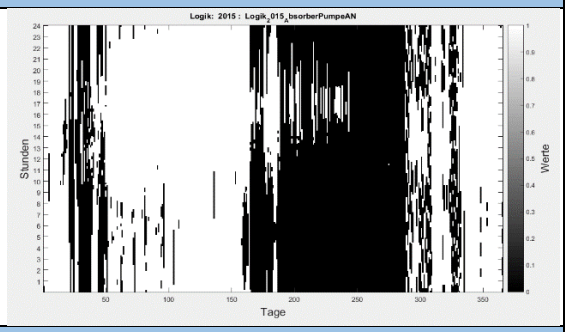
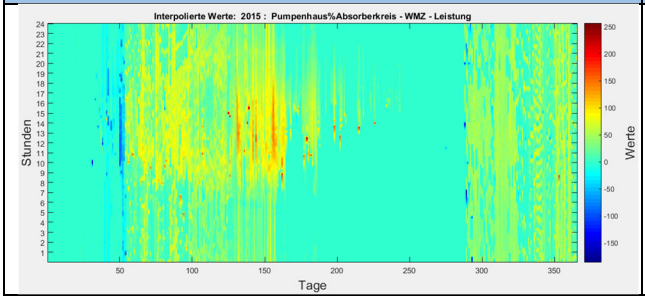
.

In der Tabelle 11 sind die verschnittenen Daten der Logik und der Heatmap der Leistung des Absorberkreislaufs zu sehen. Dazu wurde die Funktion „Manipulation2Heatmaps.m“ genutzt und die Logik mit der Absorber-Leistung multipliziert. Diese neuen Heatmaps wurden mit der Funktion „Datenanalyse.m“ ausgewertet, die Ergebnisse sind auf der nächsten Seite in Tabelle 11 zu finden.

Zur besseren Darstellung wurden die Daten in Tabelle visuell aufbereitet, so dass der Wert „0“ schwarz dargestellt ist

Tabelle 11: Aufteilung Leistung , Tabelle geht nächste Seite weiter.

Aufteilung der Leistung über die Logik	
Absorber auf WP	**Logik**
	

Absorber auf Eisspeicher	
	
Absorber auf Kollektor	
	
Absorber Pumpe An	
	
Absorber Leistung Original	-
	

Ende Tabelle 11

Zur Berechnung der Wärmemenge werden die Elemente der Matrix aufsummiert und mit 1/60 multipliziert, um auf den Wert kWh zu kommen. Für die in der Tabelle 12 (nächste Seite) berechneten Werte ist der Startwert die Minute „1“ und der Endwert n, in diesem Fall die letzte Minute 525600 des Jahres 2015. Es wurden hierbei nur die positiven Werte berücksichtigt, die man aus der Funktion Datenanalyse.m entnehmen kann.

$$\sum_{i=1}^{i=n} X_i \; kW \; Minute * \frac{1h}{60 \; Minuten} = \sum_{i=1}^{i=n} X_i \frac{1}{60} \; kWh$$

Tabelle 12: Wärmemenge des Absorbers bezogen auf die Ursprungs Heatmap

	Absorber auf Eisspeicher (1)	Absorber auf WP-Verdampfer (2)	Absorber auf Kollektor (3)	Absorber Leistung (4)	Summe (5 =1 +2+3)	Differenz (6 = 5 zu 4)
Arbeit nur positiv [KWh]	64711	93257	8049,9	171390	166017,9	-5372,1
Arbeit % auf Absorber bezogen	37,76%	54,41%	4,7%	100%	96,87%	-3,13%

Im Gegensatz zu der zeitlichen Aufteilung gibt es in Tabelle 11 (6) größere Abweichungen. Dies ist darauf zurückzuführen, dass sich als Referenz für die Leistung in der Tabelle auf die originale Heatmap der Absorberleistung (Tabelle 11: Absorber Leistung Original) bezogen wurde. Die Abweichung aus der Tabelle 11 kann folgende Ursachen haben: zum einem die Interpolation zwischen längeren Zeiträumen mit großen Differenzen der Werte und zum anderen, dass die Pumpen nicht unmittelbar die Drehzahl des Schwellenwertes erreichen und immer eine Anlauf- sowie Auslaufzeit besitzen.

Bei Tabelle 13 hingegen wurde auf den Verschnitt der Leistung über die Logik Bezug genommen (Tabelle 11: Absorber Pumpe AN), die Abweichungen sind bei der Aufteilung minimal und in der Größenordnung der zeitlichen Aufteilung. Dieses muss auch zwangsläufig so sein, da es sich analog zur zeitlichen Aufteilung verhält und alle Schaltzustände ermittelt wurden.

Tabelle 13:Absorberleistung nur auf Logik bezogen

	Absorber auf Eisspeicher (1)	Absorber auf WP-Verdampfer (2)	Absorber auf Kollektor (3)	Absorber Pumpe AN (4)	Summe (5 =1 +2+3)	Differenz (6 = 5 zu 4)
Arbeit nur positiv [KWh]	64711	93257	8049,9	166020	166017,9	-2,1
Arbeit nur positiv [KWh]	38,98%	56,17%	4,85%	100%	99,999%	-0,001%

5.3.5 Gegenüberstellung Betriebszeit und Wärmemenge

In Tabelle 14 werden die ermittelten prozentualen Anteile gegenübergestellt um Rückschlüsse auf das Verhalten der Zustände zu ziehen. Der Zustand „Absorber auf Kollektor“ bringt 4,7% der insgesamt erbrachten Wärmemenge auf, bei einer Betriebszeit von 14,65% der Absorber-Pumpe. Das Verhältnis wird sich zwar durch den Ausfall der Aufzeichnung der Absorber-Leistung (5.3.1) zu Gunsten der Zustände „Sole auf Eisspeicher“ und „Absorber auf Kollektor“ verschieben, da in diesem Bereich nur die beiden Zustände vorlagen (siehe Tabelle 11). Ob dieser Zustand „Absorber auf Kollektor“ energetisch Sinnvoll ist, müsste anhand der aufgebrachten Pumpen-Leistung und der damit einhergehenden elektrischen Arbeit untersucht werden. Der Zustand ermöglicht es aber auch, dass der Eispeicher regeneriert werden kann und sich die Grenze des Durchfrierens des Eispeichers nach hinten verschiebt.

Tabelle 14: Gegenüberstellung Betriebszeit und Wärmemenge

	Sole auf Eisspeicher (1)	**Sole auf WP-Verdampfer (2)**	**Absorber auf Kollektor (3)**	**Absorber Pumpe AN (4)**	**Summe (5 =1,2,3)**	**Differenz (5 zu 4)**
% Stunden auf Absorber bezogen	45,53%	39,82%	14,65%	100%	99,999%	-0,001%-
Arbeit % auf Absorber bezogen	37,76%	54,41%	4,7%	100%	96,87%	-3,13%

6 Fazit und Ausblick

6.1 Fazit

Durch die steigende Komplexität neuer Anlagen steigt auch die Anzahl der zu protokollierenden Datenpunkte. Damit einher geht auch ein steigender Bedarf an Analysefähigkeit. Es konnte beispielhaft gezeigt werden, wie sich durch das gezielte aussuchen von Datenpunkten mehrerer voneinander abhängiger Systeme ein Überblick für eine weitere Analyse verschafft werden kann. Anhand der Visualisierungen kann zudem beispielsweise festgestellt werden, wie die Regelung eingestellt ist, und auch Verbrauchs-/Nutzerverhalten können aus den visualisierten Daten abgeleitet werden. Ein weiterer Vorteil ist, dass die Heatmap immer in der Struktur normiert ist und der Betrachter nicht lange über die zeitliche Einordnung der Daten nachdenken muss. So lassen sich die Daten auch von mehreren Jahren gut und übersichtlich gegenüberstellen.

Die Weiterverarbeitung der Daten mit dem im Rahmen dieser Arbeit entwickelten MATLAB-Tool, wie zum Beispiel der Logik, ist nicht gebunden an die Darstellungsform der Heatmap, sondern basiert auf der Datenrekonstruktion der Rohdaten durch Interpolation und der Normierung des strukturellen Aufbaus, die zwangsläufig durch die Erzeugung einer Heatmap generiert werden. Durch die gegebene Logik können schnell Schaltzustände geprüft und durch dieselbe Darstellungsform erfasst sowie anderen Systemen zugeordnet werden.

Wie jedes andere Verfahren steht und fällt das System mit der Datenqualität und damit, wie sich das reale Verhalten aus diesen Daten rekonstruieren lässt. Aus diesem Grund sollte schon in der Planungsphase eng mit allen Beteiligten an dem Konzept des technischen Monitoring gearbeitet werden.

6.2 Ausblick

Das Werkzeug (Heatmap-Toolbox) soll Einzug in den Alltag des Ingenieurbüros erhalten. Die momentane zur Erfassung notwendige standardisierte Form der Rohdaten-Dateien, wie in Kapitel 2.3 Datenbeschaffung und Datenaufbau beschrieben, soll allgemeiner geschehen und auch gestückelte Daten, die auf mehrere Dateien verteilt sind, automatisiert eingelesen werden können. Des Weiteren soll geprüft werden, ob die Methode auf Simulationsergebnisse von Anlagen angewandt werden kann und, ob die Ergebnisse der Simulation als ideales Referenz-Bild dienen können.

7. Literaturverzeichnis

[1] VDI 6041 Facility-Management Technisches Monitoring von Gebäuden und gebäudetechnischen Anlagen, April 2015, Beuth Verlag

[2] Seite, https://upload.wikimedia.org/wikipedia/commons/0/0e/Automatisierungspyramide2.svg, In: Wikipedia, Die freie Enzyklopädie 16:16, 5. Dez. 2015, (Abgerufen 20 Dezember 2016)

[3] VDI 3814 Blatt 1, Gebäudeautomation (GA); Systemgrundlagen, 7 – November 2009, Beuth Verlag

[4] Bernd Aschendorf, Energiemanagement durch Gebäudeautomation

Grundlagen - Technologien - Anwendungen (Kapitel 5), Vieweg Springer, 2014

[5] VDI 4700, Blatt 1 Begriffe der Bau- und Gebäudetechnik, Oktober 2015 , Beuth Verlag

[6] Richtlinie zur Förderung der Nutzung von Energie aus erneuerbaren Quellen und zur Änderung und anschließenden Aufhebung der Richtlinien 2001/77/EG und 2003/30/EG Abrufbar, http://eur-lex.europa.eu/legal-content/de/ALL/?uri=CELEX%3A32009L0028 [Zugriff am 3.Januar 2017]

[7] Seite „Heatmap“. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 21:10, 25. Okt. 2016. URL: https://de.wikipedia.org/wiki/Heatmap (Abgerufen: 30. November 2016)

[8] Seite, https://de.mathworks.com/help/matlab/ref/imagesc.html (Abgerufen 20 Dezember 2016) «eigene Übersetzung»

[9] David Jonathan Fuhrländer, Analyse und Evaluation des Betriebsverhaltens eines

realen Nahwärmenetzes, 10.03.2016, Fachhochschule Flensburg

[10] Elmar Bollin, Automation regenerativer Wärme- und Kälteversorgung von Gebäuden : Komponenten, Systeme, Anlagenbeispiele, - 1. Aufl. - Wiesbaden , 2009, Vieweg + Teubner

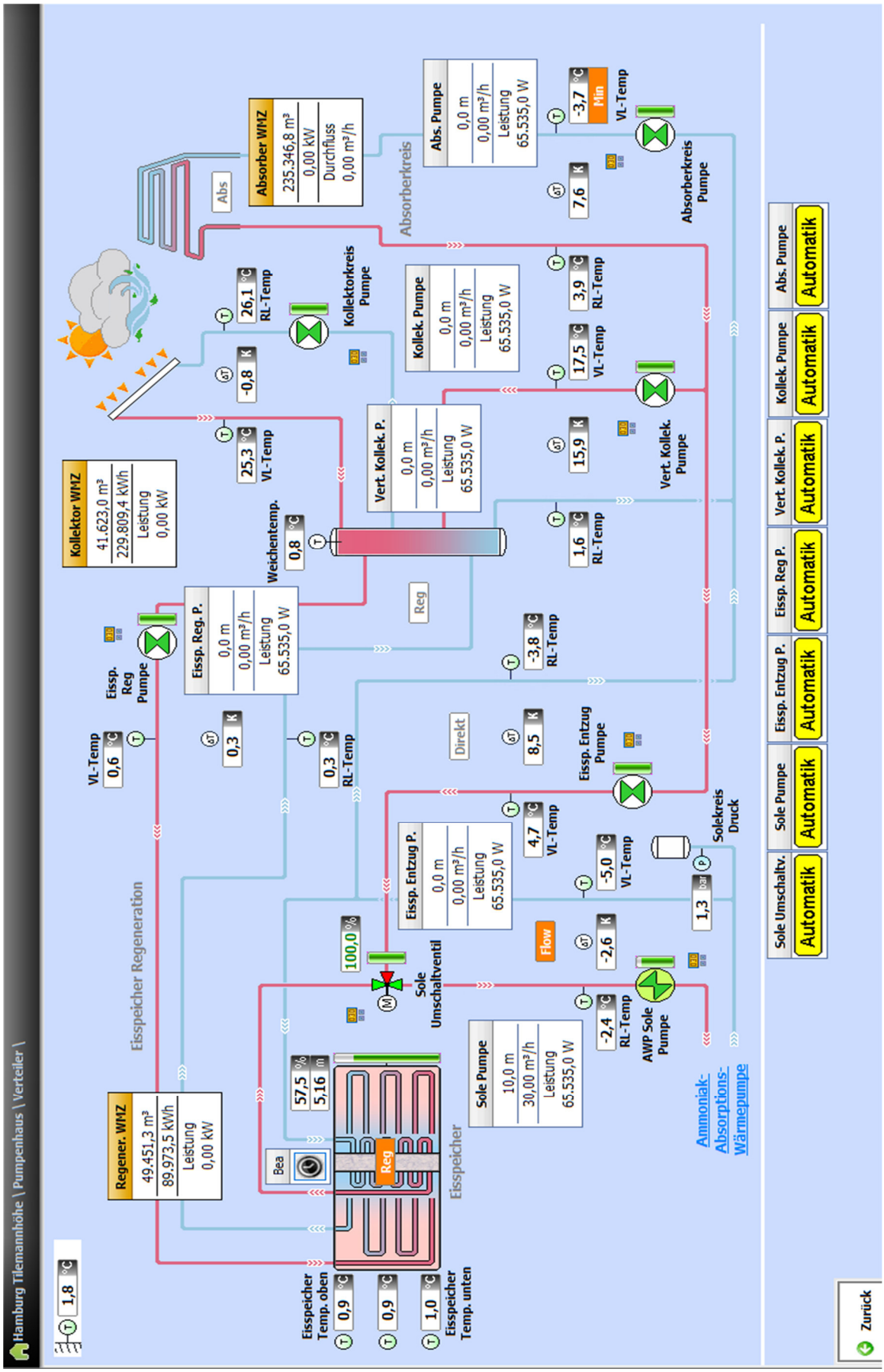
A Anhang

A.1 Merkmale von TM-Systemen

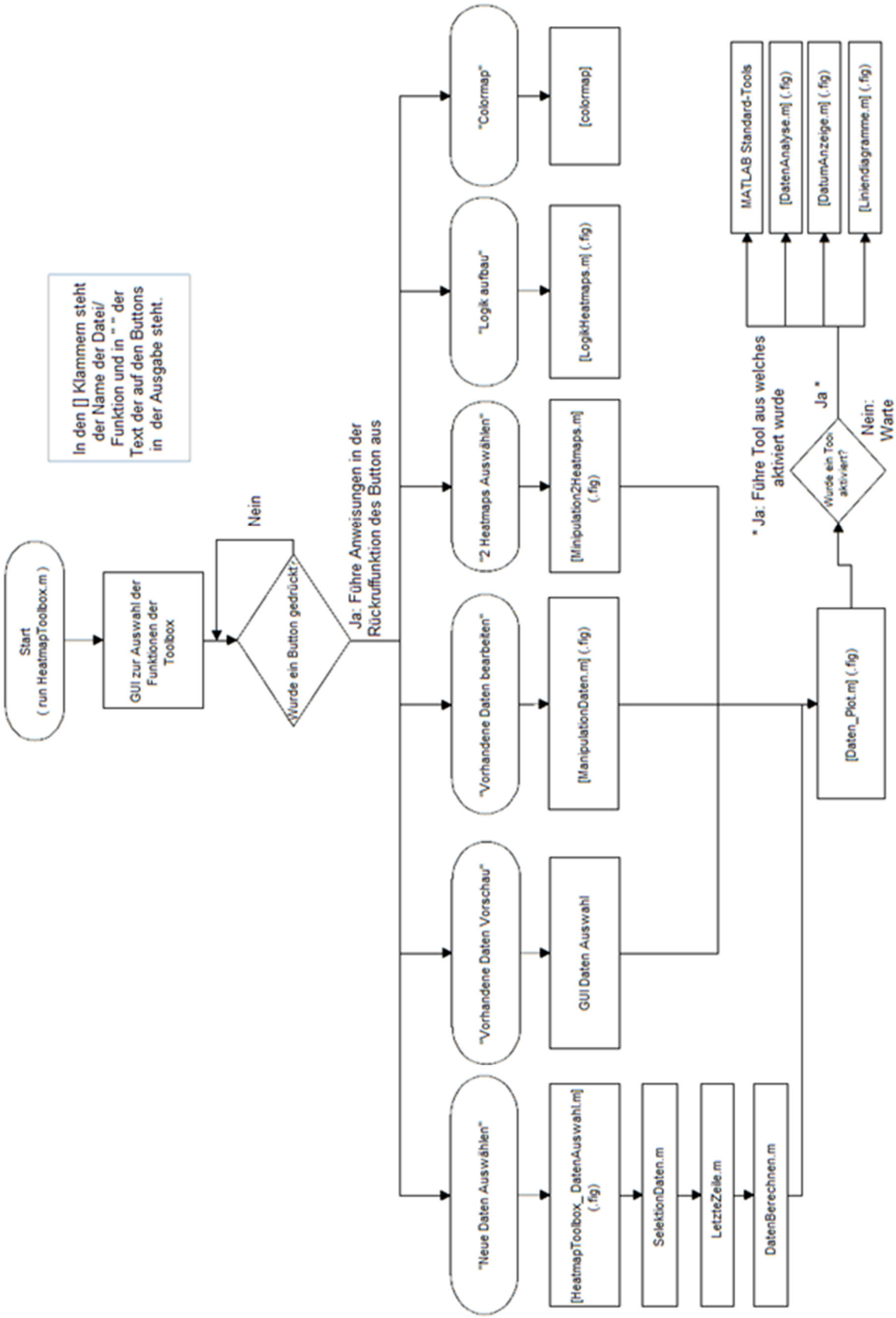
Tabelle 3. Merkmale von TM-Systemen

	Kategorie		
TM-System	**III niedrig (DIN EN 15232, Klasse C)**	**II mittel (DIN EN 15232, Klasse B)**	**I hoch (DIN EN 15232, Klasse A)**
Datenerfassung	Art: Manuell oder mobil Intervall: niedrige zeitliche Auflösung, das heißt größere Abstände, meist Wochen oder Monate	Art: meist automatisch mit fernauslesbaren Zählererfassungssystemen Intervall: mittlere zeitliche Auflösung, z. B. tage- oder wochenweise	Art: automatisiert, z. B. Fernübertragung, Bussysteme oder Prozessinformationssystem Intervall: hohe zeitliche Auflösung, z. B. minuten- oder sekundengenau
Energiemessung	• Hauptzähler • Betriebsstundenzähler (statt Energieverbräuchen)	• Hauptzähler • Unterzähler • teilweise Betriebsstundenzähler	• Hauptzähler • Unterzähler • wichtige Aggregate • alle Energieformen
Zustandssignale	Monitoring der Stellsignale (nicht tatsächliche Istsignale)	• Istsignale (im Ausnahmefall Stellsignale) • wichtigste Alarme	• Istsignale • Alarme
Bedarfserfassung	wenige bis keine Signale	wichtigste Signale an Hauptanlagen (nicht zonenbezogen)	alle Bedarfssignale
Änderungsmanagement	keine systematische Aufzeichnung	systematische Aufzeichnung von • Änderungen mit Zeit • Personenkennung • altem/neuem Wert • allen Werten summarisch nach lokalem Zugriff und Fernzugriff	systematische Aufzeichnung von • Änderungen mit Zeit • Personenkennung • altem/neuem Wert • allen Werten summarisch auf individuellen Nutzer bezogen
persönlicher Zugriff	Generalschlüssel	mindestens zwei Schlüssel (lokale Bedienung und Fernbedienung)	individueller Schlüssel für jeden Nutzer
Fernzugriff	nein	Fernzugriff auf: • Monitoringdaten • Alarme • Zeitprogramme	Fernzugriff auf alle Daten mit sehr kleiner zeitlicher Verzögerung
Datennormierung	manuell	automatisch	automatisch
Datensicherheit	Sicherung über festgelegte Zeit (z. B. 48 Stunden) bei Netzausfall oder Neustart manuelle Synchronisation	Sicherung über festgelegte Zeit (z. B. 48 Stunden) bei Netzausfall oder Neustart automatische Datenübertragung und Synchronisation	Sicherung über festgelegte Zeit (z. B. 48 Stunden) bei Netzausfall oder Neustart automatische Datenübertragung und Synchronisation
Datenqualität	wird nicht aufgezeichnet	Erfassung der Signalqualität innerhalb- oder außerhalb der Kennwerte eines Sensors	Erfassung der vollständigen Signalqualität: • Verfügbarkeit des Signals • Innerhalb- oder außerhalb der Kennwerte • Kommunikation
Auswertung	Auswertung der Energie- und Ressourcenverbräuche	Genauere Analysen sind möglich. Auswertung berücksichtigt die (wichtigsten) automatisch eingestellten Zeiten.	Zusätzlich zu den Analysen nach III und II sind aufwendige statistische Untersuchungen möglich, z. B. automatische Fehlererkennung. Auswertung berücksichtigt automatisch: • Programmierung • Wechsel der lokalen Zeitprogramme
Schnittstelle/ Datenausgabe	Berichte und Kostenzuordnung stehen erst nachträglich, oft mit hoher zeitlicher Verzögerung bereit.	neben Verbrauchserfassung z. B. auch Lastgänge oder Energiebilanzen	Zusätzlich zu den üblichen Berichten (siehe III und II) sind aufwendige statistische Untersuchungen möglich. Daten können auch direkt für MSR genutzt werden.
Potenziale	begrenztes Potenzial zur Energieeinsparung und Optimierung	erweitertes Potenzial zur Energieeinsparung und Optimierung	Aufgrund der Vielzahl der Daten können praktisch alle Einspar- und Optimierungspotenziale ausgeschöpft werden.
IT-Lösungen	geringe Anforderungen, z. B. für Kostenzuordnung	Datentransfer (Netzwerk), zentrale Bereitstellung und Aufbereitung der Daten	zwingend erforderlich für Datentransfer, Aufbereitung, Auswertung, Übergabe an MSR

A.2 Screenshot aus der Gebäudeautomation



A.3 Programmablaufplan



A.4 Programmier-Code und Datensätze die verwendet wurden

Der MATLAB Programm-Code ist in der PDF-Version dieses Dokumentes unter A.4 angehangen. Die MATLAB-Dateien, HTML-Dateien sind auf dem digitalen Datenträger vorhanden. Die verwendeten MATLAB Datensätze sind ebenfalls auf dem digitalen Datenträger zu finden.

A.4.1 Daten_Plot.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% In dieser Funktion werden die berechneten Daten in figure-Elementen
grafisch ausgegeben.
% Die Matrix IntpolWerte welche die Heatmap darstellt, bekommt noch weitere
Funktionen in Form von Buttons zum auswählen

function Daten_Plot(DatenName,JahrName, Rohdaten01, QulitaetWerte,
RohdatenWerte,IntpolWerte, Historie)

    %$ Ausgebe der Matrix IntpolWerte
    fig1 = figure;
            %$ Erstellen der Buttons für die Funktionen der
Weiterverarbeitung in dem Plott Heatmap. Nur für fig1.

                %§ Einfügen des Toolbar-Button mit Function aufruf bei
Callback
                    hToolbar = findall(fig1,'tag','FigureToolBar');
                    % Einlesen der Bild Datei für Button
                    [img,map] = imread(fullfile(matlabroot,...
                                'toolbox','matlab','icons','bze.gif'));
                    % Konvertiere Bild zu RGB fürs Icon
                    icon = ind2rgb(img,map);
                    % Erstelle ein uipushtool in der toolbar
                    tbh = findall(hToolbar,'Type','uitoolbar');
                    p = uipushtool(tbh,'TooltipString','Datenanalyse',...
                                    'Separator','on',...
                                    'HandleVisibility','off', ...
                                    'ClickedCallback',{@Datenanalyse,
DatenName, JahrName});
                                    %§§§§§§§§§§§§§§§§   %FUNCTION
Datenanalyse
                                    % Bei einem ClickedCallback von dem
Icon wird die Funktion Datenanalyse gestartet und die
                                    % Variablen DatenName und JahrName
übergeben
                    % Setze den Button
                    p.CData = icon;
                %/ Einfügen des Toolbar-Button mit Function aufruf bei
Callback

                %§ Einfügen des Toolbar-Button mit Function aufruf bei
Callback
                    % Einlesen der Bild Datei für Button
                    [img,map] = imread(fullfile(matlabroot,...
                                'toolbox','matlab','icons','bze.gif'));
```

```
                    % Konvertiere Bild zu RGB fürs Icon
                    icon = ind2rgb(img,map);
                    % Erstelle ein uipushtool in der toolbar
                    stbh = findall(hToolbar,'Type','uitoolbar');
                    sp = uipushtool(stbh,'TooltipString','Datum
Anzeige',...
                                    'ClickedCallback',{@DatumAnzeige,
JahrName});
                                    %§§§§§§§§§§§§§§§§   %FUNCTION
DatumAnzeige
                    % Setze den Button
                    sp.CData = icon;
                %/ Einfügen des Toolbar-Button mit Function aufruf bei
Callback

                %§ Einfügen des Toolbar-Button mit Function aufruf bei
Callback
                    % Einlesen der Bild Datei für Button
                    [img,map] = imread(fullfile(matlabroot,...
                                'toolbox','matlab','icons','bze.gif'));
                    % Konvertiere Bild zu RGB fürs Icon
                    icon = ind2rgb(img,map);
                    % Erstelle ein uipushtool in der toolbar
                    stbh = findall(hToolbar,'Type','uitoolbar');
                    sp =
uipushtool(stbh,'TooltipString','Liniendiagramme',...
                                    'ClickedCallback',{@Liniendiagramme,
DatenName, JahrName,IntpolWerte});
                                    %§§§§§§§§§§§§§§§§   %FUNCTION
Liniendiagramme
                    % Setze den Button
                    sp.CData = icon;
                %/ Einfügen des Toolbar-Button mit Function aufruf bei
Callback

        %/ Erstellen der Buttons für die Funktionen der Weiterverarbeitung
in dem Plott Heatmap. Nur für fig1.

            % Konfiguration, Beschriftung und Formatierung der Grafik,
Achsen und Legenden
            % Exemplarisch wird im folgenden auskommentiert was in den
Anweisungen passiert

            set(fig1, 'WindowStyle', 'docked')    % Die figure soll an dem
Fenster angedockt sein
            set(fig1, 'Name','Heatmap','NumberTitle','off');    %Ausgabe
Name, Nummer der figure nicht anzeigen
            TitelInterp=strjoin({'Interpolierte Werte: ', JahrName,':
',DatenName}); % Konstruiere Titel Name
            imagesc(IntpolWerte);        %grafische Ausgabe der Matrix
            set(gca,'ydir','normal') ;   %Drehung der y-Achse. Da sonst
oben mit der Minute 0 begonnen wird
            title(TitelInterp);          %Übergebe Titel Name
            %Beschriftung der Achsen mit Stunden statt Minuten
            ax = gca;
            ax.YTick = [60 120 180 240 300 360 420 480 540 600 660 720 780
840 900 960 1020 1080 1140 1200 1260 1320 1380 1440];
            ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
19 20 21 22 23 24};
```

```
            xlabel('Tage', 'FontSize', 16)      % Formatierung der Schrift
für die x-Achse
            ylabel('Stunden', 'FontSize', 16)   % Formatierung der Schrift
für die y-Asche
            colormap jet;                       %Definition der Coloramp
mit dem Farbskala "jet"
            h = colorbar();                     % Gibt die Farbskala mit
aus
            ylabel(h,'Werte','FontSize', 16)    % Formatierung der Schrift
für die Colorbar
            setFigDockGroup(gcf,DatenName);     % Andocken der figure an
Gruppe DatenName(Variable)
 %/ Ausgebe der Matrix IntpolWerte

    %$ Ausgabe der Matrix "Rohdaten01"
    fig2 = figure;
    set(fig2, 'WindowStyle', 'docked')
    set(fig2, 'Name','Zeitpunkt','NumberTitle','off');
    Titel01=strjoin({'Zeitpunkt Datenlog ', JahrName,': ',DatenName});
    imagesc(Rohdaten01, [0 1]);
    title(Titel01);
    set(gca,'ydir','normal');
    ax = gca;
    ax.YTick = [60 120 180 240 300 360 420 480 540 600 660 720 780 840 900
960 1020 1080 1140 1200 1260 1320 1380 1440];
    ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24};
    xlabel('Tage', 'FontSize', 16)
    ylabel('Stunden', 'FontSize', 16)
    colormap gray
    colorbar
    setFigDockGroup(DatenName);
    %/ Ausgabe der Matrix "Rohdaten01"r

    %$ Ausgabe der Matrix "RohdatenWerte"
    fig3 = figure;
    set(fig3, 'WindowStyle', 'docked');
    set(fig3, 'Name','Zeitwerte','NumberTitle','off');
    TitelWerte=strjoin({'Zeitpunkt Werte ', JahrName,': ',DatenName});
    imagesc(RohdatenWerte);
    set(gca,'ydir','normal')
    title(TitelWerte)
    ax = gca;
    ax.YTick = [60 120 180 240 300 360 420 480 540 600 660 720 780 840 900
960 1020 1080 1140 1200 1260 1320 1380 1440];
    ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24};
    xlabel('Tage', 'FontSize', 16)
    ylabel('Stunden', 'FontSize', 16)
    colormap jet;
    h = colorbar();
    ylabel(h,'Werte','FontSize', 16)
    setFigDockGroup(DatenName);
    %/ Ausgabe der Matrix "RohdatenWerte"

    %$ Ausgabe der Matrix "QulitaetWerte"
    fig4 = figure;
    set(fig4, 'WindowStyle', 'docked');
    set(fig4, 'Name','Qulität Dummie','NumberTitle','off');
    TitelWerte=strjoin({'Qulität Werte: ', JahrName,': ',DatenName});
    imagesc(QulitaetWerte);
```

```
    set(gca,'ydir','normal')
    title(TitelWerte)
    ax = gca;
    ax.YTick = [60 120 180 240 300 360 420 480 540 600 660 720 780 840 900
960 1020 1080 1140 1200 1260 1320 1380 1440];
    ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24};
    xlabel('Tage', 'FontSize', 16)
    ylabel('Stunden', 'FontSize', 16)
    colormap jet;
    h = colorbar();
    ylabel(h,'Werte','FontSize', 16)
    setFigDockGroup(DatenName);
    %/ Ausgabe der Matrix "QulitaetWerte"

    %$ Ausabe des Text-Arrays "Historie"
    fig5 = figure;
    set(fig5, 'WindowStyle', 'docked');
    set(fig5, 'Name','Historie Daten','NumberTitle','off');
    uicontrol('Style', 'text',... %Aufruf eines Text-GUIs
   'String', Historie,... %Übergabe des Arrays Historie an GUI-Element
   'Units','normalized',...
   'Position', [0.3 0.5 0.4 0.4]); %Koordinaten zur Pistion für den
    setFigDockGroup(DatenName);
    %/ Ausabe des Text-Arrays "Historie"
```

A.4.2 Datenanalyse.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% Aus der Funktion "Daten_Plot" werden von der erzeugten Heatmap 2
% Koordinaten übergebenen die mit der MATLAB-Funktion -> DataCurser
abgegriffen wurden. Damit wird aus der Matrix
% "IntpolWerte" eine Auswertung der eingeschlossenen / ausgeschlossnen
% Koordinaten Matrix Werten gestartet
%!!!! Die Daten werden mit dem Dezimaltrennzeichen . anstatt , geschrieben.
%!!!! Dies muss bei dem Export in Excel berücksichtigt werden

function Datenanalyse(src, event, DatenName, JahrName )
clc
h = gcf; % current figure handle
uiwait(helpdlg('Bitte 2 Punkte wählen und jeweils nacheinander mit Enter
bestätigen','Daten Auswertung'));

datacursormode on;

%dcm_obj = datacursormode(h)

    %$ Hole die 2 Punkte aus der Heatmap und bestätige mit "Enter"
        pause %Warte aus ENTER
        c_info = getCursorInfo(datacursormode(h));
        Position=getfield(c_info, 'Position');
        pause %Warte aus ENTER
        c_info2 = getCursorInfo(datacursormode(h));
        UrsprungDaten=get(c_info2.Target, 'CData'); % Hole Matrix
IntploWerte
```

```
        Position2=getfield(c_info2, 'Position');
    %/ Hole die 2 Punkte aus der Heatmap


    %§ Umrechnen der Koordinaten
        StartTag1=Position(1,1);
        StartTag2=Position2(1,1);
        StartMinute1=Position(1,2);
        StartMinute2=Position2(1,2);
        Jahr =str2double(JahrName); % Wandle Text zu Zahl
        %§§§§§§§§§§§§§§§§   %FUNCTION  DatumUmrechnen
        Datum1 = DatumUmrechnen(StartTag1, Jahr);
        Datum2 = DatumUmrechnen(StartTag2, Jahr);
        %§§§§§§§§§§§§§§§§   %FUNCTION  UhrzeitUmrechnen
        Uhrzeit1 = UhrzeitUmrechnen(StartMinute1);
        Uhrzeit2 = UhrzeitUmrechnen(StartMinute2);
    %/ Umrechnen der Koordinaten


        %§ Berechnung was zwischen den beiden Punkten liegt Index_[]
            %Definition des Ausschnits der Matrix
            DetltaTag=StartTag2-StartTag1+1;
            DeltaMinute=StartMinute2-StartMinute1+1;
            TagMinute=DetltaTag*DeltaMinute;
            %Deklaration ein 1 Dimensionles Arrays mit "0"en. Länge
entspricht den eingeschlossenen Werten
            AusschnittWerte=zeros(TagMinute,1);

            %Schreibe Werte aus Ausschnitt in Array
            Zaehl=1;
            for Tag=StartTag1:1:StartTag2
            for Minute=StartMinute1:1:StartMinute2
                AusschnittWerte(Zaehl,1)=UrsprungDaten(Minute,Tag);
                Zaehl=Zaehl+1;
            end
            end
            % Mathmatische Auswertung des beschriebenen Arrays
            Mittelwert=mean(AusschnittWerte);   % Mittelwert aus allen
Elementen des Arrays
            Mittelwert_Stunde=Mittelwert/60;    % Mittelwert geteilt durch
60 (1/h)
            SummeWert=sum(AusschnittWerte);     % Summe positiv und negativ
            SummeWert_Stunde=SummeWert/60;      % Summe (1/h) positiv und
negativ  positiv

            SummeWert_pos=sum(AusschnittWerte(AusschnittWerte>0));     %
Summe nur positiv
            SummeWert_Stunde_pos=SummeWert_pos/60;      % Summe (1/h) nur
porsitiv

            SummeWert_neg=sum(AusschnittWerte(AusschnittWerte<0));     %
Summe nur negativ
            SummeWert_Stunde_neg=SummeWert_neg/60;      % Summe (1/h) nur
negativ

            Minuten=TagMinute;                  % Anzahl der Minuten
            Stunden=Minuten/60;                 % Minuten Umrechnung in
Studen
            Uhrzeit_Stunde1=StartMinute1/60;    % Koordinate 1 umgerehcnet
in Uhrzeit
```

```
            Uhrzeit_Stunde2=StartMinute2/60;    % Koordinate 2 umgerehcnet
in Uhrzeit
        %/ Berechnung was zwischen den beiden Punkten liegt


    %§ Berechnung was Außerhalb der beiden Punkten liegt Index_[1]
        AusschnittWerte_Anzahl=numel(AusschnittWerte);  % Anzahl der
Elemente des Arrays aus vorherigen § Abschnitt
        UrsprungDaten_Anzahl=numel(UrsprungDaten);      % Anzahl der
Elemente Matrix "IntpolWerte" []
        Summe_UrsprungDaten=sum(sum(UrsprungDaten));    % Summe der Matrix
"IntpolWerte"
        Summe_AusschnittWerte=sum(AusschnittWerte);     % Summe des Arrays
aus vorherigen § Abschnitt
        Mittelwert_1=  (Summe_UrsprungDaten - Summe_AusschnittWerte)/
(UrsprungDaten_Anzahl-AusschnittWerte_Anzahl); % Mittelwert
        SummeWert_1=Summe_UrsprungDaten-Summe_AusschnittWerte;  % Summe der
Werte außerhalb
        SummeWert_Stunde_1=SummeWert_1/60;                      % Summe der
Werte außerhalb (1/h)
    %/ Berechnung was Außerhalb der beiden Punkten liegt Index_[1]


        %§ Berechnung was unterhalb der Punkten liegt Index_[2]
            %Definition des Ausschnits der Matrix[2]
            DetltaTag_2=StartTag2-StartTag1+1;
            DeltaMinute_2=StartMinute1;
            TagMinute_2=DetltaTag_2*DeltaMinute_2;
            AusschnittWerte_2=zeros(TagMinute_2,1);

            %Schreibe Werte aus Ausschnitt in Array
            Zaehl=1;
            for Tag_2=StartTag1:1:StartTag2
            for Minute_2=1:1:StartMinute1
                AusschnittWerte_2(Zaehl,1)=UrsprungDaten(Minute_2,Tag_2);
                Zaehl=Zaehl+1;
            end
            end

            % Mathmatische Auswertung des beschriebenen Arrays
            Mittelwert_2=mean(AusschnittWerte_2);
            Mittelwert_Stunde_2=Mittelwert_2/60;
            SummeWert_2=sum(AusschnittWerte_2);
            SummeWert_Stunde_2=SummeWert_2/60;
            Minuten_2=TagMinute_2;
            Stunden_2=Minuten_2/60;
            Uhrzeit_Stunde1_2=0;
            Uhrzeit_Stunde2_2=StartMinute1/60;
        %/ Berechnung was unterhalb der Punkten liegt Index_[2]

    %§ Berechnung was überhalb der Punkten liegt Index_[3]
        %Definition des Ausschnits der Matrix[3]
        DetltaTag_3=StartTag2-StartTag1+1;
        DeltaMinute_3=StartMinute1;
        TagMinute_3=DetltaTag_3*DeltaMinute_3;
        AusschnittWerte_3=zeros(TagMinute_3,1);

        %Schreibe Werte aus Ausschnitt in Array
        Zaehl=1;
        for Tag_3=StartTag1:1:StartTag2
```

```
        for Minute_3=1:1:StartMinute1
            AusschnittWerte_3(Zaehl,1)=UrsprungDaten(Minute_3,Tag_3);
            Zaehl=Zaehl+1;
        end
        end

        % Mathmatische Auswertung des beschriebenen Arrays
        Mittelwert_3=mean(AusschnittWerte_3);
        Mittelwert_Stunde_3=Mittelwert_3/60;
        SummeWert_3=sum(AusschnittWerte_3);
        SummeWert_Stunde_3=SummeWert_3/60;
        Minuten_3=TagMinute_3;
        Stunden_3=Minuten_3/60;
        Uhrzeit_Stunde1_3=StartMinute1/60;
        Uhrzeit_Stunde2_3=24;
    %/ Berechnung was überhalb der Punkten liegt Index_[3]

        %§ Berechnung gesamt innerhalb der Tagesgrenzen Index_[4]
            %Definition des Ausschnits der Matrix[4]
            DetltaTag_4=StartTag2-StartTag1+1;
            TagMinute_4=DetltaTag_4*1440;
            AusschnittWerte_4=zeros(TagMinute_4,1);

            %Schreibe Werte aus Ausschnitt in Array
            Zaehl=1;
            for Tag_4=StartTag1:1:StartTag2
            for Minute_4=1:1:1440
                AusschnittWerte_4(Zaehl,1)=UrsprungDaten(Minute_4,Tag_4);
                Zaehl=Zaehl+1;
            end
            end

            % Mathmatische Auswertung des beschriebenen Arrays
            Mittelwert_4=mean(AusschnittWerte_4);
            Mittelwert_Stunde_4=Mittelwert_4/60;
            SummeWert_4=sum(AusschnittWerte_4);
            SummeWert_Stunde_4=SummeWert_4/60;
            Minuten_4=TagMinute_4;
            Stunden_4=Minuten_4/60;
            Uhrzeit_Stunde1_4=0;
            Uhrzeit_Stunde2_4=24;
        %/ Berechnung gesamt innerhalb der Tagesgrenzen Index_[4]


%$ Erzeugen des Dialogs mit Tabelle
    d = dialog('Position',[300 300 450 500],'Name','Ergebnisse', 'Resize' ,
'On');

    txt = uicontrol('Parent',d,...
                'Style','text',...
                'Units', 'normalized',...
                'Position',[.25 .48 .5 .5],...
                'String','Wichtig !! Dezimal- und Tausendertrennzeichen in
Excel vorher umstellen, wenn Daten in Excel kopiert werden sollen !!');

set(txt,'Foregroundcolor','r','FontSize',10,'Fontname','Helvetica','Fontwei
ght','bold');

    t = uitable(d,'Units', 'normalized','Position',[.15 .02 .7
.8],'ColumnWidth',{150 100});
```

```
%/ Erzeugen des Dialogs mit Tabelle

    %$ Schrieben der Tabellenstruktur und der Berechneten Daten in ein
Array
        dtest = {       '<html><font color="red"><b>Daten Header', ' ';
...

                        'DatenName', DatenName;
'Jahr',JahrName;...
                        'Datum1', Datum1;
'Datum2', Datum2;...
                        'Uhrzeit1', Uhrzeit1 ;
'Uhrzeit2', Uhrzeit2; ...

                '','';'<html><font color="green"><b>!!Innerhalb Daten', '
';           ...

                        'Minuten',Minuten;
'Stunden',Stunden;...
                        'StartMinute1', StartMinute1 ;
'StartMinute2' , StartMinute2 ;...
                        'StartTag1' , StartTag1 ;
'StartTag2' , StartTag2; ...
                        'Uhrzeit_Stunde1', Uhrzeit_Stunde1;
'Uhrzeit_Stunde2', Uhrzeit_Stunde2;...
                        'Mittelwert', Mittelwert ;
'Mittelwert_Stunde', Mittelwert_Stunde; ...
                        'SummeWert', SummeWert ;
'SummeWert_Stunde' , SummeWert_Stunde;...
                '','';'SummeWert_positiv', SummeWert_pos ;
'SummeWert_Stunde_positiv' , SummeWert_Stunde_pos;...
                '','';'SummeWert_negativ', SummeWert_neg ;
'SummeWert_Stunde__negativ' , SummeWert_Stunde_neg;...



                '','';'<html><font color="green"><b>!!Außerhalb Daten', '
';     ...
                        'Mittelwert_1', Mittelwert_1 ;
'SummeWert_1' , SummeWert_1;...
                        'SummeWert_Stunde_1', SummeWert_Stunde_1 ; ...

                '','';'<html><font color="green"><b>!!Unterhalb Daten', '
';               ...
                        'Minuten_2',Minuten_2;
'Stunden_2',Stunden_2;...
                        'StartMinute1_2', 1 ;
'StartMinute2_2' , StartMinute1 ;...
                        'StartTag1_2' , StartTag1 ;
'StartTag2_2' , StartTag2; ...
                        'Uhrzeit_Stunde1_2', Uhrzeit_Stunde1_2;
'Uhrzeit_Stunde2_2', Uhrzeit_Stunde2_2;...
                        'Mittelwert_2', Mittelwert_2 ;
'Mittelwert_Stunde_2', Mittelwert_Stunde_2; ...
                        'SummeWert_2', SummeWert_2 ;
'SummeWert_Stunde_2' , SummeWert_Stunde_2;...

                '','';'<html><font color="green"><b>!!Überhalb Daten', ' ';
...
```

```
                        'Minuten_3',Minuten_3;
'Stunden_3',Stunden_3;...
                        'StartMinute1_3', StartMinute2 ;
'StartMinute2_3' , 1440 ;...
                        'StartTag1_3' , StartTag1 ;
'StartTag2_3' , StartTag2; ...
                        'Uhrzeit_Stunde1_3', Uhrzeit_Stunde1_3;
'Uhrzeit_Stunde2_3', Uhrzeit_Stunde2_3;...
                        'Mittelwert_3', Mittelwert_3 ;
'Mittelwert_Stunde_3', Mittelwert_Stunde_3; ...
                        'SummeWert_3', SummeWert_3 ;
'SummeWert_Stunde_3' , SummeWert_Stunde_3;...

                '','';'<html><font color="green"><b>!!Tagesgrenze Daten', '
';           ...
                        'Minuten_4',Minuten_4;
'Stunden_4',Stunden_4;...
                        'StartMinute1_4', 1 ;
'StartMinute2_4' , 1440 ;...
                        'StartTag1_4' , StartTag1 ;
'StartTag2_4' , StartTag2; ...
                        'Uhrzeit_Stunde1_4', Uhrzeit_Stunde1_4;
'Uhrzeit_Stunde2_4', Uhrzeit_Stunde2_4;...
                        'Mittelwert_4', Mittelwert_4 ;
'Mittelwert_Stunde_4', Mittelwert_Stunde_4; ...
                        'SummeWert_4', SummeWert_4 ;
'SummeWert_Stunde_4' , SummeWert_Stunde_4;...



                };
    %/ Schrieben der Tabellenstruktur und der Berechneten Daten in ein
Array

t.Data = dtest; %übergabe des Arrays an den Tabellen Handle

Data=get(t,'Data');%
btn = uicontrol('Parent',d,...
           'Position',[10 20 70 25],...
           'String','Speichern',...
           'Callback',{@excelspeichern, Data});



end

    function excelspeichern(src, event,Data)
%$ Daten wenn Benzuter es will, in eine Excel-Datei schreiben

    [filename, pathname] = uiputfile('*.xls', 'Bitte geben Sie ein
Dateinamen ein \n oder klicken sie auf Abbruch');
        if isequal(pathname, 0)
                %disp('Benutzer wählte ''Cancel''');
                return % Abbruch
            else
                % Hier werden keine Anweisunge ausgeführt, dient nur als
Rückgabe
                % während der Erstellung des Programmcodes
                %disp(['Pfadname ', PathName]);
                %DateiPfad =PathName;
        end
    excelName = fullfile(pathname, filename);
```

```
    xlswrite(excelName,Data);
%/ Daten wenn Benzuter es will, in eine Excel-Datei schreiben

%}

end
```

A.4.3 DatenBerechnen.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% Hier werden die aus der Excel-Tabelle gespeicherten Werte der MATLAB
Structure ausgelsen
% zu den Matrizen verarbeitet die für die Heatmaps notwendig sind berechnet
und
% gespeichert.

function DatenBerechnen()

global Daten          % kommt aus HeatmapToolbox_DatenAuswahl (ausgelesenen
Excel Daten)
global ExcelDatenInfo % kommt aus HeatmapToolbox_DatenAuswahl (Metadaten)
global AuswahlDaten   % kommt aus HeatmapToolbox_DatenAuswahl (Index welche
Datenreihen ausgewählt wuden)


%$ GUI SpeicherOrt der Daten um sie später wieder zu verweden
    PathName = uigetdir('','Auswahl Ordner zum speichern');
    if isequal(PathName, 0)
        %disp('Benutzer wählte ''Cancel''');
        return % Abbruch
    else
        % Hier werden keine Anweisunge ausgeführt, dient nur als Rückgabe
        % während der Erstellung des Programmcodes
        %disp(['Pfadname ', PathName]);
        %DateiPfad =PathName;
    end
%/ GUI SpeicherOrt der Daten um sie später wieder zu verweden

% Hole Anzahl Datenreihen
ZlSp_AuswahlDaten = size(AuswahlDaten);
AnzahlBerechnung = ZlSp_AuswahlDaten(1,2);

%$ Schleife je nachdem wie viele Datenreihen berechnet werden sollen
for StartAnzahlBer=1:1:AnzahlBerechnung

    IndexDatenInfo=AuswahlDaten(1,StartAnzahlBer);            % Hole Index
der Datenreihe die ausgewählt wurde
    DatenName= char(ExcelDatenInfo(2,IndexDatenInfo));        % Hole Name
der Datenreihe aus Metadaten

    % alle nicht erkaubten Zeichen entfernen, für das spätere Speichern der
Daten
```

```
    DatenNameSpeicher = regexprep(DatenName,' ','_');
    DatenNameSpeicher = regexprep(DatenNameSpeicher,'\','_');
    DatenNameSpeicher = regexprep(DatenNameSpeicher,'[.|%/<>?:+"*]','_');

    Jahr=ExcelDatenInfo{4,IndexDatenInfo};  % Hole Jahr als Zahl aus
Metadaten
    JahrChar=num2str(Jahr);                 % Wandle Jahr zu Zeichen
    ZeilenAnzahlInfo=cell2mat( ExcelDatenInfo(5,IndexDatenInfo)); % Hole
Zeienanzahl aus Metadaten
    SpaltenInfo=cell2mat( ExcelDatenInfo(3,IndexDatenInfo));      % Hole
Index der Spalte aus Metadaten

    % Berechne Anzahl der Tage im Jahr
    TageImMonat=eomday(Jahr, 1:12);
    AnzTage=sum(TageImMonat,2);

    % Dekalration der Matrizen als Zero-Matrizen (enthält nur Nullen)
    Rohdaten01 = zeros(1440, AnzTage);
    RohdatenWerte = zeros(1440, AnzTage);
    IntpolWerte = zeros(1440, AnzTage);
    QulitaetWerte = zeros(1440, AnzTage);

    %$ Waitbar Definition des Fortschrittbalkens
    WaitbarSteps=num2str(ZeilenAnzahlInfo);
    WaitbarText=strjoin({'Bitte Warten es müssen: ',
WaitbarSteps,'Datenpunkte ausgelsen und Interpoliert werden '});
    wait = waitbar(0,WaitbarText);
    steps = ZeilenAnzahlInfo;
    % /Waitbar Definition des Fortschrittbalkens

        %$ Schleife je nachdem wie viele Datensätze (Zeilen) eine
Datenreihe enthält
        for z=2:1:ZeilenAnzahlInfo

    %$ Waitbar Ausgabe des Fortschrittbalkens
    step = z;
    waitbar(step / steps)
    %/ Waitbar Ausgabe des Fortschrittbalkens

                %$ Hole Datum und Uhrzeit
                try
                Datum = datetime(Daten(z,SpaltenInfo),
'InputFormat','dd.MM.yyyy HH:mm:ss');
                Monat= month(Datum);
                Tag = day(Datum);
                Stunde = hour(Datum);
                Minute = minute(Datum);
                %Sekunde= second(Datum);
                catch       % Abfangen wenn eine Zeile keine Uhrzeit
enthält z.b. 00:00
                Datum = datetime(Daten(z,SpaltenInfo),
'InputFormat','dd.MM.yyyy');
                Monat= month(Datum);
                Tag = day(Datum);
                Stunde = 0;
                Minute = 1;
                end
                %/ Hole Datum und Uhrzeit

                %$ Berechnung Monat und Tag in gesamt Tag vom Jahr
```

```
                if Monat==1
                    TageMonatRechnung=0;
                else
                    % eomday() gibt Anzahl Tage der Monate zurück
                    MonateTage1_12=eomday(Jahr, 1:Monat-1);  %  -1 weil
Tage vomVormonat  gebraucht werden
                    TageMonatRechnung=sum(MonateTage1_12,2); % Summiere
Tage von Monaten
                end
                TagImJahr=TageMonatRechnung+Tag; % Summe Tage Monate + Tag
im Monat
                %/ Berechnung Monat und Tag in gesamt Tag vom Jahr

                % Berechne TagesMinute
                TagesMinute= Stunde*60 + Minute;
                if TagesMinute==0
                    TagesMinute=1;
                end

                % Hole den Wert aus Daten
                DatenWert=cell2mat(Daten(z,SpaltenInfo+1));

                % Wenn nötig Umwandlung des Dezimaltrennzeichen , in ein .
                if ischar(DatenWert)==1
                    regexprep(DatenWert,'[,]','.')
                    DatenWert=str2double(DatenWert);
                end

                % Schriebe 1 in Rohdaten01 für den Tag und Minute
                Rohdaten01(TagesMinute, TagImJahr)=1;
                % Schriebe Wert in RohdatenWerte für den Tag und Minute
                RohdatenWerte(TagesMinute, TagImJahr)= DatenWert;
                % Schriebe Wert in InterpolWerte
                IntpolWerte(TagesMinute, TagImJahr)= DatenWert;

                % Ist es die Letzte Zeile, wenn nicht weiter.
                if z~=ZeilenAnzahlInfo
                %$ Anweisung sind für die Interpolation notwendig


                    %$ Hole nächstes Datum und Uhrzeit
                    try
                    DatumNext = datetime(Daten(z+1,SpaltenInfo),
'InputFormat','dd.MM.yyyy HH:mm:ss');
                    MonatNext= month(DatumNext);
                    TagNext = day(DatumNext);
                    StundeNext = hour(DatumNext);
                    MinuteNext = minute(DatumNext);
                    catch
                    DatumNext = datetime(Daten(z+1,SpaltenInfo),
'InputFormat','dd.MM.yyyy');
                    MonatNext= month(DatumNext);
                    TagNext = day(DatumNext);
                    StundeNext = 1;
                    MinuteNext = 0;
                    end
                    %/ Hole nächstes Datum und Uhrzeit

                    %  Berechne nächste TagesMinute "TagesMinuteNext"
                    TagesMinuteNext= StundeNext*60 + MinuteNext;
```

```
                    if TagesMinuteNext==0
                        TagesMinuteNext=1;
                    end

                    %  Rechne Monat und Tag in gesamt Tag von Jahr, für
nächsten Datenpunkt um. "TagImJahrNext"
                    if MonatNext==1
                        TageMonatRechnungNext=0;
                        else
                        MonateTage1_12=eomday(Jahr, 1:MonatNext-1); %-1
weil Vormonat Tage gebraucht werden
                        TageMonatRechnungNext=sum(MonateTage1_12,2);
                    end
                    TagImJahrNext=TageMonatRechnungNext+TagNext;

                    % Hole WertNext aus Daten
                    DatenWertNext=cell2mat(Daten(z+1,SpaltenInfo+1));

                    % Wenn nötig Umwandlung des Dezimaltrennzeichen , in
ein .
                    if ischar(DatenWertNext)==1
                        regexprep(DatenWertNext,'[,]','.')
                        DatenWertNext=str2double(DatenWertNext);
                    end


                    %$ Interpolation wenn mehr als 1 Minute vergangen ist
                    if TagesMinute +1  < TagesMinuteNext &&
TagImJahr==TagImJahrNext

                        %Berechnung der Steigung zwischen den Punkten
                        DeltaX= TagesMinuteNext - TagesMinute;
                        DeltaY = DatenWertNext - DatenWert;
                        m = DeltaY/DeltaX;

                        %IntpolWerte(TagesMinuteNext, TagImJahrNext) =
DatenWertNext;
                        %QulitaetWerte(TagesMinute, TagImJahr) = abs(m);

                        %Schreiben der Werte zwischen den Punkten
                        for i=TagesMinute+1:1:TagesMinuteNext-1
                            IntpolWerte(i, TagImJahr) =  DatenWert + m;
                            %QulitaetWerte(i, TagImJahr) = abs(m);
                            DatenWert = DatenWert + m;
                        end
                    end
                    %/ Interpolation wenn mehr als 1 Minute vergangen ist


                    %$ Interpolation über Tagesgrenzen hinweg
                    if TagImJahr+1==TagImJahrNext % Sind die Datenpunkt nur
ein Tag auseinander

                        % Berechnung der Steigung zwischen den Datenpunkten
                        DeltaX= (1440 - TagesMinute)+ TagesMinuteNext;
                        DeltaY = DatenWertNext- DatenWert;
                        m = DeltaY/DeltaX;
                        WertX = DatenWert;
```

```
                        %Schreiben der Werte zwischen den Punkten
                        for i=TagesMinute+1:1:1440 % Schriebe die Daten für
aktuellen Tag
                            IntpolWerte(i, TagImJahr ) =  WertX + m;
                            WertX = WertX + m;
                        end

                        %Schreiben der Werte zwischen den Punkten
                        for i=1:1:TagesMinuteNext % Schriebe die Daten für
nächsten Tag
                            IntpolWerte(i, TagImJahrNext ) =  WertX;
                            WertX = WertX + m;
                        end

                    end
                    %/ Interpolation über Tagesgrenzen
                end
                %/ Anweisung sind für die Interpolation notwendig
        end
        %/ Schleife je nachdem wie viele Datensätze (Zeilen) eine
Datenreihe enthält

        % Anlegen der Historie
        Historie={DatenName; '----'};

        % Speichern der berechneten Variablen um sie später wieder zu
verweden

DatenNameSpeicherMat=strjoin({JahrChar,'_',DatenNameSpeicher,'.mat'}); %
Generierung des File-Namen
        DatenNameSpeicherMat = regexprep(DatenNameSpeicherMat,' ',''); %
entfernen aller Leerstellen
        DateiPfad = fullfile(PathName, DatenNameSpeicherMat);
        save(DateiPfad, 'DatenName','Jahr','Rohdaten01',
'RohdatenWerte','IntpolWerte', 'QulitaetWerte', 'Historie'  )

        Daten_Plot(DatenName, JahrChar, Rohdaten01, QulitaetWerte,
RohdatenWerte,IntpolWerte, Historie)
        close(wait) %Schließen des Fortschrittbalkens (Waitbar)
end
%$ Schleife je nachdem wie viele Datenreihen berechnet werden sollen
```

A.4.4 DatumAnzeige.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% Aus der Funktion "Daten_Plot" wird von der erzeugten Heatmap eine
% Koordinate übergebenen die mit der MATLAB-Funktion -> DataCurser
abgegriffen wurde.
% Die Koordinate wird in ein Datum und Uhrzeit umgrechnet

function DatumAnzeige(src, event, JahrName)

try % (Versuche, sonst catch)
%§ Hole CursorInfo
h = gcf; %Handle der aktuellen figure (current figure handle)
c_info = getCursorInfo(datacursormode(h));
```

```
Position=getfield(c_info, 'Position');
%/ Hole CursorInfo

%§ Speichern/Umrechnen CursorInfo
StartTag1=Position(1,1);
StartMinute1=Position(1,2);
Jahr =str2double(JahrName);
Datum1 = DatumUmrechnen(StartTag1, Jahr);
Uhrzeit1 = UhrzeitUmrechnen(StartMinute1);
%/ Speichern/Umrechnen CursorInfo

ausgabe_txt = {[ Datum1 ], [Uhrzeit1] }; % Schreibe Text

msgbox(ausgabe_txt); % Ausgabe Text
catch
    uiwait(msgbox('DatenCurser bitte aktivieren'));
    datacursormode on;
end
```

A.4.5 DatumUmrechnen.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% Die Funktion rechtner den Tag 1-365(366) anhand des Jahres in ein Datum
um
% und gibt den String Datum zurück
% Wird momentan nur genutzt von Datenanalyse

function Datum = DatumUmrechnen(Tag ,Jahr)
%function Datum: Umrechnung Minute Tag in Datum

GesamtTage=eomday(Jahr , 1:12); % Gebe Anzahl alle Tage der Moante zurück

%Schleife zur Umrechnung
TagePruef=0;
for i=1:1:12
    TagePruef=TagePruef+GesamtTage(1,i);
    if TagePruef>=Tag
        difTag=TagePruef-Tag;
        aktuelTag=GesamtTage(1,i)-difTag;
        Datum = regexprep(strjoin({num2str(aktuelTag) '.' num2str(i) '.'
num2str(Jahr)}),' ','');
        break
    end
end
end
```

A.4.6 HeatmapToolbox.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
    % $Version: 0.1  $Date: 2017/01/20
```

```
    % Grafische Startobefläche des MATLAB Programms "Heatmap Toolbox"
    % siehe Abbildung XX im Kapitel XX


    %MATLAB erstellte Funktion die anderen Funktionen in diesem .m File
    %deklariert und steuert
function varargout = HeatmapToolbox(varargin)
% HEATMAPTOOLBOX MATLAB code for HeatmapToolbox.fig
%      HEATMAPTOOLBOX, by itself, creates a new HEATMAPTOOLBOX or raises
the existing
%      singleton*.
%
%      H = HEATMAPTOOLBOX returns the handle to a new HEATMAPTOOLBOX or the
handle to
%      the existing singleton*.
%
%      HEATMAPTOOLBOX('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in HEATMAPTOOLBOX.M with the given input
arguments.
%
%      HEATMAPTOOLBOX('Property','Value',...) creates a new HEATMAPTOOLBOX
or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before HeatmapToolbox_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to HeatmapToolbox_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help HeatmapToolbox

% Last Modified by GUIDE v2.5 29-Jan-2017 20:13:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @HeatmapToolbox_OpeningFcn, ...
'gui_OutputFcn',  @HeatmapToolbox_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

    % Anweisung die vor der Erstellung des GUI ausgwführt werden sollen
```

```
% --- Executes just before HeatmapToolbox is made visible.
function HeatmapToolbox_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to HeatmapToolbox (see VARARGIN)

% Choose default command line output for HeatmapToolbox
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

    %Einlesen und Ausgabe der Beispielgrafik
    clc
    axes(handles.axes1);
    G1=imread('Cover.jpg','jpg');
    image (G1); axis image;
    axis off

% UIWAIT makes HeatmapToolbox wait for user response (see UIRESUME)
% uiwait(handles.figure1);

    %$ Ausgabe an die Kommandline in MATLAB, wird nicht genutzt
% --- Outputs from this function are returned to the command line.
function varargout = HeatmapToolbox_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
    %/ Ausgabe an die Kommandline in MATLAB, wird nicht genutzt

% --- Executes on button press in DatenAuswahl.
function DatenAuswahl_Callback(hObject, eventdata, handles)
% hObject    handle to DatenAuswahl (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    % Funktionaufruf um neue Rohdaten auszulesen
    %§§§§§§§§§§§§§§§§   %FUNCTION  HeatmapToolbox_DatenAuswahl
    HeatmapToolbox_DatenAuswahl()

% --- Executes on button press in VorhandeneDaten.
function VorhandeneDaten_Callback(hObject, eventdata, handles)
% hObject    handle to VorhandeneDaten (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %$ GUI Dateiauswahl für schon existierende Heatmap-Daten
    [filename, pathname] = uigetfile('*.mat', '.mat File Wählen: Lade
HeatMap Daten');
    if isequal(filename, 0)
        %disp('Benutzer Auswahl ''Cancel''');
        return % Abbruch
        else
        %disp(['Dateipfad ', fullfile(pathname, filename)]);
```

```
        DateiPfad = fullfile(pathname, filename);
    end
    load(DateiPfad);
    %/ GUI Dateiauswahl für schon existierende Heatmap-Daten

    JahrName=num2str(Jahr); %Wandle Zahl in Text

    %§§§§§§§§§§§§§§§§   %FUNCTION  Daten_Plot
    Daten_Plot(DatenName, JahrName, Rohdaten01, QulitaetWerte,
RohdatenWerte,IntpolWerte, Historie)

% --- Executes on button press in VorhDatenBear.
function VorhDatenBear_Callback(hObject, eventdata, handles)
% hObject    handle to VorhDatenBear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes during object creation, after setting all properties.

    %Funktionsaufruf um schon vorhandene Heatmap Daten zu bearbeiten
    %§§§§§§§§§§§§§§§§   %FUNCTION  ManipulationDaten
    ManipulationDaten();


% --- Executes on button press in Manipul2Heatm.
function Manipul2Heatm_Callback(hObject, eventdata, handles)
% hObject    handle to Manipul2Heatm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Funktionsaufruf um 2 vorhandene Heatmap Daten miteinander zu
manipulieren
    %§§§§§§§§§§§§§§§§   %FUNCTION  Manipulation2Heatmaps
    Manipulation2Heatmaps()


% --- Executes on button press in Colormap.
function Colormap_Callback(hObject, eventdata, handles)
% hObject    handle to Colormap (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    colormapeditor; % Öffnen des MATLAB Colomap-Editors für die Farbskala-
Einstellung der Heatmaps


% --- Executes on button press in Logik.
function Logik_Callback(hObject, eventdata, handles)
% hObject    handle to Logik (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Funktionsaufruf um mehrere Heatmap Daten mit einer Logik zu
untersuchen
    %§§§§§§§§§§§§§§§§   %FUNCTION  LogikHeatmaps
    LogikHeatmaps();
```

A.4.7 HeatmapToolbox_DatenAuswahl.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% Generierung des GUIs zur Rohdaten Auswahl aus einem Excel-File

    %MATLAB erstellte Funktion die anderen Funktionen in diesem .m File
    %deklariert und steuert
function varargout = HeatmapToolbox_DatenAuswahl(varargin)
% HEATMAPTOOLBOX_DATENAUSWAHL MATLAB code for
HeatmapToolbox_DatenAuswahl.fig
%      HEATMAPTOOLBOX_DATENAUSWAHL, by itself, creates a new
HEATMAPTOOLBOX_DATENAUSWAHL or raises the existing
%      singleton*.
%
%      H = HEATMAPTOOLBOX_DATENAUSWAHL returns the handle to a new
HEATMAPTOOLBOX_DATENAUSWAHL or the handle to
%      the existing singleton*.
%
%
HEATMAPTOOLBOX_DATENAUSWAHL('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in HEATMAPTOOLBOX_DATENAUSWAHL.M with the
given input arguments.
%      HEATMAPTOOLBOX_DATENAUSWAHL('Property','Value',...) creates a new
HEATMAPTOOLBOX_DATENAUSWAHL or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before HeatmapToolbox_DatenAuswahl_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
HeatmapToolbox_DatenAuswahl_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
HeatmapToolbox_DatenAuswahl

% Last Modified by GUIDE v2.5 07-Nov-2016 11:58:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @HeatmapToolbox_DatenAuswahl_OpeningFcn, ...
'gui_OutputFcn',  @HeatmapToolbox_DatenAuswahl_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

    %$ Anweisung die vor der Erstellung des GUI ausgeführt werden sollen
% --- Executes just before HeatmapToolbox_DatenAuswahl is made visible.
function HeatmapToolbox_DatenAuswahl_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to HeatmapToolbox_DatenAuswahl (see
VARARGIN)

% Choose default command line output for HeatmapToolbox_DatenAuswahl
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

    % Deklarierung der listbox1 mit leeren Text-Platzhalter und mit
    % multipler Auswahl an Daten
    txt_cell_array = {''};
    set(handles.listbox1, 'String', txt_cell_array , 'Max',2, 'Min',0);
    set(handles.AuswahlDatenOK,'Enable','off')

% UIWAIT makes HeatmapToolbox_DatenAuswahl wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
    %/ Anweisung die vor der Erstellung des GUI ausgwführt werden sollen



    %$ Ausgabe an die Kommandline in MATLAB. Wird nicht genutzt
% --- Outputs from this function are returned to the command line.
function varargout = HeatmapToolbox_DatenAuswahl_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
    %/ Ausgabe an die Kommandline in MATLAB. Wird nicht genutzt

    %$ Rohdaten Auswahl
% --- Executes on button press in DatenAuswahl.
function DatenAuswahl_Callback(hObject, eventdata, handles)
% hObject    handle to DatenAuswahl (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    % Hier werden die ausgelesenen Metadaten von der Funktion
"SelektionDaten" aus der Excel-Datei ausgelsen und an die
    % listbox1 weitergegeben
```

```
    global ExcelDatenInfo;
    global AuswahlDaten;

    %$$$$$$$$$$$$$$$$$   %FUNCTION  SelektionDaten
    ExcelDatenInfo=SelektionDaten();

    DatenSpZl=size(ExcelDatenInfo); % Hole Dimension von Metadaten um
darauß Anzahl der Datenreihen zu bestimmen
    DatenAnzahl= DatenSpZl(1,2);
    %Holt DatenNamen aus Metadaten für die "listbox1"
    for i=1:1:DatenAnzahl
    txt_cell_array(i)=  ExcelDatenInfo(2,i);
    end
    % Cell Array wird an listbox1 übergeben
    set(handles.listbox1, 'String', txt_cell_array , 'Max',2, 'Min',0);

    % Button "AuswahlDatenOK" aktiviert
    AuswahlDaten = 1;
    set(handles.AuswahlDatenOK,'Enable','on')

    %/ Rohdaten Auswahl

    %$ Listbox zum Auswählen der Datenreihen aus Rohdaten
% Enthählt keine Anweisung, wird aber von anderen Funktionen genutzt
% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1

function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
    %/ Listbox zum Auswählen der Datenreihen aus Rohdaten

% --- Executes on button press in AuswahlDatenOK.
function AuswahlDatenOK_Callback(hObject, eventdata, handles)
% hObject    handle to AuswahlDatenOK (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    global AuswahlDaten
    % Die ausgewählten/markierten Datenindizes aus der Listbox werden
gespeichert
    AuswahlDaten=get(handles.listbox1, 'Value');
    %$$$$$$$$$$$$$$$$$   %FUNCTION  DatenBerechnen
    DatenBerechnen()
```

A.4.8 LetzteZeile.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% Generierung des GUIs zur Rohdaten Auswahl aus einem Excel-File

    %MATLAB erstellte Funktion die anderen Funktionen in diesem .m File
    %deklariert und steuert
function varargout = HeatmapToolbox_DatenAuswahl(varargin)
% HEATMAPTOOLBOX_DATENAUSWAHL MATLAB code for
HeatmapToolbox_DatenAuswahl.fig
%      HEATMAPTOOLBOX_DATENAUSWAHL, by itself, creates a new
HEATMAPTOOLBOX_DATENAUSWAHL or raises the existing
%      singleton*.
%
%      H = HEATMAPTOOLBOX_DATENAUSWAHL returns the handle to a new
HEATMAPTOOLBOX_DATENAUSWAHL or the handle to
%      the existing singleton*.
%
%
HEATMAPTOOLBOX_DATENAUSWAHL('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in HEATMAPTOOLBOX_DATENAUSWAHL.M with the
given input arguments.
%      HEATMAPTOOLBOX_DATENAUSWAHL('Property','Value',...) creates a new
HEATMAPTOOLBOX_DATENAUSWAHL or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before HeatmapToolbox_DatenAuswahl_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
HeatmapToolbox_DatenAuswahl_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
HeatmapToolbox_DatenAuswahl

% Last Modified by GUIDE v2.5 07-Nov-2016 11:58:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @HeatmapToolbox_DatenAuswahl_OpeningFcn, ...
'gui_OutputFcn',  @HeatmapToolbox_DatenAuswahl_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

    %$ Anweisung die vor der Erstellung des GUI ausgeführt werden sollen
% --- Executes just before HeatmapToolbox_DatenAuswahl is made visible.
function HeatmapToolbox_DatenAuswahl_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to HeatmapToolbox_DatenAuswahl (see
VARARGIN)

% Choose default command line output for HeatmapToolbox_DatenAuswahl
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

    % Deklarierung der listbox1 mit leeren Text-Platzhalter und mit
    % multipler Auswahl an Daten
    txt_cell_array = {''};
    set(handles.listbox1, 'String', txt_cell_array , 'Max',2, 'Min',0);
    set(handles.AuswahlDatenOK,'Enable','off')

% UIWAIT makes HeatmapToolbox_DatenAuswahl wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
    %/ Anweisung die vor der Erstellung des GUI ausgwführt werden sollen


    %$ Ausgabe an die Kommandline in MATLAB. Wird nicht genutzt
% --- Outputs from this function are returned to the command line.
function varargout = HeatmapToolbox_DatenAuswahl_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
    %/ Ausgabe an die Kommandline in MATLAB. Wird nicht genutzt

    %$ Rohdaten Auswahl
% --- Executes on button press in DatenAuswahl.
function DatenAuswahl_Callback(hObject, eventdata, handles)
% hObject    handle to DatenAuswahl (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    % Hier werden die ausgelesenen Metadaten von der Funktion
"SelektionDaten" aus der Excel-Datei ausgelsen und an die
```

```
    % listbox1 weitergegeben

    global ExcelDatenInfo;
    global AuswahlDaten;

    %$$$$$$$$$$$$$$$$$   %FUNCTION  SelektionDaten
    ExcelDatenInfo=SelektionDaten();

    DatenSpZl=size(ExcelDatenInfo); % Hole Dimension von Metadaten um
darauß Anzahl der Datenreihen zu bestimmen
    DatenAnzahl= DatenSpZl(1,2);
    %Holt DatenNamen aus Metadaten für die "listbox1"
    for i=1:1:DatenAnzahl
    txt_cell_array(i)=  ExcelDatenInfo(2,i);
    end
    % Cell Array wird an listbox1 übergeben
    set(handles.listbox1, 'String', txt_cell_array , 'Max',2, 'Min',0);

    % Button "AuswahlDatenOK" aktiviert
    AuswahlDaten = 1;
    set(handles.AuswahlDatenOK,'Enable','on')

    %/ Rohdaten Auswahl

    %$ Listbox zum Auswählen der Datenreihen aus Rohdaten
% Enthählt keine Anweisung, wird aber von anderen Funktionen genutzt
% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1

function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
    %/ Listbox zum Auswählen der Datenreihen aus Rohdaten

% --- Executes on button press in AuswahlDatenOK.
function AuswahlDatenOK_Callback(hObject, eventdata, handles)
% hObject    handle to AuswahlDatenOK (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    global AuswahlDaten
    % Die ausgewählten/markierten Datenindizes aus der Listbox werden
gespeichert
    AuswahlDaten=get(handles.listbox1, 'Value');
    %$$$$$$$$$$$$$$$$$   %FUNCTION  DatenBerechnen
    DatenBerechnen()
```

A.X

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% Aufruf aus DatenSelektion.m wenn Datenreihe leere Zeilen am Ende der
% Datenreihe enthält und gibt NeuZeilenAnzahl zurück (Kommt vor weil alle
Datenreihen die selbe Dimension haben, da m x n Matrix )
function [NeuZeilenAnzahl] = LetzteZeile(ZeilenAnzahl,n)

global Daten

NeuZeilenAnzahl = round(ZeilenAnzahl/1.2);
ZeileLeer = Daten{NeuZeilenAnzahl,1+n};
PruefeNeu = isnan(ZeileLeer); % Wenn keine Nummer Wahr 1
PruefeNeu01 = PruefeNeu(1,1);

    %$ Kopfschleife so lange keine Zahl vorhanden
    % Teile Zeilen bis auf Zahlen getroffen wird
    while 1
        if PruefeNeu01 ==0
            break % Beende Schleife weil Zahl
        end
    NeuZeilenAnzahl = round(NeuZeilenAnzahl/1.1); % Teile Zeilenanzahl
durch 1.1 und runde auf
    ZeileLeer = Daten{NeuZeilenAnzahl,1+n}  ;
    PruefeNeu = isnan(ZeileLeer);   % Wenn keine Nummer Wahr 1
    PruefeNeu01 = PruefeNeu(1,1);
    end
    %/ Kopfschleife so lange keine Zahl vorhanden

    %$ Kopfschleife so lange Zahl vorhanden
    while 1
        if PruefeNeu01 == 1
            break % Beende Schleife weil keine Zahl
        end
        NeuZeilenAnzahl = NeuZeilenAnzahl +1; % Erhöhe Zeile um 1
        ZeileLeer = Daten{NeuZeilenAnzahl,1+n}  ;
        PruefeNeu = isnan(ZeileLeer);   % Wenn keine Nummer Wahr 1
        PruefeNeu01 = PruefeNeu(1,1);
    end
    %/ Kopfschleife so lange Zahl vorhanden

NeuZeilenAnzahl = NeuZeilenAnzahl -1; %  -1 da letzte Schleife erst
verlassen wird wenn keine Zhal gefunden wurde
```

A.4.9 Liniendiagramme.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20
```

```
%Ist eine Dummy-Funktion die die Daten "IntpolWerte" in ein Liniendiagramm
schreiben soll

function Liniendiagramme(src, event, DatenName, JahrName, IntpolWerte )

% Berechne gesamt Tage im Jahr
Jahr =str2double(JahrName);
TageImMonat=eomday(Jahr, 1:12);
AnzTage=sum(TageImMonat,2);
xAchse=1:1:AnzTage;

%$ Tage bis zum Monatsende
Jan=TageImMonat(1,1);
Feb=Jan + TageImMonat(1,2);
Maerz= Feb + TageImMonat(1,3);
April= Maerz + TageImMonat(1,4);
May= April + TageImMonat(1,5);
Juni= May + TageImMonat(1,6);
Juli= Juni + TageImMonat(1,7);
August=  Juli + TageImMonat(1,8);
Sept= August + TageImMonat(1,9);
Okt = Sept + TageImMonat(1,10);
Nov = Okt + TageImMonat(1,11);
Dez = Nov + TageImMonat(1,12);
%/ Tage bis zum Monatsende

%$ Minuten bis zum Monatsende
Jan2=(TageImMonat(1,1))*1440;
Feb2=(Jan + TageImMonat(1,2))*1440;
Maerz2= (Feb + TageImMonat(1,3))*1440;
April2= (Maerz + TageImMonat(1,4))*1440;
May2= (April + TageImMonat(1,5))*1440;
Juni2= (May + TageImMonat(1,6))*1440;
Juli2= (Juni + TageImMonat(1,7))*1440;
August2=  (Juli + TageImMonat(1,8))*1440;
Sept2= (August + TageImMonat(1,9))*1440;
Okt2 = (Sept + TageImMonat(1,10))*1440;
Nov2 = (Okt + TageImMonat(1,11))*1440;
Dez2 = (Nov + TageImMonat(1,12))*1440;
%/ Minuten bis zum Monatsende

% Mathematische Berechnung der gesamten Werte
TageswerteMittel = mean(IntpolWerte);
size(TageswerteMittel)
TageswerteSumme = sum(IntpolWerte);
fprintf('Weiter \n')
TageswerteSumme_Stunde = TageswerteSumme/60;

% Umschrieben von der Matrix IntpolWerte in 1.Dimensionales Array
ZlSp_IntpolWerte= size(IntpolWerte);
Sp_IntoplWerte = ZlSp_IntpolWerte(1,2); % Tage im Jahr
MinutenJahr = Sp_IntoplWerte *1440;
WertMinute = zeros(1,MinutenJahr);

xAchse2=1:1:MinutenJahr; % Abzissen Schrittweite

%1.Dimensionales Array beschreiben mit Werten
z=1;
for t=1:1:Sp_IntoplWerte
```

```
for i=1:1:1440
    WertMinute(1,z) = IntpolWerte(i,t);% Nur Umsschrieben in
1.Dimensionales Array
    if z==1
        WertMinuteSumme(1,z)=IntpolWerte(i,t);
        %WertStundeSumme(1,z)=IntpolWerte(i,t)/60;
    else
        WertMinuteSumme(1,z)=WertMinuteSumme(1,z-1)+
IntpolWerte(i,t);%Werte aufsummieren
        %WertStundeSumme(1,z)=WertStundeSumme(1,z-1)+
(IntpolWerte(i,t)/60);%Werte aufsummieren
    end
    z=z+1;
end
end
WertStundeSumme=WertMinuteSumme/60;
    %$ Ausgabe Liniendiagramm Tages-Mittelwerte
        fig1 = figure;
        set(fig1, 'WindowStyle', 'docked')
        set(fig1, 'Name','Tageswerte Mittel','NumberTitle','off');
        Titel=strjoin({'Tageswerte: ', JahrName,': ',DatenName});
        plot(xAchse ,TageswerteMittel)
        title(Titel);
        xlabel('Tage', 'FontSize', 16)
        ylabel('Werte', 'FontSize', 16)
        xlim([0 AnzTage])
        ax = gca;
        ax.XTick = [ Jan Feb Maerz April May Juni Juli August Sept Okt Nov
Dez];
        %Jan Feb Maerz April May Juni Juli August Sept Okt Nov Dez
        ax.XTickLabel = {'Jan' 'Feb' 'Mär' 'Apr' 'Mai' 'Jun' 'Jul' 'Aug'
'Sep' 'Okt' 'Nov' 'Dez'};
        DockTitel=strjoin({'Tageswerte_', JahrName,DatenName});
        setFigDockGroup(gcf,DockTitel);  % Andocken der figure an Gruppe
DockTitel
    %$ Ausgabe Liniendiagramm Tages-Mittelwerte

    %$ Ausgabe Liniendiagramm Tages-Summe
        fig2 = figure;
        set(fig2, 'WindowStyle', 'docked')
        set(fig2, 'Name','Tageswerte Summe','NumberTitle','off');
        Titel=strjoin({'Tageswerte Summe: ', JahrName,': ',DatenName});
        plot(xAchse ,TageswerteSumme)
        title(Titel);
        xlabel('Tage', 'FontSize', 16)
        ylabel('Werte', 'FontSize', 16)
        xlim([0 AnzTage])
        ax = gca;
        ax.XTick = [ Jan Feb Maerz April May Juni Juli August Sept Okt Nov
Dez];
        %Jan Feb Maerz April May Juni Juli August Sept Okt Nov Dez
        ax.XTickLabel = {'Jan' 'Feb' 'Mär' 'Apr' 'Mai' 'Jun' 'Jul' 'Aug'
'Sep' 'Okt' 'Nov' 'Dez'};
        DockTitel=strjoin({'Tageswerte_', JahrName,DatenName});
        setFigDockGroup(gcf,DockTitel);  % Andocken der figure an Gruppe
DockTitel
    %/ Ausgabe Liniendiagramm Tages-Summe

    %$ Ausgabe Liniendiagramm Tages-Summe (1/h)
        fig3 = figure;
        set(fig3, 'WindowStyle', 'docked')
```

```
        set(fig3, 'Name','Tageswerte Summe h ','NumberTitle','off');
        Titel=strjoin({'Tageswerte Summe h : ', JahrName,': ',DatenName});
        plot(xAchse ,TageswerteSumme_Stunde)
        title(Titel);
        xlabel('Tage', 'FontSize', 16)
        ylabel('Werte', 'FontSize', 16)
        xlim([0 AnzTage])
        ax = gca;
        ax.XTick = [ Jan Feb Maerz April May Juni Juli August Sept Okt Nov
Dez];
        %Jan Feb Maerz April May Juni Juli August Sept Okt Nov Dez
        ax.XTickLabel = {'Jan' 'Feb' 'Mär' 'Apr' 'Mai' 'Jun' 'Jul' 'Aug'
'Sep' 'Okt' 'Nov' 'Dez'};
        DockTitel=strjoin({'Tageswerte_', JahrName,DatenName});
        setFigDockGroup(gcf,DockTitel);  % Andocken der figure an Gruppe
DockTitel
    %/ Ausgabe Liniendiagramm Tages-Summe (1/h)

    %$ Ausgabe Liniendiagramm Tages-Summe
        fig4 = figure;
        set(fig4, 'WindowStyle', 'docked')
        set(fig4, 'Name','Plot ','NumberTitle','off');
        Titel=strjoin({'Plot : ', JahrName,': ',DatenName});
        plot(xAchse2 ,WertMinute)
        title(Titel);
        %xlabel('Minuten', 'FontSize', 16)
        ylabel('Werte', 'FontSize', 16)
        xlim([0 MinutenJahr])
        ax = gca;
        ax.XTick = [ Jan2 Feb2 Maerz2 April2 May2 Juni2 Juli2 August2 Sept2
Okt2 Nov2 Dez2];
        %Jan Feb Maerz April May Juni Juli August Sept Okt Nov Dez
        ax.XTickLabel = {'Jan' 'Feb' 'Mär' 'Apr' 'Mai' 'Jun' 'Jul' 'Aug'
'Sep' 'Okt' 'Nov' 'Dez'};
        DockTitel=strjoin({'Tageswerte_', JahrName,DatenName});
        setFigDockGroup(gcf,DockTitel);  % Andocken der figure an Gruppe
DockTitel
    %/ Ausgabe Liniendiagramm Tages-Summe (1/h)


    %$ Ausgabe Liniendiagramm WertMinuteSumme
        fig4 = figure;
        set(fig4, 'WindowStyle', 'docked')
        set(fig4, 'Name','WertMinuteSumme ','NumberTitle','off');
        Titel=strjoin({'WertMinuteSumme : ', JahrName,': ',DatenName});
        plot(xAchse2 ,WertMinuteSumme)
        title(Titel);
        %xlabel('Minuten', 'FontSize', 16)
        ylabel('Werte', 'FontSize', 16)
        xlim([0 MinutenJahr])
        ax = gca;
        ax.XTick = [ Jan2 Feb2 Maerz2 April2 May2 Juni2 Juli2 August2 Sept2
Okt2 Nov2 Dez2];
        %Jan Feb Maerz April May Juni Juli August Sept Okt Nov Dez
        ax.XTickLabel = {'Jan' 'Feb' 'Mär' 'Apr' 'Mai' 'Jun' 'Jul' 'Aug'
'Sep' 'Okt' 'Nov' 'Dez'};
        DockTitel=strjoin({'Tageswerte_', JahrName,DatenName});
        setFigDockGroup(gcf,DockTitel);  % Andocken der figure an Gruppe
DockTitel
    %/ Ausgabe Ausgabe Liniendiagramm WertMinuteSumme
```

```
    %$ Ausgabe Liniendiagramm WertMinuteSumme
        fig4 = figure;
        set(fig4, 'WindowStyle', 'docked')
        set(fig4, 'Name','WertStundeSumme ','NumberTitle','off');
        Titel=strjoin({'WertStundeSumme : ', JahrName,': ',DatenName});
        plot(xAchse2 ,WertStundeSumme)
        title(Titel);
        %xlabel('Minuten', 'FontSize', 16)
        ylabel('Werte', 'FontSize', 16)
        xlim([0 MinutenJahr])
        ax = gca;
        ax.XTick = [ Jan2 Feb2 Maerz2 April2 May2 Juni2 Juli2 August2 Sept2
Okt2 Nov2 Dez2];
        %Jan Feb Maerz April May Juni Juli August Sept Okt Nov Dez
        ax.XTickLabel = {'Jan' 'Feb' 'Mär' 'Apr' 'Mai' 'Jun' 'Jul' 'Aug'
'Sep' 'Okt' 'Nov' 'Dez'};
        DockTitel=strjoin({'Tageswerte_', JahrName,DatenName});
        setFigDockGroup(gcf,DockTitel);  % Andocken der figure an Gruppe
DockTitel
    %/ Ausgabe Ausgabe Liniendiagramm WertMinuteSumme

end
```

A.4.10 LogikHeatmaps.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

    % Hier können schon gespeicherte Daten (Heatmaps) mit logischen
Operatoren
    % verknüpft werden und dsa Ergebniss ausgegeben werden.


    %$ MATLAB erstellte Funktion die anderen Funktionen in diesem .m File
    % deklariert und steuert
function varargout = LogikHeatmaps(varargin)
% LOGIKHEATMAPS MATLAB code for LogikHeatmaps.fig
%      LOGIKHEATMAPS, by itself, creates a new LOGIKHEATMAPS or raises the
existing
%      singleton*.
%
%      H = LOGIKHEATMAPS returns the handle to a new LOGIKHEATMAPS or the
handle to
%      the existing singleton*.
%
%      LOGIKHEATMAPS('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in LOGIKHEATMAPS.M with the given input
arguments.
%
%      LOGIKHEATMAPS('Property','Value',...) creates a new LOGIKHEATMAPS or
raises the
```

```
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before LogikHeatmaps_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to LogikHeatmaps_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help LogikHeatmaps

% Last Modified by GUIDE v2.5 29-Jan-2017 03:37:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @LogikHeatmaps_OpeningFcn, ...
'gui_OutputFcn',  @LogikHeatmaps_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
    %/ MATLAB erstellte Funktion die anderen Funktionen in diesem .m File
    % deklariert und steuert


    %$ Anweisung die vor der Erstellung des GUI ausgeführt werden sollen
% --- Executes just before LogikHeatmaps is made visible.
function LogikHeatmaps_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to LogikHeatmaps (see VARARGIN)

% Choose default command line output for LogikHeatmaps
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes LogikHeatmaps wait for user response (see UIRESUME)
% uiwait(handles.figure1);
        global JahrPruef
        JahrPruef = 0;
    %/ Anweisung die vor der Erstellung des GUI ausgeführt werden sollen
```

```
    %$ Ausgabe an die Kommandline in MATLAB. Wird nicht genutzt
% --- Outputs from this function are returned to the command line.
function varargout = LogikHeatmaps_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
    %/ Ausgabe an die Kommandline in MATLAB. Wird nicht genutzt


    %$ Auswahl von gespeichrten Datensätzen
% --- Executes on button press in pushbutton_Auswahl.
function pushbutton_Auswahl_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_Auswahl (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global Heat
    global JahrPruef

    %$ GUI Dateiauswahl
        [filename, pathname] = uigetfile('*.mat', '.mat File Wählen: Lade
HeatMap Daten');
        if isequal(filename, 0)
            %disp('Benutzer wählte ''Cancel''');
            return %Abbruch
            else
            %disp(['Dateipfad', fullfile(pathname, filename)]);
            DateiPfad = fullfile(pathname, filename);
        end
        load(DateiPfad)
     %/ GUI Dateiauswahl

        % Hole schon ausgewählte Datensätze aus der Listbox3 um Index ür
"Heat" zu berechnen
        Elemente=size(get(handles.listbox3_Anzeige,'String'));
        ElementAnzahl =Elemente(1,1);
        Index_Heatmaps= (ElementAnzahl/4) +1; % Eine Datensatz für die
Logik besteht aus 4 Elementen
        Heat{1,Index_Heatmaps}=IntpolWerte;

        %$ Prüfe ob Heatmap Daten vom selben Jahr sind
            if JahrPruef == 0;
            JahrPruef=Jahr;
            JahrString= num2str(Jahr);
            set(handles.Jahr, 'String', JahrString );
            else
                if JahrPruef==Jahr;
                    %msgbox('Jahr OK');
                else
                    msgbox('Jahr unterschiedlich! ');
                    return % Abbruch
                end
            end
        %/ Prüfe ob Heatmap Daten vom selben Jahr sind

        %Übergebe ausgewählten Heatmap an listbox1_Datensatz
```

```
        set(handles.listbox1_Datensatz, 'String', DatenName , 'Max',2,
'Min',0);
    %/ Auswahl von gespeichrten Datensätzen


    %$ listbox1_Datensatz um ausgewählten Datensatz anzuzeigen
% --- Executes on selection change in listbox1_Datensatz.
function listbox1_Datensatz_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1_Datensatz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
listbox1_Datensatz contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox1_Datensatz

% --- Executes during object creation, after setting all properties.
function listbox1_Datensatz_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1_Datensatz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
    %/ listbox1_Datensatz um ausgewählten Datensatz anzuzeigen

    %$ listbox2_Bedingung enthählt Operatoren ==, <, > ... zum auswählen
% --- Executes on selection change in listbox2_Bedingung.
function listbox2_Bedingung_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2_Bedingung (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
listbox2_Bedingung contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox2_Bedingung

% --- Executes during object creation, after setting all properties.
function listbox2_Bedingung_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2_Bedingung (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
    %/ listbox2_Bedingung enthählt Operatoren ==, <, > ... zum auswählen


    %$ Eingabe Feld für den Wert zum vergleichen
function edit1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
    %/ Eingabe Feld für den Wert zum vergleichen


    %$ listbox3_Anzeige um bestätigten Datensatz anzuzeigen
% --- Executes on selection change in listbox3_Anzeige.
function listbox3_Anzeige_Callback(hObject, eventdata, handles)
% hObject    handle to listbox3_Anzeige (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox3_Anzeige
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox3_Anzeige

% --- Executes during object creation, after setting all properties.
function listbox3_Anzeige_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox3_Anzeige (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
    %/ listbox3_Anzeige um bestätigten Datensatz anzuzeigen

    %$ Listbox 4 enthählt Logik UND, ODER ... zum auswählen
% --- Executes on selection change in listbox4_Verknuepfung.
function listbox4_Verknuepfung_Callback(hObject, eventdata, handles)
% hObject    handle to listbox4_Verknuepfung (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
listbox4_Verknuepfung contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox4_Verknuepfung
```

```
% --- Executes during object creation, after setting all properties.
function listbox4_Verknuepfung_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox4_Verknuepfung (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
    %$ Listbox 4 enthählt Logik UND, ODER ... zum auswählen

    %$ pushbutton2_Eingabe
    % pushbutton2_Eingabe speichert Ausgewählten Datensatz (Heatmap) mit
    % Logik-Verknüpfung und schreibt sie in die Listbo3_Anzeige und Edit2
Feld um die Logik noch nachzubearbeiten
% --- Executes on button press in pushbutton2_Eingabe.
function pushbutton2_Eingabe_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2_Eingabe (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

        %Hole die Ausgewählte Bedningung aus der Listbox2_Bedingung
        index_selected_Bedingung = get(handles.listbox2_Bedingung,'Value');
        file_list_Bedingung = get(handles.listbox2_Bedingung,'String');
        Bedingung=file_list_Bedingung(index_selected_Bedingung,:);

        %Hole die Ausgewählte Verknüpfung aus der listbox4_Verknuepfung
        index_selected_Verknuepfung =
get(handles.listbox4_Verknuepfung,'Value');
        file_list_Verknuepfung =
get(handles.listbox4_Verknuepfung,'String');
        Verknuepfung=file_list_Verknuepfung(index_selected_Verknuepfung,:);
        Verknuepfung= Verknuepfung{1,1};

        %Hole Name des Datensatz (Heatmap)
        Name = get(handles.listbox1_Datensatz, 'String');
        %Hole Wert aus edit1 Feld
        Wert = get(handles.edit1, 'String');
        Wert = Wert{1,1};

        %Hole Namen listbox3_Anzeige und hänge neu  ausgewählten Namen dran
        Anzeige=get(handles.listbox3_Anzeige,'String');
        Anzeige{end+1}=Name;

        % Lösche Leerzeichen aus Logik (kommen durch Array Dimension
zusatande)
        Anzeige{end+1}=regexprep(Bedingung,' ','');
        Anzeige{end+1}=regexprep(Wert,' ','');
        Anzeige{end+1}=regexprep(Verknuepfung,' ','');

        %$ Abfangen Datensatz ausgewählt und Wert eingegeben wurde
            if sum(isstrprop(Wert, 'alpha')) == 0
                try
                set(handles.listbox3_Anzeige, 'String', Anzeige , 'Max',2,
'Min',0);
                set(handles.edit2, 'String', Anzeige, 'Max', 2);
```

```
                catch
                msgbox('Kein Datensatz ausgewählt, bitte auswählen!');
                end
            else
               msgbox('Kein Wert eingegeben, bitte eingeben!');
            end
        %/ Abfangen Datensatz ausgewählt und Wert eingegeben wurde
    %/ pushbutton2_Eingabe



    %$ Anzeige aller ausgewählte Datensätze mit Möglichkeit Logik-Aufbau zu
ändern
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
    %/ Anzeige aller ausgewählte Datensätze mit Möglichkeit Logik-Aufbau zu
ändern




% --- Executes on button press in pushbutton3_Berechnen.
function pushbutton3_Berechnen_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3_Berechnen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
        global Heat
        global LogikRechnung
        try
        %Hole Element Anzahl und berechne Anzahl Heatmaps die verwendet
werden
        Elemente=size(get(handles.edit2,'String'));
        ElementAnzahl =Elemente(1,1);
        Index_Heatmaps= (ElementAnzahl/4);

        %$ Schleife bis Anzahl Heatmaps erreicht ist
        for n=1:1: Index_Heatmaps
            %Berechne Zähler z zum indizieren
            if n~=1
                z=((n-1)*4)+1; % z Springt mit jeder Heatmap+Logik um 4
(weil 4 Elemnete pro Heatmap) 5,9,13...
```

```
            else
                z=1; %Start Index 1
            end

            % Hole Logik-Aufbau aus Edit2 Fenster
            Anzeige=get(handles.edit2,'String');
            % Hole Bedingung und lösche Leerzeichen
            Bedingung(1,n) =  regexprep(Anzeige(z+1,:),' ','');

            %$ Wenn Eingabe Zahl ist Hole Wert
                if sum(isstrprop(Anzeige{z+2,:}, 'alpha')) == 0
                    %Wandle Text-Zahl in Double-Zahl
                    Wert(1,n) = str2double(Anzeige(z+2,:));
                else
                    close (gcf)
                    msgbox('Wert nicht korrekt! Abbruch');
                    return
                end
            %$ Wenn Eingabe Zahl ist Hole Wert

            % Hole Verknüfung und lösche Leerzeichen
            Verknuepfung(1,n) = regexprep(Anzeige(z+3,:),' ','');
        end
        %/ Schleife bis Anzahl Heatmaps erreicht ist

    %$ Schleife Anzahl Heatmaps
    for n=1:1: Index_Heatmaps

            %$ Vergleiche welche Bedingung vorliegt und Bilde Heatmap mit
Logik 0 1
            switch Bedingung{1,n}
                  case '=='
                    Logik{1,n}=Heat{1,n} == Wert(1,n);
                    %fprintf('Bedingung == \n');
                  case '<'
                    %
                    Logik{1,n}=Heat{1,n}< Wert(1,n);
                    fprintf('Bedingung < \n');
                  case '>'
                    Logik{1,n}=Heat{1,n}> Wert(1,n);
                    fprintf('Bedingung > \n');
                  case '<='
                    Logik{1,n}=Heat{1,n}<= Wert(1,n);
                    %fprintf('Bedingung <= \n');
                  case '>='
                   Logik{1,n}=Heat{1,n}>= Wert(1,n);
                   %fprintf('Bedingung >=  \n');
                  case '~='
                    Logik{1,n}=Heat{1,n}~= Wert(1,n);
                    %fprintf('Bedingung ~= \n');
                  otherwise
                    close (gcf)
                    msgbox('Bedingung nicht korrekt! Abbruch');
                    return
            end
            %/ Vergleiche welche Bedingung vorliegt und Bilde Heatmap mit
Logik 0 1


                        % Deaktiv. Nur zum überprüfen des Codes notwendig
```

```
                        %$Daten Plot Test
                        fig1 = figure;
                        set(fig1, 'WindowStyle', 'docked')
                        set(fig1, 'Name','Heatmap','NumberTitle','off');
                        TitelInterp=strjoin({'Interpolierte Werte: ',
'JahrName',': ','DatenName'});
                        imagesc(Logik{1,n});
                        set(gca,'ydir','normal') ;   %reverse that y-axis!
man that took me forever to find!
                        title(TitelInterp);
                        ax = gca;
                        ax.YTick = [60 120 180 240 300 360 420 480 540 600
660 720 780 840 900 960 1020 1080 1140 1200 1260 1320 1380 1440];
                        ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24};
                        xlabel('Tage', 'FontSize', 16)
                        ylabel('Stunden', 'FontSize', 16)
                        colormap gray;
                        h = colorbar();
                        ylabel(h,'Werte','FontSize', 16)
                        setFigDockGroup(gcf,'DatenName');  % Andocken der
figure an Gruppe DatenName


                        fig2 = figure;
                        set(fig2, 'WindowStyle', 'docked')
                        set(fig2, 'Name','Zeitpunkt','NumberTitle','off');
                        Titel01=strjoin({'Zeitpunkt Datenlog ',
'JahrName',': ','DatenName'});
                        imagesc(Heat{1,n});
                        title(Titel01);
                        set(gca,'ydir','normal');
                        ax = gca;
                        ax.YTick = [60 120 180 240 300 360 420 480 540 600
660 720 780 840 900 960 1020 1080 1140 1200 1260 1320 1380 1440];
                        ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24};
                        xlabel('Tage', 'FontSize', 16)
                        ylabel('Stunden', 'FontSize', 16)
                        colormap jet
                        colorbar
                        setFigDockGroup('DatenName');  % dock fig to a new
user group

                        %/DAten Plot Test

    end

        if Index_Heatmaps ==1
            LogikRechnung = Logik{1,1};
        end



        %$ Wenn mehr als 1 Heatmap einbezogen sind
        if Index_Heatmaps >1
            %$ Vergleiche welche Verknüpfung vorliegt und Bilde aus den
beiden neue Logik 0 1
              switch Verknuepfung{1,1}
```

```
                    case '&'
                      LogikRechnung= Logik{1,1} & Logik{1,2};
                      fprintf('&')
                    case '|'
                      LogikRechnung= Logik{1,1} | Logik{1,2};
                      fprintf('|')
                    case 'xor'
                      LogikRechnung= xor(Logik{1,1},  Logik{1,2});
                      fprintf('xor')
                    otherwise
                      close (gcf)
                      msgbox('1 Verknüpfung nicht korrekt! Abbruch');
                      return
                end
            %/ Vergleiche welche Verknüpfung vorliegt und Bilde aus den
beiden neue Logik 0 1
        end
        %/ Wenn mehr als 1 Heatmap einbezogen sind


            %$ Wenn mehr als 2 Heatmaps einbezogen sind
            if Index_Heatmaps > 2

            %$ Schleife wenn mehr als 1 Verknüpfung
               for n=3:1:Index_Heatmaps

                    %$ Vergleiche welche Verknüpfung vorliegt und Bilde
aus den beiden neue Logik 0 1
                    switch Verknuepfung{1,n-1}
                        case '&'
                          LogikRechnung= LogikRechnung & Logik{1,n};
                          fprintf('&')
                        case '|'
                          LogikRechnung= LogikRechnung | Logik{1,n};
                          fprintf('|')
                        case 'xor'
                          LogikRechnung= xor (LogikRechnung ,
Logik{1,n});
                          fprintf('xor')
                        otherwise
                          close (gcf)
                          msgbox('2 Verknüpfung nicht korrekt! Abbruch');
                          return
                    end
                    %/ Vergleiche welche Verknüpfung vorliegt und Bilde
aus den beiden neue Logik 0 1
                end

            end
            %/ Wenn mehr als 2 Heatmaps einbezogen sind

            %$ Ausgabe der berechneten Logik
                fig1 = figure;
                set(fig1, 'WindowStyle', 'docked')
                set(fig1, 'Name','Heatmap','NumberTitle','off');
                TitelInterp=strjoin({'Logik : ', 'Name erst beim Speichern
vergeben'});
                imagesc(LogikRechnung);
                set(gca,'ydir','normal') ;   %reverse that y-axis! man that
took me forever to find!
                title(TitelInterp);
```

```
                ax = gca;
                ax.YTick = [60 120 180 240 300 360 420 480 540 600 660 720
780 840 900 960 1020 1080 1140 1200 1260 1320 1380 1440];
                ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24};
                xlabel('Tage', 'FontSize', 16)
                ylabel('Stunden', 'FontSize', 16)
                colormap gray;
                h = colorbar();
                ylabel(h,'Werte','FontSize', 16)
                setFigDockGroup(gcf,'DatenName');  % Andocken der figure an
Gruppe DatenName
            %/ Ausgabe der berechneten Logik
        catch
            msgbox('Keine Logik Datei')
        end

    %$ Speichern der Logik
% --- Executes on button press in speichern.
function speichern_Callback(hObject, eventdata, handles)
% hObject    handle to speichern (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
        global JahrPruef
        global LogikRechnung

        %Aufruf des Dialogs zum Speichern der Logik
        LogikSpeichern= get(handles.edit2,'String');
        [DateiName,PathName] = uiputfile('*.mat','Save Workspace As');
        DateiPfad = fullfile(PathName, DateiName);
        save(DateiPfad, 'LogikRechnung','LogikSpeichern','JahrPruef');
    %/ Speichern der Logik


    %$ Öffnen einer schon vorhandenen Logik-Heatmap
% --- Executes on button press in Vorh_Logik.
function Vorh_Logik_Callback(hObject, eventdata, handles)
% hObject    handle to Vorh_Logik (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %$ GUI Dateiauswahl
        [filename, pathname] = uigetfile('*.mat', '.mat File Wählen: Lade
HeatMap Daten');
        if isequal(filename, 0)
            disp('User selected ''Cancel''');
            else
            disp(['User selected ', fullfile(pathname, filename)]);
            DateiPfad = fullfile(pathname, filename);
        end
        load(DateiPfad);
    %/ GUI Dateiauswahl
    filename= regexprep(filename, '.mat', '');
        try %  Wenn in Anweisung fehler auftritt weil keine Logik

        %Hole Daten aus geladenen Logik-Datensatz
        JahrString= num2str(JahrPruef);
        set(handles.Jahr, 'String', JahrString )
        set(handles.edit2, 'String', LogikSpeichern, 'Max', 2)
```

```
        DatenName=JahrString;
        JahrName=JahrString;
        %$ Ausgabe Logik in figure
            fig1 = figure;

                %$ Erstellen der Buttons für die Funktionen der
Weiterverarbeitung in dem Plott Heatmap. Nur für fig1.

                    %§ Einfügen des Toolbar-Button mit Function aufruf bei
Callback
                        hToolbar = findall(fig1,'tag','FigureToolBar');
                        % Einlesen der Bild Datei für Button
                        [img,map] = imread(fullfile(matlabroot,...
                                    'toolbox','matlab','icons','bze.gif'));
                        % Konvertiere Bild zu RGB fürs Icon
                        icon = ind2rgb(img,map);
                        % Erstelle ein uipushtool in der toolbar
                        tbh = findall(hToolbar,'Type','uitoolbar');
                        p =
uipushtool(tbh,'TooltipString','Datenanalyse',...
                                            'Separator','on',...
                                            'HandleVisibility','off', ...
                                           'ClickedCallback',{@Datenanalyse,
DatenName, JahrName});
                                            %§§§§§§§§§§§§§§§§   %FUNCTION
Datenanalyse
                                            % Bei einem ClickedCallback von
dem Icon wird die Funktion Datenanalyse gestartet und die
                                            % Variablen DatenName und JahrName
übergeben
                        % Setze den Button
                        p.CData = icon;
                    %/ Einfügen des Toolbar-Button mit Function aufruf bei
Callback


                    %§ Einfügen des Toolbar-Button mit Function aufruf bei
Callback
                        hToolbar = findall(fig1,'tag','FigureToolBar');
                        % Einlesen der Bild Datei für Button
                        [img,map] = imread(fullfile(matlabroot,...
                                    'toolbox','matlab','icons','bze.gif'));
                        % Konvertiere Bild zu RGB fürs Icon
                        icon = ind2rgb(img,map);
                        % Erstelle ein uipushtool in der toolbar
                        tbh = findall(hToolbar,'Type','uitoolbar');
                        p =
uipushtool(tbh,'TooltipString','DatumAnzeige',...
                                            'HandleVisibility','off', ...
                                            'ClickedCallback',{@DatumAnzeige,
JahrName});
                                            %§§§§§§§§§§§§§§§§   %FUNCTION
Datenanalyse
                                            % Bei einem ClickedCallback von
dem Icon wird die Funktion Datenanalyse gestartet und die
                                            % Variablen DatenName und JahrName
übergeben
                        % Setze den Button
                        p.CData = icon;
                    %/ Einfügen des Toolbar-Button mit Function aufruf bei
Callback
```

```
                %§ Einfügen des Toolbar-Button mit Function aufruf bei
Callback
                    hToolbar = findall(fig1,'tag','FigureToolBar');
                    % Einlesen der Bild Datei für Button
                    [img,map] = imread(fullfile(matlabroot,...
                                'toolbox','matlab','icons','bze.gif'));
                    % Konvertiere Bild zu RGB fürs Icon
                    icon = ind2rgb(img,map);
                    % Erstelle ein uipushtool in der toolbar
                    tbh = findall(hToolbar,'Type','uitoolbar');
                    p =
uipushtool(tbh,'TooltipString','DatumAnzeige',...
                                'HandleVisibility','off', ...

'ClickedCallback',{@Liniendiagramme, DatenName, JahrName,LogikRechnung});
                                %§§§§§§§§§§§§§§§§   %FUNCTION
Datenanalyse
                                % Bei einem ClickedCallback von
dem Icon wird die Funktion Datenanalyse gestartet und die
                                % Variablen DatenName und JahrName
übergeben
                    % Setze den Button
                    p.CData = icon;
                %/ Einfügen des Toolbar-Button mit Function aufruf bei
Callbac

            %/ Erstellen der Buttons für die Funktionen der
Weiterverarbeitung in dem Plott Heatmap. Nur für fig1.

            % Konfiguration, Beschriftung und Formatierung der Grafik,
Achsen und Legenden
            % Exemplarische Beschreibung ist in Daten_Plot.m zu finden
            set(fig1, 'WindowStyle', 'docked')
            set(fig1, 'Name','Heatmap','NumberTitle','off');
            TitelInterp=strjoin({'Logik: ', JahrString,': ',filename});
            imagesc(LogikRechnung);
            set(gca,'ydir','normal') ;
            title(TitelInterp);
            ax = gca;
            ax.YTick = [60 120 180 240 300 360 420 480 540 600 660 720 780
840 900 960 1020 1080 1140 1200 1260 1320 1380 1440];
            ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
19 20 21 22 23 24};
            xlabel('Tage', 'FontSize', 16)
            ylabel('Stunden', 'FontSize', 16)
            colormap gray;
            h = colorbar();
            ylabel(h,'Werte','FontSize', 16)
            setFigDockGroup(gcf,filename);  % Andocken der figure an Gruppe
DatenName

        %/Ausgabe Logik in figure

        catch
            msgbox('Keine Logik-Daten! Abbruch');
        end
    %/ Öffnen einer schon vorhandenen Logik-Heatmap
```

A.4.11 Manipulation2Heatmaps.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20


% Hier können 2 Heatmaps mit einander über Mathematische Opperatoren +,-,*
% etc. miteinader kombiniert und das Ergbeniss gespeichert werden


    %$ MATLAB erstellte Funktion die anderen Funktionen in diesem .m File
    % deklariert und steuert
function varargout = Manipulation2Heatmaps(varargin)
% MANIPULATION2HEATMAPS MATLAB code for Manipulation2Heatmaps.fig
%      MANIPULATION2HEATMAPS, by itself, creates a new
MANIPULATION2HEATMAPS or raises the existing
%      singleton*.
%
%      H = MANIPULATION2HEATMAPS returns the handle to a new
MANIPULATION2HEATMAPS or the handle to
%      the existing singleton*.
%
%      MANIPULATION2HEATMAPS('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in MANIPULATION2HEATMAPS.M with the given
input arguments.
%
%      MANIPULATION2HEATMAPS('Property','Value',...) creates a new
MANIPULATION2HEATMAPS or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before Manipulation2Heatmaps_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Manipulation2Heatmaps_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Manipulation2Heatmaps

% Last Modified by GUIDE v2.5 07-Nov-2016 14:26:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @Manipulation2Heatmaps_OpeningFcn, ...
'gui_OutputFcn',  @Manipulation2Heatmaps_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
    %/ MATLAB erstellte Funktion die anderen Funktionen in diesem .m File
    % deklariert und steuert


    %$ Anweisung die vor der Erstellung des GUI ausgeführt werden sollen
% --- Executes just before Manipulation2Heatmaps is made visible.
function Manipulation2Heatmaps_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Manipulation2Heatmaps (see VARARGIN)

% Choose default command line output for Manipulation2Heatmaps
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes Manipulation2Heatmaps wait for user response (see UIRESUME)
% uiwait(handles.figure1);

        % Speicher Button deaktiviert
        set(handles.Speichern2Heatmaps,'Enable','off')
    %/ Anweisung die vor der Erstellung des GUI ausgeführt werden sollen


   %$ Ausgabe an die Kommandline in MATLAB. Wird nicht genutzt
% --- Outputs from this function are returned to the command line.
function varargout = Manipulation2Heatmaps_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
    %/ Ausgabe an die Kommandline in MATLAB. Wird nicht genutzt


    %$ Wähle 1 Datensatz (Heatmap)
% --- Executes on button press in Heat1.
function Heat1_Callback(hObject, eventdata, handles)
% hObject    handle to Heat1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
        global Heat1_Jahr
        global Heat1
        global Heat1_DatenName
        global Heat1_Historie

        %$ GUI Dateiauswahl
```

```
            [filename, pathname] = uigetfile('*.mat', '.mat File Wählen:
Lade HeatMap Daten');
            if isequal(filename, 0)
                %disp('Benutzer wählte ''Cancel''');
                return
                else
                %disp(['DateiPfad ', fullfile(pathname, filename)]);
                DateiPfad = fullfile(pathname, filename);
            end
            load(DateiPfad);
        %/ GUI Dateiauswahl

        % Schreibe Daten aus der geladenen Datei in global Variablen
        try % Für Heatmap Datensatz
            Heat1=IntpolWerte;
            Heat1_DatenName=DatenName;
            Heat1_Jahr=num2str(Jahr);
            Heat1_Historie= Historie;
        catch % Für Logik DatenSatz
            Heat1=LogikRechnung;
            Heat1_DatenName='Logik';
            Heat1_Jahr=num2str(JahrPruef);
            Heat1_Historie= LogikSpeichern;
        end
        %Schreibe DatenName in Textfeld
        set(handles.TextHeatmap1, 'String', Heat1_DatenName);
    %/ Wähle 1 Datensatz (Heatmap)


    %$ Wähle 2 Datensatz (Heatmap)
% --- Executes on button press in Heat2.
function Heat2_Callback(hObject, eventdata, handles)
% hObject    handle to Heat2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

        global Heat2_Jahr
        global Heat2
        global Heat2_DatenName
        global Heat2_Historie


        %$ GUI Dateiauswahl
        [filename, pathname] = uigetfile('*.mat', '.mat File Wählen: Lade
HeatMap Daten');
        if isequal(filename, 0)
            %disp('Benutzer wählte ''Cancel''');
            return
            else
            %disp(['Dateipfad ', fullfile(pathname, filename)]);
            DateiPfad = fullfile(pathname, filename);
        end
        load(DateiPfad);
        %/ GUI Dateiauswahl

        try
        % Schreibe Daten aus der geladenen Datei in global Variablen
            Heat2=IntpolWerte;
            Heat2_DatenName=DatenName;
            Heat2_Jahr=num2str(Jahr);
```

```
            Heat2_Historie= Historie;
        catch % Für Logik DatenSatz
            Heat2=LogikRechnung;
            Heat2_DatenName='Logik';
            Heat2_Jahr=num2str(JahrPruef);
            Heat2_Historie= LogikSpeichern;
        end
        %Schreibe DatenName in Textfeld
        set(handles.TextHeatmap2, 'String', Heat2_DatenName);

    %/ Wähle 2 Datensatz (Heatmap)


    %$ Berechne die 2 Heatmaps und gebe sie aus
% --- Executes on button press in Vorschau2Heat.
function Vorschau2Heat_Callback(hObject, eventdata, handles)
% hObject    handle to Vorschau2Heat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
            global Heat1
            global Heat2
            global HeatNeu
            global Heat1_DatenName
            global Heat2_DatenName
            global Heat1_Jahr
            global Heat2_Jahr
            global Operator

            %Hole Wert von den Radiobuttons
            Plus= get(handles.Plus2Heat, 'Value');
            Minus= get(handles.Minus2Heat, 'Value');
            Divi= get(handles.Divi2Heat, 'Value');
            Multi= get(handles.Multi2Heat, 'Value');

                %$ Vergleiche welcher Opperator verwendet wurde und
berechne die neuen Werte
                    if Plus==1
                            HeatNeu=Heat1 + Heat2;
                            Operator='+';
                    elseif Minus==1
                            HeatNeu=Heat1 - Heat2;
                            Operator='-';
                    elseif  Multi == 1
                            size(Heat1)
                            size(Heat1)

                            SpZl_Heat1=size(Heat1);
                            TagEnde=SpZl_Heat1(1,2);

                            for t=1:1:TagEnde
                                for r=1:1:1440
                                HeatNeu(r,t)=Heat1(r,t) * Heat2(r,t);
                                end
                            end
                            Operator='*';
                    elseif Divi == 1
                            HeatNeu=Heat1 / Heat2;
                            Operator='/';
                    end
```

```
                %/ Vergleiche welcher Opperator verwendet wurde und
berechne die neuen Werte


                %$ Gebe die Urspungs und neu berechnete Heatmap aus
                    fig1 = figure;
                    set(fig1, 'WindowStyle', 'docked')
                    set(fig1, 'Name','Ursprung 1','NumberTitle','off');
                    TitelInterp=strjoin({'Ursprung 1: ',Heat1_Jahr,'
',Heat1_DatenName});
                    imagesc(Heat1);
                    set(gca,'ydir','normal') ;   %reverse that y-axis! man
that took me forever to find!
                    title(TitelInterp);
                    ax = gca;
                    ax.YTick = [60 120 180 240 300 360 420 480 540 600 660
720 780 840 900 960 1020 1080 1140 1200 1260 1320 1380 1440];
                    ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24};
                    xlabel('Tage', 'FontSize', 16)
                    ylabel('Stunden', 'FontSize', 16)
                    %Prüfe ob Heatmap oder Logik, passe colormap an
                    if strcmp(Heat1_DatenName, 'Logik');
                        colormap gray;
                    else
                        colormap jet;
                    end
                    h = colorbar();
                    ylabel(h,'Werte','FontSize', 16)
                    % set(h, 'ylim', [0 5])
                    setFigDockGroup(gcf,'Manipulation2Heatmaps');  % dock
fig to a new user group

                    fig2 = figure;
                    set(fig2, 'WindowStyle', 'docked')
                    set(fig2, 'Name','Ursprung 2','NumberTitle','off');
                    TitelInterp=strjoin({'Ursprung 2: ',Heat2_Jahr,'
',Heat2_DatenName});
                    imagesc(Heat2);
                    set(gca,'ydir','normal') ;   %reverse that y-axis! man
that took me forever to find!
                    title(TitelInterp);
                    ax = gca;
                    ax.YTick = [60 120 180 240 300 360 420 480 540 600 660
720 780 840 900 960 1020 1080 1140 1200 1260 1320 1380 1440];
                    ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24};
                    xlabel('Tage', 'FontSize', 16)
                    ylabel('Stunden', 'FontSize', 16)
                     %Prüfe ob Heatmap oder Logik, passe colormap an
                    if strcmp(Heat1_DatenName, 'Logik');
                        colormap gray;
                    else
                        colormap jet;
                    end
                    h = colorbar();
                    ylabel(h,'Werte','FontSize', 16)
                    setFigDockGroup(gcf,'Manipulation2Heatmaps');  % dock
fig to a new user group
                    % set(h, 'ylim', [0 5])
```

```
                    fig3 = figure;
                    set(fig3, 'WindowStyle', 'docked')
                    set(fig3, 'Name','Neu Berechnete
Werte','NumberTitle','off');
                    TitelInterp=strjoin({'Neu Berechnete Werte aus 2
Heatmaps: '});
                    imagesc(HeatNeu);
                    set(gca,'ydir','normal') ;   %reverse that y-axis! man
that took me forever to find!
                    title(TitelInterp);
                    ax = gca;
                    ax.YTick = [60 120 180 240 300 360 420 480 540 600 660
720 780 840 900 960 1020 1080 1140 1200 1260 1320 1380 1440];
                    ax.YTickLabel = {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24};
                    xlabel('Tage', 'FontSize', 16)
                    ylabel('Stunden', 'FontSize', 16)
                    colormap jet;
                    h = colorbar();
                    ylabel(h,'Werte','FontSize', 16)
                    setFigDockGroup(gcf,'Manipulation2Heatmaps');  % dock
fig to a new user group

                %/ Gebe die Urspungs und neu berechnete Heatmap aus

    % Aktiviere Speicher-Button
    set(handles.Speichern2Heatmaps,'Enable','on')

    %/ Berechne die 2 Heatmaps und gebe sie aus


    % --- Executes on button press in Speichern2Heatmaps.
    function Speichern2Heatmaps_Callback(hObject, eventdata, handles)
    % hObject    handle to Speichern2Heatmaps (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
        global HeatNeu
        global Heat1_Jahr
        Jahr=Heat1_Jahr;
        global Heat1_DatenName
        global Heat2_DatenName
        global Operator
        global Heat1_Historie
        global Heat2_Historie


        SpZl_AnzTage=size(HeatNeu);
        AnzTage=SpZl_AnzTage(1,2);

        % Schreibe neue Heatmap in Datensatz Schema. Nur IntpolWerte
        % enthält Daten. Anderen Matrizen belieben 0 da diese Werte nicht
        % berechenbar sind
        IntpolWerte=HeatNeu;
        Rohdaten01 = zeros(1440, AnzTage);
        RohdatenWerte = zeros(1440, AnzTage);
        QulitaetWerte = zeros(1440, AnzTage);


        %GUI Speichern der Variablen um sie später wieder zu verweden
```

```
        [FileName, PathName] = uiputfile('ManipulierteHeatmap.mat',
'Speichern als...') ;
        if isequal(FileName, 0)
            %disp('Benutzer wählte ''Cancel''');
            return
            else
            %disp(['dateipfad ', fullfile(PathName, FileName)]);
            DateiPfad = fullfile(PathName, FileName);
        end

        DatenName=FileName;
        DatenName = regexprep(DatenName,'.mat','');

        %$ Schreibe Historie
        for Heat1_Zaehl=1:1:numel(Heat1_Historie)
            Historie(Heat1_Zaehl,1) = Heat1_Historie(Heat1_Zaehl,1)
        end
        Historie{end+1}=Operator
        for Heat2_Zaehl=1:1:numel(Heat2_Historie)
            Historie(end+1) = Heat2_Historie(Heat2_Zaehl,1)
        end
        %/ Schreibe Historie

        save(DateiPfad, 'DatenName','Jahr','Rohdaten01',
'RohdatenWerte','IntpolWerte', 'QulitaetWerte', 'Historie'  )
```

A.12 ManipulationDaten.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

function varargout = ManipulationDaten(varargin)
% MANIPULATIONDATEN MATLAB code for ManipulationDaten.fig
%      MANIPULATIONDATEN, by itself, creates a new MANIPULATIONDATEN or
raises the existing
%      singleton*.
%
%      H = MANIPULATIONDATEN returns the handle to a new MANIPULATIONDATEN
or the handle to
%      the existing singleton*.
%
%      MANIPULATIONDATEN('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in MANIPULATIONDATEN.M with the given input
arguments.
%
%      MANIPULATIONDATEN('Property','Value',...) creates a new
MANIPULATIONDATEN or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ManipulationDaten_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ManipulationDaten_OpeningFcn via
varargin.
%
```

```
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ManipulationDaten

% Last Modified by GUIDE v2.5 08-Feb-2017 12:32:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ManipulationDaten_OpeningFcn, ...
                   'gui_OutputFcn',  @ManipulationDaten_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before ManipulationDaten is made visible.
function ManipulationDaten_OpeningFcn(hObject, eventdata, handles,
varargin)

        %Globals aus Matlab save file
        global DatenName
        global IntpolWerte
        global Jahr
        global QulitaetWerte
        global Rohdaten01
        global RohdatenWerte
        global Historie
        %/Globals aus Matlab save file

        global JahrName
        global filename
        global pathname
        global DateiPfad
        global WertMin
        global WertMax
        %GUI Dateiauswahl
        [filename, pathname] = uigetfile('*.mat', '.mat File Wählen: Lade
HeatMap Daten');
        if isequal(filename, 0)
            disp('User selected ''Cancel''');
            else
            disp(['User selected ', fullfile(pathname, filename)]);
            DateiPfad = fullfile(pathname, filename);
        end
        load(DateiPfad);
        %/ GUI Dateiauswahl
```

```
        JahrName=num2str(Jahr)
        WertMax=max(max(RohdatenWerte));
        WertMin=min(min(RohdatenWerte));

        if sum(sum(RohdatenWerte))==0
        WertMax=max(max(IntpolWerte));
        WertMin=min(min(IntpolWerte));
        end

        fprintf('OpenFun\n')

        WertMin_Char=num2str(WertMin);
        WertMax_Char=num2str(WertMax);


        set(handles.Speichern,'Enable','off')


% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ManipulationDaten (see VARARGIN)

% Choose default command line output for ManipulationDaten
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

        %WertMax_Char=getappdata(test,'Uebergabe')
        %handles.myNewVariable1
        %handles.myNewVariable2
        TextMinMax=strjoin({'Min-Wert;',WertMin_Char,'Max-Wert:',
WertMax_Char});
        set(handles.text4, 'String', TextMinMax);

        set(handles.ManipulationDatenName, 'String', DatenName);
        set(handles.MinNorm, 'String',WertMin_Char );
        set(handles.MaxNorm, 'String',WertMax_Char );
        set(handles.MinAbs, 'String',WertMin_Char );
        set(handles.MaxAbs, 'String',WertMax_Char );
        set(handles.WertMathOper, 'String','' );
% UIWAIT makes ManipulationDaten wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ManipulationDaten_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
function MinNorm_Callback(hObject, eventdata, handles)
% hObject    handle to MinNorm (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
MinNorm as text
%        str2double(get(hObject,'String')) returns
contents of MinNorm as a double

        valMinNorm = get(hObject, 'Value')
        TextMinNorm = get(hObject, 'String')


% --- Executes during object creation, after setting
all properties.
function MinNorm_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MinNorm (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
        fprintf('Create Min\n')



function MaxNorm_Callback(hObject, eventdata, handles)
% hObject    handle to MaxNorm (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
MaxNorm as text
%        str2double(get(hObject,'String')) returns
contents of MaxNorm as a double
        valMaxNorm = get(hObject, 'Value')
        TextMaxNorm = get(hObject, 'String')

% --- Executes during object creation, after setting
all properties.
function MaxNorm_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxNorm (see GCBO)
```

```
                    % eventdata  reserved - to be defined in a future
version of MATLAB
                    % handles    empty - handles not created until after
all CreateFcns called

                    % Hint: edit controls usually have a white background
on Windows.
                    %       See ISPC and COMPUTER.
                    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
                        set(hObject,'BackgroundColor','white');
                    end
                            fprintf('Create Max\n')
                            valMaxNorm = get(hObject, 'Value')
                            TextMaxNorm = get(hObject, 'String')

                    % --- Executes on button press in Normierung.
                    function Normierung_Callback(hObject, eventdata,
handles)
                    % hObject    handle to Normierung (see GCBO)
                    % eventdata  reserved - to be defined in a future
version of MATLAB
                    % handles    structure with handles and user data (see
GUIDATA)
                        % Hier Kommt Funktion zum Normieren der Werte
                        global DatenName
                        global JahrName
                        global Rohdaten01
                        global QulitaetWerte
                        global RohdatenWerte
                        global IntpolWerte
                        global Historie
                        valNormierung = get(hObject, 'Value')
                        if valNormierung ==1
                            fprintf('ValNormierung Start')
                            WertMax=str2double(get(handles.MaxNorm,
'String'))
                            WertMin=str2double(get(handles.MinNorm,
'String'))
                        end
                        SpZl_IntpolWerte=size(IntpolWerte)
                        TagEnde=SpZl_IntpolWerte(1,2)
                        IntpolWerte(1,1)=WertMin;
                        IntpolWerte(1440,TagEnde)=WertMax;
                        Daten_Plot(DatenName, JahrName, Rohdaten01,
QulitaetWerte, RohdatenWerte,IntpolWerte, Historie)
                        set(handles.Speichern,'Enable','on')




function MinAbs_Callback(hObject, eventdata, handles)
% hObject    handle to MinAbs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MinAbs as text
%        str2double(get(hObject,'String')) returns contents of MinAbs as a
double
```

```
        valMaxNorm = get(hObject, 'Value')
        TextMaxNorm = get(hObject, 'String')

% --- Executes during object creation, after setting all properties.
function MinAbs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MinAbs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function MaxAbs_Callback(hObject, eventdata, handles)
% hObject    handle to MaxAbs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MaxAbs as text
%        str2double(get(hObject,'String')) returns contents of MaxAbs as a
double
        valMaxNorm = get(hObject, 'Value');
        TextMaxNorm = get(hObject, 'String');

% --- Executes during object creation, after setting all properties.
function MaxAbs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxAbs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in DatenAbs.
function DatenAbs_Callback(hObject, eventdata, handles)
% hObject    handle to DatenAbs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    % Hier Werden Daten abgeschnitten nach einem vordefinierten Wert

    global DatenName
    global JahrName
    global Rohdaten01
    global QulitaetWerte
    global RohdatenWerte
    global IntpolWerte
    global Historie
```

```
    %! Hole Min und/oder Max Wert zum Abschneiden der Daten
    valDatenAbs = get(hObject, 'Value');
    if valDatenAbs ==1
        WertMax=str2double(get(handles.MaxAbs, 'String'));
        WertMin=str2double(get(handles.MinAbs, 'String'));
    end

    %! Hole Größe der Variable IntpolWerte
    SpZl_IntpolWerte=size(IntpolWerte);
    TagEnde=SpZl_IntpolWerte(1,2);

    %$ Überprüfe Elemente der Matrix und schneide gegebenenfalls ab
    for t=1:1:TagEnde
        for r=1:1:1440
            if IntpolWerte(r,t) < WertMin
               IntpolWerte(r,t)=WertMin;
            end
            if IntpolWerte(r,t) > WertMax
               IntpolWerte(r,t)=WertMax;
            end
        end
        fprintf('Tag %d', t)
    end
    %/ Überprüfe Elemente der Matrix und schneide gegebenenfalls ab

    Daten_Plot(DatenName, JahrName, Rohdaten01, QulitaetWerte,
RohdatenWerte,IntpolWerte, Historie)
    %! Setze Button "Speichern" aktiv
    set(handles.Speichern,'Enable','on')


% --- Executes on button press in Plus.
function Plus_Callback(hObject, eventdata, handles)
% hObject    handle to Plus (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Plus


% --- Executes on button press in Minus.
function Minus_Callback(hObject, eventdata, handles)
% hObject    handle to Minus (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Minus


% --- Executes on button press in Divi.
function Divi_Callback(hObject, eventdata, handles)
% hObject    handle to Divi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Divi


% --- Executes on button press in Multi.
```

```
function Multi_Callback(hObject, eventdata, handles)
% hObject    handle to Multi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Multi

% --- Executes on button press in Multi.
function Absolut_Callback(hObject, eventdata, handles)
% hObject    handle to Multi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Multi



function WertMathOper_Callback(hObject, eventdata, handles)
% hObject    handle to WertMathOper (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of WertMathOper as text
%        str2double(get(hObject,'String')) returns contents of WertMathOper
as a double


% --- Executes during object creation, after setting all properties.
function WertMathOper_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WertMathOper (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in MathOper.
function MathOper_Callback(hObject, eventdata, handles)
% hObject    handle to MathOper (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %! Vergelich der Opperatoren anhand des Ausgewählten Radio Buttons
    %! und Berechnung und Plotten der neuen Matrix

    global DatenName
    global JahrName
    global Rohdaten01
    global QulitaetWerte
    global RohdatenWerte
    global IntpolWerte
    global Historie

    valMathOper = get(hObject, 'Value');
```

```
    Wert_MathOper=str2double(get(handles.WertMathOper, 'String'));
    Plus= get(handles.Plus, 'Value');
    Minus= get(handles.Minus, 'Value');
    Multi= get(handles.Multi, 'Value');
    Divi= get(handles.Divi, 'Value');
    Absolut= get(handles.Absolut, 'Value');
    %! Hole Größe der Variable IntpolWerte
    SpZl_IntpolWerte=size(IntpolWerte);
    TagEnde=SpZl_IntpolWerte(1,2);


        if Plus==1
            MathOper = '+';
            for t=1:1:TagEnde
                for r=1:1:1440
                    IntpolWerte(r,t)=IntpolWerte(r,t) + Wert_MathOper;
                end
            end
        elseif Minus==1
            MathOper = '-';
            for t=1:1:TagEnde
                for r=1:1:1440
                IntpolWerte(r,t)=IntpolWerte(r,t) - Wert_MathOper;
                end
            end
        elseif  Multi == 1
            MathOper = '*';
            for t=1:1:TagEnde
                for r=1:1:1440
                IntpolWerte(r,t)=IntpolWerte(r,t) * Wert_MathOper;
                end
            end
        elseif Divi == 1
            MathOper = '/';
            for t=1:1:TagEnde
                for r=1:1:1440
                IntpolWerte(r,t)=IntpolWerte(r,t) / Wert_MathOper;
                end
            end
            elseif Absolut == 1
                MathOper = 'Betrag';
                IntpolWerte=abs(IntpolWerte)
        end



    Daten_Plot(DatenName, JahrName, Rohdaten01, QulitaetWerte,
RohdatenWerte,IntpolWerte, Historie)
    set(handles.Speichern,'Enable','on')

    %! Schreibe Historie
    String_Wert_MathOper = num2str(Wert_MathOper);
    Historie{end+1}=MathOper;
    Historie{end+1}=String_Wert_MathOper;


% --- Executes on button press in Speichern.
function Speichern_Callback(hObject, eventdata, handles)
% hObject    handle to Speichern (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

        %! Speichern der neuen Matrix mit Historie welche Opperation
        %durchgeführt wurde
        global DateiPfad

        %Globals aus Matlab save file
        global DatenName
        global IntpolWerte
        global Jahr
        global QulitaetWerte
        global Rohdaten01
        global RohdatenWerte
        global Historie
        %/Globals aus Matlab save file

        %GUI Speichern der Variablen als m. File
        [FileName, PathName] = uiputfile(DateiPfad, 'Speichern als...') ;
        if isequal(FileName, 0)
            disp('User selected ''Cancel''');
            return
            else
            disp(['User selected ', fullfile(PathName, FileName)]);
            DateiPfad = fullfile(PathName, FileName);
        end
        save(DateiPfad, 'DatenName','Jahr','Rohdaten01',
'RohdatenWerte','IntpolWerte', 'QulitaetWerte', 'Historie' )
        %ENDE GUI Speichern der Variablen um sie später wieder zu verweden
```

A.13 SelektionDaten.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% Diese Funktion wird von HeatmapToolbox_DatenAuswahl.m aufgerufen und
% dient der Auswahl eines Excel-Files mit Rohdaten

function [ExcelDatenInfo]= SelektionDaten()
global Daten;

% $GUI Dateiauswahl der Excel-Datei
    [filename, pathname] = uigetfile('*.xlsx', 'Excel Datei Wählen');
    if isequal(filename, 0)
    else
        DateiPfad = fullfile(pathname, filename);
    end
%/ GUI Dateiauswahl der Excel-Datei

% Excel Daten in Array abgelegen
[num, txt, raw] = xlsread(DateiPfad);
Daten = raw;

%$ Prüfe ob Daten Spalten gerade oder ungerade sind.
% Wenn die Spalten ungerade sind ist es ein erstes Indiz, dass die Daten
unvollständig sind.
```

```
ZeSp=size(Daten);
ZeilenAnzahlUngeprueft= ZeSp(1,1);
SpaltenAnzahlUngeprueft=ZeSp(1,2);
PruefeDatensaetze=rem(SpaltenAnzahlUngeprueft,2);
if PruefeDatensaetze==0
    % Daten Satz ist ok
else
    msgbox('Datsatz hat ungerade Anzahl an Splaten! Bitte Prüfen!
Abbruch');
    return
end
%/ Prüfe ob Daten Spalten gerade oder ungerade sind.


%$ Hole: Index, Name, SpalteDatum, Jahr (Metadaten)
    % Aufbau: ExcelDatenInfo=cell(5,Datensaetze) % Index, Name,
SpalteDatum, ZeileEnde, Jahr
i=1;
for n=2:2:SpaltenAnzahlUngeprueft % Schrittweite 2 weil eine Datenreihen 2
Spalten besitzt (Datum, Wert)
    ExcelDatenInfo{1,i}= i; %DatenIndex;
    ExcelDatenInfo{2,i}= Daten{1,n}; %DatenName
    ExcelDatenInfo{3,i}= n-1;%SpalteDatum
    ExcelDatenInfo{4,i}=year(datetime(Daten(2,n-1),
'InputFormat','dd.MM.yyyy HH:mm:ss'));% Jahr
    i=i+1;
end

%$ Hole Anzahl der Zeilen die Daten enthalten
i=1;
for n=0:2:SpaltenAnzahlUngeprueft-1
    ZeileLeer= Daten{ZeilenAnzahlUngeprueft,1+n};
    PruefStart = isnan(ZeileLeer);
    if PruefStart(1,1)==1
        %§§§§§§§§§§§§§§§§   %FUNCTION  LetzteZeile
        ZeilenAnzahl =  LetzteZeile(ZeilenAnzahlUngeprueft,n);
   else
        % Letzte Zeile enthält ein Datum
        ZeilenAnzahl = ZeilenAnzahlUngeprueft;
    end
ExcelDatenInfo{5,i}=ZeilenAnzahl;
i=i+1;
end
%/ Hole Anzahl der Zeilen die Daten enthalten
```

A.14 UhrzeitUmrechnen.m

```
% Programmiert von Simon Navarro für BZE Ökoplan, Hummelsbütteler Weg 36,
22339 Hamburg
% $Version: 0.1  $Date: 2017/01/20

% Funktion soll Minuten in Uhrzeit umrechnen und als String zurück geben
function Uhrzeit = UhrzeitUmrechnen(Minute)
Stunde = Minute/60;
StundeUhrzeit = floor(Stunde);
MinuteUhrzeit = rem(Stunde,1)*60;
```

```
Uhrzeit = regexprep(strjoin({num2str(StundeUhrzeit) ':'
num2str(MinuteUhrzeit) }) ,' ', '');
end
```

A.5 Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Navarro

Vorname: Simon

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Digitale Weiterverarbeitung und Analyse von Monitoringdaten im Anlagenbau mittels Generierung von Heatmap-Diagrammen

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

Hamburg	27.02.2017	S. Navarro
Ort	Datum	Unterschrift im Original