
TECHNISCHE UNIVERSITÄT BERLIN

**Fachgebiet Energieversorgungsnetz und Integration Erneuerbarer Energien**

**STUDIENARBEIT**

# Objektorientierte Implementierung und Simulation einer Kältelast mit Kältespeicher im Energieversorgungsnetz

vorgelegt von Juri STEBLAU
Matr.-Nr: 300244

19. August 2011

**Korrektoren:**

| | | | |
|---|---|---|---|
| Gutachter: | Dipl.-Ing. Felix KLEIN | - | (TU-Berlin) |
| Betreuer: | Prof.Dr.-Ing. Kai STRUNZ | - | (TU-Berlin) |

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# MATLAB®-Code Verzeichnis

# Abkürzungsverzeichnes

| | |
|---|---|
| NK | Normalkühlung |
| NK_KR_V | normalgekühltes Kühlregal an Verbundanlage |
| NK_KT_S | normalgekühlte Kühltruhe steckerfertig |
| NK_KZ | normalgekühlte Kühlzelle an Vergundanlage |
| OOP | objektorientierte Programmierung |
| TK | Tiefkühlung |
| TK_KZ | tiefgekühlte Tiefkühlzelle Kälteaggregat |
| TK_TKT_S | tiefgekühlte Tiefkühltruhe steckerfertig |

Kapitel 1

# Einleitung

Die Erforschung der Ursachen und der Folgen des Klimawandels, die wachsende Schwierigkeit bei der Bereitstellung der konventionellen Energien, die Neubewertung der Risiken und der technischen Mitteln bei der Endlagerung von Abfällen der Atomindustrie, die besorgniserregende Erkenntnis der bisherigen Fehlbewertung der Atomsicherheit treiben die Entwicklung der erneuerbaren Energien zu einer grundlegenden Energieform in Deutschland noch schneller voran.

Der Umstieg auf alternative Energien ist mit einigen grundlegenden Problemen verbunden. An einigen Orten ist der Einsatz dieser Technik aus politischer, technischer, ökonomischer oder ökologischer Sicht nicht möglich. Aus diesem Grund weichen oft die Stromerzeugung und der Strombedarf zeitlich und räumlich voneinander ab. Windkraft im Meer, Wasserkraft in den Bergen, Sonnenkraft in in den südliche Regionen, Geothermie in seismisch aktiven Gebieten. Desweiteren kann nur ein Teil der erneuerbaren Energien direkt vom Menschen beeinflusst werden. Besonders die Menge der durch Sonne und Wind gewonnenen Energie schwankt abhängig von der Wetterlage. Die Integration dieser Energie in das Netz führt zur erhöhten Bereitstellung an Regelenergie. Diese Herausforderung wird zur Zeit hauptsächlich durch übermäßige Belastung der zur Ausregelung geeigneten konventionellen thermischen Kraftwerke übernomen. Durch den regelungsbedingten ineffizienten Teillastbetrieb und wiederholte An- und Abfahrvorgänge sinkt der Wirkungsgrad und hat einen höheren Verschleißdieser Kraftwerke zu Folge.

Aus langfristiger Sicht werden Investitionen in Lastmanagement und in Energiespeicher unverzichtbar. Effiziente Lastmanagemant und Energiespeicherkonzepte können helfen diesen Problemen langfristig entgegenzuwirken.

Die Intention der Arbeit ist es, im Ramen der Bereitstellung der Regelleistung für erneuerbaren Energien durch den Einsatz der Verfahren und Techniken des objektorientierten Entwurfs ein MATLAB® -Programm zur Simulation des variablen Lastverhaltens von Kältelasten mit Kältespeichern im Energieversorgungsnetz zu entwerfen. An erster Stelle im Kapitel 2 wird der Einsatz der Suparmärkte als Kältespeischer diskutiert. Es folgt eine kurze Einführung in die objektorientierte Programmierung mit MATLAB® . Anschließend werden die physikalischen Grundlagen der Kältetechnik erklärt. Im Kapitel 3 wird der Lösungsweg vom objektorientierten Entwurf bis zum Programm-Code erklärt.

Desweiteren wird mit Hilfe der Kältelast spezifizierender Konfigurationsdateien[1] das variable Verhalten der bei Variation der Anzahl der Kältespeicher im Enerergieversorgungsnetz mit dem Programm simuliert und der Einfluss auf das Energieversorgungsnetz untersucht. Es werden ausschließlich Kälteanlagen modelliert, die in Supermärkten zum Einsatz kommen. Das Ergebnis der Simulation, der Verbrauch einer variabel geführten Supermarktkette auf der Basis der einzelnen in den Supermärkten eingesetzten Kälteanlagen, wird im Kapitel 4 vorgestellt.

[1] Der Entwurf der Konfigurationsdateien basiert auf der Spezifikation der Parameter durch Caroline Möller. Der Titel ihrer Arbeit, die ebenfalls im Fachgebiet Energieversorgungsnetze und Integration erneuerbarer Energien an der TU Berlin entstand und die Spezifikation einer Kältelast zum Inhalt hat, heißt: Spezifikation und Simulation einer Kältelast mit Kältespeicher im Energieversorgungsnetz.

Kapitel 2

# Theoretische Problembetrachtung

## Inhaltsangabe



## 2.1 Erneuerbare Energien im Energieversorgungsnetz

Die erneuerbaren Energien übernehmen in Deutschland mit jedem Jahr einen größeren Anteil an der Elektrizitätsversorgung. Mehr als die Hälfte [4] der erneuerbaren Energie wird durch Wind und Photovoltaik erzeugt. Diese Energiequellen sind jedoch in hohem Grad witterungsabhängig und können stark fluktuieren. Dadurch kann die Menge der zur Verfügung stehenden Energie von der Nachfrage zeitlich enorm abweichen. Dies kann zur Folge haben, dass der sicheren Betrieb des Stromnetzes bei einer konstanten Frequenz von 50 Hz gefährden. Zusätzliche Bereitstellung an Regelleistung wird somit notwendig. Außerdem erfolgt die Erstellung der Fahrpläne für den Einsatz der konventionellen Kraftwerke gezwungenermaßen auf den fehlerbehafteten Vortagsprognosen über die Menge an Leistung aus erneuerbaren Energien. Die Güte der Prognose bestimmt dabei direkt den Regel- und den Reservebedarf.

Eine exakte Prognose über die Leistung aus Wind- und Photovoltaikanlagen kann nur auf Grundlage exakter Wettervorhersagen erfolgen. In der Literatur wird eine mittlere Genauigkeit für Day-Ahead Windleistungsprognosen von 7 % [18] und für Leistungsprognosen aus Photovoltaikanlagen im Bereich von $3,5\,\% - 4,4\,\%$ [9] angegeben. Darüberhinaus können im Verlauf eines Tages zeitweise deutliche Abweichungen von diesen Werten auftreten. Durch den gedrosselten Betrieb der Anlagen ist eine Möglichkeit zur Kompensation der Prognosefehler gegeben. Der gedrosselte Anteil kann dadurch jedoch nicht genutzt werden. Die Bereitstellung der Regelleistung durch konventionelle oder andere Kraftwerke erscheint an dieser Stelle als sinnvoll. Eine Alternative zu den obengenannten Methoden bieten Energiespeicher,

da sie bei Überangebot Energie speichern können und bei Unterversorgung Energie in das Netz einspeisen können.

Variable Gestalltung der Netzlast kann weiterhin zur Reduzierung der obengenannten Problematik beitragen.

Dies bedeutet, dass beispielsweise Großverbraucher in Zeiten geringen Energiebedarfs zeitweilig vom Netz genommen und die dadurch freiwerdenden Kapazitäten anderwertig eingesetzt werden. Dies hätte zur Folge, dass sowohl Regelungsverluste als auch der Regelungsaufwand verringert und somit Kosten gespart werden könnten. Es ist vorstellbar, dass Anlagen zur Kälteerzeugung aufgrund der thermischen Trägheit zu günstigen Zeiten die Temperatur zusätzlich senken können und zu ungünstigen Zeiten die Kühltätigkeit auf das Minimum herunterfahren. Hiermit ist also das Interesse begründet, das Lastverlagerungspotential in der Kälteerzeugung zu untersuchen.

## 2.2 Kälteerzeugung im Supermarkt: Potentiale und Besonderheiten

Im Mittel entfallen 60 % [5,8] des Stromverbrauchs in einem Supermarkt auf das Kühlen und Tiefkühlen von Produkten. Der durchschnittliche Verbrauch in einem Supermarkt liegt zwischen 300 MWh und 500 MWh im Jahr [8]. Für die kommenden Jahren liegt die Schätzung für das Wachstum des gesamten Kältebedarfs für Deutschland bei 100 % [14]. Der Anteil des Bedarfs an elektrischer Energie vom Gesamtverbrauch eines Industrielandes für Kälteerzeugung in den Supermärkten wird in der Literatur für Australien mit 1 % [10] und in Schweden mit 2 % [2,5] angegeben. In Deutschland liegt der Verbrauch mit 6294 $GWh$ für das Jahr 1999 bei rund $1,27\,\%$ [17]. Die Größenordnung des Verbrauchs für die Produktlagerung in Supermärkten und die Möglichkeit diesen zeitlich zu verschieben, macht die Supermärkte für die Regelung besonders interessant. Jedoch unterliegt der Einsatz bestimmten Beschränkungen und Randbedingungen (siehe Abschnitt 2.4).

Die Mindesthaltbarkeit für bestimmte Produkte kann nur durch Lagerung dieser Produkte in für sie festgelegtem Kühltemperaturbereichen garantiert werden. Dieser Temperaturbereich ist je nach Bedarf und Anwendung in Normalkühlung (NK) über 0°C und in Tiefkühlung (TK) unter 0°C unterteilt. Sowohl auf der nationalen als auch auf der internationalen Ebene existieren Auflagen, die die maximale Temperaturen bei der Lagerung von Nahrungsmitteln vorschreiben. Die fundamentale Verordnung ist die EG 853/2004. Aus dieser Regelung geht hervor, dass normalgekühlte Produkte im Temperaturbereich von 0 bis +8°C je nach Nahrungsmittel und tiefgekühlte Produkte mindestens bei −18°C gekühlt werden müssen.

Die Kälteerzeugung im Supermarkt kann räumlich generell in zwei Bereiche unterteilt werden, dem Verkaufsbereich und dem Warenlager. Außerdem kommen Kühleinheiten zum Einsatz, die einer Verbundkälteanlage angehören oder steckerfer-

tig zur Verfügung stehen. Im Verkaufsbereich kommen überwiegend Kühltruhen im Tiefkühlbereich und Kühltruhen sowie Kühlregale im Niederkühlbereich zum Einsatz. Zur Ausstattung zählen Geräte in offener sowie durch Türen verschließbarer Ausführung. Größtenteils werden die offenen Modelle außerhalb der Öffnungszeiten durch Decken und Rollos zwecks Energieeinsparung verschlossen [8].

## 2.3 Objektorientierte Programmierung mit MATLAB®

Seit Ende des letzten Jahrhunderts wird in der Fachliteratur für Informatik die Ansicht geteilt, dass der Einsatz von objektorientierten Techniken Programme hervorbringt, die im Vergleich *einfacher erweiterbar, besser testbar* und *besser wartbar* sind [3]. Dabei wird ein Verfahren angewendet, nachdem große Systeme in kleinere Teile des Ganzen zerlegt werden. Programme lassen sich dadurch im Allgemeinen mit weniger Aufwand und kleinerer Fehlerwahrscheinlichkeit programmieren. Inspiriert durch die Vorgänge aus der realen Welt, werden die Abläufe durch operierende Objekte dargestellt. Auf diesem Wege können die Aufträge erledigen und vergeben werden. Die wesentlichen Eigenschaften der objektorientierten Programmierung, kurz OOP, sind die Datenkapselung, die Polymorphie und die Vererbung.[1]

### Allgemeine Erläuterungen

**Klasse** Eine Klasse ist ein Instrument der Programmierung zur Erfassung von charakteristischen Eigenschaften zusammenhängender Objekte. Die Definition der Strukturen der Objekte erfolgt durch Klassen.

**Objekt** Ein Objekt ist ein konkretes Exemplar einer Klasse.

**Eigenschaften** Eigenschaften sind Variablen, die für jedes Objekt existieren.

**Methoden** Methoden sind objektlokale Funktionen.

**Datenkapselung** Man spricht von Kapselung, wenn Objekte den Zugriff auf ihre Daten kontrollieren.

**Polymorphie** Können unterschiedliche Objekte auf eine gleiche Nachricht unterschiedlich reagieren, spricht man von Polymorphie.

**Vererbung** Die Vererbung ermöglicht durch Veränderung der bestehenden Klassen neue Klassen zu erstellen. Die grundlegenden Programmteile der bestehenden Klasse werden zwangsläufig übernomen.

[1] Ausführliche Informationen dazu findet man z.B. in [3], [12], [13] oder [19].

## Erläuterungen zur OOP-Syntax[2]

**classdef** Neue Klassen werden mit der Anweisung `classdef` eingeleitet. Der Name der Klasse wird unmittelbar nach der Anweisung eingetragen. Der Klassen-Block endet mit der Anweisung `end`.

**properties** Die Anweisung `properties` beginnt den Eigenschaften-Block. Abgeschlossen wird der Block mit der Anweisung `end`. Innerhalb einer Klasse können mehrere Blöcke existieren. Möchte man das Verhalten eines Eigenschaften-Blocks zusätzlich verändern, zum Beispiel neue Zugriffsarten oder Zugriffsrechte vergeben, ist der Einsatz von Attributen zweckmäßig.

```
properties (attribut1, attribut2, etc.)
end
```

**methods** Die Anweisung `methods` beginnt den Methoden-Block und mit `end` wird er geschlossen. Analog zu dem Eigenschaften-Block können in einer Klasse mehrere Methoden-Blöcke existieren. Soll das Verhalten eines Methoden-Blocks zusätzlich verändern, ist der Einsatz von Attributen zweckmäßig.

```
methods (attribut1, attribut2, etc.)
end
```

## Klassen und Objekte mit MATLAB®

Klassen werden in MATLAB® mit Hilfe einer Datei mit der Endung *.m* definiert, die den selben Namen wie die Klasse hat. Im Quelltextbeispiel MATLAB®-Code 2.1 wird eine Beispielklasse **Class_name** mit einem Eigenschaften-Block und einem Methoden-Block definiert. Die Datei muss demnach **Class_name.m** heißen. In die erste Zeile der Datei kommt die Anweisung `classdef`, die die Klasse einleitet. Werden der Klasse zusätzliche Verhaltensmuster zugewiesen, folgt die Setzung der Attribute in Klammern. Anschließend kommt der Name der Klasse. Wird die neue Klasse eine durch Vererbung abgeleitete einer anderen Klasse, so folgt das Kleinerzeichen und der Name der Superklasse. In den anschließenden Zeilen werden der Eigenschaften-Block und der Methoden-Block definiert.

Objekte einer Klasse werden durch eine Konstruktor-Methode erzeugt. Diese Funktion wird im ersten Methoden-Block an erster Stelle definiert. Sie hat den selben Namen wie die Klasse. Über die Konstruktor-Funktion können bereits bei der Erzeugung der Objekte Werte an ausgewählte Attribute übergeben werden. Erfolgt

[2] Es werden nur Schlüsselwörter vorgestellt, die essentiell sind. Eine tiefgreifende Darstellung würde den Rahmen einer Studienarbeit bei Weitem übersteigen. Ausführliche Informationen dazu findet man z.B. in [16] oder http://www.mathworks.com/help/.

MATLAB®-Code 2.1: Beispiel Klassendefinition

```
classdef (Attributes) Class_name < Super_class % class definition
    properties (Attributes) % first property block
        PropertyName = [];
    end % end of properties block
    % additionally here can be another properties block with
    % specifying by another Attributes
    methods (Attributes)
        function obj = Class_name(obj,a) % consturctor
            obj.PropertyName = a;
        end
        function [X] = second_function(obj)
            X = obj.PropertyName + 1;
        end
    end
    % additionally here can be another methods block with
    % specifying by another Attributes
end % end of classdef block
```

der Aufruf einer Konstruktor-Funktion einer Klasse fehlerfrei, so ist das Ergebnis des Aufrufes ein Objekt dieser Klasse. Der Aufruf eines Konstruktors unterscheidet sich nicht von dem Aufruf gewöhnlicher MATLAB®-Funktionen. Im folgenden Quelltextbeispiel wird im Command-Window durch den Aufruf der Konstruktor-Funktion ein Objekt **object** der im MATLAB®-Code 2.1 vorgestellten Klasse erzeugt.

```
>> object = Class_name(input_data);
```

Methoden, die applikationsspezifische Operationen mit einem Datensatz durchführen sollen, werden nach der Konstruktor-Funktion definiert. Die meisten dieser Funktionen nutzen das Objekt als Eingabeargument, zum Beispiel: `second_function(obj)`. Der Zugriff auf die Variablen der Objekteigenschaften erfolgt durch Referenzierung auf das Objekt. Im Quelltextbeispiel MATLAB®-Code 2.1 wird das wie folgt vorgestellt: `obj.PropertyName`. Wenn ein Zugriff auf Objektfunktionen erlaubt ist, kann darauf im folgenden Ablauf zugegriffen werden:

```
>> function_return = object.second_function;
```

Die Referenz auf den Namen des Objekts muss dabei stets erfolgen.

## 2.4 Einführung in die Kältetechnik

An dieser Stelle wird ein zusammenfassender Überblick über die mathematischen und physikalischen Zusammenhänge gegeben, die bei der Entwicklung eines Modells

einer Kälteanlage in einem Modellsupermarkt zwingend beachtet werden müssen.

In den Supermärkten werden Kühlgeräte zur Lagerung der Ware bis zum Verkauf an den Endkunden eingesetzt. Um die Haltbarkeit dieser Ware für den Mindestzeitraum zu gewährleisten, wird diese bei niedrigen Temperaturen gehalten. Körper mit unterschiedlicher Temperatur sind bestrebt, wenn sie thermisch von einander nicht vollkommen isoliert sind, durch gegenseitige Wechselwirkung ihre Temperaturen anzugleichen. Infolge dessen wird ein Wärmegleichgewicht erreicht. Der natürliche Wärmefluss findet selbstständig immer vom Körper mit der höheren Temperatur in die Richtung des Körpers mit der niedrigeren Temperatur statt. Um eine negative Temperaturänderung herzustellen und diese auch zu halten, muss die eindringende Wärmeenergie ständig in derselben Höhe abgeführt werden, damit die Temperatur konstant bleibt. Diese Energiemenge pro Zeiteinheit wird als Kälteleistung bezeichnet. Eine Abweichung von dieser Menge führt zum Steigen der Temperatur, wenn weniger und zum Sinken der Temperatur wenn mehr abgeführt wird. Um diesen Kühlkreislauf aufrecht zu erhalten, muss Leistung aufgewendet werden[3].

In der Abbildung 2.1 auf der Seite 9 ist die Prinzipdarstellung der weitverbreiteten Kompressionskältemaschine dargestellt. Die Ware, die symbolisch durch einen Kohlkopf dargestellt ist, wird mit einer Temperatur zwischen ±0 und +8°C gelagert. Der Temperaturunterschied zwischen dem Innen- und Außenbereich ist maßgebend für den Verlust an Kälte (Wärmeausgleich). Der Ausgleich führt dazu, dass Kälteenergie für den Lagerungs- und Kühlungsprozess verloren geht.

Die Aufnahme der Wärmeenergie und der spezifischen Wärmekapazität dieser Substanzmasse ergibt die Temperaturdifferenz $\Delta t$.

$$\Delta t = \frac{Q}{m \cdot c} \tag{2.1}$$

| | |
|---|---|
| $\Delta t$ | Temperaturdifferenz in Kelvin K |
| $Q$ | Verlust an Kälte in in kJ |
| $m$ | Substanzmasse zur Aufnahme der Kälteenergie in kg |
| $c$ | Spezifische Kältekapazität (Wärmekapazität) der Substanzmasse in $\frac{\text{kJ}}{\text{kg}\cdot\text{K}}$ |

Die installierte Kälteleistung multipliziert mit der täglichen Betriebszeit muss gleich dem stündlichen Kältebedarf multipliziert mit der Tagesstundenzahl sein.

$$\dot{Q}_0^{)} = \frac{24}{\tau_B} \cdot \dot{Q}_0 \tag{2.2}$$

[3]Eine detailierte Beschreibung dieser Prozesse in einer Kompressionskälteanlage und Spezifikation ist nicht Gegenstand dieser Arbeit. Ausführliche Informationen dazu findet man z.B. in [2,11,15].

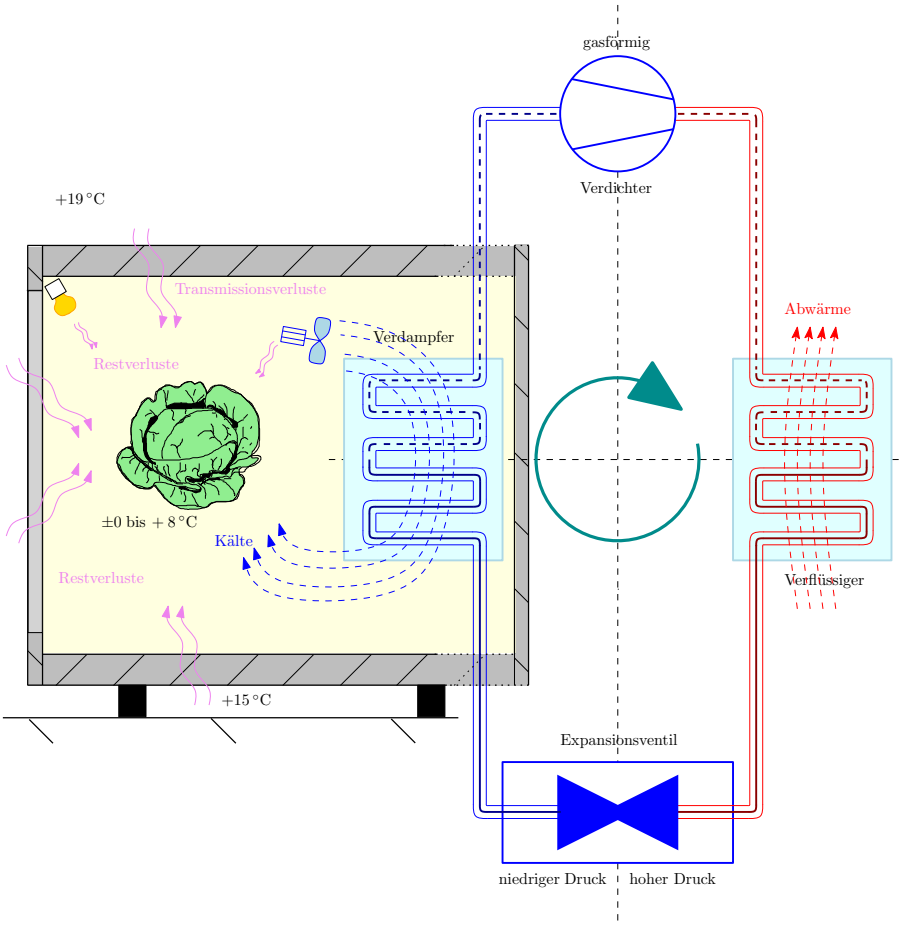


Abbildung 2.1: Prinzipdarstellung Kompressionskälteanlage

| | |
|---|---|
| $\dot{Q}_0^{)}$ | Installierte Kälteleistung in kW |
| $\tau_B$ | Tägliche Betriebszeit in h |
| $\dot{Q}_0$ | Stündlicher Kälteleistungsbedarf in kW |

Die Transmissionswärmeleistung wird aus der Multiplikation der Fläche der wärmeübertragenden Wände mit ihrem spezifischen Wärmedurchgangskoeffizient und der Temperaturdifferenz zwischen der Kühlraumtemperatur und der Umgebungstemperatur berechnet.

$$\dot{Q}_{Tr} = A \cdot k \cdot \Delta t \tag{2.3}$$

| | |
|---|---|
| $\dot{Q}_{Tr}$ | Transmissionswärmeleistung in kW |
| $A$ | Fläche in $m^2$ |

$k$ Wärmedurchgangskoeffizient in $\frac{\mathrm{W}}{\mathrm{m}^2 \cdot \mathrm{K}}$

$\Delta\, t$ Temperaturdifferenz in K

Der Zusammenhang zwischen der aufgewendeten elektrischen Antriebsleistung $P$ eines Verdichters in einer Kompressionskälteanlage und der genutzten stündlichen Kälteleistung $\dot{Q}_0$ wird durch die Kältezahl $\varepsilon$ wiedergegeben.

$$\varepsilon = \frac{\dot{Q}_0}{P} \tag{2.4}$$

$\varepsilon$ Leistungszahl (Einheitenlos)

$\dot{Q}_0$ Stündlicher Kälteleistungsbedarf in kW

$P$ Elektrische Verdichterantriebsleistung in einer Kompressionskälteanlage in kW

Die Öffnungszeit des Supermarkts hat einen spürbaren Einfluss auf die Größe der Kälteverluste. In der Literatur wird der Nachtverbrauch mit 10 % bis 20 % des Tagesverbrauchs angegeben [6]. In der Nacht fallen keine zusätzlichen Verluste an, zum Beispiel durch Licht, Körperwärme oder Türöffnungszeiten, sodass nur Transmissionsverluste bei der Berechnung beachtet werden. Leistungsbedarf definiert die Menge an Leistung, der benötigt wird, um den Verlust an Kälte zu ersetzen. Die Abweichung zwischen Tagesbedarf und Nachtbedarf an Leistung wird im weiteren Verlauf Tagesmehrbedarf genannt.

$$\dot{Q}_{Nacht} = \dot{Q}_{Tr} \tag{2.5}$$

$\dot{Q}_{Nacht}$ Leistungsbedarf in der Nacht in kW

$\dot{Q}_{Tr}$ Transmissionswärmeleistung in kW

Der Kälteleistungsbedarf am Tag ergibt sich aus der Summe des Tagesmehrbedarfs und der Transmissionswärmeleistung. Aus Gründen der Vereinfachung wird der Tagesmehrbedarf als weitgehend konstant angenommen.

$$\dot{Q}_{Tag} = \dot{Q}_{mehr} + \dot{Q}_{Tr} \tag{2.6}$$

$\dot{Q}_{Tag}$ Leistungsbedarf am Tag in kW

$\dot{Q}_{mehr}$ Tagesmehrbedarf am Tag in kW

$\dot{Q}_{Tr}$ Transmissionswärmeleistung in kW

Die Verlustkältememenge im Kühlbereich ist aufgrund der obengenannten Gründe je nach Tageszeit unterschiedlich.

$$Q_v = \begin{cases} 1h \cdot \dot{Q}_{Nacht}, & \text{Außerhalb der Öffnungszeiten} \\ 1h \cdot \dot{Q}_{Tag}, & \text{Innerhalb der Öffnungszeiten} \end{cases} \tag{2.7}$$

$Q_v$ Verlustkältemenge in kJ

$\dot{Q}_{Tag}$ Leistungsbedarf am Tag in kW

$\dot{Q}_{Nacht}$ Leistungsbedarf in der Nacht in kW

$h$ Zeiteinheit für Stunden

Ist für eine Kälteanlage der stündlicher Kälteleistungsbedarf bekannt, so wird der Tagesmehrbedarf an Kälteleistung ermittelt, indem vom Produkt des Kälteleistungsbedarfs mit dem Faktor für Kältebedarfsabsenkung die mittlere Transmissionswärmeleistung abgezogen wird.

$$\dot{Q}_{mehr} = \dot{Q}_0 \cdot K - \overline{\dot{Q}}_{Tr} \tag{2.8}$$

$\dot{Q}_{mehr}$ Tagesmehrbedarf in kW

$\dot{Q}_0$ Stündlicher Kälteleistungsbedarf in kW

$K$ Faktor für Kältebedarfsabsenkung (Einheitenlos)

$\overline{\dot{Q}}_{Tr}$ Mittlere Transmissionswärmeleistung in kW

Die Berechnung der mittleren Transmissionswärmeleistung erfolgt durch die Multiplikation der Differenz zwischen der Umgebungstemperatur und der mittleren Kühlraumtemperatur mit dem Wärmedurchgangskoeffizienten und der Fläche der wärmeübertragenden Wände.

$$\overline{\dot{Q}}_{Tr} = A \cdot k \cdot \left(t_{amb} - \overline{t}_{KR}\right) \tag{2.9}$$

$\overline{\dot{Q}}_{Tr}$ Mittlere Transmissionswärmeleistung in kW

$A$ Fläche in $\mathrm{m}^2$

$k$ Wärmedurchgangskoeffizient in $\frac{\mathrm{W}}{\mathrm{m}^2 \cdot \mathrm{K}}$

$t_{amb}$ Umgebungstemperatur in ° C

$\bar{t}_{KR}$ Mittlere Kühlraumtemperatur in ° C

Ist der Kälteleistungsbedarf nicht bekannt, wie zum Beispiel bei steckerfertigen Geräten, kann der Mehrbedarf am Tag über den Wert Verdichterarbeit pro 24 Stunden $W_{Verd_{24}}$ ermittelt werden.

Das Produkt aus dem spezifischen Energieverbrauch mit dem Faktor für Kältebedarfsabsenkung, dem Verdichteranteil ergibt die Verdichterarbeit.

$$v_{mod} = K \cdot v \tag{2.10}$$

$$W_{Verd_{24}} = W_{spez_{24}} \cdot v_{mod} \tag{2.11}$$

$W_{Verd_{24}}$ Verdichterarbeit pro 24 Stunden in $\frac{\text{kWh}}{\text{24h}}$

$W_{spez_{24}}$ Spezifischer Energieverbrauch pro 24 Stunden in $\frac{\text{kWh}}{\text{24h}}$

$K$ Faktor für Kältebedarfsabsenkung (Einheitenlos)

$v$ Verdichteranteil (Einheitenlos)

$v_{mod}$ Modifizierter Verdichteranteil (Einheitenlos)

Im Folgenden muss die Verdichterarbeit zum Ersetzen der zusätzlichen Kälteenergieverlustmenge in der Öffnungszeit berechnet werden, die in einem weiteren Schritt in Tagesmehrbedarf $\dot{Q}_{mehr}$ umgewandelt wird.

$$W_{mehr} = W_{Verd_{24}} - \frac{\overline{\dot{Q}}_{Tr}}{\varepsilon} \cdot 24 \tag{2.12}$$

$W_{mehr}$ Verdichterarbeit zum Ersetzen der zusätzlichen Kälteenergieverluste $Q_{mehr}$ in der Öffnungszeit in kWh

$W_{Verd_{24}}$ Verdichterarbeit pro 24 Stunden in $\frac{\text{kWh}}{\text{24h}}$

$\overline{\dot{Q}}_{Tr}$ Mittlere Transmissionswärmeleistung in kW

$\varepsilon$ Leistungszahl (Einheitenlos)

Die Umrechnung von $W_{mehr}$ in $\dot{Q}_{mehr}$ erfolgt mit Hilfe der Leistungszahl $\varepsilon$. Die Verdichterarbeit $W_{mehr}$ wird mit $\varepsilon$ multipliziert. Es wird angenommen, dass der Mehrbedarf in den rund zwölf Stunden der Öffnungszeit entsteht.

$$\dot{Q}_{mehr} = \frac{W_{mehr}}{12} \cdot \varepsilon \tag{2.13}$$

$\dot{Q}_{mehr}$ Tagesmehrbedarf in kW

$W_{mehr}$ Verdichterarbeit zum Ersetzen der zusätzlichen Kälteenergieverluste $Q_{mehr}$ in der Öffnungszeit in kWh

$\varepsilon$ Leistungszahl (Einheitenlos)

Nun kann der stündliche Leistungsbedarf des Supermarkts in MW berechnet werden.

$$P_{dem} = \frac{\dot{Q}_v}{\varepsilon} \tag{2.14}$$

$P_{dem}$ Stündlicher Leistungsbedarf des Supermarkts (demand) MW

$\dot{Q}_v$ Verlustkälteleistung in kW

$\varepsilon$ Leistungszahl (Einheitenlos)

Kapitel 3

# Modellierung

## Inhaltsangabe



Folgende explizite Anforderungen sind an das Programm gestellt:

- Der Entwurf des Programms erfolgt mit Hilfe der objektorientierter Verfahren und Techniken.
- Die Speicherung der für das Beschreiben der Modelle erforderlichen Parameter findet in einer Konfigurationsdatei statt.
- Der elektrische Energieverbrauch wird aufgrund der Modellparameter der Kältelasten sowie des gefahrenen Einsatzes der Kältelasten berechnet.
- Die Topologie des elektrischen Netzwerkes wird berücksichtigt.
- Die Möglichkeit einer eindeutigen Zuweisung der Verbrauchswerte für folgende Verursacher wird realisiert:
  - Die berechneten Werte für jede Kälteanlage werden einzeln gespeichert, wenn mehrere Anlagen simuliert werden.
  - Die berechneten Werte für jede Supermarktkette werden einzeln gespeichert, wenn der Verbrauch mehrerer Supermarktketten simuliert wird.
- Eine Möglichkeit zur graphischen Auswertung der Berechnung wird erstellt.

## 3.1 Erstellung des OOP-Modells

Der durchschnittliche Energieverbrauch der Kälteanlagen je Supermaktkette kann aufgrund der technischen Ausführung unterschiedlich ausfallen. Der Energieverbrauch entsteht an definierten Punkten im Netz. An den einzelnen Knotenpunkten können mehrere Kältelasten angeschlossen sein.

In der Abbildung 3.1 wird ein einfaches Energieversorgungsnetz mit vier Knoten dargestellt. Am Knoten eins ist eine regenerative elektrische Energiequelle, in diesem Fall ein Windpark, angeschlossen. Weitere konventionelle elektrische Energiequellen befinden sich an den Knoten zwei und drei. Die passiven Lasten befinden sich am Knoten zwei und vier. Der Kältespeicher ist am Knoten zwei angeschlossen. Im Bild wird der Kältespeicher Supermarkt durch ein Gebäude mit Einkaufswagen dargestellt. Die Knoten sind untereinander durch Leitungen verbunden.

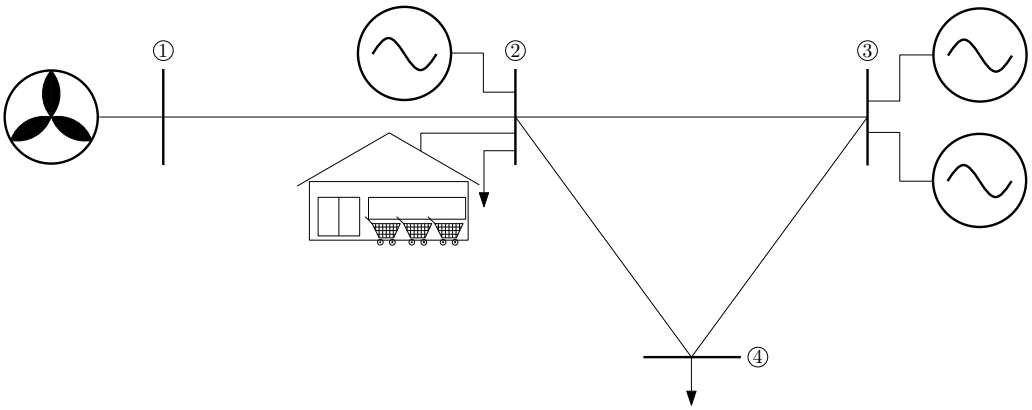


Abbildung 3.1: Vier Knoten Beispiel

In der Realität kann ein Energieversorgungsnetz durch die Variation der Knotenzahl und die Vermaschung eine beliebig komplizierte Form aufweisen.

In Anbetracht der im Abschnitt 2.3 vorgestellten Verfahren erscheint der Ansatz der OOP bei der Erstellung eines Programms zur Simulation einer Kälteslast mit Kältespeicher im Energieversorgungsnetz auf der Basis des vorangegangenen Beispiels (Abbildung 3.1 auf der Seite 15) als sinnvoll. In der Abbildung 3.2 auf der Seite 16 wird in Form eines Klassendiagramms das fertige Klassen-Modell dargestellt und die Abhängigkeit unter Klassen visualisiert.

Das Ergebnis der Abstraktion sind vier Klassen. In der Klasse **Refrigerator** sind Eigenschaften und Methoden zusammengefasst, die das Modell einer Kältelast beschreiben. Darum wird nachfolgend die Klasse **Refirgerator** explizit erläutert.

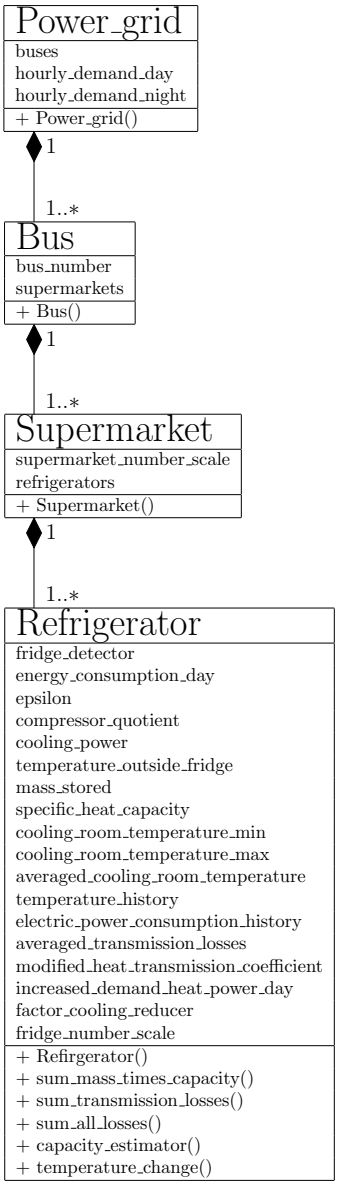


Abbildung 3.2: Klassendiagramm Modellkonstrukt

## Refrigerator

An dieser Stelle werden die Eigenschaften und die Methoden der Klasse **Refrigerator** explizit erklärt.

### Attribute

**fridge_detector** Art der Anlage. Die Berechnung der Verluste der Verbundanlagen unterscheidet sich von der der steckerfertigen Geräte.

**energy_consumption_day** Höhe des durchschnittlichen elektrischen Energieverbrauchs.

**epsilon** Leistungszahl.

**compressor_quotient** Anteil des Verdichters am Gesamtverbrauch der Kühlanlage.

**cooling_power** Kälteleistungsbedarf (Herstellerangaben).

**temperatur_outside_fridge** $1 \times M$-Array mit Außentemperaturwertet für die Umgebung der Kälteanlage. $M$ stimmt mit der Anzahl der Wände überein, für die ihre Fläche bekannt ist.

**mass_stored** $1 \times N$-Array mit den Massewerten, der zur Kühlung vorhandenen Substanzmasse. $N$ stimmt mit der Anzahl der Massen überein, für die die spezifische Wärmekapazität unterschiedlich ist.

**specific_heat_capacity** $1 \times N$-Array mit den Werten für spezifische Wärmekapazitäten der Substanzmassen.

**cooling_room_temperature_min** Niedrigste Temperatur, die durch Kühlung erreicht werden darf.

**cooling_room_temperature_max** Höchste Temperatur, die durch Kühlung erreicht werden darf.

**averaged_cooling_room_temperature** Durchschnittliche Temperatur im Normalbetrieb.

**temperature_history** $n \times m$-Array mit $n, m \in \mathbb{Z}_0^+$, wobei $n$ für die Anzahl der Werte des berechneten Temperaturverlaufs und $m$ für die Anzahl der simulierten Tage steht. Die Werte sind das Ergebnis der Berechnung der Funktion **main_equation**. Eine Spalte des Arrays stellt den Temperaturverlauf an genau einem Tag dar.

**electric_power_consumption_history** $n \times m$-Array mit $n, m \in \mathbb{Z}_0^+$, wobei $n$ für die Anzahl der Werte des berechneten Verbrauchs und $m$ für die Anzahl der simulierten Tage steht. Die Werte sind das Ergebnis der Berechnung der Funktion **main_equation**. Eine Spalte des Arrays stellt den Verlauf des berechneten Verbrauchs an genau einem Tag dar.

**averaged_transmission_losses** Summe aller durchschnittlichen Transmissionsverluste im Normalbetrieb (vgl. Gleichung (2.9) auf der Seite 11).

**modified_heat_transmission_coefficient** $1 \times M$-Array mit Werten aus der elementeweisen Multiplikation des Vektors mit den Werten der spezifischen Wärmedurchgangskoeffizienten mit dem Vektor mit Werten für die Durchgangsflächen (vgl. Gleichung (2.10) auf der Seite 12).

**increased_demand_heat_power_day** Mehrbedarf am Tag aufgrund der zusätzlichen Verluste durch Licht, Menschenkörperwärme, Türöffnungszeiten und weiteren Einflüssen. Die Berechnung erfolgt nach einer Auswahl zwischen der Gleichung (2.8) und der Gleichung (2.11) bis Gleichung (2.13) wie auf den Seiten 11 und 12 beschrieben.

**factor_cooling_reducer** Faktor für Kältebedarfsabsenkung infolge der Luftfeuchtigkeit und der Umgebungstemperatur.

**fridge_number_scale** Anzahl der Anlagen mit der identischen technischen Ausführung, der Beladung und dem Fahrplan.

### Methoden

**Refrigerator(fridge_config,number_steps,days)** Konstruktor der Klasse. Bei dem Aufruf dieser Funktion werden folgende Argumente übergeben:

**fridge_config** $1 \times 2$-Cell Array mit Modellparametern[1].

**number_steps** Anzahl der Zeitschritte, die einen Tag beschreiben.

**days** Anzahl der zu simulierenden Tage.

Bei der Initialisierung eines Objekts werden einem Teil der oben vorgestellten Attribute der Klasse **Refrigerator** Werte aus dem **fridge_config** -Array direkt übergeben. Andere Werte werden einmalig von der Konstruktor-Methode berechnet oder im Laufe der Simulation verändert.

Folgende Attribute werden einmalig bei der Initialisierung berechnet:

**averaged_transmission_losses** Summe aller durchschnittlichen Transmissionsverluste im Normalbetrieb (vgl. Gleichung (2.9) auf der Seite 11).

**modified_heat_transmission_coefficient** $1 \times M$-Array mit Werten aus der elementeweisen Multiplikation des Vektors mit Werten der spezifischen Wärmedurchgangskoeffizienten mit dem Vektor mit den Werten für die Durchgangsflächen (vgl. Gleichung (2.10) auf der Seite 12).

[1] Die Form des Arrays ist äquivalent zu genau einer Zeile der in MATLAB®-Code 3.2 auf der Seite 33 vorgestellten Arrays, die die Modelle der Supermärkte abbilden.

**increased_demand_heat_power_day** Mehrbedarf am Tag , aufgrund der zusätzlichen Verluste durch Licht, Menschenkörperwärme, Türöffnungszeiten und weiteren Einflüsse. Die Berechnung erfolgt nach einer Auswahl zwischen der Gleichung (2.8) und der Gleichung (2.11) bis Gleichung (2.13) wie auf der Seiten 11 und 12 beschrieben.

Folgende Attribute werden im Laufe der Simulation verändert:

**temperature_history** $n \times m$-Array mit $n, m \in \mathbb{Z}_0^+$, wobei $n$ für die Anzahl der Werte des berechneten Temperaturverlaufs und $m$ für die Anzahl der simulierten Tage steht. Die Werte sind das Ergebnis der Berechnung der Funktion **main_equation**. Eine Spalte des Arrays stellt den Temperaturverlauf an genau einem Tag dar.

**electric_power_consumption_history** $n \times m$-Array mit $n, m \in \mathbb{Z}_0^+$, wobei $n$ für die Anzahl der Werte des berechneten Verbrauchs und $m$ für die Anzahl der simulierten Tage steht. Die Werte sind das Ergebnis der Berechnung der Funktion **main_equation**. Eine Spalte des Arrays stellt den Verlauf des berechneten Verbrauchs an genau einem Tag dar.

Folgende Attribute sind aufgrund der Annahmen dieser Studie konstant, können aber bei der einer Modifizierung der Annahmen im Laufe der Simulation verändert werden:

**temperature_outside_fridge** Befindet sich der Kältespeicher außerhalb geschlossener Räume und im direkten Kontakt mit der Außentemperatur, muss die Temperaturänderung die im Tagesverlauf in Abhängigkeit von der Jahreszeit entsteht, beachtet werden.

**mass_stored** Wenn nicht mehr Angenommen wird, dass die Substansmasse im Supermarkt im Laufe des Tages konstant ist, sondern mit dem Konsumverhalten der Bevölkerung in Abhängigkeit steht, muss eine Anpassung erfolgen.

**sum_mass_times_capacity()** Der Rückgabewert dieser Funktion ist die berechnete Summe aus der elementeweisen Multiplikation aller Substanzmassen mit den jeweiligen spezifischen Wärmekapazitäten. Der Funktion werden keine explizite Argumente übergeben. Die Funktion greift direkt auf die Attribute **specific_heat_capacity** und **mass_stored** des Objekts zu.

**sum_transmission_losses()** Es gibt zwei Rückgabewerte dieser Funktion. Der erste Wert ist die Summe der Transmissionsverluste (vgl. Gleichung (2.9) auf der Seite 9) durch alle Durchgangsflächen für den aktuellen Zeitpunkt und für die aktuelle Kühlraumtemperatur. Desweiteren wird die Summe der Transimissionsverluste berechnet, die bei der oberen Grenze des zugelassenen Temperaturbereichs entsteht. Die Kenntnis hinsichtlich dieser Verluste ist wichtig, um

die Zeitspanne abschätzen zu können bis die obere Temperaturgrenze erreicht wird (vgl. Gleichung (3.5) auf der Seite 24).

Die Übergabe folgender Argumente an die Funktion ist erforderlich:

**number_steps** Laufvariable. Gesamtzahl der zu simulierenden Zeitschritte für einen Tag.

**count_number_steps** Der aktuelle Wert von **number_steps**.

**count_number_day** Der aktuelle Wert von der Laufvariable **number_days**.

**sum_all_losses()** Es gibt zwei Rückgabewerte dieser Funktion. Der erste Rückgabewert beinhaltet die Summe aller Verluste an Kälte (vgl. Gleichung (2.7) auf der Seite 11). Außerhalb der Öffnungszeiten entfällt der Anteil an zusätzlichen Verlusten, die durch Licht, offene Türen etc. anfallen. Durch eine alternative Verzweigung, deren Bedingung die Zugehörigkeit des aktuellen Wertes der Laufvariable **count_number_steps** zu den Öffnungszeiten ist, wird die Berücksichtigung der zusätzlichen Verluste sichergestellt. Der zweite Rückgabewert ist die Zeitspanne bis zum Erreichen der kritischen maximalen Temperatur (vgl. Gleichung (3.5) auf der Seite 24).

Die Übergabe folgender Argumente an die Funktion ist erforderlich:

**number_steps** Laufvariable. Gesamtzahl der zu simulierenden Zeitschritte für einen Tag.

**count_number_steps** Der aktuelle Wert von **number_steps**.

**count_number_day** Der aktuelle Wert von der Laufvariable **number_days**.

**number_days** Laufvariable. Gesamtzahl der zu simulierenden Tage.

**capacity_estimator()** Die Funktion hat einen Rückgabewert. Berechnet wird die maximal abzuführende Wärmeenergiemenge $Q_{max}$ (vgl. die Gleichung (3.11) auf der Seite 30).

Die Übergabe folgender Argumente an die Funktion ist erforderlich:

**count_number_steps** Der aktuelle Wert von **number_steps**.

**count_number_day** Der aktuelle Wert von der Laufvariable **number_days**.

**Q_losses** Wert der aktuellen Verlustkältemenge als Ergebnis der Funktion **sum_all_losses()** (vgl. Gleichung (2.7) auf der Seite 11).

**temperature_change()** Die Funktion hat einen Rückgabewert. Berechnet wird die Temperatur der Substanzmasse, die durch Kühlung am Ende eines Zeitschritts erreicht wird (vgl. die Gleichung (3.9) auf der Seite 26).

Die Übergabe folgender Argumente an die Funktion ist erforderlich:

**number_steps** Laufvariable. Gesamtzahl der zu simulierenden Zeitschritte für einen Tag.

**count_number_steps** Der aktuelle Wert von **number_steps**.

**count_number_day** Der aktuelle Wert von der Laufvariable **number_days**.

**Q_losses** Wert der aktuellen Verlustkältemenge als Ergebnis der Funktion **sum_all_losses()** (vgl. Gleichung (2.7) auf der Seite 11).

Alle Klassen haben einen unterschiedlichen Status. Die Klasse **Power_grid** vertritt die Rolle des ganzen Systems, also des Energieversorgungsnetzes, welches sich aus den charakteristischen Teilen, den Knoten (**Bus**) zusammensetzt. Die Existenz eines Knotens außerhalb eines Netzes ergibt keinen Sinn. Dieser Zusammenhang zwischen Klassen wird im Klassendiagramm in der Abbildung 3.2 durch eine Linienverbindung/Komposition verdeutlicht. Auf der Seite des Ganzen endet die Linie mit einem ausgefüllten Rhombus. Das komplette Modell ist nach diesem Prinzip aufgestellt. Das kleinste Element des Ganzen stellt die Kälteanlage (**Refrigerator**) dar.

Aufgrund des Modells wird das elektrische Verhalten einer Supermarkt-Kältelast im Energieversorgungsnetz durch miteinander kooperierende Objekte dargestellt. Wird eine Funktion aufgerufen, die den Start der Simulation veranlasst, müssen die Objekte aus der Konfigurationsdatei erzeugt werden.

In der Abbildung 3.3 auf der Seite 22 wird mit Hilfe eines Sequenzdiagramms der Nachrichtenaustausch hervorgehoben, der bei der Erstellung der Objekte durchlaufen wird. Auf der Zeitachse, die senkrecht von oben nach unten verläuft, wird mit Hilfe der Pfeile der Zugriff auf die Methoden der Objekte hingewiesen. Ist eine Methode des Objekts für einen Zeitraum aktiv, wird dieser Zustand durch einen grauen Balken auf dem Zeitstrahl des zugehörigen Objekts visualisiert. Der Aufruf der Funktionen der Objekte wird durch einen Pfeil mit ausgefüllter Spitze dargestellt. Der Pfeil geht vom Zeitstrahl eines Objekts aus, der die Funktion aufruft und endet mit der Spitze im Objekt, zu dem diese Funktion gehört. Der Name der aufgerufenen Funktion steht über dem Pfeil. Wird von dieser Funktion etwas zurückgegeben, so ist der Pfeil gestrichelt. Die Rückgabewerte stehen über dem Pfeil.

In der Funktion **runCooling**, die den Start der Simulation veranlasst und steuert, wird der Konstruktor der Klasse **Power_grid** aufgerufen. Als Ergebnis des Aufrufs bekommt die Funktion **runCooling** ein Objekt der Klasse **Power_grid** mit allen Unterobjekten, die den zu simulierenden Fall beschreiben. Die Konstruktor-Funktionen der Klassen, die einen untergeordneten Status besitzen, werden bei der

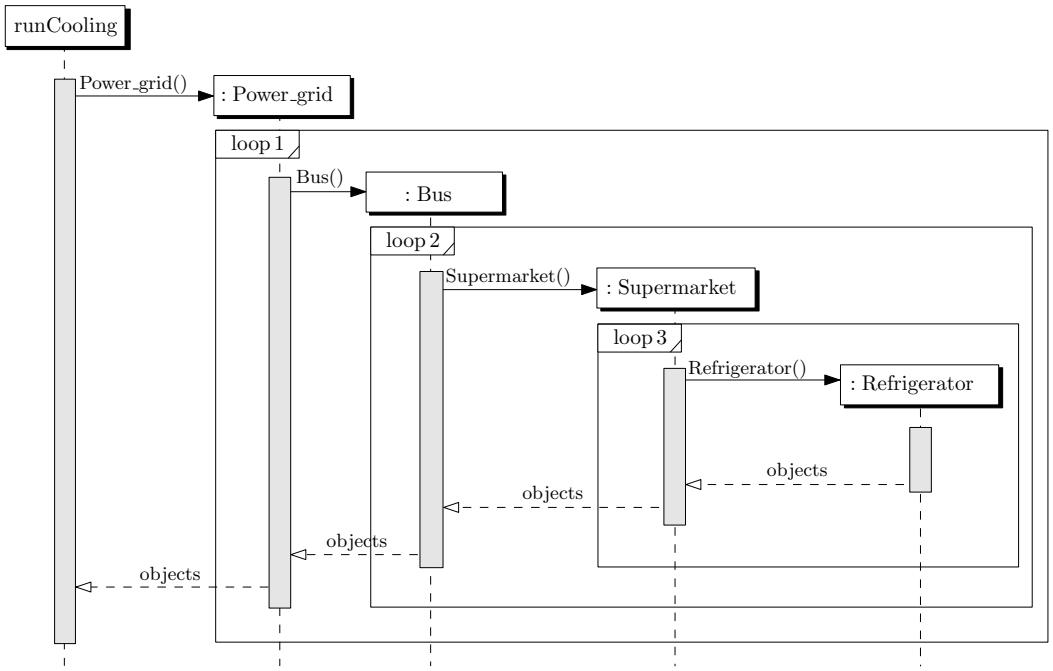


Abbildung 3.3: Sequenzdiagramm Modellkonstrukt

Erstellung der übergeordneten Objekte aufgerufen. Die Anzahl der erzeugten Objekte hängt davon ab, wie oft die Konstruktor-Funktion der jeweiligen Klasse aufgerufen wird. Diese Zahlen werden in Konfigurationsdateien abhängig von dem untersuchten Fall festgelegt. Die ausführliche Erklärung der Konfigurationsdateien findet im Abschnitt 3.2 statt.

## Bereitstellung der Leistung zur Kälteerzeugung

Das augenblickliche Gleichgewicht zwischen dem Kältebedarf und der zur Verfügung stehenden Kälteleistung sichert die Einhaltung der Soll-Temperatur der Kühlgüter. Werden die Kälteanlagen als Energiespeicher zur Regelung eingesetzt, so führt das, aufgrund der fluktuierenden zur Verfügung stehenden Kälteleistung, zu Temperaturschwankungen. Die engen Grenzen des zulässigen Temperaturbandes dürfen nicht überschritten werden.

Verschiedene Konzepte für den Einsatz der Kälteanlagen als Energiespeicher im Energieversorgungsnetz sind denkbar.

Im folgenden Abschnitt wird die Implementierung eines möglichen Kooperationskonzepts zwischen einer Supermarktkette und einem Windpark vorgestellt.

Die prognostizierten und die tatsächlich vom Windpark produzierten elektrischen Windleistungsdaten der Vattenfall-Regelzone (heute 50Hertz-Regelzone) aus dem Jahr 2005 werden als Grundlage-Datensatz verwendet. Im Datensatz sind Stundenwerte von Windleistung gespeichert, weshalb in diesem Fall auch von Energie

gesprochen werden kann.

Der Windpark ist bestrebt, die Verpflichtung über die Bereitstellung von Leistung beim Netzbetreiber zu erfüllen. Abweichungen, die aufgrund der fehlerhaften Windprognosen entstehen, sollen durch die Kooperation mit Supermarktketten ausgeglichen werden.

Es wird angenommen, dass der Prognosefehler von der Windleistungsvorhersage für 24 Stunden den 15 % entspricht. Der Windparkbetreiber meldet beim Netzbetreiber die untere Grenze an, also 85 % von der Windleistungsvorhersage. Der Supermarkt weist einen stündlichen Bedarf auf, der unbedingt gedeckt werden muss. Ist der stündliche Bedarf größer als 15 % der Prognose, wird die Differenz aus dem Netz gezogen. Diese Abweichung wird beim Netzbetreiber zur Erstellung des Lastfahrplans mitgeteilt. Gibt es eine Abweichung von der Prognose, so versucht der Supermarkt diese zu absorbieren.

Die vom Windparkbetreiber dem Supermarkt versprochene Leistung wird berechnet, indem von der Windleistungsvorhersage die 15 % bestimmt werden oder nur der stündliche Bedarf, wenn er kleiner als die 15 % ist.

$$P_{sm} = \min\{0,15 \cdot P_{DA},\ P_{dem}\} \tag{3.1}$$

$P_{sm}$ Leistung, die vom Windparkbetreiber dem Supermarkt versprochen wird in MW

$P_{DA}$ Vorhersage der Windleistung (day ahead) in MW

$P_{dem}$ Stündlicher Leistungsbedarf des Supermarkts (demand) MW

Die vom Windparkbetreiber beim Netzbetreiber anzumeldende Leistung wird berechnet, indem von der Windleistungsvorhersage die 85 % bestimmt werden oder der stündliche Bedarf des Supermarkts abgezogen wird, wenn er kleiner als die 15 % ist.

$$P_{net} = P_{DA} - P_{sm} \tag{3.2}$$

$P_{net}$ Leistung, die vom Windparkbetreiber beim Netzbetreiber angemeldet wird in MW

$P_{DA}$ Vorhersage der Windleistung (day ahead) in MW

$P_{dem}$ Stündlicher Leistungsbedarf des Supermarkts (demand) MW

Die Leistung, die insgesamt den Kälteanlagen zur Verfügung steht, berechnet sich nach der Gleichung (3.4). Der Ausdruck $P_{RT} - P_{net}$ bedeutet die überschüssige Windleistung. Der Ausdruck $P_{dem} - P_{sm}$ bedeutet die vom Supermarkt beim Netzbetreiber nicht abgemeldete Leistung.

$$P_{load} = P_{RT} - P_{net} + P_{dem} - P_{sm} \tag{3.3}$$
$$= P_{RT} - P_{DA} + P_{dem} \tag{3.4}$$

$P_{load}$ Vorhandene Leistung zum Laden in MW

$P_{RT}$ Windleistungsdaten. Windleistung, die tatsächlich vom Windpark generiert wurde in MW

$P_{net}$ Leistung, die vom Windparkbetreiber beim Netzbetreiber angemeldet wird in MW

$P_{dem}$ Stündlicher Leistungsbedarf des Supermarkts (demand)

$P_{sm}$ Leistung, die vom Windparkbetreiber dem Supermarkt versprochen wird in MW

### Implementierung

Die Implementierung des Kooperationskonzepts erfolgt in einer neuen Klasse **Cooling_strategy**. Diese Klasse hat die Aufgabe, jeder Kühlstelle in jedem Supermarkt im Netz die aufgrund des Kooperationskonzepts und der Temperaturbeschränkungen optimale Menge an Kälteleistung zuzuweisen. Die zur Verfügung stehende Leistung wird auf alle Kühlstellen gleichmäßig aufgeteilt bis keine Leistung mehr vorhanden ist oder die Minimaltemperatur erreicht wird. Alle Kühlstellen sollen zu jeder Stunde mindestens soviel Leistung zugewiesen bekommen, dass genau diese eine Stunde überbrückt werden kann, ohne, dass die Höchsttemperatur überschritten wird. Die Zeit bis zur Höchsttemperatur darf also nie unter eine Stunde fallen. Um diese Zeitbeschränkung zu erfüllen, muss ermittelt werden, ob die verbleibende Zeit bis zur oberen Schranke nicht unter eine Stunde fällt, wenn keine Kühlleistung vorhanden ist. Diese Zeit kann nach der Gleichung (3.5) abgeschätz werden.

$$\tau_{krit}(i) = \frac{m \cdot c \cdot (t_{max} - t(i))}{Q_{v_{ln}}(i)} \tag{3.5}$$

$$Q_{v_{ln}}(i) = \begin{cases} 1h \cdot \overline{\dot{Q}}_{Tr_{ln}}(i), & \text{außerhalb der Öffnungszeiten} \\ 1h \cdot (\overline{\dot{Q}}_{Tr_{ln}}(i) + \dot{Q}_{mehr}(i)), & \text{innerhalb der Öffnungszeiten} \end{cases}$$

$Q_{v_{ln}}(i)$ Verlustkältemenge zur Stunde $i$ in kJ

$\tau_{krit}(i)$ Zeit bis zur kritischen Temperatur zur Stunde $i$ in h

$m$ Substanzmasse zur Aufnahme der Wärmeenergie in kg

$c$ Spezifische Wärmekapazität der Substanzmasse in $\frac{\text{kJ}}{\text{kg}\cdot\text{K}}$

$t_{max}$ Obere Temperaturgrenze in °C

$t(i)$ Temperatur zur Stunde $i$ in °C

$\overline{\dot{Q}}_{Tr_{ln}}(i)$ Logarithmierter Mittelwert der Transmissionswärmeleistung zum aktuellen Zeitpunkt in kW

$\dot{Q}_{mehr}(i)$ Tagesmehrbedarf zum aktuellen Zeitpunkt in kW

Die Differenz zwischen der Transmissionswärmeleistung für den aktuellen Zeitpunkt wenn die Kühlraumtemperatur minimal wäre, und der Transmissionswärmeleistung zum aktuellen Zeitpunkt geteilt durch den Logarithmus aus der Division der beiden Leistungen, ergibt den logarithmierten Mittelwert der Transmissionswärmeleistung.

$$\overline{\dot{Q}}_{Tr_{ln}}(i) = \frac{\dot{Q}_{Tr_{max}}(i) - \dot{Q}_{Tr}(i)}{\log\left(\frac{\dot{Q}_{Tr_{max}}(i)}{\dot{Q}_{Tr}(i)}\right)} \tag{3.6}$$

$\overline{\dot{Q}}_{Tr_{ln}}(i)$ Logarithmierter Mittelwert der Transmissionswärmeleistung zum aktuellen Zeitpunkt in kW

$\dot{Q}_{Tr_{max}}(i)$ Transmissionswärmeleistung für den aktuellen Zeitpunkt, wenn die Kühlraumtemperatur minimal wäre in kW

$\dot{Q}_{Tr}(i)$ Transmissionswärmeleistung für den aktuellen Zeitpunkt kW

$$\dot{Q}_{Tr_{max}}(i) = A \cdot k \cdot (t_{max} - t(i)) \tag{3.7}$$

$\dot{Q}_{Tr_{max}}(i)$ Transmissionswärmeleistung für den aktuellen Zeitpunkt, wenn die Kühlraumtemperatur minimal wäre in kW

$A$ Fläche in $\text{m}^2$

$k$ Wärmedurchgangskoeffizient in $\frac{\text{W}}{\text{m}^2\cdot\text{K}}$

$t_{max}$ Obere Temperaturgrenze in °C

$t(i)$ Temperatur zur Stunde $i$ in °C

$$\dot{Q}_{Tr}(i) = A \cdot k \cdot (t_{amb}(i) - t(i)) \tag{3.8}$$

| | |
|---|---|
| $\dot{Q}_{Tr}(i)$ | Aktuelle Transmissionswärmeleistung in kW |
| $A$ | Fläche in $m^2$ |
| $k$ | Wärmedurchgangskoeffizient in $\frac{W}{m^2 \cdot K}$ |
| $t_{amb}(i)$ | Aktuelle Umgebungstemperatur in °C |
| $t(i)$ | Aktuelle Kühlraumtemperatur in °C |

Um den Temperaturausgleich bei den Lebensmitteln im Algorithmus zu berücksichtigen, werden deshalb die zu- und abgeführten Wärmeenergiemengen bei der Berechnung der Temperatur für jeden Zeitschritt stets mit dem Faktor 0,8 multipliziert. Die Gleichung zur stündlichen Berechnung der aktuellen Temperatur ist damit:

$$t(i+1) = 0.8 \cdot \frac{Q_v - Q_{ab}}{m \cdot c} h + t(i) \tag{3.9}$$

| | |
|---|---|
| $t$ | Temperatur in °C |
| $Q_v$ | Eindringende Verlustwärmemenge in kJ |
| $Q_{ab}$ | Abführende Wärmemenge in kJ |

wobei $Q_v$ die aktuell eindringende Wärmeenergie und $Q_{ab}$ die abgeführte Wärmeenergie ist.

In der Abbildung 3.4 auf der Seite 27 wird in einem Klassendiagramm die Beziehung zwischen dem bestehenden Modell und der Klasse **Cooling_strategy** veranschaulicht. Eine Klasse **Cooling_strategy** steht in der direkten Verbindung mit der Klasse **Refrigerator**, für die sie die zur Kühlung erforderliche Leistung berechnet. Ein Objekt der Klasse **Cooling_strategy** kann mit einem bis mehreren Objekten der Klasse **Refrigerator** in Verbindung stehen.

In der Abbildung 3.5 auf der Seite 31 ist der Programmstrukturplan des Kooperationskonzepts abgebildet. Der Algorithmus zeigt den Ablauf beim „Auffüllen“ der Kühlstellen für genau einen Zeitschritt der Simulation. Die Umsetzung des Algorithmus erfolgt in der Funktion **strategy_calculator()**.

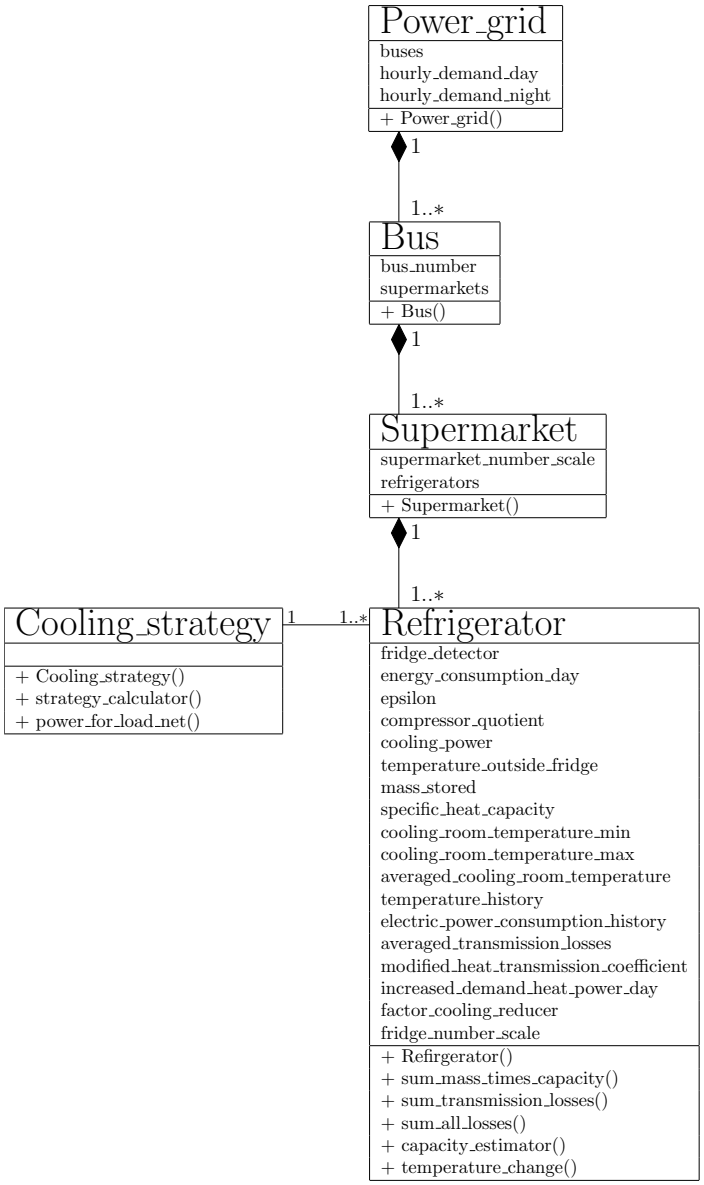


Abbildung 3.4: Klassendiagramm Kooperationskonzept

**strategy_calculator()** Es gibt fünf Rückgabewerte dieser Funktion. Der erste Rückgabewert ist ein $n \times 11$-Array, wobei $n$ für die Anzahl der im gesamten Netz vorhandenen Kühlzellen steht. Dieser Array wird zu jedem Zeitschritt (Stunde) der Simulation neu erzeugt. In den Spalten des Arrays werden alle Informationen gespeichert, die durch Berechnungen in der Funktion entstanden sind und weitere Informationen, die zur eindeutigen Zuordnung dieser Ergebnisse zu der jeweiligen Kälteeinheit beitragen. Die Energie zum Kühlen wird für die jeweiligen Kühleinheiten z.B. in der Spalte sechs gespeichert. Der zweite Ausgabewert ist der stündliche Leistungbedarf $P_{dem}$ für das Gesamt-

netz. Die für das Gesamtnetz berechnete vorhandene Leistung zum Laden $P_{load}$ ist der dritte Ausgabewert. Der vierte Wert ist die vom Windparkbetreiber beim Netzbetreiber angemeldete Leistung $b$ für das Gesamtnetz. Der fünfte Ausgabewert ist die dem Supermarkt vom Windparkbetreiber versprochene Leistung. Die letzten drei Ausgaben berechnen sich nach den Formeln, die in der Gleichung (3.4) auf der Seite 24 zu sehen sind.

Die Übergabe folgender Attribute an die Funktion ist erforderlich.

**buses** Referenz zum Attribut **buses** vom Objekt der Klasse **Power_grid**

**hourly_demand_day** Referenz zum Attribut **hourly_demand_day** vom Objekt der Klasse **Power_grid**

**hourly_demand_night** Referenz zum Attribut **hourly_demand_night** vom Objekt der Klasse **Power_grid**

**count_number_day** Der aktuelle Wert von der Laufvariable number_day

**count_number_steps** Der aktuelle Wert von der Laufvariable number_steps

**gen_out_day_T** Daten der tatsächlich vom Windpark generierten Leistung für den untersuchten Zeitraum

**gen_out_day_A** Windleistungsprognosedaten für den untersuchten Zeitraum

**promised_gen_output_DA_gros** untere Grenze des angenommenen Bandes für Abweichung der Windleistungsprognosedaten von realen Werten für den untersuchten Zeitraum

**number_day** Laufvariable. Gesamtzahl der zu simulierenden Tage.

**number_steps** Laufvariable. Gesamtzahl der zu simulierenden Zeitschritte für einen Tag.

Der Auffüllvorgang findet zwischen den stündlichen Temperaturberechnungen statt. Diese Handlung wird schrittweise wiederholt, um die gleichmäßige Verteilung der Leistung auf alle Anlagen zu ermöglichen. Zur Lösung dieser Aufgabe im Programm ist eine **while**-Schleife mit einer Abbruchbedingung geeignet. Innerhalb der **while**-Schleife wird der Algorithmus, der in der Abbildung 3.5 zu sehen ist, umgesetzt.

Im ersten Schritt des Algorithmus findet eine Abfrage statt, ob Leistung zum Kühlen der Anlagen zur Verfügung steht. Der Abfrage liegt eine Berechnung (vgl. Gleichung (3.4) auf der Seite 24) zugrunde, die in der Funktion **power_for_load_net()** implementiert ist.

**Fall 1** Die Menge der Leistung ist nicht größer Null. Es erfolgt die nächste Abfrage nach den Kühlstellen, die eine Stunde ohne Leistung aus dem Netz nicht

überstehen können, ohne die kritische Temperatur zu überschreiten. Diese Anlagen werden mit der zusätzlichen, nicht angemeldeten Leistung aus dem Netz geladen, sodass sie genau eine Stunde überstehen, ohne die obere Temperaturgrenze zu verletzen. Der Vorgang endet anschließend, und es kann zum nächsten Zeitschritt der Simulation übergegangen werden.

Sind keine solche Kühlstellen vorhanden, endet der Vorgang und es kann zum nächsten Zeitschritt der Simulation übergegangen werden.

**Fall 2** Die Menge der Leistung ist größer Null. In diesem Fall werden alle Kühlstellen gefunden, deren Kapazität (vgl. die Gleichung (3.10)) noch nicht ausgeschöpft ist und deren Zeit bis zum Erreichen der kritischen Temperatur am kleinsten ist. Anschließend wird überprüft, ob eine Wärmeenergie abgeführt werden kann, die genau $\frac{1}{60} \cdot Q_v$[2] entspricht, ohne als Folge die Überschreitung der minimalen Temperatur zu verursachen. Ist das der Fall, wird die Kapazität dieser Kühlstelle gleich Null gesetzt, damit diese Kühlstelle beim nächsten Durchgang der **while**-Schleife nicht beachtet wird.

Ist das Abführen der Wärmeenergiemenge ohne Verletzung der Temperaturgrenzen möglich, so wird anschließend die Kapazität, die zur Verfügung stehende Gesamtleistung und die Leistung, die zur Kühlung dieser Kühltruhe bereitgestellt wird, um diese Leistungsmenge ergänzt. Die Zeit $\tau_{krit}$ wird um einen Zeitschritt ergänzt. Der Zyklus der **while**-Schleife setzt sich fort.

$$Q_{Kap} = Q_{max} - \frac{1}{60} Q_v \tag{3.10}$$

$$P_{load} = P_{load} - \frac{1}{60} \frac{Q_v}{3,6 \cdot 10^6 \cdot \varepsilon}$$

$$\tau_{krit} = \tau_{krit} + \frac{1}{60}$$

$$Q_{cooling} = Q_{cooling} + \frac{1}{60} Q_v$$

| | |
|---|---|
| $Q_{Kap}$ | Wärmeenergiemenge, die zusätzlich abgeführt werden kann, ohne dass die minimale Temperaturgrenze verletzt wird in kJ |
| $Q_{max}$ | Maximal abzuführende Wärmeenergiemenge in kJ |
| $Q_v$ | Eindringende Verlustwärmemenge in kJ |
| $P_{load}$ | Vorhandene Ladeleistung in MW |
| $\tau_{krit}$ | Zeit bis zur kritischen Temperatur in h |
| $Q_{cooling}$ | Wärmeenergiemenge, die der Kühlzelle zur Kühlung zugewiesen wird in kJ |

[2] $Q_v$ ist die eindringende Verlustwärmemenge in kJ.

$$Q_{max} = m \cdot c \cdot (t(i) - t_{min}) + Q_v \tag{3.11}$$

| | |
|---|---|
| $Q_{max}$ | Maximal abzuführende Wärmeenergiemenge in kJ |
| $m$ | Substanzmasse zur Aufnahme der Wärmeenergie in kg |
| $c$ | Spezifische Wärmekapazität der Substanzmasse in $\frac{\text{kJ}}{\text{kg}\cdot\text{K}}$ |
| $Q_v$ | Verlustkältemenge in kJ |
| $t_{min}$ | Obere Temperaturgrenze in ° C |
| $t(i)$ | Temperatur zur Stunde $i$ in ° C |

Der genaue Ablauf des Informationsflusses, der zwischen den Objekten im Verlauf der Simulation des Kooperationskonzepts stattfindet, wird im Sequenzdiagramm in der Abbildung 3.6 auf der Seite 32 grafisch erläutert.

## 3.2 Handhabung des Programms

### 3.2.1 Systemanforderungen

Zur Benutzung des Programms muss aufgrund der gewählten Programmiersprache und der objektorientierten Programmierweise folgendes gelten:

- Im Rechner muss die Software MATLAB® der Version 5.0 oder höher[3] eingerichtet sein.

Die Anforderungen an die Hardware sind durch die eingesetzte MATLAB®-Version bestimmt.

### 3.2.2 Installation

Das Programm richtet sich in seiner Installation und der Benutzung nach dem allgemein gültigen Gebrauch der MATLAB® M-files[4].

### 3.2.3 Aufruf der Simulation

#### Vorbereitung der Input-Information

Zum Starten des Programms werden folgende Konfigurationsdateien[5] benötigt:

[3] MATLAB® verfügbar über The MathWorks, Inc. (http://www.mathworks.com).

[4] Ausführliche Information dazu findet man z.B. in [1].

[5] Bei der Simulation wurden die Konfigurationsdateien verwendet, die im MATLAB®-Code A.1, MATLAB®-Code A.2 und MATLAB®-Code A.3 dargestellt sind.

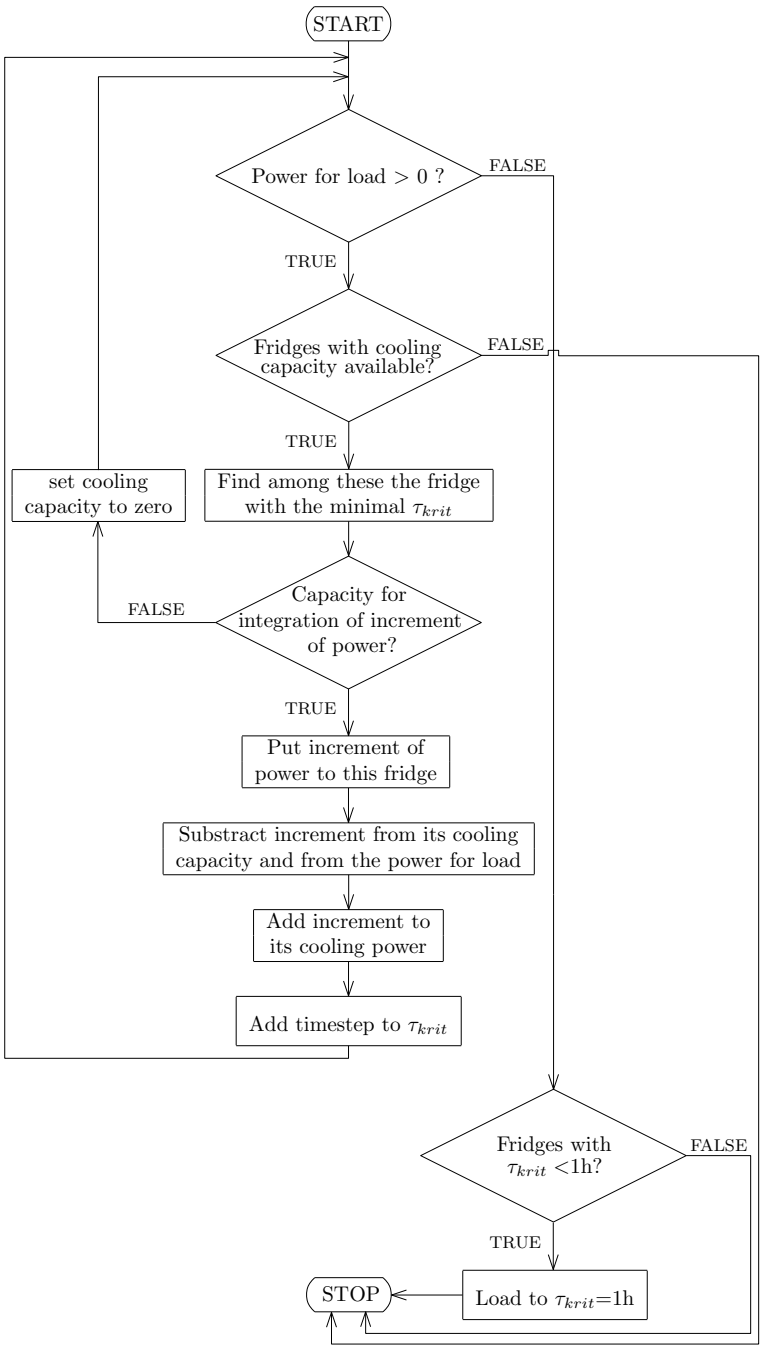


Abbildung 3.5: Flussdiagramm

- config_grid.m
- config_supermarkets.m
- config_fridges.m

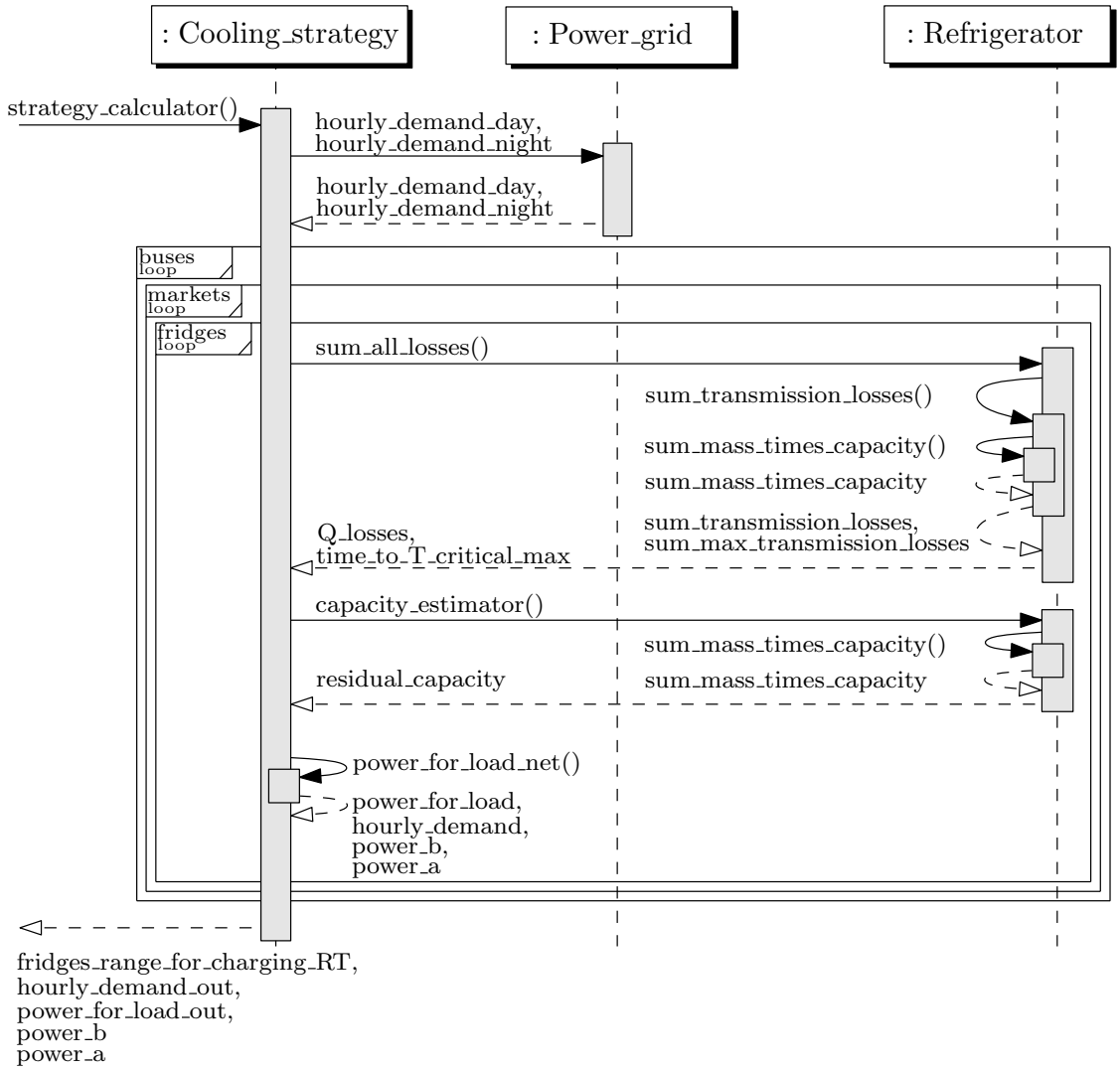


Abbildung 3.6: Sequenzdiagramm Kooperationsstrategie

### config_grid.m

Es ist wichtig, dass die topologischen Eigenschaften des Netzes im Programm berücksichtigt werden. Damit das Programm mit den für die Simulation notwendigen Informationen, die das Energienetz beschreiben, versorgt wird, ist die Form, die in MATLAB®-Code 3.1 vorgestellt wird, für die **config_grid.m**-Datei zwingend:

In der **config_grid.m**-Datei wird ein $n \times 2$-Cell Array ($n \in \mathbb{Z}_0^+$) definiert, der die Information über die Topologie des Netzes sowie die Verteilung der Kältelasten im Netz beinhaltet. Ein Cell Array ist ein Speicherobjekt, der verschiedene Datentypen unterschiedlicher Größe aufnehmen kann [1, Teil 2, Seite 15]. Ein Cell Array wird mit dem Befehl $c = cell(\ldots)$ oder mit Hilfe von geschweiften Klammern $c = \{\ldots\}$, wie in diesem Fall, erzeugt. Jede Zeile des Cell Arrays **configuration_grid**, der

MATLAB®-Code 3.1: config_grid.m

```
%%  Bus,  Supermarkets,   Number of Supermarkets
configuration_grid = {...
        1 , {Aldi        ,        1500;      ...
             Netto       ,         500};     ...
        2 , {0           ,           0};     ...
        3 , {0           ,           0};     ...
        4 , {0           ,           0}      ...
       };
```

in der **config_grid.m**-Datei definiert werden muss, steht für ein Knotenpunkt im Netz. In der ersten Spalte wird die Nummer des Knotens bzw. Busses gespeichert. In die zweite Spalte werden die Arten der Kältelasten und deren Anzahl am jeweiligen Knoten festgelegt. Es ist wiederum ein $m \times 2$-Cell Array ($m \in \mathbb{Z}_0^+$). Jede Zeile dieses Cell Arrays ist für eine eigene Art Supermarktkette reserviert. In die erste Spalte kommt der Name der Supermarktkette deren Eigenschaften in der **config_supermarkets.m**-Datei (vgl. MATLAB®-Code 3.2) gespeichert sind. In der zweiten Spalte wird die Anzahl der Supermärkte einer Kette festgelegt, die an dem bestimmten Knoten simuliert werden soll.

Um Fehler auf dieser Ebene zu vermeiden, muss die **config_grid.m**-Datei unbedingt die oben vorgestellte Form beibehalten.

**config_supermarkets.m**

MATLAB®-Code 3.2: config_supermarkets.m

```
%%       Kind of fridge     Anzahl
Aldi = {...
        { NK_KT_S       ,    1 };      ...
        { NK_KR_V       ,    2 };      ...
        { TK_TKT_S      ,    1 }       ...
       };
Netto = {...
        { NK_KT_S       ,    5 };      ...
       };
```

Die Kältelast in einem Supermarkt bilden die einzelnen je nach Einsatzzweck speziell dafür konstruierten Kühleinheiten. Die Anzahl, die technischen Eigenschaften und die Betriebsweise dieser Kältemaschinen können in der Realität Unterschiede aufweisen. In der Konfigurationsdatei **config_supermarkets.m** (vgl. MATLAB®-Code 3.2) wird festgelegt, wie die einzelnen Supermarktketten aus verschiedenen

Kälteeinheiten zusammengesetzt sind. Die Gruppierung der einzelnen Kühleinheiten zu einem Modellsupermarkt erfolgt in der **config_supermarkets.m** -Datei durch Definition eines oder mehrerer $i \times 2$-Cell ($i \in \mathbb{Z}_0^+$) Arrays, die jeweils einen Supermarkt abbilden. Jede Zeile eines solchen Cell Arrays ist für je eine Art Kühleinheit reserviert. In die erste Spalte kommt der Name der Kühlzelle, deren Eigenschaften in der **config_fridges.m**-Datei (vgl. MATLAB® -Code 3.3 auf der Seite 34) gespeichert sind. In der zweiten Spalte wird die Anzahl für diese Kühleinheiten, die in dem Supermarkt betrieben werden, festgelegt.

### config_fridges.m

Die Konfigurationsdatei **config_fridges.m** kann als eine Datenbank für Kälteeinheiten aufgefasst werden. In dieser Datenbank werden Informationen nach einem bestimmten Muster gruppiert und gespeichert, sodass jede Gruppe eine bestimmte Kühleinheit abbildet. Im Programm kann aus je einem dieser physikalisch fundierten Modellen ein Kühleinheit-Objekt erzeugt werden.

Am Beispiel des Modells einer steckerfertigen Normalkühltruhe, abgekürzt NK_KT_S (vgl. MATLAB® -Code 3.3 auf der Seite 34), wird im Folgenden die Eingabeform eines solchen Datenbankeintrages in der Datei **config_fridges.m** erläutert. Zur Beschreibung einer Kühleinheit ist es erforderlich, die Annahmen in einen $1 \times 14$-Cell Array zu erfassen. Die Größe und die Art des Arrays ist vorgegeben durch die Auslegung des Modells zur Anzahl und dem Typus der modellbeschreibenden Annahmen. Die Struktur eines Cell Arrays ermöglicht neben der zusammenhängenden Speicherung verschiedener Datentypen einen durch MATLAB® darauf direkten Zugriff.

MATLAB® -Code 3.3: config_fridges.m

```
 % fridge configuration parameters
 NK_KT_S = { ...
1          1, ... % 1 if plugin module or 2 if combine fridge
2          4.7e3, ... % energy consumption per day Wh/24h
3          2, ... % epsilon power quotient
4          0.66, ... % compressor quotient
5          0 ... % installed cooling power in W
6          [16.4 6.7], ... % area_wall
7          [0.38 0.38], ... % heat_transmission_coefficient
8          [19 15], ... % temperature_outside
9          [200 200], ... % masse_stored
10         [2.3 3.52], ... % specific_mass_capacity
11         -6, ... % temperature_min in Celsius
12         2, ... % temperature_max in Celsius
13         1, ... % averaged cooling room temperature in Celsius
14         0, ... % refrigerating capacity
        };
```

**Spalte eins** [6] Die Unterscheidung in steckerfertige Kälteanlagen und Kälteverbundanlagen im Programm ist notwendig, da bei der Berechnung der Kälteverluste unterschiedliche Verfahren zugrunde gelegt werden. Diese Differenzierung erfolgt durch die Zuweisung einer Eins für die steckerfertige Einheit und einer Zwei für eine Einheit an einer Verbundanlage in der ersten Spalte im Array. Die Berechnug der Verluste wird im Kapitel 2 erklärt.

**Spalte zwei** Die druchschnittliche elektrische Energieaufnahme ist eine Kennzahl in Watt pro 24 Stunden, die durch die Hersteller mit Hilfe eines genormten Verfahrens ermittelt wird und in den technischen Blättern angegeben werden muss. Diese Angabe wird bei der Bestimmung von Kähltemehrverlusten am Tag (vgl. Gleichung (2.8) auf der Seite 11) verwendet.

**Spalte drei** In diese Spalte wird die Leistungszahl $\varepsilon$ eingetragen, die den Zusammenhang zwischen der Kälteleistung und der elektrischen Leistung herstellt.

**Spalte vier** Der Anteil des Verdichters an dem Tagesenergieverbrauch der Kühleinheiten wird in diese Spalte eingetragen. Aus der Gleichung (2.11) im Kapitel 2 geht hervor, wie diese Kennzahl in die Berechnung eingeht.

**Spalte fünf** Werden anschlussfertige Kälteaggregate zur Kühlung von Räumen verwendet, so wird die installierte Kälteleistung in Kilowatt in die Spalte fünf anstatt einer Null eingetragen.

**Spalte sechs** Für die Transmissionsverlustberechnung sind unter anderem die Flächengrößen und die jeweiligen Wärmedurchgangskoeffizienten der einzelnen Wände maßgebend[7]. Für je eine Flächengröße in Quadratmeter ist in der Spalte sieben des NK_KT_S-Arrays ein Spaltenplatz im $1 \times w$-Zeilenarray reserviert. $w$ steht für die Anzahl der Wände.

**Spalte sieben** In dieser Spalte werden in einem weiteren $1 \times w$-Zeilenarray die mit den Flächen korrespondierende Wärmedurchgangskoeffiziente in gleicher Reihenfolge gespeichert.

**Spalte acht** Die Temperatur außerhalb der Kühleinheit bestimmt die Größe der Transmissionsverluste. In dieser Spalte werden nach dem gleichen Prinzip und in der gleichen Form wie im vorhergegangenen Eintrag die Außentemperatur für jede Wand eingetragen.

**Spalte neun** Die Abhängigkeit der Temperaturänderung von der in einer Kühleinheit deponierten Masse an Waren und deren spezifische Wärmekapazität

[6] Aus Gründen der Übersicht sind im MATLAB®-Code 3.3 auf der Seite 34 der Zeilenvektor NK_KT_S als Spaltenvektor dargestellt. Es wird jedoch weiterhin auf Spalten verwiesen, da er in MATLAB® als Zeilenvektor implementiert ist.

[7] Der mathematische Zusammenhang wird im Kapitel 2 in der Gleichung (2.3) deutlich gemacht.

wird aus der Gleichung (2.1) im Kapitel 2 ersichtlich. In der Realität werden in einem Kühlschrank gewöhnlich mehrere verschiedene Lebensmittel gekühlt. Soll gemischte Beladung simulieren werden, so sind die Massen in der Spalte zehn im NK_KT_S in der gleichen Art und Weise wie in der Spalte sechs sieben oder acht einzutragen. Der Eintrag erfolgt in Kilogramm.

**Spalte zehn** Zu jeder in der Spalte neun eingetragenen Massezahl muss in dieser Spalte nach dem Beispiel der Spalten neun und zehn, die spezifische Wärmekapazität eingetragen werden.

**Spalte elf** In der Realität fällt die Spanne für die Variation die Innentemperatur in einem Lebensmittelkühlschrank eher gering aus, da die Temperatur auf einem bestimmten Niveau gehalten werden muss, damit die Lebensmittel maximal lange frisch bleiben können. Möchte man die Kälte im Lebensmittel speichern, darf die Lebensmitteltemperatur einen bestimmten Bereich nicht verlassen. In dieser Spalte wird die untere Temperaturgrenze in Grad Celsius festgelegt.

**Spalte zwölf** In dieser Spalte wird die obere Temperaturgrenze in Grad Celsius festgelegt.

**Spalte dreizehn** Die Bedeutung[8] der mittleren Kühlraumtemperatur ist im Kapitel 2 erklärt.

**Spalte vierzehn** Zur Beachtung der Zeit, bis die Kälte aufgrund der Änderung der Kühlleistug nach Außen durch die Kühlraumabgrenzung entweichen kann, wird ein Faktor zwischen Null und Eins eingeführt, um den Kälteverlust zu verkleinern.

## Dynamische diskrete Simulation

Mit Hilfe der Simulation sollen Erkenntnisse über reale Systeme gewonnen werden. Die Umwandlung der elektrischen Energie in thermische Energie ist ein Prozess, der in einem hohen Grad von der Zeit abhängig ist. Die Simulation zeitabhängiger Systeme, wird als dynamische Simulation bezeichnet. Das Modell wurde so aufgestellt, dass zu jedem Zeitpunkt ein Systemzustand bestimmt und verändert wird. Ein neuer Systemzustand wird im nächsten Zeitschritt bestimmt, sodass die Auswirkung der Veränderung, die vorgenommen wurde, erst im nächsten Zeitschritt sichtbar gemacht werden kann. Weist die Simulation das obengenannte Verhalten auf, so spricht man von einer dynamischen diskreten Simulation. Die Funktionen, die den Systemzustand der Kälteanlagen zu jedem Zeitpunkt berechnen, werden von der Funktion **runCooling.m** aufgerufen.

[8] Alle Modellannahmen wurden in Zusammenarbeit mit Caroline Möller entwickelt [11].

**runCooling.m**

Zum Ausführen der Simulation dient die MATLAB® Funktions-Datei **runCooling.m**. Vor dem Start der Simulation mussen die Konfigurationsdateien, deren Zusammensetzung im vorangeganden Abschnitt erklährt wurden (vlg. auch MATLAB® -Code A.1, MATLAB® -Code A.2, MATLAB® -Code A.3) definiert und in MATLAB® Workspace geladen werden. Die **runCooling**, die im MATLAB® Command-Window aufgerufen werden kann, benötigt anschließend als Eingabeparameter die Referenz auf nun im MATLAB® Workspace vorhanden Cell-Array **configurations_grid** (vgl. MATLAB® -Code 3.1). Die Diese Datei hat die Aufgabe im ersten Schritt, die Konstruktor-Funktionen der einzelnen Klassen zum Erzeugen des OOP-Modells aufzurufen. Anschließend werden die Vattenfall(50Herz)-Windleistungsdaten geladen. Die Berechnung der Systemzustände erfolgt nun stundenweise für eine gewünschte Anzahl an Tagen. Die Tagesanzahl wird in der Zeile 13 der **runCooling.m**-Datei festgelegt (vgl. MATLAB® -Code A.4 auf der Seite 53). Die Berechneten Temperatur- und Verbrauchswerte werden in den jeweiligen Objekten gespeichert. Die Leistungswerte, die auf Grund des Kooperationskonzepts entstehen, werden in einem $(d \times h \times v)$-Array namens **power_grid_results** gespeichert.

$d$ steht für die Anzahl der Tage

$h$ steht für die Anzahl der Stunden an einem Tag

$v$ steht für die Anzahl der berechneten Lestungswerte, die für weitere Verwendung, z.B. in einem Leistungsflußberechnungsprogramm verwendet werden können

Die Anzahl ist auf $v = 8$ festgelegt. Die Speicherung der Daten erfolgt für das gesamte Netz und wird nicht nach Knoten oder Supermärkten unterschieden. Diese Behandlung erfolgt aus dem Grund, dass durch die Umsetzung des Kooperationskonzepts die Zuweisung der Menge an Leistung für die einzelnen Kühlstellen nicht von der Netztopologie abhängig ist. Der Verbrauch der einzelnen Kühleinheiten wird in den jeweiligen Objekten gesichert.

Die zeitlichen Verläufe folgender Leistungswerte in MW werden in dem Array **power_grid_results** unter diesen Adressen abgelegt.

$v = 1$ Leistungsbedarfs $P_{dem}$ für das Gesamtnetz.

$v = 2$ Die dem Supermarkt angebotene Leistung $P_{load}$.

$v = 3$ Die vom Windparkbetreiber beim Netzbetreiber angemeldete Leistung $P_{net}$.

$v = 4$ Die vom Windparkbetreiber dem Supermarkt versprochen Leistung $P_{sm}$.

$v = 5$ Der durch die Kooperation und die reale Produktion der Windleistung entstandene Regelleistungsbedarf ($P_{RT} - P_{net}$).

$v = 6$ Der Anteil der Leistung $P_{load}$, der an die Speicher verteilt wurde, $P_{involved}$.

$v = 7$ Die gesamte an den Kältespeicher verteilte Leistung $P_v$.

$v = 8$ Die aufgrund der Kooperation zusätzlich entstandene und an die Speicher aus dem Netz verteilte Regelleistung: $P_{additionally} = P_v - P_{involved}$.

Die graphische Ausgabe dieser Werte wird durch die Funktion **plotCooling.m** realisiert.

## Anschauung der berechneten Daten

Zur inhaltlichen Auswertung einer Simulation hat sich graphische Darstellung der Ergebnisse als sehr hilfreich erwiesen.

### plotCooling.m

Nach dem Durchlauf der Simulation kann mit der Funktion **plotCooling.m** die graphische Ausgabe durchgeführt werden. Wird die Funktion z.B. im MATLAB® Command-Window aufgerufen, so muss ihr an erster Stelle die Referenz auf die **power_grid_results** übergeben werden. An zweiter Stelle wird die Referenz auf das Objekt der Klasse **Power_grid** mit allen Unterobjekten übergeben. Anschließend kommt die Angaben über den Knoten, den Supermarkt und den Tag, für den die graphische Ausgabe erfolgen soll. Die Zeitverläufe der Leistungswerte beziehen sich jedoch auf das Gesamtnetz.

Befindet man sich in dem Verzeichnis, wo das Programm gespeichert ist, so führ die Eingabe folgender Kommandos im MATLAB® Command-Window zum Start der Simulation und im Anschluss zu der graphischen Ausgabe (vgl.die Abbildung 3.7 auf der Seite 39) der Werte für den Knoten eins, Supermarkt eins und den vierten Tag.

```
>> clear all, config_fridges, config_supermarkets, config_grid
>> [g,m]=runCooling(config_grid_);
>> plotCooling(m,g,1,1,4)
```

Das Graphik-Fenster ist optisch in drei Spalten unterteilt. In den ersten beiden Spalten sind in den insgesamt sechs Subplots die zeitlichen Leistungsverläufe abgebildet. In der dritten Spalte sind die Temperaturverläufe der fünf Kühlanlagen eines Modellsupermarkts zu sehen.


Abbildung 3.7: Ausgabe Tag 4

Kapitel 4

# Simulation und Ergebnisse

## Inhaltsangabe



Im folgenden Kapitel werden die Ergebnisse der Simulation vorgestellt und ausgewertet. Die Daten für prognostizierte und die tatsächlich generierte elektrische Leistung durch Windenergie in der Vattenfall-Regelzone (heute 50Hertz-Regelzone) für das Jahr 2005 bilden die Datengrundlage für die Implementierung des Kooperationsalgorithmus. Der Kooperationsalgorithmus wurde in den vorangegangenen Kapiteln vorgestellt. Die Annahmen, die die jeweiligen Kälteeinheiten spezifizieren, wurden von Caroline Möller im Rahmen ihrer Diplomarbeit entwickelt [11]. Die Substanzmasse in den Kühlbereichen wurde aus Gründen der Vereinfachung als Konstant angenommen (Es wird angenommen, dass die Kauf der Ware durch permanentes Nachliefern neutralisiert wird). Aus diesem Grund darf die Temperaturänderung nur aufgrund der zugeführten Ladeleistung erfolgen. Für drei zufällig ausgewählte aufeinander folgende Tage des Jahres wird die Stätigkeit des Temperaturverlaufes eine Kühleinheit des Modellsupermarktes revidiert. Zusätzlich wird der Einfluss der Kooperation auf die Ladeleistung der Modellsupermarktkette untersucht. Zum Abschluss wird durch Variation der Parameter in den Konfigurationsdateien das Programm auf mögliche Skalierbarkeit untersucht. Durch das objektorientierte Modell entsteht das Potenzial, die erforderliche Speicherkapazität durch unterschiedliche Maßnahmen zu erhöhen z.B. durch die Variation der Anzahl von Objekten der Klassen Supermarkt und Refrigerator. Im ersten Schritt wird untersucht, wie die Verdoppelung der Speicherkapazität auf die Regelleistung den Energieverbrauch der Supermarktkette auswirkt. Im zweiten Schritt wird überprüft, ob die Art und Weise, wie die Speicherkapazität erhöht wurde, einen Einfluss auf das Verhalten des Systems nimmt. Im vorangegangenen Kapitel wurde erklärt, wie mit Hilfe der Konfigurationsdateien die Anzahl der Supermärkte und der Kühleinheiten im System verändert werden kann. Aufgrund des im Kooperationsmodell umgesetzten Verfahrens der gleichmäßigen Aufteilung der vorhandenen Ladeleistung auf alle Kälteeinheiten im Netz wird erwartet, dass die Art und Weise wie man Anzahl der Kälteeinheiten erhöht, keinen Einfluß auf das Ergebnis der Simulation haben dürfen, solange die Anzahl, die Beladung und die technische Ausführung übereinstimmen. Eine Bestätigung der Vermutung würde bedeuten, dass die implementierte Kommu-

nikation zwischen Objekte nach gewünschter Weise funktioniert. Dadurch entsteht die Perspektive, mit dem Programm den Einsatz der Kältespeicher im elektrischen Energieversorgungsnez mit weiteren Lastmanagement-Algorithmen und Ausführungen zu untersuchen.

## 4.1 Auswertung

Die Simulation wird für vier Fälle durchgeführt (vgl. Tabelle 4.1 auf der Seite 42). Im im Fall 1 werden die Multiplikatoren in den Konfigurationsdateien MATLAB®-Code 3.1 und MATLAB®-Code 3.2 so verändert, dass die Anzahl der Supermärkte 1500 der aktuellen realen Situation für das Gebiet Berlin-Brandenburg stark nahe kommt [11]. Das Modellsupermarkt enthält dabei fünf verschiedene Kühleinheiten. Das Programm erzeugt mit Hilfe dieser Konfigurationsdateien ein Objekt der Klasse Supermarkt und fünf Objekte der Klasse Refrigerator. Durch die Multiplikatoren wird mit Hilfe der sechs Objekte, das Verhalten 7500 unterschiedlicher Kälteeinheiten mit je 1500 Einheiten simuliert (vgl. Fall 1 in der Tabelle 4.1). Der Modellsupermarkt hat einen angenommenen durchschnittlichen Verbrauch von von 75,1384 MWh im Jahr [11]. Durch die Kooperation erhöht sich der Verbrauch auf 75,4479 MWh um rund 0,00441 %. Für das gesamte Netz mit 1500 Supermärkten sind das 464,25 MWh an Mehrbedarf im Jahr. Der Wert für die Regellenergie $W_{wo}$ führ das ganze Jahr 2005 (vgl. die Gleichung (4.1) auf der Seite 42) ohne die Kooperation lag bei rund 3,6 $10^6$ MWh. Durch die Kooperation konnte in der Simulation der Wert um rund ein Prozent mit 37,458 MWh gesenkt werden. Trägt der Winparkbetreiber die Kosten des Mehrbedarfs für den Supermarkt, dann ist durch die Einsparung an Regelenergie die Kooperationsrendite um die 7968,5 Prozent möglich (vgl. Tabelle 4.1). Für die Verdoppelung der Anzahl der Kältespeicher ist das Ergebnis der Simulation auch eine Verdoppelung der Einsparung an Regelleistung. Eine negative Folge ist die Verdoppelung des Mehrbedarfs für den Supermarkt. Dadurch ist keine Änderung der Kooperationsrendite beobachtet worden.

Zum besseren Verständnis, welche Dateien und wie zur Simulation jeweiligen Fälle verädert wurden sind sie im Anhang zusätzlich abgebildet. Die Konfigurationsdatein, mit denen Fall 1 simuliert wurden, sind in der MATLAB®-Code 3.1 und MATLAB®-Code 3.2 zu sehen. Die Konfigurationsdatein, mit denen Fall 2 simuliert wurden, sind in der MATLAB®-Code A.12 und MATLAB®-Code 3.2 zu sehen. Die Konfigurationsdatein, mit denen Fall 3 simuliert wurden, sind in der MATLAB®-Code 3.1 und MATLAB®-Code A.10 zu sehen. Die Konfigurationsdatein, mit denen Fall 4 simuliert wurden, sind in der MATLAB®-Code A.11 und MATLAB®-Code 3.2 zu sehen.

Aus der Tabelle 4.1 ist ersichtlich, dass die Vergrösserung der Speicherkapazi-

| | Fall 1 | | Fall 2 | | Fall 3 | | Fall 4 | |
|---|---|---|---|---|---|---|---|---|
| | Super-markt | Refri-gera-tor | Super-markt | Refri-gera-tor | Super-markt | Refri-gerator | Super-markt | Refri-gerator |
| Multiplikator | 1500 | 1 | 3000 | 1 | 1500 | 2 | 1500 | 1 |
| Objektenanzahl | 1 | 5 | 1 | 5 | 1 | 5 | 2 | 10 |
| Gesamtzahl im Netz | 1500 | 7500 | 3000 | 15000 | 1500 | 15000 | 3000 | 15000 |
| Gesamtzahl der Objekte | 6 | | 6 | | 6 | | 12 | |
| $W_{r_{wo}}$ in $10^6$ MWh | 3,6116 | | | | | | | |
| $W_{r_w}$ in $10^6$ MWh | 3,5742 | | 3,5381 | | 3,5381 | | 3,5381 | |
| $\lvert W_{r_{wo}} - W_{r_w} \rvert$ in MWh | 37.458 | | 73.530 | | 73.649 | | 73.530 | |
| $W_{dem}$ in MWh | 112.707,62 | | 225.415,23 | | 225.415,23 | | 225.415,23 | |
| $W_{coop}$ in MWh | 113.171,86 | | 226.353,68 | | 226.355,31 | | 226.353,68 | |
| $(W_{dem} - W_{coop})$ in MWh | 464,24 | | 938.450 | | 940,080 | | 938.450 | |
| Kooperationsgewinn in % | 7968,5 | | 7935,3 | | 7934,3 | | 7935,3 | |

Tabelle 4.1: Simulation Fälle

tät durch die Konfigurationsdateien funktioniert. Für beliebig große Energieversorgungsnetze können dadurch Simulationen durchgeführ werden.

Die Regelenergie berechnete sich dadurch, dass der Summe die Stundenwerte für die vom Windparkbetreiber tatsächlich generierte Leistungswerte über das ganze Jahr die Summe der Prognosestundenwerte über das Ganze Jahr abgezogen. Anschließs end wird daraus der Betrag genommen.

$$W_{r_{wo}} = \lvert \sum_{d=1}^{365} \sum_{h=1}^{24} (P_{RT\,dh} - P_{DA\,dh}) \rvert \tag{4.1}$$

$W_{r_{wo}}$ Regelenergie ohne Kooperation zum Ausgleich der Prognosefehler.

$P_{RT}$ Windleistungsdaten. Windleistung, die tatsächlich vom Windpark generiert wurde in MW

$P_{DA}$ Vorhersage der Windleistung (day ahead) in MW

$d$ Laufvariable für die Tage

$h$ Laufvariable für die Stunden

Die durch die Kooperation entstanden Regelenergie wird berechnet, indem zuerst der Betrag aus der Differenz zwischen der tatsächlich vom Windpark generierten Leistung summiert über das Jahr die auch über das Jahr summierte Leistung abgezogen wird, mit der der Windparkbetreiber sich beim Netzbetreiber angemeldet hat. Konnte der Supermarkt Winleistung integrieren, ohne zusätzliche Leistung aus dem Netz zu beziehen so muss der Betrag um diese auch über das ganze Jahr summierte Leistung abziehen. Wurde vom Supermarkt zusätzlich Leistung aus dem Netz

bezogen, so wird die restliche Regelenergie um die über das ganze Jahr summierte zusätzliche Regelleistung erhöht.

$$W_{r_w} = |\sum_{d=1}^{365}\sum_{h=1}^{24}(P_{RT_{dh}} - P_{net_{dh}})| - \sum_{d=1}^{365}\sum_{h=1}^{24}(P_{involved_{dh}} - P_{additionally_{dh}}) \tag{4.2}$$

$W_{r_{wo}}$ Regelenergie ohne Kooperation zum Ausgleich der Prognosefehler.

$P_{RT}$ Windleistungsdaten. Windleistung, die tatsächlich vom Windpark generiert wurde in MW

$P_{DA}$ Vorhersage der Windleistung (day ahead) in MW

$d$ Laufvariable für die Tage

$h$ Laufvariable für die Stunden

Der Betrag der Differenz $|W_{r_{wo}} - W_{r_w}|$ bildet den Wert der eingesparten Regelenergie.

Der durchschnittliche Jahresverbrauch für alle Supermärkte $W_{dem}$ ist die Summe des durchschnittlichen stündlichen Bedarfs $P_{dem}$ für alle Stunden und Tage des Jahres.

$$W_{dem} = \sum_{d=1}^{365}\sum_{h=1}^{24} P_{dem_{dh}} \tag{4.3}$$

$W_{dem}$ Der durchschnittliche Jahresverbrauch für die Supermaktkette

$P_{dem}$ Stündlicher Leistungsbedarf des Supermarkts (demand)

Der durchschnittliche Jahresverbrauch für alle Supermärkte $W_{dem}$ ist die Summe des durchschnittlichen stündlichen Bedarfs $P_{dem}$ für alle Stunden und Tage des Jahres.

$$W_{coop} = \sum_{d=1}^{365}\sum_{h=1}^{24} P_{v_{dh}} \tag{4.4}$$

$W_{coop}$ Der Jahresverbrauch für die Supermaktkette infolge des Kooperation

$P_v$ Der stündliche Leistungsverbrauch des Supermarkts aufgrund der Kooperation

Die Differenz $W_{dem} - W_{coop}|$ bildet den Wert des Jahresmehrbedarfs an Energie aufgrund der Kooperation.

In der Abbildung 4.1 wird der zeitliche Verlauf der Leistungsdaten des Widparkbetreibers und die daraus für die Kooperation ermittelte Werte über drei Tage vom 14.1.2005 bis 16.1.2005 stündlich aufgetragen. Die Prognose über die Windleistung[1] am ersten Tag von Mitternacht und bis zum Mittag war relativ konstant zwischen 3000 MW und 4000 MW angegeben. Daraus wurde die Leistung $P_{net}$[2] (vgl. Gleichung (3.2) auf der Seite 23), die beim Netzbetreiber aufgrund des Kooperationskonzepts anzumelden ist, berechnet. Die tatsächlich vom Windpark generierte Leistung[3] liegt zeitweise mit rund 2500 MW unter der Prognose. Die Leistung, mit der der Windpark sich beim Supermarkt und beim Netzbetreiber auf Grund der hohen Prognosewerte verpflichtet, kann nicht geliefert werden. In der Abbildung 4.2 auf der Seite Abbildung 4.2 ist im dritten Subplot zu erkennen, dass der Windparkbetreiber sich beim Supermarkt mit der Lieferung des kompletten durchschnittlichen stündlichen Bedarfs[4] verpflichtet[5] hat. Infolge dessen hat sich der Supermarkt beim Netzbetreiber komplett als Last abgemeldet[6]. Im Kühlbereich darf die Temperatur nicht über die obere Grenze steigen. Zum Einhalten dieser Bedingung muss der Supermarkt zusätzlich Leistung[7] aus dem Netz ziehen. Für den Netzbetreiber Verursacht diese unangemeldete Last einen zusätzlichen Regelaufwand. Der Verlauf dieser Energie ist im Subplot in der Abbildung 4.2 enthalten. Es ist zu erkennen, dass der Bedarf des Supermarkts[8] an diesem Tag komplett durch die zusätzliche Leistung aus dem Netz gedeckt wird. Die Temperatur in den Kühlbereichen tiefgekühlte Tiefkühltruhe steckerfertig und Tiefkühl Kühlzellen bleibt in diesem Zeitraum bei der maximal zulässigen Temperatur von -18 ° C. In den Stunden 27 bis 30 liegen die Werte für tatsächlich generierte Leistun über der Prognose. In diesem Zeitbereich ist auch ein deutlicher Anstieg der Ladeleistung des Supermarkts zu beobachten. Die Temperatur wurde dadurch gesenkt. Es wurde keine zusätzliche Leistung aus dem Netz beansprucht. Die Aufladung wurde komplett durch die Windleistung vorgenommen[9]. Die Prognose wurde der realen Werte für den neuen Tag angepasst. Der Tag 15.1.2005 zeichnet sich durch eine deutliche Windschwäche aus, sodass der Supermarkt mit dem größten Teil seines Bedarfs sich nicht beim Netzbetreiber abgemeldet hat. Der Windpark konnte aber auch die Differenz nicht bedienen, sodass

[1] In der Abbildung als *gen_ output_ DA* bezeichnet.
[2] In der Abbildung als *power_ net* bezeichnet
[3] In der Abbildung als *gen_ output_ DA* bezeichnet.
[4] In der Abbildung als *demand* bezeichnet.
[5] In der Abbildung wird diese Leistung als *power promised* bezeichnet (vgl. die Gleichung (3.1) auf der Seite 23).
[6] In der Abbildung als *firm commitment* bezeichnet
[7] In der Abbildung mit *additionally power bezeichnet* bezeichnet.
[8] In der Abbildung als *charge* bezeichnet.
[9] In der Abbildung wird die Leistung, die ausschließlich vom Windpark beansprucht wird, als **wind only** bezeichnet

nur die aus dem Netz bestellte Leistung verbraucht werden konnte. Diese Menge ist jedoch kleiner als der durchschnittliche stündliche Bedarf. Aus diesem Grund steigt die Temperatur in den Kühlbereichen. In den Stunden zwischen 36 und 40 ist die bestellte Leistung so groß, dass ein Teil der Kühlzellen damit versorgt wird, und die Temperatur konstant gehalten werden kann. In den Stunden 43 bis 60 liegt die tatsächliche generierte Windleistung deutlich über der Prognose, sodass der Supermarkt komplett von der Windversorgung gespeist wurde. Die Speicherkapazität wurde vollständig ausgenutzt und die Temperatur auf das Minimun heruntergekühlt. Aus diesem Grund konnte der Supermarkt in der Stunde 60, als wieder keine Leistung geliefert werden konnte, vollständig auf Leistung verzichten. In den folgenden Stunden wird aufgrund der Temperaurbeschränkungen die zusätzliche Leistung aus dem Netz beansprucht, bis wieder Mehrangebot an Windleistung gibt. Die Temperatur bleibt stehts in den definierten Grenzen und reagiert gemäß den Erwartungen auf die Netzsituation. Die vollständige Übersicht über alle Kühlzellen des Modellsupermarktes sowie über die Leistungsferläufe im gesamten Netz für diese Tage kann aus der Abbildung A.1, der Abbildung A.2 und der Abbildung A.3 bezogen werden. Der Unterschiede im Temperaturverlauf sind dadurch zu begründen, dass einzelnen Kühleinheiten sich durch die Beladung, die Abdichtung gegen Kälteenergieverlust etc. unterscheiden.

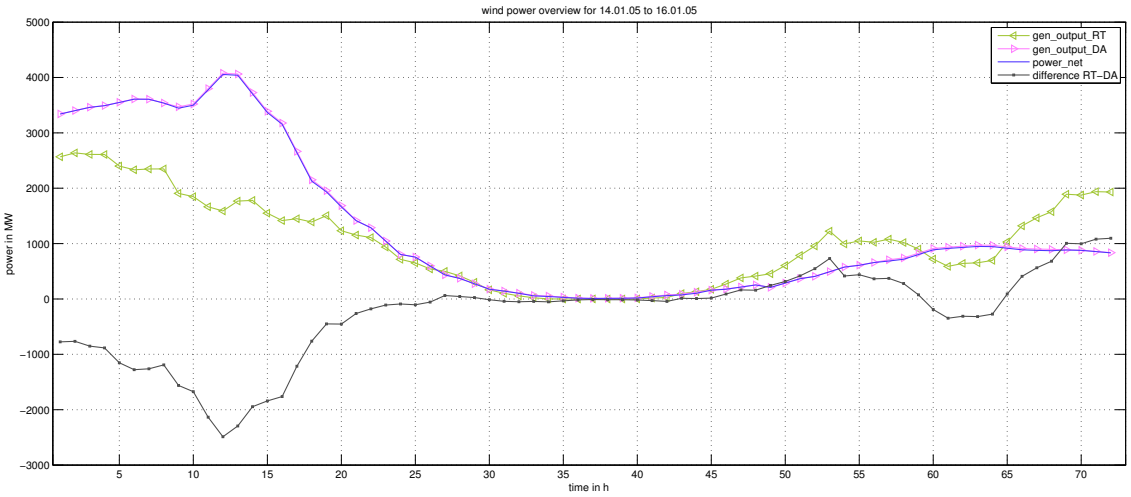


Abbildung 4.1: Leistungswerte Windparkbetreiber für die Tage 14.1.05-16.1.05

## Aussichten

Es gib viele Möglichgeiten, wie das Programm verbessert werden kann. An erster Stelle können zur realitätsgetreuen Darstellung der Abläufe und Verfahren im Kältetechnikbetrieb die Erstellung der Spezifikationsvariablen auf der Basis von messtech-

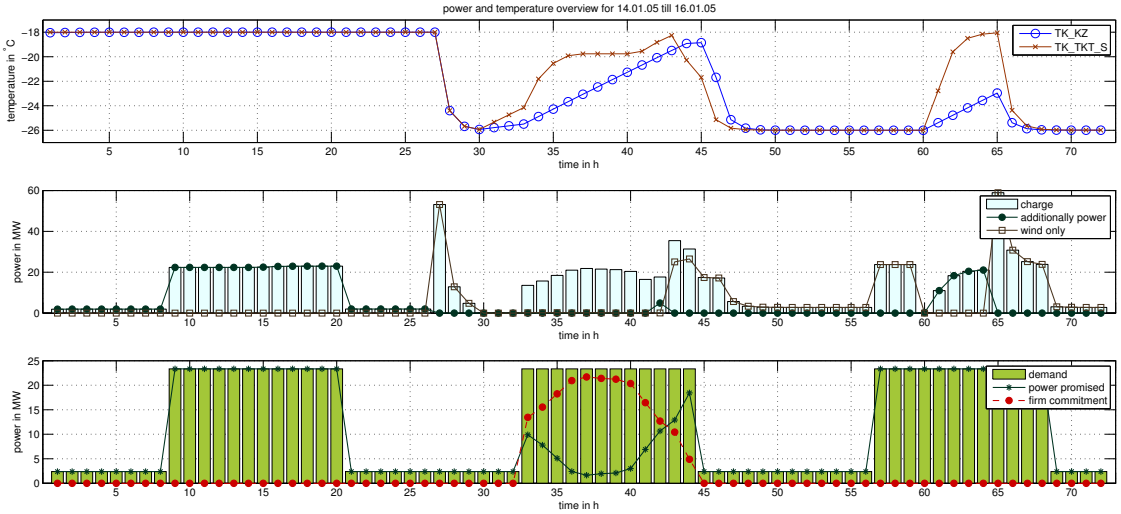


Abbildung 4.2: Temperatur und Leistungswerte für die Tage 14.1.05-16.1.05

nischen Verfahren zu verwirklichen. Mögliche Schwachstellen des Modells könnten dadurch korrigiert werden. Weitere Klassen zur Simulation weiterer verschiedener Kälteverlustquellen können implementiert werden. Der Einfluss der Vereisung des und die sinkende Leistungszahl des Verdampfers sowie die Beschränkung der Leistungsaufnahme durch technische Grenzen wurden nicht betrachtet. Das Kaufverhalten, also der Verlust an Substanzmasse zur Speicherung der Kälteenergie und der Einfluss der Jahreszeiten würden einen genaueren Abbild der Realität darstellen. Das Programm könnte einen Grundstein für weitere Studien auf diesem Gebiet bilden.

Kapitel 5

# Zusammenfassung

In dieser Arbeit wurde durch den Einsatz der Verfahren und Techniken des objektorientierten Entwurfs ein MATLAB®-Programm zur Simulation des variablen Lastverhaltens von Kältelasten mit Kältespeichern im Energieversorgungsnetz entworfen. Die Grundvorstellung der objektorientierten Programmierung, die Abläufe durch operierende Objekte darzustellen die Aufträge erledigen und auch untereinander vergeben können, wurde zur Entwicklung eines Simulationstools benutzt. Die Implementierung unterschiedlicher Modelle von Kälteanlagen, die der fluktuierenden Bereitstellung der elektrischen Leistung aus Windkraftanlagen folgen können, wurde durchgeführt. Ein objektorientiertes Modell des Systems wurde entwickelt und in Programm umgesetzt. Mit Hilfe der unifizierten Konfigurationsdateien können Spezifikationen, Beschränkungen und Randbedingungen unterschiedlicher Modelle gleichzeitig gespeichert werden. Dadurch können Objekte verschiedener Klassen erzeugt werden. Diese können auf das System unterschiedlich einwirken und so die Berechnungen beeinflussen. Die spezifischen Eigenschaften der Kälteeinheiten, wie die Speicherfähigkeit an Kälteenergie, die unterschiedliche Beladung und die Bestimmung des Verlustes an Kälte durch den Temperaturunterschied zwischen dem Kühlbereich und der Umgebungstemperatur oder durch zusätzliche Wärmestrahlung, konnten im Programm implementiert und für die Simulation verwendet werden. Die zeitliche Abhängigkeit des Prozesses wurde mit Hilfe des diskreten dynamischen Verfahrens umgesetzt. Dabei wird für jeden diskreten Schritt des simulierten Zeitraums ein Systemzustand bestimmt und verändert. Die Auswirkung der Veränderung kann erst im nächsten Schritt bestimmt werden. Durch die Ermittlung der Ladestades, der momentanen Verluste und aktuellen Temperatur für jede Kühleinheit, war es möglich die Veränderungen so zu begrenzen, dass die Temperaturgrenzen nicht überschritten wurden. Auf dieser Grundlage können die Kälteeinheiten als flexibler Verbraucher eingesetzt werden. Die Temperatur im Kühlbereich einer Kühleinheit bildet die Hauptbeschränkung, die unbedingt einzuhalten ist. Soll eine Kälteeinheit der fluktuierenden elektrischer Leistung folgen, so muss die Einschaltfrequenz des Verdichtermotors auf eine dem Normalbedrieb abweichende Form erfolgen. Eine möglicher Kooperationskonzept zwischen Windparkerzeuger und der Supermarktkette, der den Einsatz des Speichers steuert, wurde in einer Extraklasse implementiert. Auf der Basis Windleistungsprognosedaten und den realen Windleistungsdaten der Vattenfall-Regelzone (heute 50Hertz) für das Jahr 2005 wurde die Simulation für 1500 Modellsupermärkte durchgeführt. Die Zahl 1500 kommt der Anzahl der Supermärkte im Bereich der früheren Vattenfall-Regelzone nahe (vgl. [11]).

Die Simulation hat ergeben, dass die Einsparung an Regelenergie 1 % der Gesamtregelenergie für das Jahr betragen wurde. Eine Verdoppelung der Anzahl der Speicher auf 3000 Modellsupermärkte hatte eine Verdoppelung der Einsparung an Regelenergie zur Folge. Würde die Kooperation unter diesen Bestimmungen entstehen und würde der Windparkbetreiber die Kosten des aufgrund der Kooperation entstandenen Jahresmehrbedarfs an Leistung für den Supermarkt übernehmen, so würde der Windparkbetreiber aufgrund der durch Kooperation eingesparten Regelleistung eine Kooperationsrendite von 7968,5 % erfahren. Diese Ergebnisse haben aber eine ausschließlich theoretische Basis. Durch Messungen überprüfte Annahmen des Modells könnten stark zur Verbesserung des Programms beitragen und die Aussagekraft über die Ergebnisse der Simulation fundieren.

# Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzenden Quellen wörtlich und inhaltlich entnommen Stellen als solche kenntlich gemacht habe. Außerdem habe ich die Arbeit in gleicher oder ähnlicher Form keiner anderen Hochschule oder Prüfungsstelle vorgelegt.

Berlin, 19. August 2011

Juri Steblau

# Danksagung

Mein ganz besonderer Dank gilt der Caroline, ohne ihre Vorschläge und Ideen bei der Spezifikation der Kältelast und der Entwicklung des Kooperationskonzepts wäre diese Arbeit und das Programm in dieser Form nicht entstanden.

Ich danke ...

...Felix für die interessante Aufgabenstellung und die Betreuung.

...Peter, Lutz, Roman, Marcel, Daniel, Andreas, Jouni und allen meinen zahlreichen Kommilitonen für gute Arbeitsatmosphäre und viele anregende und aufschlussreiche Diskussionen.

...Siarhei für sein unermüdliches Interesse und Kenntnisse in mathematischen, technischen und den programmiertechnischen Bereichen.

...meinen Eltern für die Unterstützung jeglicher Art. Meiner Schwester Julia danke ich für ihre Hilfe bei der Korrektur der Rechtschreibfehler.

...Natasha für die Inspiration und ihr Feingefühl.

Anhang A

# Anhang

---

MATLAB®-Code A.1: config_grid.m

```
%%         Bus,  Supermarkets,            number of Supermarkets
config_grid_ = {...
        1 , {Aldi        ,        1500};     ...
        2 , {0           ,           0};     ...
        3 , {0           ,           0};     ...
        4 , {0           ,           0}      ...
          };
```

MATLAB®-Code A.2: config_supermarkets.m

```
Aldi = {...
        {NK_KT_S       ,           1};     ...
        {NK_KR_V       ,           1};     ...
        {TK_TKT_S      ,           1};     ...
        {NK_KZ         ,           1};     ...
        {TK_KZ         ,           1}     ...
       };
```

MATLAB®-Code A.3: config_fridges.m

```
NK_KT_S = { ...
    1, ... % kind of fridge: one plug in or two combine fridge
    1.88e4, ... % energy consumption per day Wh/24h
    2, ... % epsilon
    0.66, ... % compressor quotient
    0, ... % cooling power in W
    [16.4 6.7], ... % area_wall
    [0.38 0.38], ... % heat_transmission_coefficient
    [19 15], ... % temperature_outside
    [200 200]... % masse_stored
    [2.3 3.52], ... % specific_heat_capacity for each mass
    0, ... % temperature_min in C
    4, ... % temperature_max in C
    2, ... % averaged cooling room temperature in C
    0.75 ... % factor for reducing of cooling demand
    };
```

```
TK_TKT_S = { ...
    1, ... % kind of fridge: one plug in or two combine fridge
    8.88e4, ... % energy consumption per day Wh/24h
    1.5, ... % epsilon
    0.66, ... % compressor quotient
    0, ... % cooling power in W
    [36.9 16.8] ... % area_wall
    [0.38 0.38], ... % heat_transmission_coefficient
    [19 15], ... % temperature_outside
    [480 840 360 756]... % masse_stored
    [1.76 1.76 1.38 1.91], ... % specific_mass_capacity
    -26, ... % temperature_min in C
    -18, ... % temperature_max in C
    -22, ... % averaged cooling room temperature in C
    0.75}; % factor for reducing of cooling demand

NK_KR_V = { ...
    2, ... % kind of fridge: one plug in or two combine fridge
    0, ... % energy consumption per day Wh/24h
    2, ... % epsilon
    0, ... % compressor quotient
    28.3e3, ... % cooling power in W
    [40.2 68.8 24.3], ... % area_wall
    [1.53 0.62 0.38], ... % heat_transmission_coefficient
    [19 19 15], ... % temperature_outside
    [1210 1050 2520], ... % masse_stored
    [2.51 2.3 3.85], ... % specific_mass_capacity
    2, ... % temperature_min in C
    8, ... % temperature_max in C
    5, ... % averaged cooling room temperature in C
    0.75}; % factor for reducing of cooling demand

NK_KZ = { ...
    2, ... % kind of fridge: one plug in or two combine fridge
    0, ... % energy consumption per day Wh/24h
    2, ... % epsilon
    0, ... % compressor quotient
    830, ... % cooling power in W
    [30.7], ... % area_wall
    [0.26], ... % heat_transmission_coefficient
    [15], ... % temperature_outside
    [1440 2304], ... % masse_stored
    [2.51 3.85], ... % specific_mass_capacity
    1, ... % temperature_min in C
    5, ... % temperature_max in C
    3,  ... % averaged cooling room temperature in C
    1}; % factor for reducing of cooling demand

TK_KZ = { ...
```

```
    2, ... % kind of fridge: one plug in or two combine fridge
    0, ... % energy consumption per day Wh/24h
    1.5, ... % epsilon
    0, ... % compressor quotient
    1060 ... % cooling power in W
    [30.7], ... % area_wall
    [0.21], ... % heat_transmission_coefficient
    [15], ... % temperature_outside
    [1522 1384], ... % masse_stored
    [1.76 1.76], ... % specific_mass_capacity
    -26, ... % temperature_min in C
    -18, ... % temperature_max in C
    -22, ... % averaged cooling room temperature in C
    1}; % factor for reducing of cooling demand
```

MATLAB® -Code A.4: runCooling.m

```
function [ grid, ...
        grid_power_results ...
        ] = runCooling(config_grid) %% this function estimates:
% a) r = load_matrix for power flow simulator
% b) grid_system = supermarket system whit all estimated values and
% parameters
% c) b = matrix whit provided and integrated electric power
% here are indices of time the simulation should run
number_steps = 24; % do not change
number_days = 365;
% object grid system class Power_grid initialised
grid = Power_grid(config_grid, number_steps, number_days);
% object strategy class Cooling_strategy initialised
strategy = Cooling_strategy;

%%--------------load Data-----------------%%
load 'gen_output_RT'
load 'gen_output_DA'
% this is the real power that wind power can supply
gen_output_real_T = gen_output_RT(4:7,:,1:number_days);
gen_output_day_A = gen_output_DA(4:7,:,1:number_days);
promised_gen_output_DA_gross = ...
   sum(gen_output_DA(4:7,:,1:number_days) * 0.15);
%%--------------Data loaded-----------------%%

grid_power_results = zeros(number_steps, number_days, 8);

%% loop run down the "w" days of year
for w = 1 : number_days
   %% loop run down the "l" time steps of the day
   for l = 1 : number_steps
       % invoke function every time step
```

```
[   fridges_range_for_load_RT, ...
    hourly_demand_out, ...
    power_for_load_out, ...
    power_net, ...
    power_sm] = ...
        strategy.strategy_calculator( ...
        grid.buses, ...
        grid.hourly_demand_day, ...
        grid.hourly_demand_night, ...
        w, ...
        l, ...
        gen_output_real_T(:,l,w), ...
        gen_output_day_A(:,l,w), ...
        promised_gen_output_DA_gross(:,l,w), ...
        number_steps, ...
        number_days);
% hourly_demand
grid_power_results(l,w,1) = hourly_demand_out;
% power available result of cooperation
grid_power_results(l,w,2) = power_for_load_out;
% power promised to the net
grid_power_results(l,w,3) = power_net;
% power promised to the supermarket
grid_power_results(l,w,4) = power_sm;
% excess wind power
grid_power_results(l,w,5) = sum(gen_output_real_T(:,l,w)) ...
         - power_net;



for m = 1:length(fridges_range_for_load_RT(:,1))

b = fridges_range_for_load_RT(m,2); % number_bus
s = fridges_range_for_load_RT(m,3); % number_steps
r = fridges_range_for_load_RT(m,4); % number_fridge

    % TEMPERATURE CALCULATION
    grid.buses(b).supermarkets(s).refrigerators(r). ...
    temperature_change( ...
    number_steps, ...
    l, ... % count_step
    w, ... % count_day
    fridges_range_for_load_RT(m,5), ... % cooling
    fridges_range_for_load_RT(m,6)); % losses
    % overall electric power without additionally source power
    grid_power_results(l,w,6) = grid_power_results(l,w,6) + ...
       fridges_range_for_load_RT(m,11);
    % overall power
    grid_power_results(l,w,7) = grid_power_results(l,w,7) + ...
```

```
                fridges_range_for_load_RT(m,10);
        end

        % additionally power from source
        grid_power_results(l,w,8) = grid_power_results(l,w,7) - ...
                 grid_power_results(l,w,6);
    end
  message = ['day:' num2str(w) ' was simulated!'];
  disp(message)
 end
end
```

MATLAB® -Code A.5: Cooling_strategy.m

```
classdef Cooling_strategy < handle
 % NEW_COOLING_STRATEGY Summary of this class goes here
 %   Detailed explanation goes here
 properties
 end

 methods

  function obj = Cooling_strategy
  end % function constructor

  function [ fridges_range_for_charging_RT, ...
        hourly_demand_out, ...
        power_for_load_out, ...
        power_net, ...
        power_sm] = ...
            strategy_calculator( obj, ...
               buses, ...
               hourly_demand_day, ...
               hourly_demand_night, ...
               count_number_days, ...
               count_number_steps, ...
               gen_out_day_T, ...
               gen_out_day_A, ...
               promised_gen_output_DA_gros, ...
               number_steps, ...
               number_days)

  % This function gives a matrix which contains rang of
  % refrigerators refer to size up time to the critical temperature
  cooling = 0; %
  count_bus = 0; % initialisation runner
  time_step = 1/60; % incrementally charging
  electric_power_consumption = 0;
  wind_power_integrated = 0; % initialisation runner
```

```
fridges_range_for_charging_RT = zeros(1,11);

for b = 1:length(buses) % each bus
   count_bus = count_bus + 1; % update runner
   count_supermarket = 0; % initialisation runner
   for s = 1 : length(buses(b).supermarkets) %each supermarket
       count_supermarket = count_supermarket + 1;
       count_refrigerator = 0;
       for r = 1:length([buses(b).supermarkets(s).refrigerators])

     % before the estimation of temperature for the next step
     % can be started, the decision for the power, which
     % will be adjudged for each fridge, has be affected
     [Q_losses, time_to_T_critical_max] = ...
        buses(b).supermarkets(s).refrigerators(r). ...
             sum_all_losses(...
             number_steps, ...
             count_number_steps, ...
             count_number_days, ...
             number_days);

     % this function estimates the heat power the fridge can
     % maximum integrate for the moment
     residual_capacity = buses(b). ...
       supermarkets(s). ...
       refrigerators(r). ...
       capacity_estimator( ...
           count_number_steps, ...
           count_number_days, ...
           Q_losses);

           if sum (fridges_range_for_charging_RT) == 0 % securing

               count_refrigerator = count_refrigerator + 1;
               fridges_range_for_charging_RT = [...
                   time_to_T_critical_max ...
                   count_bus ...
                   count_supermarket ...
                   count_refrigerator ...
                   cooling ...
                   Q_losses ...
                   time_step ...
                   buses(b).supermarkets(s).refrigerators(r). ...
                   epsilon ...
                   residual_capacity ...
                   electric_power_consumption ...
                   wind_power_integrated ...
                   ];

           else
```

```
                count_refrigerator = count_refrigerator + 1;
                fridges_range_for_charging_RT = [...
                    fridges_range_for_charging_RT; ...
                    time_to_T_critical_max ...                  %1
                    count_bus ...                               %2
                    count_supermarket ...                       %3
                    count_refrigerator ...                      %4
                    cooling ...                                 %5
                    Q_losses ...                                %6
                    time_step ...                               %7
                    buses(b).supermarkets(s).refrigerators(r). ...
                    epsilon ...                             %8
                    residual_capacity ...                       %9
                    electric_power_consumption ...             %10
                    wind_power_integrated ...                  %11
                    ];
            end
        end
    end
end

% condition for load fridges
MODUS = 'load';

[ power_for_load, ...
  hourly_demand, ...
  power_net, ...
  power_sm] = obj.power_for_load_net( ...
              count_number_steps, ...
              gen_out_day_T, ...
              gen_out_day_A, ...
              promised_gen_output_DA_gros, ...
              hourly_demand_day, ...
              hourly_demand_night);

hourly_demand_out = hourly_demand;
power_for_load_out = power_for_load;
while strcmp('load', MODUS)

    table_to_restock = 0;

  if power_for_load > 0
    for c = find(fridges_range_for_charging_RT(:,9) > 0)
      if table_to_restock(1) == 0
      table_to_restock = [c, fridges_range_for_charging_RT(c,:)];
      else
      table_to_restock = [table_to_restock; c, ...
      fridges_range_for_charging_RT(c,:)];
      end
    end
```

```
if length(table_to_restock) > 1

[~,t] = min(table_to_restock(:,2));
n = table_to_restock(t,1);

% heat from cooling (power demand) for one more minute
transitional_storage_power_for_one_more_minute = ...
fridges_range_for_charging_RT(n,7) * ...
fridges_range_for_charging_RT(n,6);
  % this request secures that the fridge do not overload
  if fridges_range_for_charging_RT(n,9) - ...
     (fridges_range_for_charging_RT(n,6) * time_step) < 0
            %
     fridges_range_for_charging_RT(n,9) = 0;
  else
     % save the time period that was integrated
     fridges_range_for_charging_RT(n, 1) = ...
     fridges_range_for_charging_RT(n, 1) + ...
     fridges_range_for_charging_RT(n, 7);

     % update residual_capacity
     fridges_range_for_charging_RT(n, 9) = ...
     fridges_range_for_charging_RT(n, 9) - ...
     time_step * fridges_range_for_charging_RT(n,6);
     % update power_for_load
     power_for_load = power_for_load - ...
     (transitional_storage_power_for_one_more_minute / ...
     fridges_range_for_charging_RT(n, 8) / 3.6e6) * ...
     buses(fridges_range_for_charging_RT(n, 2)). ...
     supermarkets(fridges_range_for_charging_RT(n, 3)). ...
     supermarket_number_scale * ...
     buses(fridges_range_for_charging_RT(n, 2)). ...
     supermarkets(fridges_range_for_charging_RT(n, 3)). ...
     refrigerators(fridges_range_for_charging_RT(n,4)). ...
     fridge_number_scale;
     % update cooling
     fridges_range_for_charging_RT(n, 5) = ...
     fridges_range_for_charging_RT(n,5) + ...
     time_step * fridges_range_for_charging_RT(n,6);
     % update of electric_power_consumption
     fridges_range_for_charging_RT(n,10) = ...
     (fridges_range_for_charging_RT(n,5) / ...
     fridges_range_for_charging_RT(n,8) / 3.6e6) * ...
     buses(fridges_range_for_charging_RT(n, 2)). ...
     supermarkets(fridges_range_for_charging_RT(n, 3)). ...
     supermarket_number_scale * ...
     buses(fridges_range_for_charging_RT(n, 2)). ...
     supermarkets(fridges_range_for_charging_RT(n, 3)). ...
     refrigerators(fridges_range_for_charging_RT(n,4)). ...
```

```
            fridge_number_scale;

            %  power consumption without additionally source power
            fridges_range_for_charging_RT(n,11) = ...
              fridges_range_for_charging_RT(n,10);
        end
      else
         MODUS = 'do_not_load';
      end
     else
      for v = 1 : length(fridges_range_for_charging_RT(:,1))
          if fridges_range_for_charging_RT(v,1) < 1
             % estimates the rest of the time_crit to one hour
             s = 1 - fridges_range_for_charging_RT(v,1);
             % power cooling updated
             fridges_range_for_charging_RT(v,5) = ...
             fridges_range_for_charging_RT(v,5) + ...
             s * fridges_range_for_charging_RT(v,6);
             % update of electric_power_consumption
             fridges_range_for_charging_RT(v,10) = ...
             (fridges_range_for_charging_RT(v,5) / ...
             fridges_range_for_charging_RT(v,8) / 3.6e6) * ...
             buses(fridges_range_for_charging_RT(v, 2)). ...
             supermarkets(fridges_range_for_charging_RT(v, 3)). ...
             supermarket_number_scale * ...
             buses(fridges_range_for_charging_RT(v, 2)). ...
             supermarkets(fridges_range_for_charging_RT(v, 3)). ...
             refrigerators(fridges_range_for_charging_RT(v,4)). ...
             fridge_number_scale;
          else
              MODUS = 'do_not_load';
          end
      end
      MODUS = 'do_not_load';
    end
   end
end

% this function estimates power_for_load_nett
function [power_for_load, hourly_demand, power_net, power_sm] = ...
              power_for_load_net(obj, ...
              count_number_steps, ...
              gen_out_day_T, ...
              gen_out_day_A, ...
              promised_gen_output_DA_gros, ...
              hourly_demand_day, ...
              hourly_demand_night)

     if count_number_steps < 9 || count_number_steps > 20
         hourly_demand = hourly_demand_night;
```

```
      else
          hourly_demand = hourly_demand_day;
      end
      if promised_gen_output_DA_gros > hourly_demand
          power_sm = hourly_demand;
          power_net = sum(gen_out_day_A) - hourly_demand;
      else
          power_sm = promised_gen_output_DA_gros;
          power_net = sum(gen_out_day_A) - promised_gen_output_DA_gros;
      end
      if sum(gen_out_day_T) > sum(power_net)
          power_for_load = sum(gen_out_day_T) - power_net + ...
          hourly_demand - power_sm;
      else
          power_for_load = hourly_demand - power_sm;
      end

  end
 end
end
```

MATLAB® -Code A.6: Power_grid.m

```
classdef Power_grid < handle % handle call by reference
    %Power_grid Summary of this class goes here
    % Detailed explanation goes here
    properties
    buses %
    hourly_demand_day %
    hourly_demand_night %
    end

    methods
    % invoke constructor class Power_grid
        function obj = Power_grid( ...
                configuration_grid, ...
                number_steps, ...
                days ...
                )
    % in configuration_grid all data about which how many
    % supermarkets are connected on which bus invoce
    % constructor class Bus and save into
    % properties buses of object Power_grid
        for i = 1 : configuration_grid{end,1}
            obj.buses = [obj.buses ...
                Bus({configuration_grid{i,2}}, ...
                i, ...
                number_steps, ...
                days)];
```

```
        end
    obj.hourly_demand_day = 0;
    obj.hourly_demand_night = 0;
% %     here hourly_demand_day and %_night calculation starts
%       % observance of all supermarkets on each bus
            for b = 1 : length(obj.buses)

        % observance of all refrigerators of each supermarket
                for s = 1 : length(obj.buses(b).supermarkets)
            super = obj.buses(b).supermarkets(s); % substitution
                    for r = 1 : length(super.refrigerators)

            refr = super.refrigerators(r); % substitution

            obj.hourly_demand_day = obj.hourly_demand_day + ...
            (refr.increased_demand_heat_power_day ...
            / 3.6 + refr.averaged_transmission_losses) * ...
            refr.fridge_number_scale * ...
            super.supermarket_number_scale ...
            / refr.epsilon;

            obj.hourly_demand_night = obj.hourly_demand_night + ...
            refr.averaged_transmission_losses * ...
            refr.fridge_number_scale * ...
            super.supermarket_number_scale ...
            / refr.epsilon;
                    end
                end
             end
            obj.hourly_demand_day = obj.hourly_demand_day / 1e6;
            obj.hourly_demand_night = obj.hourly_demand_night / 1e6;
        end
    end
end
```

MATLAB®-Code A.7: Bus.m

```
classdef Bus < handle % have to be handle class becouse of the speed
 %%Bus Summary of this class goes here
 %   Detailed explanation goes here
 %%
 properties
     bus_number % the number of the bus
     supermarkets % all supermarkets which are connected into bus
 end % end properties
 %%
 methods
 % the constuctor of Bus class
     function obj = Bus( ...
```

```
                supermarkets_connected, ...
                bus_number, ...
                number_steps, ...
                days ...
            )

        obj.bus_number = bus_number; % save bus_number
        for n = 1 : length([supermarkets_connected{1}{:,2}])
            if supermarkets_connected{1}{n,2} ~= 0
                obj.supermarkets = [obj.supermarkets ...
                    Supermarket( ...
                    {supermarkets_connected{1}{n,1}}, ...
                    supermarkets_connected{1}{n,2}, ...
                    number_steps, ...
                    days)];
            else
                obj.supermarkets = [obj.supermarkets []];
            end
        end
    end
    %%
  end % methods end
end
```

MATLAB® -Code A.8: Supermarket.m

```
classdef Supermarket < handle
 %SUPERMARKET Summary of this class goes here
 %   Detailed explanation goes here
 %%
 properties
     supermarket_number_scale
     refrigerators
 end
 %%
 methods
   function obj = Supermarket( ...
      fridges, ...
      supermarket_number_scale, ...
      number_steps, ...
      days)

       obj.supermarket_number_scale = supermarket_number_scale;
       for n = 1 : length(fridges{1})
           obj.refrigerators = [obj.refrigerators ...
               Refrigerator( ...
                 fridges{1}{n},  ...
                 number_steps, ...
                 days )];
```

```
      end % end for
   end % end function Supermarket
 end % end methods
end % end classdef
```

MATLAB® -Code A.9: Refrigerator.m

```
classdef Refrigerator < handle
 %% REFRIGERATOR class created from Juri Steblau 09.03.10
 %   this class is a part of an program calls SuperM which simulates an
 %   supermarket as a cooling energy storage
 %%
 properties
     fridge_detector % the identification number of refrigerator
     energy_consumption_day % the maxim on power refrigerator can use
     epsilon % the ...
     compressor_quotient
     cooling_power
     temperature_outside_fridge
     mass_stored % the mass of stored product
     specific_heat_capacity % for each mass
     cooling_room_temperature_min % the minimum on temperature
     cooling_room_temperature_max % the maximum on temperature
     averaged_cooling_room_temperature
     temperature_history % temperature into fridge
     electric_power_consumption_history % the power consumption real
     averaged_transmission_losses
     modified_heat_transmission_coefficient
     increased_demand_heat_power_day % rest_day_power
     factor_cooling_reducer
     fridge_number_scale % factor for scale number fridges
 end % properties end
 %%
 methods
  function obj = Refrigerator( fridge_config, ...
                               number_steps, ...
                               days ) % fridge constructor
      % save the object properties
      obj.fridge_detector = fridge_config{1,1}{1}; % kind of fridge
      obj.energy_consumption_day = fridge_config{1,1}{2}; % in Wh/h
      obj.epsilon = fridge_config{1,1}{3};
      obj.compressor_quotient = fridge_config{1,1}{4};
      obj.cooling_power =  fridge_config{1,1}{5};
      obj.temperature_outside_fridge = fridge_config{1,1}{8};
      obj.mass_stored = fridge_config{1,1}{9};
      obj.specific_heat_capacity =fridge_config{1,1}{10};
      obj.cooling_room_temperature_min = fridge_config{1,1}{11};
      obj.cooling_room_temperature_max = fridge_config{1,1}{12};
      obj.averaged_cooling_room_temperature = fridge_config{1,1}{13};
```

```
      obj.temperature_history = zeros(number_steps, days);
      obj.electric_power_consumption_history =  ...
          zeros(number_steps, days);
      obj.temperature_history(1,1) =  ...
          obj.averaged_cooling_room_temperature;
      obj.factor_cooling_reducer = fridge_config{1,1}{14};
      obj.fridge_number_scale = fridge_config{2};
      obj.modified_heat_transmission_coefficient = ...
          fridge_config{1,1}{6} .* fridge_config{1,1}{7}; % Watt/K end
      % estimation of averaged transmission losses
      obj.averaged_transmission_losses = sum( ...
          obj.modified_heat_transmission_coefficient .* ...
          ( obj.temperature_outside_fridge - ...
          obj.averaged_cooling_room_temperature ));

      % this function estimates the rest of the power
      if obj.fridge_detector == 1 % the one means, PLUG IN FRIDGE
        % this function estimates the rest of the power
        % this function estimates the electrical energy of the fridge
        % for 24h in kJ (eigentlich Leistung)
        obj.increased_demand_heat_power_day = ...
          obj.energy_consumption_day * obj.compressor_quotient * ...
          obj.factor_cooling_reducer * obj.epsilon * 3.6 / ...
          12 - 2 * 3.6 * obj.averaged_transmission_losses;
      else
        % COMBINE FRIDGE
        obj.increased_demand_heat_power_day = (obj.cooling_power * ...
           obj.factor_cooling_reducer - ...
           obj.averaged_transmission_losses) * 3.6; %
      end % if end

  end % function constructor end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% ordinary object specific functions %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  function r = sum_mass_times_capacity(obj)
  %% this function estimates a part of a main_equation
      % estimates mass specific heat-capacity
       r = sum(obj.specific_heat_capacity .* ...
    obj.mass_stored); % kJ/K
  end % function sum_mass_times_capacity end

  function [r, r_c_max]  = sum_transmission_losses(obj, ...
            number_steps, ...
            count_number_steps, ...
            count_number_day)
  %% this function sums transmission losses of the wall
```

```
    % estimates sum of all wall transmission losses into fridge
    r = 0;
    r_c_max = 0;

    for i = 1:length(obj.modified_heat_transmission_coefficient)
     % it is necessary if temperature outside the fridges is constant
     % to get vector for each time step
     if length(obj.temperature_outside_fridge(i)) == 1
     n_temperature_outside_fridge = ...
     obj.temperature_outside_fridge(i) + zeros(1,number_steps);
     end % if end

     % VERY IMPORTANT estimates losses
     r = r + obj.modified_heat_transmission_coefficient(i) ...
     * (n_temperature_outside_fridge(count_number_steps) - ...
     obj.temperature_history(count_number_steps, count_number_day));
     % estimates maximal losses
     r_c_max = r_c_max + ...
     obj.modified_heat_transmission_coefficient(i) ...
     * (n_temperature_outside_fridge(count_number_steps) - ...
     obj.cooling_room_temperature_max);

    end % for end

    r = r * 3.6; % transmission losses in kJ (in actual fact, POWER)

end % function sum_transmission_losses end

function [r, time_to_T_critical_max]  = sum_all_losses(obj, ...
          number_steps, ...
          count_number_steps, ...
          count_number_day, ...
          number_days)
%% this function estimates the sum of the losses which can be
%  the losses can be positive or negative

    % sum all fridge losses for step
    [sum_transmission_losses, sum_max_transmission_losses] = ...
              obj.sum_transmission_losses( ...
              number_steps, ...
              count_number_steps, ...
              count_number_day);

    % here the time_no_cooling or time to the temperature go critical
    % will be estimated

    % logarithmic estimation of maximal transmission losses
    Q_T_log_max = (sum_max_transmission_losses - ...
         sum_transmission_losses) / ...
         log(sum_max_transmission_losses / ...
```

```
          sum_transmission_losses);

    % estimation of time till maximal temperature will be achieved
    time_to_T_critical_max = ((obj.cooling_room_temperature_max - ...
         obj.temperature_history(...
         count_number_steps, ...
         count_number_day)) * ...
         obj.sum_mass_times_capacity) / ...
         (Q_T_log_max * 3.6);

    % inspect what kind of day hour is it. Necessary because of
    % different level of losses
    if count_number_steps <  9 || count_number_steps >  20
        % here only the transmission losses influencing the
        % temperature
        r = sum_transmission_losses; %
    else
        % here in addition to the transmission losses the static
        % losses  of day activities influencing the temperature
        r = sum_transmission_losses + ...
            obj.increased_demand_heat_power_day;

        time_to_T_critical_max = ...
              ((obj.cooling_room_temperature_max - ...
              obj.temperature_history(count_number_steps, ...
              count_number_day)) * obj.sum_mass_times_capacity) / ...
              (Q_T_log_max * 3.6  + ...
              obj.increased_demand_heat_power_day);
    end % if else end
end % function sum_all_losses end

function r = capacity_estimator(obj, ...
          count_number_steps, ...
          count_number_day, ...
          Q_losses)
%% this function estimates the heat power, which can be
% stored into the fridge if the fridge temperature is T(i).

    r = Q_losses - (obj.cooling_room_temperature_min - ...
       obj.temperature_history(count_number_steps, ...
       count_number_day)) * obj.sum_mass_times_capacity; % in kJ

end % function capacity_estimator end

function r = temperature_change(obj, ...
          number_steps, ...
          count_number_steps, ...
          count_number_day, ...
          cooling, ...
          Q_losses)
```

```
  %% this function estimates the temperature in the fridge
      Q = 0.8 * (Q_losses - cooling);
      %% this is the main equation
      r = Q / obj.sum_mass_times_capacity + ...
      obj.temperature_history(count_number_steps, ...
            count_number_day);
      %%
      if count_number_steps == number_steps
        obj.temperature_history(1,count_number_day + 1) = r;
        obj.electric_power_consumption_history(1, ...
            count_number_day + 1) = cooling / obj.epsilon;
      else
        obj.temperature_history(count_number_steps + 1, ...
           count_number_day) = r;
        obj.electric_power_consumption_history( ...
           count_number_steps + 1, count_number_day) =  ...
           cooling / obj.epsilon;
      end % if else end
  end % function main equation end
 end % methods end
end % class end
```

MATLAB® -Code A.10: Variation Multipliaktor Kühlzellen

```
%
Aldi = {...
        {NK_KT_S        ,           2};     ...
        {NK_KR_V        ,           2};     ...
        {TK_TKT_S       ,           2};     ...
        {NK_KZ          ,           2};     ...
        {TK_KZ          ,           2}     ...
       };
```

MATLAB® -Code A.11: Variation Anzahl Supermärkte

```
%%          Bus,  Supermarkets,           number of Supermarkets
config_grid_ = {...
        1 , {Aldi        ,        1500;      ...
             Aldi        ,        1500};     ...
        2 , {0           ,           0};     ...
        3 , {0           ,           0};     ...
        4 , {0           ,           0}      ...
          };
```

MATLAB® -Code A.12: Variation Multiplikator Supermärkte

```
%%          Bus,  Supermarkets,           number of Supermarkets
config_grid_ = {...
        1 , {Aldi        ,        3000};     ...
        2 , {0           ,           0};     ...
        3 , {0           ,           0};     ...
        4 , {0           ,           0}      ...
          };
```

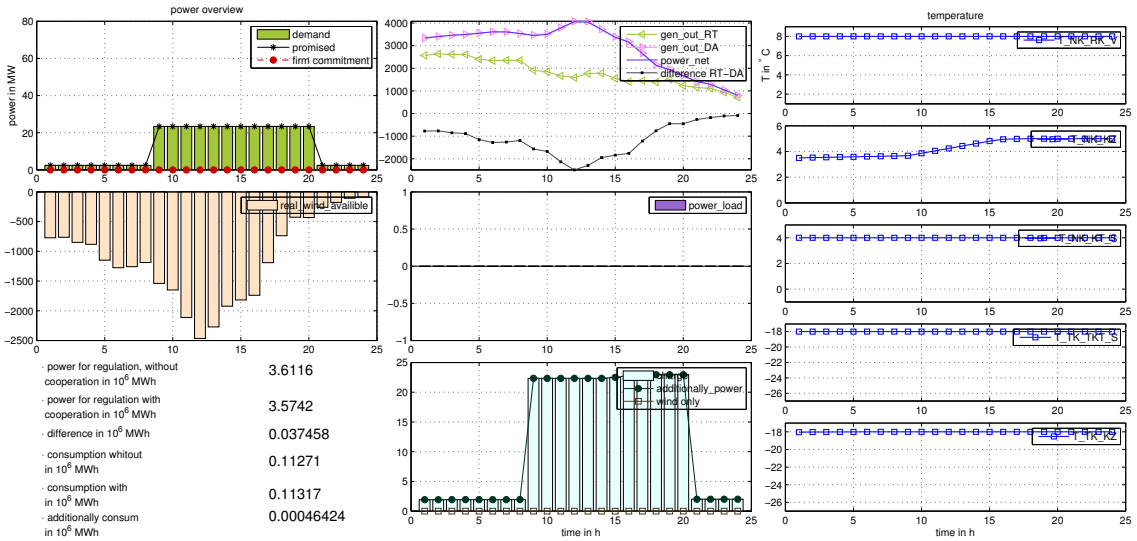


Abbildung A.1: Ausgabe für den Tag 14.01.2005

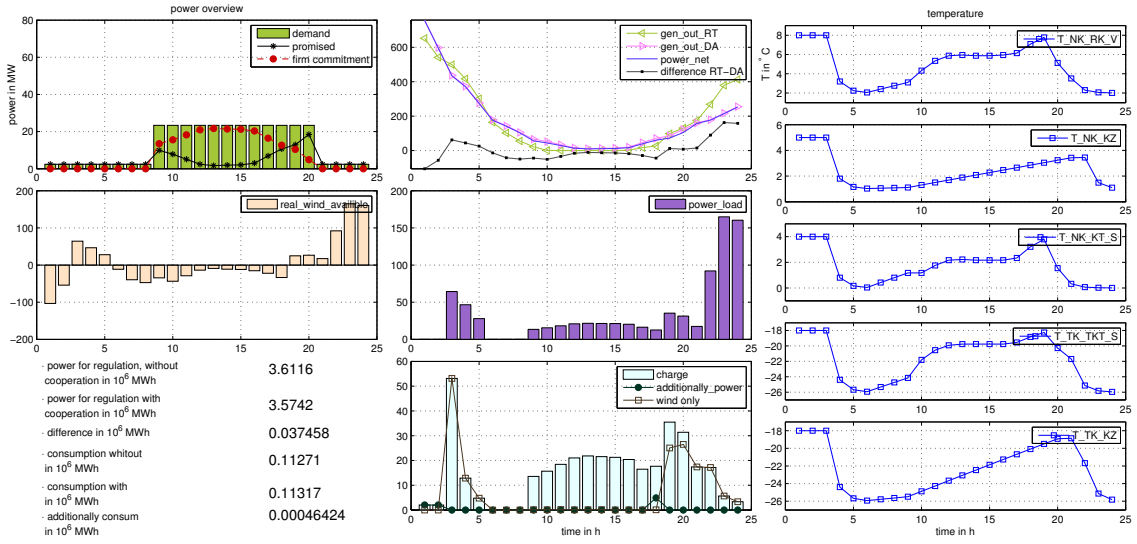


Abbildung A.2: Ausgabe für den Tag 15.01.2005

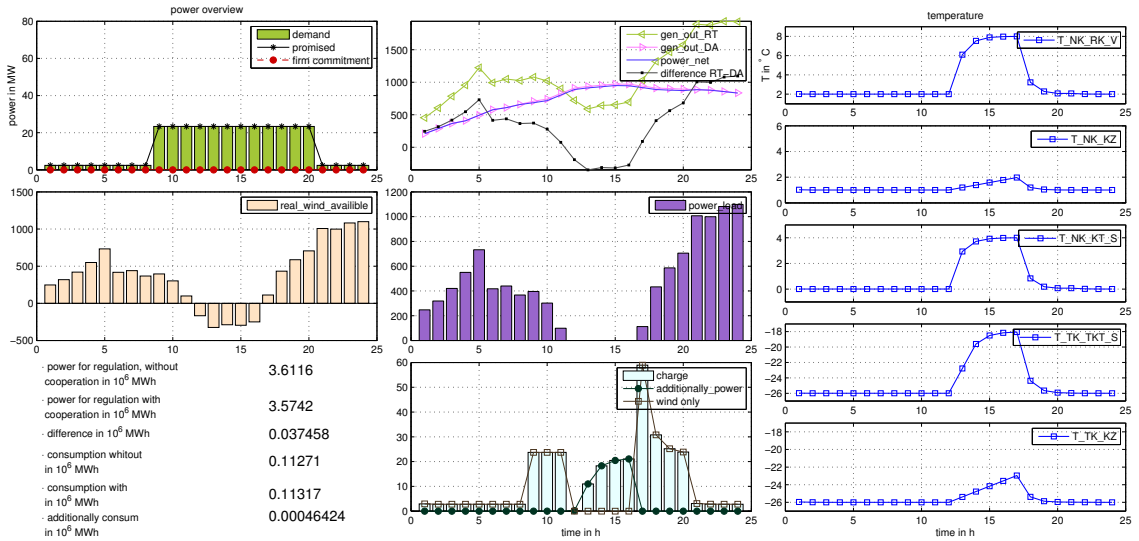


Abbildung A.3: Ausgabe für den Tag 16.01.2005

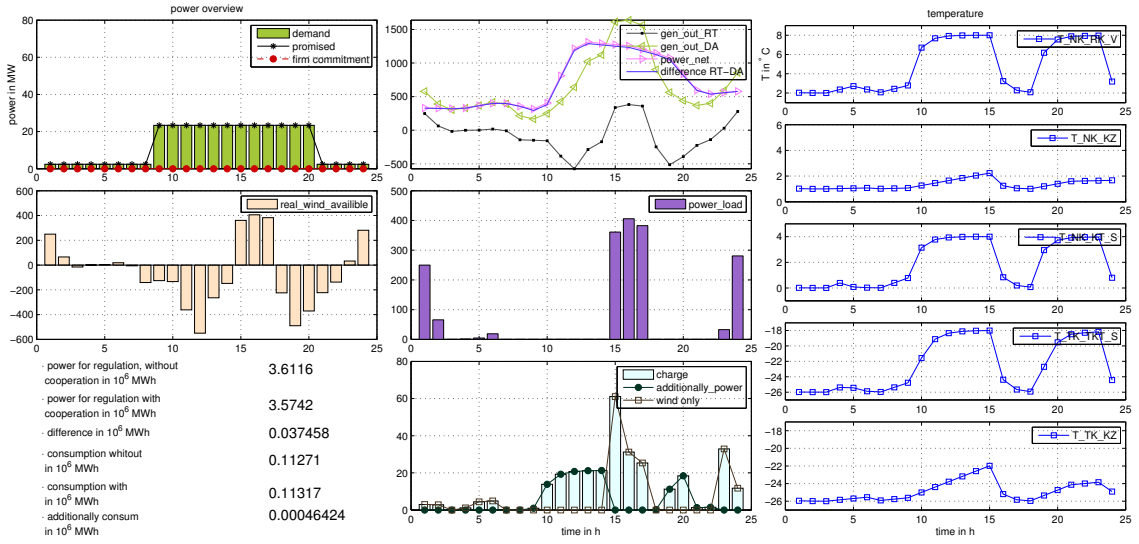


Abbildung A.4: Fall 1, Tag 176

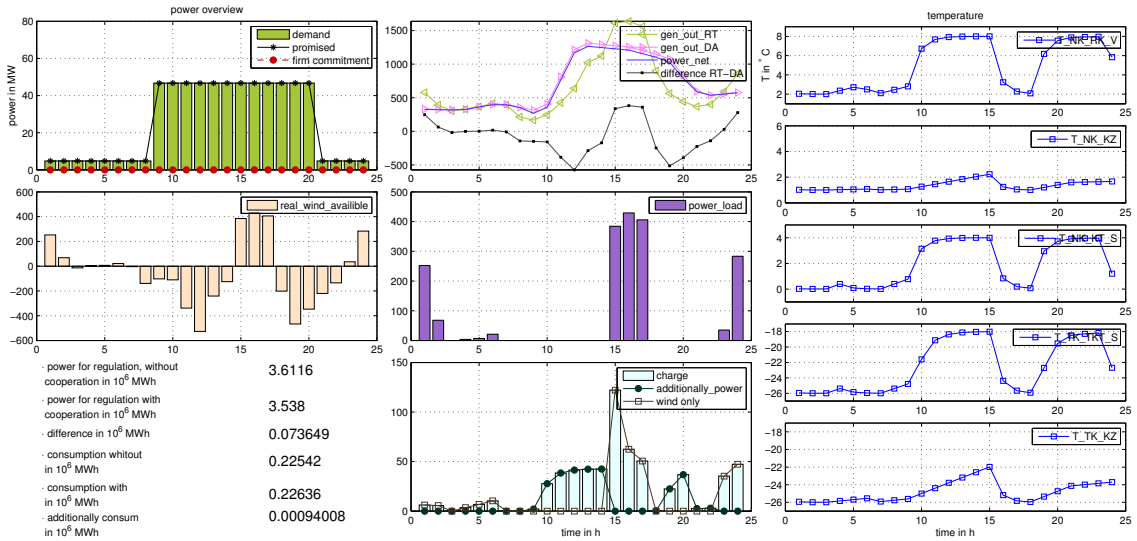


Abbildung A.5: Fall 2, Tag 176

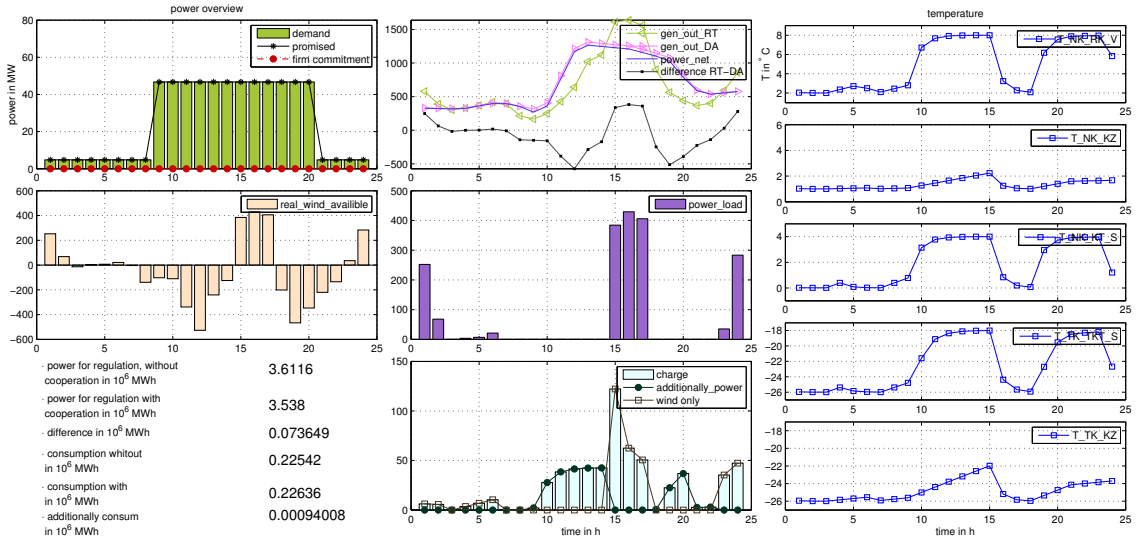


Abbildung A.6: Fall 3, Tag 176

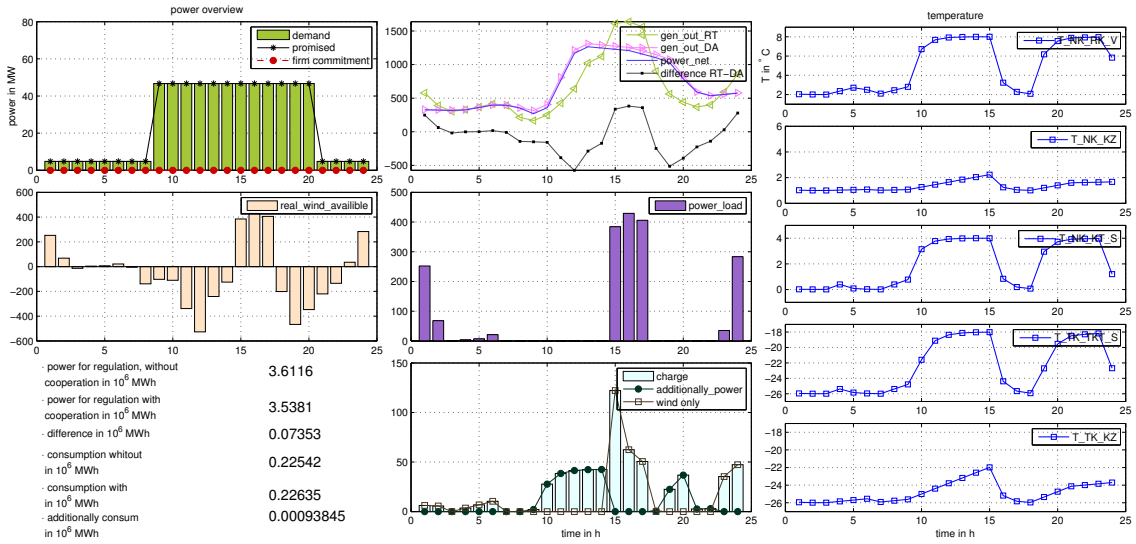


Abbildung A.7: Fall 4, Tag 176

# Literaturverzeichnis


[1] Angermann, Beuschel, Rau und Wohlfarth. *MATLAB - Simulink - Stateflow: Grundlagen, Toolboxen, Beispiele*. Oldenbourg, München, 6., aktualis. aufl. edition, 2009. 30, 32, 73

[2] Jaime Arias. *Energy Usage in Supermarkets - Modelling und Field Measurements*. text, Royal Institute of Technology, KTH, Energy Technology ; Stockholm, 2005. 4, 8, 73

[3] Bernhard Lahres und Gregor Rayman. *Objektorientierte Programmierung. das umfassende Handbuch*. Galileo computing. Galileo Press, Bonn, 2., aktualisierte und erw. aufl. edition, 2009. 5, 73

[4] Naturschutz und Reaktorsicherheit (BMU) Bundesministerium für Umwelt. Grafiken und Tabellen mit Daten zur Entwicklung der erneuerbaren Energien in Deutschland im Jahr 2010 . URL: http://www.erneuerbare-energien.de/inhalt/42038/2720/ (Zugriff am 18. August 2011). 3

[5] EnergieAgentur.NRW. Energieeffizienz im Lebensmittel-Einzelhandel. URL: http://www.energieagentur.nrw.de/unternehmen/page.asp?TopCatID=3695&CatID=3721&RubrikID=3743 (Zugriff am 20. Juli 2011). 4, 73

[6] Michael Kauffeld. *Stand der Technik von Supermarktkälteanlagen. Umwelteinfluss und Entwicklungspotential*. 2008. 10

[7] Martin Kleimaier. Netzintegration von Strom aus erneuerbaren Energiequellen: Zunehmende dezentrale Einspeisung erfordert eine Umrüstung der Netze. *Energy 2.0*, 1(7):45–47, 2008. 73

[8] F.A.T.M. Ligthart. Untersuchung zur Möglichkeit einer Abdeckung von Kühl- und Tiefkühlmobiliar in Supermärkten . Technical Report ECN-E–08-009, ECN Energy in the Built Environment, 2008. 4, 5

[9] Elke Lorenz, Johannes Hurka, and Detlev Heinemann. Solarleistungsvorhersage zur netzintegration von solarstrom. *24. Symposium Photovoltaische Solarenergie : 04. - 06. März 2009, Kloster Banz, Bad Staffelstein*, page 6, 2009. 3, 73

[10] Sean McGowan. Supermarket refrigeration going natural. *HVAC&R Nation*, pages 8–9, 2007. 4, 73

[11] Caroline Möller. Spezifikation und Simulation einer Kältelast mit Kältespeicher im Energieversorgungsnetz. Diplomarbeit, Technische Universität Berlin, Berlin, Juli 2010. 8, 36, 40, 41, 47, 73

[12] P. Pepper. *Programmieren lernen: Eine grundlegende Einführung mit Java.* Springer, 2008. 5

[13] Arnd Poetzsch-Heffter. *Konzepte Objektorientierter Programmierung: Mit Einer Einführung in Java.* Springer, Berlin, 2009. 5, 73

[14] J Probst. Ein Netzwerk für effiziente Kältetechnik. URL: http://www.energieagentur.nrw.de/unternehmen/page.asp?TopCatID=3695&CatID=3721&RubrikID=3743 (Zugriff am 20. Juli 2011), 2009. 4

[15] Risto Ciconkov und Arnd Hilligweg. Simulationsprogramme für Kälteanlagen Einsatz in Ausbildung und Praxis. *Technik im Bau, Fachzeitschrift für Technische Gebäudeausrüstung*, 77(3):64–70, 2004. 8, 73

[16] W. Schweizer. *MATLAB kompakt.* Oldenbourg Wissensch.Vlg, 2009. 6

[17] F. Steimle, H. Kruse, E. Wobst, and et al. Energiebedarf für die technische Erzeugung von Kälte. Technical Report Statusbericht Nr. 22, Deutscher Kälte- und Klimatechnische Verein (DKV) e.V., Stuttgart, 2002. 4

[18] Steve Völler. *Optimierte Betriebsführung von Windenergieanlagen durch Energiespeicher Elektronische Ressource.* Universitätsbibliothek Wuppertal, Wuppertal, 2010. 3, 73

[19] Michael Weigend. *Objektorientierte Programmierung mit Python 3. Einstieg, Praxis, professionelle Anwendung.* mitp-Verl., Heidelberg u.a., 4., aktualisierte aufl. edition, 2010. 5, 73