Christoph G. Trabert

Smart Operation of Smart Grids –
Network Risk Limiting Dispatch for
Integrating Renewable Power

Master Thesis

Power Systems Laboratory,
Dept. Information Technology and Electrical Engineering
Swiss Federal Institute of Technology (ETH) Zurich

Precourt Institute for Energy, Dept. Civil and Environmental Engineering
Stanford University

Electrical Engineering and Computer Science
University of California, Berkeley

PSL 1344

Supervisors
Prof. Pravin Varaiya (UC Berkeley)
Prof. Ram Rajagopal (Stanford University)
Prof. Göran Andersson (ETH Zurich)
Andreas Ulbig (ETH Zurich)

May 2014
For my father and grandfather.
Abstract

High renewable energy penetration increases the uncertainty of production and hence uncertainty of the net load demand. Therefore it imposes potentially high costs for the transmission system operator (TSO). In this thesis, a multistage stochastic dispatch problem is considered, namely risk limiting dispatch (RLD). With this approach the TSO decides the production, i.e. its own generation or purchased forward energy, over a time interval at each stage. The accumulated energy blocks must suffice to match the net load demand at each time $t$. Hence, the operating risk (i.e. reliability) of the power system is managed, while at the same time the expected costs to fulfill the net load demand can be minimized.

First, the fundamentals for an RLD strategy in uncongested networks is derived. Afterwards, a network applicable extension is developed: network risk limiting dispatch (NRLD), which also accounts for line constraints in a distributed network and especially for networks with one congested link.

The tractability of two computational approaches is analyzed with a discretization based dynamic programming approach, including the derivation of an analytic solution for speeding up computation, as well as a sample average approximation.
Acknowledgment

Attribution error is a beautiful idea from the field of psychology. It is the false believe that we got to where we are by the things that have done ourselves. In other words, when we succeed, we believe it is by our own special talent. The fact of the matter is that none of us exists alone. We are dwarfs standing on the shoulders of giants.

First of all, I would like to show my deepest gratitude to my supervisor Andreas Ulbig for his dedicated support and encouragement throughout this project. At the same time, I want to express great appreciation to Prof. Pravin Varaiya, who gave me important fundamental as well as advanced insights into the topic and helped me to stay focused.

Furthermore, my profound thanks go to Prof. Ram Rajagopal and his PhD student, Junjie Qin, for long, enlightening and inspiring discussions especially about algorithmic implementations, leading to understanding and intuition.

Additionally, I thank Prof. Göran Andersson and the Power Systems Laboratory (PSL) for providing an excellent framework to undertake the project-related research while being at ETH Zurich as well as abroad.

Some other particularly helpful giants were:

Prof. Ashutosh Nayyar,
Prof. Kameshwar Poolla,
Dr. Bernd Koch,
Marcus Hildmann,
Christoph Baumgart,
Matthias Dietzfelbinger,
David Krammer,
Sarah Belkner,
Dr. Andrea Trabert.

The project would not have been finished without them. I am genuinely grateful.

\footnote{Dwarfs standing on the shoulders of giants (Latin: \textit{nanos gigantum humeris insidentes}) is a metaphor with a contemporary interpretation meaning "one who discovers by building on previous discoveries", e.g. as Isaac Newton remarked in a letter to his rival Robert Hooke dated February 5, 1676 \cite{1}.}
# Contents

Abstract iii  
Acknowledgment v  
Contents viii  
List of Figures x  
List of Symbols xii  

## 1 Introduction 1  
1.1 Motivation 1  
1.2 Goal 1  
1.3 Content 2  
1.4 Notation 3  

## 2 Mathematical Tools 5  
2.1 Convex Optimization 5  
2.1.1 Introduction 5  
2.1.2 Convexity 6  
2.1.3 Lagrange Duality Theory 7  
2.1.4 Optimality Conditions 9  
2.2 Dynamic Programming 10  
2.2.1 Introduction 10  
2.2.2 Dynamic Programming Algorithm 12  
2.2.3 Dealing with Constraints 13  
2.2.4 Speeding Up Dynamic Programming 13  
2.3 Sample Average Approximation 14  
2.4 Aspects of Statistics 14  

## 3 Risk Limiting Dispatch 17  
3.1 Introduction 17  
3.2 Conceptual Framework 18  
3.3 Theoretical Derivations 20
## CONTENTS

3.3.1 Basic Insights .................................................. 20  
3.3.2 Advanced Insights ............................................ 22  
3.4 Discussion Of Assumptions ...................................... 23  
  3.4.1 Prices ....................................................... 23  
  3.4.2 Net Demand ................................................ 27  
3.5 Example In Two Stages .......................................... 27  
3.6 Example In Three Stages ....................................... 28  

4 Network Risk Limiting Dispatch ................................. 31  
  4.1 Introduction .................................................. 31  
  4.2 Two Stages ................................................... 32  
  4.3 Structure Of $J^R$ – An Analytic Approach ................ 34  
  4.4 Probability Distribution ...................................... 39  
  4.5 Three Or More Stages ........................................ 39  
  4.6 N-bus Network With Single Congested Line ................ 43  
  4.7 N-bus Network With Multiple Congested Lines .......... 44  

5 Computational Tractability ...................................... 53  
  5.1 Dynamic Programming Based Approach ...................... 53  
  5.2 Sample Based Approach ...................................... 54  
  5.3 Computational Tools .......................................... 56  
  5.4 Results ....................................................... 56  
  5.4.1 Dynamic Programming Based Approach .................. 56  
  5.4.2 Sample Based Approach .................................. 60  

6 Conclusion and Outlook ........................................... 63  
  6.1 Conclusion ................................................... 63  
  6.2 Future Research ............................................. 64  

A Source Code ......................................................... 65  

B DVD Contents ....................................................... 73  

Bibliography .......................................................... 75
# List of Figures

1.1 Forecast error vs. forecast horizon. ........................................ 2

2.1 Illustration of truncation at $[-2, 2]$. ........................................... 16

3.1 Intraday spot price in Germany in 2012. ................................. 25

3.2 Day-ahead price in Germany in 2012. ........................................ 25

3.3 Difference between intra-day and day-ahead prices in 2012 in Germany. .................................................. 26

3.4 Supply and demand. ............................................................. 26

4.1 Setup with two nodes and one connecting line in between. ....... 33

4.2 Representation of $J^R$ (with $x_i = d_i - g_i$ for $i = \{1; 2\}$). .... 35

4.3 Two-dimensional representation of the ten different regions. ......... 37

4.4 Two-dimensional representation of reduced set of five regions. .... 38

4.5 Probability distribution. ........................................................... 39

4.6 Exemplary probability distribution, entry wise multiplied with value of cost function $J^R$. .................................................. 40

4.7 Representation of $J^R(x)$ a smoothed version of $J^R(x)$. ........ 41

4.8 Representation of $J^H(x)$, based on the functions $J^R(x)$ and $\hat{J}^R(x)$. .................................................. 42

4.9 Representation of $J^D(x)$, based on the expectation $\hat{J}^H(x)$. .... 42

5.1 Ternary tree, not recombining, to illustrate growth rate. ............. 55

5.2 Computation time for various numbers of stages (DP approach). .......................... 57

5.3 Relative efficiency gain factor of using the analytic solution. ....... 58

5.4 Normalized computation time for various numbers of stages, two nodes (DP approach). .................................................. 58

5.5 Computation time for various numbers of stages and discretization granularities, two nodes (DP approach). .......................... 59

5.6 Mean total cost (realized), for two and three stages for a dynamic programming implementation, with exponential fit. .......... 60

5.7 Computational run time for various numbers of samples, for three stages in two nodes (sample based approach). .................... 61

5.8 Mean total cost, for three stages, with exponential fit (sample based approach). .................................................. 61
List of Symbols

$c_{ij}$   Line power capacity from bus $i$ to bus $j$.
$f$       Power flow between two specific nodes.
$l$       Actual demand.
$w$       Renewable (uncontrolled) generation.
$d$       Net demand (i.e. $d = l - w$).
$\epsilon$ Prediction error (of net demand).
$q$       Generalized price.
$g$       Generation level at first stage.
$g^D$     Day ahead generation.
$g^H$     Hour ahead generation.
$g^R$     Real time (fast ramping) generation level (last stage).
$\hat{d}^D$ Day ahead forecast (available at day ahead stage).
$\hat{d}^H$ Hour ahead forecast (available at hour ahead stage).
$d$       Real time forecast (available at real time).
$\epsilon^{DH}$ Difference between day ahead and hour ahead forecast.
$\epsilon^{HR}$ Difference between hour ahead and real time net demand realization.
$\alpha$  First stage marginal generation cost.
$\alpha^D$ Day ahead marginal generation cost.
$\alpha^H$ Hour ahead unit marginal generation cost.
$\alpha^R$ Real time unit marginal generation cost.
$x$       System state (e.g. accumulated generation).
$d$       System input or decision (e.g. generation).
$V(\cdot)$ Generalized value function.
$J(\cdot)$ Minimum cost function.
$J^D(\cdot)$ Minimum day ahead cost function.
$J^H(\cdot)$ Minimum hour ahead cost function.
$J^R(\cdot)$ Minimum real time cost function.
$E[\cdot]$ Stochastic expectation.
$P[\cdot]$ Stochastic probability.
Chapter 1

Introduction

1.1 Motivation

Governments in many countries seek a drastic reduction of fossil-fueled electricity generation reducing carbon and other emissions in order to combat more local environmental pollution. A significant contribution from the electricity sector is crucial in order to meet this ambitious goal. However, an increasing renewable energy sources (RES) share in the generation mix brings up challenges due to their intermittent nature (cf. [2]). Despite these challenges, increasing shares of partly fluctuating RES are envisioned to play the major role in electricity generation in the upcoming decades. How power systems should best be upgraded in order to accommodate fluctuating RES integration and power market activities is a highly relevant and still open research question.

One key option to tackle the challenges of RES integration is a demand response and storage equipped smart grid, which manages and controls the increasingly complex power system.

Some of the smart grid components, such as renewable energy sources as well as consumer choice and smart appliances, increase uncertainties in both demand and supply. In contrast, other smart grid components provide more detailed information about the power system and improve controllability of power in-feed and out-feed, by sensors, smart meters, demand response as well as advanced communication means. Making use of the full potential for both the overall system efficiency and carbon reduction is not possible using conventional power grid operating paradigms such as the $N - 1$ criterion.

1.2 Goal

The goal of this master thesis is to derive an operating paradigm, which treats generation as a heterogeneous commodity of intermittent and stochastic power in-feed, while managing the risk of production and consumption uncertainty.
CHAPTER 1. INTRODUCTION

Figure 1.1: Forecast error vs. forecast horizon\textsuperscript{3}. Source: Iberdrola Renovables.

Specifically, a so called ”Risk limiting dispatch” (RLD) strategy shall be derived, which reduces the operating risk (related to reliability)\textsuperscript{2} and/or reduces the minimum expected cost (related to economics). This can be achieved by recognizing that risk from current decisions can be mitigated by future decisions in a multistage power market, e.g. 24 h-ahead, 1 h-ahead, 15 min-ahead and 5 min-ahead. Less time ahead means more accurate information, cf. figure 1.1. Additionally, after thoroughly explaining how RLD works, this operating paradigm should be applied to a network with more than one node. Specifically, line constraints between nodes or buses, should be respected by optimizations based on a DC power flow model.

To conclude, the objective of this master thesis is to

1. derive a network applicable RLD strategy, 
2. develop and implement scalable RLD algorithms, and
3. analyze the efficiency and effectiveness of these algorithms.

1.3 Content

The content of this thesis is divided into the following parts. First, after this introduction, essential mathematical tools, which are needed for understanding

\textsuperscript{2}e.g. by setting a Value of Lost Load (VOLL)

\textsuperscript{3}The zero error at zero forecast horizon comes from assuming real-time communication of actual load, production and hence, net demand.
1.4. **NOTATION**

Some of the derivations, are outlined in chapter 2. Subsequently, some basic derivations as well as examples of pure risk limiting dispatch are presented in chapter 3. Then, risk limiting dispatch is extended to network risk limiting dispatch in chapter 4, introducing the possibility of power flows between nodes in an abstract power grid. Afterwards, the computational approaches, which were theoretically introduced in chapter 2, are applied and extensively described in chapter 5, where also their computational tractability is analyzed.

### 1.4 Notation

Some general remarks on used notation:

- Vector variables are written in **bold italic** typeface.

- Capital letters (e.g. $D$) correspond to **stochastic variables**, while their small written counterparts (e.g. $d$) mean the corresponding **realization**.

- A hat marks a variable as a forecast (e.g. $\hat{d}$).

- An asterisk indicates the optimal value of a variable in an optimization problem (e.g. $g^*$).

- Subscript $i$ of a variable usually associates this variable to bus or node $i$. 
Chapter 2

Mathematical Tools

In this chapter, essential characteristics about mathematical tools, which were used for derivations in this thesis, are explained. The fundamentals of convex optimization as well as the basics of used optimization approaches, like dynamic programming and a sample based approach, are introduced. Furthermore, some peripheral cornerstones about probability distributions are laid out.

2.1 Convex Optimization

2.1.1 Introduction

An optimization problem generally means to maximize (or minimize) an objective function, while ensuring constraints on state or decision variables to remain satisfied. A formulation of a generic optimization problem is the following

\[
\inf_{x} f(x) \quad \text{s.t.} \quad x \in S \subseteq Z
\]

where \(x\) collects the decision variables, \(Z\) stands for the domain of decision variables and \(S \subseteq Z\) refers to the feasible set of decisions.

The function \(f: Z \to \mathbb{R}\) assigns a cost \(f(x) \to \mathbb{R}\) to each decision \(x\).

Often, a notation will be used, where the optimal cost function value is called \(J^* = \inf_{x \in S} f(x)\); in this case \(J^*\) represents the greatest lower bound of \(f(x)\) over the set \(S\). If \(J = -\infty\), we call the problem unbounded below; if the the set \(S\) is empty, we call it infeasible, and by convention \(J^* = +\infty\). The case \(S = Z\) refers to an unconstrained problem.

If there exists a decision \(x\), which yields \(f(x) = J^*\), then it is called a (global) optimizer \(x^*\) of

\[
J^* = \min_{x \in S} f(x).
\]

In a continuous optimization, \(Z\) is a finite-dimensional subset of a Euclidean vector space \(\mathbb{R}^s\), and the subset of feasible vectors is defined by a list of real-
valued functions, equalities $h_i(x)$ and inequalities $f_i(x)$, i.e.

$$\begin{align*}
\min & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \quad h_i(x) = 0, \quad i = 1, \ldots, p \\
& \quad x \in Z,
\end{align*}$$

where the domain $Z$ is the intersection of the domains of the cost function and the constraint functions. However, usually $x \in Z$ is implicitly defined by the optimization problem and can be omitted.

A point $\bar{x} \in \mathbb{R}^s$ is a feasible solution for the problem, if it belongs to $Z$ and therefore satisfies all inequality and equality constraints. Furthermore, we call the $i$-th inequality constraint an \textit{active constraint} at $x = \bar{x}$, if $g_i(\bar{x}) = 0$. If $g_i(\bar{x}) < 0$, it is called an \textit{inactive constraint}. Equality constraints are always active [3].

### 2.1.2 Convexity

The fundamental characteristic of convexity is that \textit{local optimizers} always are \textit{global optimizers}. As stated in [3], a set $S \in \mathbb{R}^s$ is \textit{convex}, if it has the property

$$\lambda x_1 + (1 - \lambda) x_2 \in S \quad \forall x_1, x_2 \in S, \lambda \in [0, 1].$$

Furthermore, a function $f : S \to \mathbb{R}$ is convex if $S$ is convex and

$$f(\lambda x_1 + (1 - \lambda) x_2) \leq \lambda f(x_1) + (1 - \lambda) f(x_2) \quad \forall x_1, x_2 \in S, \lambda \in [0, 1].$$

The relation

$$f(\lambda x_1 + (1 - \lambda) x_2) < \lambda f(x_1) + (1 - \lambda) f(x_2) \quad \forall x_1, x_2 \in S, \lambda \in (0, 1)$$

refers to \textit{strict convexity}.

Various operations on convex sets or functions preserve their corresponding property of convexity. A detailed list can be found in [4]. The most important are the following.

1. The empty set is a convex set.

2. The intersection of an arbitrary (even in infinite) number of convex sets is a convex set.

3. If $f_1, f_2, \ldots, f_N$ are convex functions, then $\sum_{i=1}^N \alpha_i f_i$ (with $\alpha_i \geq 0$ for $i = 1, \ldots, N$) is a convex function.

4. The composition of a convex function $f(x)$ with an affine mapping $x = Az + b$ generates a convex function $f(Az + b)$ in $z$. 
5. A linear function \( f(x) = c'x + d \) is both convex and concave.

6. A quadratic function \( f(x) = x'Qx + 2r'x + s \) is convex, if and only if \( Q \preceq 0 \) (positive semi-definite).

7. A quadratic function \( f(x) = x'Qx + 2r'x + s \) is strictly convex, if and only if \( Q \succ 0 \) (positive definite).

### 2.1.3 Lagrange Duality Theory

#### The Lagrangian

A standard optimization problem

\[
\begin{align*}
\min & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \quad h_i(x) = 0, \quad i = 1, \ldots, p
\end{align*}
\]

with variable \( x \in \mathbb{R} \) is considered. It is assumed, that its domain \( D \) is nonempty, not necessarily convex, and the optimal value of problem 2.1 is \( p^* \).

In Lagrangian duality, the constraint functions of the problem are added to the objective function. The Lagrangian is hence defined as

\[
L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x).
\]

It can be seen, that \( \lambda_i \) is multiplied with \( f_i(x) \) as the \( i \)th inequality constraint function, while analogously \( \nu_i \) is multiplied with \( h_i(x) \), the \( i \)th equality constraint function. In this setup, \( \lambda \) and \( \nu \) are called the dual variables or Lagrange multiplier vectors of problem 2.1, following [4].

#### The Lagrange Dual Function

The Lagrange dual function \( g(\lambda, \nu) \) is defined as the minimum value of the Lagrangian over \( x \):

\[
g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) = \inf_{x \in D} \left( f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x) \right).
\]

Imagining the Lagrangian unbounded below in \( x \), the dual function hence takes value \(-\infty\).

Note, that the dual function is the (pointwise) infimum of a family of affine functions of \((\lambda, \nu)\). Therefore it is concave, even when the original problem 2.1 is not convex [4].
Lower Bounds On Optimal Value

The dual function reveals a property, which helps to reduce the optimization effort: It yields lower bounds on the optimal value $p^*$ of problem 2.1. In fact, for any $\lambda \succeq 0$ and any $\nu$, it holds

$$g(\lambda, \nu) \leq p^*. \quad (2.2)$$

This can be shown, by considering $\tilde{x}$ as a feasible point, i.e., $f_i(\tilde{x}) \leq 0$ and $h_i(\tilde{x}) = 0$, and given $\lambda \succeq 0$. It follows

$$\sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{i=1}^{p} \nu_i h_i(\tilde{x}) \leq 0,$$

as each term of the first sum is negative or zero, and each term of the second sum is zero. This yields

$$L(\tilde{x}, \lambda, \nu) = f_0(\tilde{x}) + \sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{i=1}^{p} \nu_i h_i(\tilde{x}) \leq f_0(\tilde{x}).$$

Therefore

$$g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) \leq L(\tilde{x}, \lambda, \nu) \leq f_0(\tilde{x}).$$

As $g(\lambda, \nu) \leq f_0(\tilde{x})$ is true for every feasible point $\tilde{x}$, inequality 2.2 follows immediately [4].

The Lagrange Dual Problem

After revealing inequality 2.2 (there exists a lower bound that depends on some parameters $\lambda, \nu$) we can state the problem, what actually is the best lower bound that can be obtained from the Lagrange dual function. This yields the following optimization problem, the so called Lagrange dual problem of the (primal) problem 2.1:

$$\max \quad g(\lambda, \nu)$$

$$\text{s.t.} \quad \lambda \succeq 0. \quad (2.3)$$

In this context, $(\lambda^*, \nu^*)$ are the optimal Lagrange multipliers, if they are optimal for 2.3, and yield the optimal value $d^*$. Independently of whether the primal problem is convex, the Lagrange dual problem is a convex optimization problem, since the objective is concave and to be maximized under a convex constraint set [4]. In [5] is given more insight and interpretations of the Lagrangian multipliers, for example for shadow pricing and locational marginal pricing.
2.1. CONVEX OPTIMIZATION

Weak Duality

By definition in section 2.1.3, \( d^* \) is the best lower bound on \( p^* \) that can be obtained from the Lagrange dual function. Hence

\[
d^* \leq p^*,
\]

which is called weak duality (and true even for a non-convex primal problem). Accordingly, \( p^* - d^* \) is referred to as the optimal duality gap. The bound 2.4 can sometimes be used to find a lower bound on an optimal value for difficult (even non-convex) problems. It is often the case, that \( d^* \) can be found efficiently [4].

Strong Duality

If the optimal duality gap equals zero, i.e. it holds

\[
d^* = p^*,
\]

then the problem features strong duality. The best bound obtained from the Lagrange dual function is then tight. However, this characteristic does not hold in general. One important class of problems, where strong duality usually, but not always holds, is a primal problem 2.1, which is convex, i.e. of the form

\[
\begin{align*}
\min & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \quad Ax = b,
\end{align*}
\]

with \( f_0, \ldots, f_m \) being convex. A certificate of strong duality would for example be given by Slater’s condition, which is beyond the scope of the described mathematical background in this thesis.

In fact, with all inequalities being affine and equalities being linear, i.e. any LP in standard or inequality form, feasibility of the primal problem implies strong duality [4].

2.1.4 Optimality Conditions

We saw the standard optimization problem

\[
\begin{align*}
\min & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \quad h_i(x) = 0, \quad i = 1, \ldots, p
\end{align*}
\]

with the corresponding Lagrangian

\[
L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x).
\]
Written as vectors with \( \mathbf{x} \in \mathbb{R}^n \) yields

\[
\min_{\mathbf{x}} f_0(\mathbf{x}) \\
\text{s.t. } \mathbf{f}(\mathbf{x}) \leq 0, \\
\mathbf{h}(\mathbf{x}) = 0.
\] (2.8)

Accordingly, the Lagrangian changes to

\[
L(\mathbf{x}, \lambda, \nu) = f_0(\mathbf{x}) + \lambda^T \mathbf{f}(\mathbf{x}) + \nu^T \mathbf{h}(\mathbf{x}),
\]

with \( \mathbf{x} = [x_1 \ x_2 \ \ldots \ x_n]^T \),
\( \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) & \ldots & f_m(\mathbf{x}) \end{bmatrix}^T \) and
\( \mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) & h_2(\mathbf{x}) & \ldots & h_p(\mathbf{x}) \end{bmatrix}^T \).

Let \( \mathbf{x}^* \) and \( (\lambda^*, \nu^*) \) be any primal and dual optimal points with zero duality gap. As \( \mathbf{x}^* \) minimizes \( L(\mathbf{x}, \lambda^*, \nu^*) \), the gradient vanishes in this point. Therefore, the following conditions are necessary for optimality:

\[
\nabla L(\mathbf{x}^*, \lambda^*, \nu^*) = \nabla f_0(\mathbf{x}^*) + \nabla \mathbf{f}(\mathbf{x}^*)^T \lambda^* + \nabla \mathbf{h}(\mathbf{x}^*)^T \nu^* = 0, \\
f(\mathbf{x}^*) \leq 0, \\
h(\mathbf{x}^*) = 0, \\
\lambda_i^* f_i(\mathbf{x}^*) = 0, \quad \lambda_i^* \geq 0, \quad i = 1, \ldots, m.
\] (2.9)

These so called Karush-Kuhn-Tucker (KKT) conditions must be satisfied for any optimization problems (even of non-convex nature) with differentiable objective and constraint functions for which strong duality is obtained.

In case the primal problem is convex, the KKT conditions are sufficient (and not only necessary) for the corresponding points to be primal and dual optimal and ensure zero duality gap [4].

### 2.2 Dynamic Programming

#### 2.2.1 Introduction

We want to make multiple decisions in stages to minimize a cost, which captures unwanted outcomes. It is important to note, that such decisions cannot be viewed in an isolated fashion: The desired low present cost generally have to be balanced with undesirably high future costs.

Dynamic programming accounts for this trade-off. Assuming optimality of subsequent decision stages, at each stage a decision is made based on the sum of present cost and expected future cost.
2.2. DYNAMIC PROGRAMMING

Following [6], the key ingredients to dynamic programming problem are:

1. An underlying discrete-time dynamic system

\[ x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots, N - 1, \]

where \( k \) indexes discrete time, \( x_k \in S_k \) is a state (in state space), \( u_k \in C_k \) is a control or decision variable (in control space), \( w_k \in D_k \) is a random parameter (e.g. disturbance; in disturbance space), \( N \) is a time horizon and \( f_k \) is a function describing the system dynamics.

2. Independent random variables \( w_k \), where the probability distribution of \( w_k \) might depend on \( x_k \) and \( u_k \), i.e. \( w_k \sim P_r(\cdot|x_k, u_k) \)

3. A constraint on \( u_k \), generally \( u_k \in U_k(x_k) \), e.g. \( u_k \geq 0 \).

4. A cost function additive over time

\[ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k), \]

where \( g_N(x_N) \) is a terminal cost occurring at the end of the process and \( g_k \) is a given nonlinear function.

As the \( w_k \) are random, typically one might formulate the optimization problem considering the expected cost

\[ \mathbb{E}_{w_k} \left[ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right], \]

5. Finally, a class of policies or control laws comprising a set of functions

\[ \Pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}, \]

where \( \mu_k \) maps states \( x_k \) into controls \( u_k = \mu_k(x_k) \), such that \( \mu_k(x_k) \in U(x_k) \) for all \( x_k \in S_k \).

Given a policy \( \pi \in \Pi \), the expected cost of starting at state \( x_0 \) is

\[ J_\pi(x_0) = \mathbb{E}_{w_k} \left[ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right]. \]

An optimal policy minimizes that cost

\[ J^{*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0), \]

where \( \Pi \) is the set of all admissible policies. The optimal cost function, as well depending on \( x_0 \), is denoted as

\[ J^{*}(x_0) := J^{*}(x_0). \]
2.2.2 Dynamic Programming Algorithm

The dynamic programming (DP) algorithm is based on a simple idea, the principle of optimality, which was originally popularized as a systematic tool by Bellman, e.g. in [7].

**Principle of Optimality** Let \( \pi = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\} \) be an optimal policy for the basic problem, and assume that when using \( \pi^* \), a given state \( x_i \) occurs at time \( i \) with positive probability. Consider the subproblem whereby we are at \( x_i \) at time \( i \) and wish to minimize the "cost-to-go" from time \( i \) to time \( N \)

\[
E \left[ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right].
\]

Then the truncated policy \( \{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\} \) is optimal for this subproblem (from [6]).

In other words, the "tail" portion of an optimal schedule must be optimal.

**The DP Algorithm** For every initial state \( x_0 \), the optimal cost \( J^*(x_0) \) of the basic problem is equal to \( J_0(x_0) \), given by the last step of the following algorithm, which proceeds backward in time from period \( N-1 \) to period 0:

\[
J_N(x_N) = g_N(x_N)
\]

\[
J_k(x_k) = \min_{u_k \in U_k(x_k)} E \left[ g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right],
\]

for \( k = 0, 1, \ldots, N-1 \), where the expectation is taken with respect to the probability distribution of \( w_k \), which depends on \( x_k \) and \( u_k \). Furthermore, if \( u_k^* = \mu_k^*(x_k) \) minimizes the right side of equation 2.10 for each \( x_k \) and \( k \), the policy \( \pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\} \) is optimal (from [6]).

\( J_k(x_k) \) is hence the optimal cost for an \((N-k)\)-stage problem starting at state \( x_k \) and time \( k \), and ending at time \( N \). Consequently, \( J_k(x_k) \) is called the cost-to-go at state \( x_k \) and time \( k \), while \( J_k \) refers to the cost-to-go function at time \( k \).

Regarding the computational burden, a proportional relation to the number of possible values of \( x_k \) can be established. This might lead to excessive computational requirements for complex problems, however, DP is the only general approach for sequential optimization under uncertainty (which also might be serving as the basis for more practical suboptimal approaches).

To wrap it up, a general DP algorithm works as follows:

- **Initialization:**
  \[
  J_N(x_N) = g_N(x_N) \quad \forall x_N \in S_N
  \]
2.2. DYNAMIC PROGRAMMING

- Recursion:
  \[ J_k(x_k) = \min_{u_k \in U_k(x_k)} \mathbb{E} [g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))], \]
  where for each recursion step, the optimization is performed over all possible values \( x_k \in S_k \). It is not known a priori, which states will be visited during backtracking.

- The dynamics \( f_k(x_k, u_k, w_k) \) are described, with the following example function for a state update:
  \[ x_{k+1} = x_k + u_k - w_k, \]
  where \( x_k \) is the state at time \( k \), \( u_k \) is the input at the same time and \( w_k \) refers to a (usually random) update error (i.e. disturbance).

2.2.3 Dealing with Constraints

Dynamic Programming algorithms generally do not comprise the functionality of satisfying constraints. However, one can formulate the dynamic program with solving a constrained LP with an appropriate solver at each (generally recursive) step.

2.2.4 Speeding Up Dynamic Programming

Dynamic Programming uses variables, i.e. referring to state, decision and outcome, in potentially very high dimensionality, depending on the number of recursive steps. Generally, this leads to exponential growth of computational effort to solve a problem. However, several methods might speed up computations by reducing this effect.

- There exist techniques to use preprocessing with estimation of lower and upper bounds to reduce the problem to a core problem. The core problem is then solved by standard dynamic programming\(^4\). Experimental performance tests using random numbers revealed potentially very large speed-ups due to reducing the dynamic programming problem to a dynamic programming core problem.

- Another method is to reuse precomputed solutions, formulated in a problem-customized way over various steps by writing them in a table. Furthermore, similar problems can be solved in an approximated fashion by finding the nearest neighbor of a concurrent problem in the set of precomputed problems. Potentially, interpolation within such a table is possible, and a-priori-knowledge about the structure of such a table can also be employed.

A promising approach is also to use Approximate Dynamic Programming. It incorporates forward induction into the concept of dynamic programming and hence reduces the dimensionality, i.e. the need of looping over all possible states. More specifically it generates (random) samples of what might happen in the next step, based on various sources of random information, e.g. with methods related to Monte Carlo approaches. Please be referred to [8] and [9] for further details.

2.3 Sample Average Approximation

Sample average approximation approaches are linked to Monte Carlo methods. They rely on the law of large numbers; typically one runs simulations many times in order to obtain the distribution of an unknown probabilistic system. These methods are especially well fitting to modeling phenomena with significant uncertainty in inputs, such as the calculation of risk in business as well as engineering.

Generally, the following steps are carried out during a Monte Carlo simulation (cf. [10]):

1. Define a domain of possible inputs.
2. Generate inputs randomly from a probability distribution over the domain.
3. Perform a deterministic computation on the inputs.
4. Aggregate the results.

In a later chapter it will be shown, how a sample average approximation approach can be applied to a specific example.

2.4 Aspects of Statistics

In this section a few fundamental concepts of statistics and probability are very briefly introduced. Probability forms the basis of any of the random processes which play a role in this project: It is a measure of the likelihood of a certain event to occur.

Probability Distribution

A probability distribution assigns a probability to each measurable subset of the possible outcomes of a random experiment. A probability distribution can either be univariate or multivariate. A univariate distribution returns the probabilities of a single random variable for various values; a multivariate distribution, also
called a \textit{joint probability distribution}, gives the probabilities of a random vector, taking on various combinations of values. Important and commonly encountered univariate probability distributions include the binomial distribution, the hypergeometric distribution, and the normal or Gaussian distribution [11]. One must distinguish between discrete and continuous probability distributions. In the discrete case, one can easily assign a probability to each possible value. However, when a random variable takes values from a continuum, probabilities can be nonzero only if they refer to intervals. If the random variable is real-valued, the cumulative distribution function (CDF) gives the probability that the random variable is not larger than a given value. Furthermore, the CDF is the integral of the probability density function (PDF) provided that this function exists [11].

The \textit{expected value} of a probability distribution refers to the weighted average of all possible values, using their probabilities as their weights or the corresponding continuous analog.

\textit{Standard deviation} is another important aspect of probability distributions: It is the square root of the variance, and hence a measure of dispersion. It shows how much variation from the average exists. A low standard deviation indicates that the data points tend to be very close to the mean (also called expected value); a high standard deviation indicates that the data points are spread out over a large range of values [12].

\textbf{Probability Distribution Truncation}

A set of discrete points, on which various calculations, e.g. solving LPs, are conducted, i.e. the solution space, is always bounded (truncated) to be computational tractable. However, a normal distribution of a random variable is generally not confined to a given set. This might lead to inaccuracies, especially at the borders of the set. More specifically, truncation and non-truncation involve different model assumptions, in particular, whether the randomness, e.g. forecast error updates, is bounded. Consequently, the probability distribution is truncated to fit the given set. A one dimensional example of that is given in figure 2.1.
Figure 2.1: Illustration of truncation at $[-2, 2]$. 
Chapter 3

Risk Limiting Dispatch

The existing power system is designed to provide electric power to meet the (forecasted) electric load demand. Uncertainties arise from outages and unpredicted fluctuations in demand, and (especially) in renewable generation. A consequence might be a loss of load event, when the generation level does not meet the current demand, and a part of the load demand generation is required to be shedded. To minimize the probability of such events, the Transmission System Operator (TSO) schedules an additional reserve capacity to compensate the uncertainty due to generation uncertainties and load forecast errors. Typically, the provided reserve capacities are in the range of $3\sigma$, with $\sigma$ being the standard deviation of net demand forecasting error, amounting to about 1...2% of total load (e.g. in California, USA). This level of uncertainty is subject to increase due to strong efforts in integrating renewable energy into the grid, causing an increasing need of additional reserve capacity.

3.1 Introduction

Demand response, i.e. the management of customer consumption of electricity in response to supply conditions, is considered one of the measures, that improves efficiency delivering a significant carbon reduction and might be even cost saving. However, the high penetration of renewable generation and demand response are a tremendous challenge onto the power system’s operation. In a traditional power system, power was generated by a small number of (controllable) large generators with high inertia, matching a (passive) load demand, with a statistically predictable aggregate behavior, anywhere and anytime. This, actually existing system, was not designed for variable stochastic renewable generation and interactive demand response.

To accommodate the vital changes, a functionally enriched (in communications and control) power system is required, e.g. as outlined in [13]. Accompanying the induced hardware changes, a new operating paradigm of the system
is required to extensively utilize the carbon emission reduction potential. If a renewable generation capacity is treated as an "equivalent" of a conventional generator to fit into traditional operating principles, a large reserve capacity needs to be provided [14]. Conventional wisdom is that this will seriously affect grid operation when renewable penetration is more than 20%. Furthermore, the \((N - 1)\) worst case dispatch approach seems to be outdated; the future power grid will provide tighter feedback between supply and demand for a secure and efficient integration of distributed (renewable) generation and demand response. Consequently, a new operating paradigm is proposed in [13], called risk-limiting dispatch (RLD). More recently, the certain aspects of risk limiting dispatch are elaborated in [15] and [16].

### 3.2 Conceptual Framework

The new operating paradigm is enabled by a two-way low-latency communication network, providing real-time information about supply and demand, and considering the stochastic nature of renewable sources as well as demand response. It is based on the principle of limiting the risk in operation, i.e. the risk of not meeting operating constraints and avoiding potentially cascading blackouts. To safely and securely operate a power system, two basic properties have to be fulfilled (cf. [13]):

1. **Power balance constraints:** The power system needs to be in power balance, at all times, at all points. Mathematically this could be expressed by a set of differential and algebraic equations (depending on the focused time-scale)

   \[
   \begin{align*}
   \dot{x} &= f(x, u, p) \\
   0 &= g(x, u, p)
   \end{align*}
   \]

   where equation 3.1 refers to the dynamics of the generator, with \(p\) being the load demand and equipment status vector, \(u\) the control variables vector and \(x\) the state variable vector, e.g. voltage magnitudes, phasor angles, and frequency. The algebraic equation 3.2 on the other hand reflects the steady-state power balance.

2. **Operating limit constraints:** The power system is required to satisfy operating limits, e.g. line flow limits, voltage limits

   \[
   h(x, u, p) \leq 0.
   \]

   The power balance constraints and the operating limit constraints together form the operating constraints of the grid. The control problem then is set up as follows: While maintaining the operating constraints (equations 3.1 – 3.3), \(u\) is
tuned, such that an objective function $F$ is optimized, e.g. minimizing total cost or maximizing social welfare:

$$\max_u F(x, u, p) \quad \text{s.t.} \quad \dot{x} = f(x, u, p)$$
$$g(x, u, p) = 0$$
$$h(x, u, p) \leq 0,$$  \hspace{1cm} (3.4)

where $u(p)$ and $x(u(p), p)$ depend on the parameters $p$. For pure contingency analysis this problem might be reduced to a standard *power flow problem*, as transients could then be neglected, i.e.:

$$\max_u F(x, u, p) \quad \text{s.t.} \quad f(x, u, p) = 0$$
$$g(x, u, p) = 0$$
$$h(x, u, p) \leq 0.$$  \hspace{1cm} (3.5)

Note, that parameter $p$ incorporates variables of stochastic nature, e.g. renewable generation and demand response. Hence, it might be considered as a vector of random variables. Consequently, $u(p)$ and $x(u(p), p)$ are functions of these random variables and therefore are random themselves.

In a simple embodiment of the model, the transmission constraints are ignored and only the *adequacy of generation* to meet the load demand. Then, the operating constraints (equations 3.1 – 3.3) can be expressed by the simple inequality

$$S - D \geq 0.$$  \hspace{1cm} (3.6)

The *degree* of not meeting the operating constraints can be defined as severity index $n$ (cf. [13]) for the simple adequacy model

$$n = D - S.$$  \hspace{1cm} (3.7)

Analogously, for the standard power flow model, the corresponding severity index $n$ might be defined as

$$n(p) = \max_i h_i(x(u(p), p), u(p), p),$$  \hspace{1cm} (3.8)

where $h_i$ is the $i$-th component of the vector function $h$.

Now, it might be interesting to define the *operating risk*. This measure could for example now be taken as a real-valued function $R(n(p))$ of the random variable $n(p)$, which is generally the degree of not meeting the operating constraints.
CHAPTER 3. RISK LIMITING DISPATCH

In one realization, the corresponding operating risk might be the probability that \( n(p) \) is positive

\[
R(n(p)) = P\{n(p) > 0\},
\]

which is also known as a loss-of-load probability (LOLP) used in conventional reliability assessment in generation planning. The risk limiting dispatch consequently forms as

\[
\max F(x, u, p) \quad \text{s.t.} \quad R(n(p)) \leq \epsilon,
\]

where \( \epsilon \) is the acceptable risk level.

The probability distribution of the random variable \( p \) (the parameters) is derived from real-time measurements and forecasts. It therefore is a conditional probability distribution \( P\{p \mid Y_t\} \) with \( Y_t \) consisting of all current and past sensor readings.

3.3 Theoretical Derivations

3.3.1 Basic Insights

For the following derivations, some assumptions are specified. Let us take stages \( k \), with a day-ahead stage \((k = 1)\), an hour-ahead stage \((k = 2)\), and a realization time \((k = 3)\). In the last stage the actual demand is apparent, unlike in the preceding stages, where only a distribution of the demand is known. Furthermore, the inputs or decisions are called \( u_k \) and refer to the “amount to be purchased/produced (or sold)”. The state \( x_k \) means the “power on stock”.

The dynamics are consequently described by

\[
\begin{align*}
  x_1 &= 0 \\
  x_2 &= x_1 + u_1 = u_1 \\
  x_3 &= x_2 + u_2 = x_1 + u_1 + u_2 = u_1 + u_2.
\end{align*}
\]

The demand is referred to by a random variable \( D \) with expectation \( \hat{d} \). Demand forecasts are assumed to be disturbed by noise \( w_k \), and the corresponding demand realization is then called \( d \).

The penalty function is taken, with \( K \) as a scaling factor\(^5\), as in

\[
K(d - x_3)^2.
\]

The value function consists of the price \( p_i u_i \) as well as the penalty function. In the following it is described, how the two stages are dependent on each other. They are evaluated in a backwards manner (as in standard dynamic programming).

\(^5\)Other penalty functions like \( K \max(d - x_3, 0) \) are also possible, but for the ease of derivation, only a quadratic cost function is used here.
3.3. THEORETICAL DERIVATIONS

Second Stage \( x_3 = x_2 + u_2 \)

\[
\begin{align*}
\min_{u_2} & \quad p_2 u_2 + K (x_2 + u_2 - d)^2 \\
\iff & \quad p_2 + 2K (x_2 + u_2 - d) = 0 \\
\iff & \quad u_2 = -\frac{p_2}{2K} + d - x_2,
\end{align*}
\]

hence, the value function at stage 2 gets

\[
V_2(x_2, d) := \min_{u_2} p_2 u_2 + K (x_2 + u_2 - d)^2
\]

\[
= -\frac{p_2^2}{4K} + p_2 (d - x_2).
\]

Note here, that equation 3.12 is linear in \( x_2 \) and hence convex in \( x_2 \). For convexity with respect to \( u_2 \) in function 3.11 it holds, that the first summand \( p_2 u_2 \) is linear in \( u_2 \) and hence (non strictly) convex. The second summand \( K (x_2 + u_2 - d)^2 \) is strictly convex in \( u_2 \), like a quadratic function \( f(x)^2 \) is convex in \( \mathbb{R} \). Following [4], a positive weighted sum of a (non strictly) convex and a (strictly) convex function is a (strictly) convex function.

First Stage \( x_2 = x_1 + u_1 \)

Suppose \( x_1 = 0 \) yielding \( x_2 = u_1 \), and furthermore that \( p_1 \) and the probability density function \( f(d) \sim N(\hat{d}, \sigma^2) \) are known. The realization \( d \) of \( D \) is not known at this stage.

The value function hence yields the following first stage problem.

\[
\begin{align*}
\min_{u_1} & \quad p_1 u_1 + \mathbb{E}_D [V_2(x_2, D)] \\
\min_{u_1} & \quad p_1 u_1 + \mathbb{E}_D [V_2(u_1, D)] \\
= & \quad \min_{u_1} p_1 u_1 + \int_0^\infty V_2(u_1, d) f(d) \, dd \\
= & \quad \min_{u_1} p_1 u_1 - \frac{p_2^2}{4K} - p_2 u_1 + p_2 \int_0^\infty df(d) \, dd \\
= & \quad \min_{u_1} p_1 u_1 - \frac{p_2^2}{4K} - p_2 u_1 + p_2 \hat{d} \\
= & \quad \min_{u_1} u_1 \cdot (p_1 - p_2) - \frac{p_2^2}{4K} + p_2 \hat{d},
\end{align*}
\]

which yields

\[
\begin{align*}
\quad u_1 &= +\infty \quad \text{if} \quad p_2 > p_1 \\
\quad u_1 &= -\infty \quad \text{if} \quad p_2 < p_1.
\end{align*}
\]
CHAPTER 3. RISK LIMITING DISPATCH

This result is only of theoretical matter. It basically says to buy an infinite amount of energy in the first stage, and then sell it again in the second stage, if the second stage prices are higher than first stage prices. If that was possible in reality, an infinite amount of profit would result.

Concerning convexity of function 3.13 in $u_1$, please note its affine character in $u_1$, revealing that it is (non strictly) convex with respect to $u_1$.

3.3.2 Advanced Insights

To get to a more applicable model, the pricing is now revisited and worked out in more detail. Hence, we assume a different price for buying and selling (for each stage): $p_1^b$ and $p_1^s$ for selling and buying unit price in stage one, and correspondingly $p_2^b$ and $p_2^s$ for stage two. Referring to [15], the following order is assumed:

$$p_2^b > p_1^b > p_1^s > p_2^s,$$

assuring that buying is more expensive than selling; and furthermore, that buying prices increase and selling prices decrease, as the time of delivery approaches.$^6$

Second Stage $x_3 = x_2 + u_2$

As before, but with different prices for selling and buying:

$$\begin{align*}
\min_{u_2} p_2(u_2)u_2 + K(x_2 + u_2 - d)^2 \\
\iff p_2(u_2) + 2K(x_2 + u_2 - d) = 0 \\
\iff u_2 = -\frac{p_2(u_2)}{2K} + d - x_2,
\end{align*}$$

with $p_2(u_2) = \begin{cases} p_2^s & \text{if } u_2 < 0, \\ p_2^b & \text{if } u_2 \geq 0. \end{cases}$

The minimization of function 3.14 needs convexity to make it computationally feasible. Its convexity will be shown in the following:

The first summand is locally lipschitz, i.e. differentiable almost everywhere, hence everywhere left- and right sided differentiable. Then it is (strictly) convex, if and only if its derivative is (strictly) monotonically increasing. This is true, if and only if $p_2^s < p_2^b$ is assumed. The second summand $K(x_2 + u_2 - d)^2$ is strictly convex in $u_2$, as before, like a quadratic function $f(x)^2$ is convex in $\mathbb{R}$. A positive weighted sum of a (non strictly) convex and a (strictly) convex function is a (strictly) convex function. Hence, function 3.14 is convex under the given price

---

$^6$If prices are assumed to be known in advance, this ordering has to be applied, otherwise unrealistic results might occur, and convexity of the problem is. If prices are seen stochastic, their respective ordering is not important.
3.4. DISCUSSION OF ASSUMPTIONS

3.4.1 Prices

The assumption, that generation costs are increasing with a decreasing time to execution might be justified with the fact, that large and inertial power plants, e.g. nuclear power plants, generally have to be powered up with a longer time horizon in advance than faster and more flexible power plants, e.g. gas fired
power plants. In this sense, more flexibility comes typically with higher generation costs. That basically describes the dispatch from a perspective of a TSO. Other justifications have mathematical origins, as the given price scheme ensures convexity of the algorithm and as well avoids trivialities, i.e. yielding infinite solutions.

However, from a market point of view, the decreasing selling price / increasing buying price assumption (e.g. in equation 3.17) is not generally true, cf. figures 3.1, 3.2 and 3.3, where day-ahead spot market and intra-day spot market prices are compared, for the full year 2012 in Germany\textsuperscript{7}.

Electricity prices are driven by their respective demand and supply. Hence, an unexpected over-supply of wind power may, in fact, decrease the intra-day spot price with respect to the initial corresponding day-ahead spot price. On the contrary, an unexpected under-supply, however, might increase the intra-day spot price over the initial corresponding day-ahead spot price (cf. figure 3.4).

An accordingly even more generalized optimization would implement a stochastic (and not deterministic) view on the pricing. Then the decreasing selling price and increasing buying price assumption would not be needed anymore (cf. to [18]). To support this, in [19], an analysis of the impacts of RES in-feeds on the spot market prices imposes is presented. Particularly, the dependency of spot prices on wind turbine and PV electricity in-feeds are studied. However, this is beyond the scope of this thesis and an interesting topic for future research.

\textsuperscript{7}It is the same delivery product: Power/Energy Delivery [in MW/MWh] for a given hour of the day procured first, on the day-ahead spot market or second, on the intra-day spot market (always for the same delivery hour [time-slot]).
3.4. DISCUSSION OF ASSUMPTIONS

Figure 3.1: Intraday spot price in Germany, average (black) and median (blue) for one day (24h), full year 2012.

Figure 3.2: Day-ahead price in Germany, average (black) and median (blue) for one day (24h), full year 2012.
CHAPTER 3. RISK LIMITING DISPATCH

Figure 3.3: Difference between intra-day and day-ahead prices in 2012, average (black) and median (blue) in Germany; note the negative difference about half of the time.

Figure 3.4: Supply ($S$) and demand ($D$), with respect to quantity ($Q$) and the corresponding price ($P$). Note an increased price settlement ($P_1 \rightarrow P_2$) for an increased demand ($D_1 \rightarrow D_2$), from [17].
3.4.2 Net Demand

In this thesis it is generally assumed that demand refers to net demand\(^8\). Net demand denotes the actual electricity demand (e.g. of an aggregation of households in a region), subtracted a corresponding occurrence of renewable energy in-feed. Hence, net demand is the electric demand, which actively has to be produced (e.g. by conventional power plants) to fulfill all electricity needs:

\[
\text{NetDemand} = \text{Demand} - \text{RenewableInfeed}.
\]

It should be noted, that net demand might be a negative number in rare cases. This happens, if the renewable energy production happens to exceed the actual demand, and hence needs to be curtailed or discarded. Net demand is of stochastic nature, as both the demand and the renewable in-feed are non-deterministic.

3.5 Example In Two Stages

Now, a simple example is derived, making use of the assumptions made before, e.g. on prices, seeking a generic linear program (LP) formulation. Considered are two stages, the first is assumed to be 24h ahead a specified delivery interval, e.g. five minutes, and the second stage is at real-time\(^9\).

The (random) net demand \(d\), which is to be satisfied, is known at real-time. In Stage 1, the TSO dispatches generation \(g\) based on information about random loads and random renewable production following a given probability distribution. However, by definition, \(g \geq 0\), i.e. the dispatch is required to be purchase or generation only.

In Stage 2, the TSO dispatches generation \(g^R\) after observing the actual loads and demands to balance the network.

A simple illustrative example is given in the following, where \(\alpha\) represents the day-ahead price for the specified delivery interval (with generation \(g\)), and \(\alpha^R\) corresponds to the price at the same delivery interval at real-time (with generation \(g^R\)).

- **Stage 2 (real-time)**

  \[
  J(\alpha^R, d - g) = \min_{g^R} F(\alpha^R)
  = \min_{g^R} \alpha^R \cdot (g^R)^+
  \]

  \[
  s.t. \quad g^R + g - d = 0
  \]

\(^8\)Also known as residual demand.

\(^9\)You could also see it as a 15min ahead or 1h ahead, but suppose to have perfect information about real-time net demand.
The basic idea is to tune $g^R$ to optimize the objective function\textsuperscript{10} $F$, while maintaining operating constraints.

- **Stage 1** (day-ahead)

  \[
  V^*(\hat{d}) = \min_{g \geq 0} \left[ \alpha^T g + \mathbb{E}[J(\alpha^R, d - g) | \hat{d}] \right].
  \] (3.18)

  After the 2nd stage problem is solved for basically any possible outcome of the 1st stage problem, i.e. $\mathbb{E}[J(\alpha^R, d - g) | \hat{d}]$ is computed, the first stage problem can effectively be optimized.

As you can see in this example, it is important for the concept that the sensing system is able to provide better forecasts (of net demands) at each stage. Otherwise stages without better information would be useless, and decisions were zero (assumed increasing prices), as the expectation $\mathbb{E}[J(\alpha^R, d - g) | \hat{d}]$ would not change compared to the preceding stage and therefore the minimizer for equation 3.18 would be zero. The dispatch from the (cheaper) stage before would remain optimal.

### 3.6 Example In Three Stages

Now a case with three stages is considered. In the following the mathematical representation of this case is laid out. Notationwise, imagine "D" as day-ahead (stage 1), "H" as hour-ahead (stage 2) and "R" as real-time (stage 3). The general optimization problem looks like the following

\[
\min_{g^D, g^H, g^R} \quad \mathbb{E}[\alpha^D g^D + \alpha^H g^H + \alpha^R (g^R)^+] \\
\text{s.t.} \quad d = g^D + g^H + g^R
\]

In the real-time stage, there is given $g^D, g^H$, determining the accumulated dispatch. Apart from that, the problem remains unchanged compared to the two-stage version.

\[
J^R(d - g^D - g^H) = \min_{g^R} \alpha^R (g^R)^+ \\
\text{s.t.} \quad d = g^D + g^H + g^R \\
\Leftrightarrow \quad g^{R*} = d - g^D - g^H
\]

\textsuperscript{10}The objective function $F$ represents social welfare or total cost or any other cost measure.
In the *hour-ahead* stage, there is given a day-ahead dispatch as well as an hour-ahead forecast (incl. corresponding distribution) of the demand \((g^D, \hat{\epsilon}^H)\). Mathematically the problem reads as

\[
J^H(\hat{\epsilon}^H - g^D) = \min_{g^H} \alpha^H g^H + \mathbb{E}[J^R(\hat{\epsilon}^H + \epsilon^{HR} - g^D - g^H)]. \quad (3.19)
\]

In the *day-ahead* stage, there is given the day-ahead forecast only \((\hat{\epsilon}^D)\), with its corresponding distribution. The problem structure is the same as in the hour-ahead version, i.e.

\[
J^D(\hat{\epsilon}^D) = \min_{g^D} \alpha^D g^D + \mathbb{E}[J^H(\hat{\epsilon}^D + \epsilon^{DH} - g^D)]. \quad (3.20)
\]

It is worth noting here that the general problem has a nested structure in the number of stages. For solving the day-ahead stage, an expectation of the hour-ahead stage has to be computed; consequently, for solving each corresponding hour-ahead stage, the real-time problem has to be solved, to achieve an expectation of the hour-ahead costs. Solving the hour-ahead problem is analogue to solving the two-stage problem, which is outlined in chapter 3.5.
Chapter 4

Network Risk Limiting Dispatch

In [20] is explained, how the stochastic economic dispatch (RLD) problem can be approached in a two-stage fashion, for both an uncongested and a congested network setup. First an outline of this work is given, while in a second part of this chapter a generalization to a multistage problem is attempted.

4.1 Introduction

A power system naturally consists of numerous producers, consumers and – especially important – transmission lines in between them. Together, these components build a power system. Naturally, collapsing all these interconnected power nodes\(^{11}\) results in losing model accuracy and, for example, network flow constraints can not be captured any more, and therefore might be violated by optimization algorithms based on a simple one-node model.

Hence, to get a more applicable approach to limit the risk of not meeting operating constraints, i.e. power balance and operating limits constraints, an underlying network model is inevitable.

For the purpose of modeling a setup with interconnected nodes or buses, namely a network risk limiting dispatch (NRLD) approach, a DC power flow model is considered (cf. [22] and [23]).

As we will see later in this chapter, often various network setups with comparable line ratings can be reduced to a two-node network, as usually only one transmission line is congested. Although a few lines might operate near their capacity limit, usually only one line is actually congested: That is the bottleneck of the network.

Now, a random net demand \(d_i\) occurs at bus \(i\) and is known at real-time (as before). In Stage 1, the TSO dispatches generation \(g_i\) based on information about random loads and random renewable production at each bus \(i\) (following a given probability distribution). Still, by definition, the first stage dispatch

\(^{11}\)The term power nodes is used here like it was introduced for example in [21].
CHAPTER 4. NETWORK RISK LIMITING DISPATCH

$g_i \geq 0$ is required to be positive, i.e. purchase or production.
In Stage 2, the TSO dispatches generation $g_i^R$ after observing the actual loads and demands to balance the network.
The demand has an underlying probabilistic distribution. The prices, however, can be assumed deterministic, as we consider fixed prices for the dispatch of electric generators in each node. We think of being a TSO (i.e. monopolistic), without a market and no possibility to sell energy, other then dispose it (for free)\(^{12}\).
The structure of marginal prices in networks, also referred to as *nodal pricing* or *locational marginal prices*, is indicated in the literature, such as [24], [25] and especially [5].

### 4.2 Two Stages

Here, the economic dispatch is modeled as a two-stage dispatch problem under uncertainty. Without loss of generality, it is assumed that the first stage is associated with the day-ahead market, and the second stage corresponds with the real-time market. In the day-ahead market, the TSO purchases energy at the generators, which are connected to various buses in the network, and incorporating forecasts and error distributions for loads and renewable generation for the different buses.
In the real-time market, the second stage, the dispatch decisions are made based on the current values loads, generated power as well as line constraints.

A simple illustrative example is given in the following. The costs are assumed linear, as usual:

\[
c_i(g_i) = \alpha_i g_i,
\]

\[
q_i(g_i^R) = \alpha_i^R (g_i^R)^+,
\]

where also energy disposal is modeled to be "for free". Similar to the non-network case, $\alpha_i$ represents the day-ahead price for the specified delivery interval (with generation $g_i$), and $\alpha_i^R$ corresponds to the price at the same delivery interval at real-time (with generation $g_i^R$).
To avoid trivialities, it is assumed that

\[
\alpha_i \leq \alpha_k^R \quad 1 \leq i, k \leq n.
\]

The forecast model follows

\[
d = \hat{d} + \epsilon,
\]

\(^{12}\)Please note that this is a generally unrealistic assumption for markets other than e.g. the Californian power market.
where \( \mathbf{d} = [\hat{d}_1 \quad \hat{d}_2]^T \) is the first stage forecast. Moreover, \( \mathbf{e} = [\epsilon_1 \quad \epsilon_2]^T \) refers to a zero mean Gaussian distributed random vector with covariance matrix \( \Sigma_e = \sigma_e^2 \Sigma_e' \), where \( \Sigma_e' \) is a given error correlation matrix (\( e \sim N(0, \sigma_e^2 \Sigma_e') \)). The Gaussian error assumption is justified by recent studies, e.g. [26]; for typical variances utilized here, the error is negligible.

Furthermore, \( \mathbf{d} \) and \( \mathbf{e} \) are assumed independent, and the standard deviation \( \sigma_e^2 \) is assumed to be given for each bus or region.

The Gaussian error assumption is justified by recent studies, e.g. [26]; for typical variances utilized here, the error is negligible.

The **DC Optimal Power Flow** (DC-OPF) problem is stated as in [20] with the nodal setup represented in figure 4.1. First, the RT-dispatch (2nd stage) \( \mathbf{g}^R \) is computed

\[
J(\alpha^R, \mathbf{d} - \mathbf{g}) = \min_{\alpha^R} \alpha^R^T (\mathbf{g}^R)^+
\]

\[
= \min_{\alpha^R} \alpha_1^R \cdot (\mathbf{g}_1^R)^+ + \alpha_2^R \cdot (\mathbf{g}_2^R)^+
\]

s.t. \( \mathbf{g}_1^R + \mathbf{g}_1 - \mathbf{d}_1 - f = 0 \)
\( \mathbf{g}_2^R + \mathbf{g}_2 - \mathbf{d}_2 + f = 0 \)
\( c_{21} \leq f \leq c_{12} \). \hspace{1cm} (4.1)

The constraints in 4.1 form a convex set of all two-dimensional vectors \( \mathbf{g}^R \). This can be seen by the following argument: Setting \( x_1 = d_1 - g_1 \) and \( x_2 = d_2 - g_2 \) to some fixed value \( x_1 \) and \( x_2 \) yields \( g_1^R = x_1 + f \) and \( g_2^R = x_2 - f \). This is one single point on a \( g_1^R-g_2^R \)-plane, if \( |f| = 0 \). Relaxing this condition to \( |f| \leq c(>0) \), and assuming \( c_{21} = -c \) and \( c_{12} = c \), a bounded line with slope \(-1\) symmetrical to \( x_1 \) and \( x_2 \) evolves. A bounded line is a convex set.

In more general terms, the set of constraints in problem 4.1 form a convex set of all two-dimensional vectors \( \mathbf{g}^R \), by the reason that every single constraint can be rewritten as two affine inequalities, each of them describing a half plane. The intersection of convex sets, i.e. the half planes, form a convex set. Considering \( J(\alpha^R, \mathbf{d} - \mathbf{g}) = J(\alpha^R, \mathbf{x}) \), it can be shown by an analogous argument as convexity is ensured for the problem in \( \mathbf{g}^R \) that convexity holds with respect to \( \mathbf{x} \) as well.

The corresponding 1st stage problem (Day-Ahead Stochastic Power Flow) with
solution $g$ is then:

$$V^*(\hat{d}) = \min_{g \geq 0} \left[ \alpha^T g + E[J(\alpha^R, d - g) \mid \hat{d}] \right]. \quad (4.2)$$

The dependence on $\hat{d}$ illustrates, that generation $g$ is chosen based on the prediction of the actual demand $d$, which is available at first stage. Additionally, problem 4.2 is convex in $g$ by the following argument. The first part is linear in $g$ and hence convex. Suppose, the second part (the expectation) is a combination of two probabilities of outcomes:

$$E[J(\alpha^R, d - g) \mid \hat{d}] = p_1 J(\alpha^R, d - g_1) + p_2 J(\alpha^R, d - g_2).$$

As $J(\alpha^R, d - g)$ is convex in $g$, the combination preserves convexity. This is also true for a combination of infinitely many probabilities, hence an integral, or the expectation of $J$.

Algorithmically, this approach breaks down to the following steps

1. Determine granularity of discretization of $d - g$. This does not mean to set a discretization of the last stage dispatch $g^R$.

2. Determine

$$E[J(\alpha^R, d - g) \mid \hat{d}] = \int J(\alpha^R, \hat{d} + \epsilon - g) \cdot N(\epsilon) d\epsilon$$

for all values of the discretization of $d - g$. For this purpose $\epsilon$ also has to be discretized.

3. Determine $V^*(\hat{d})$ by minimizing

$$\min_{g \geq 0} \left[ \alpha^T g + E[J(\alpha^R, d - g) \mid \hat{d}] \right]$$

over all determined $E[J(\alpha^R, d - g) \mid \hat{d}]$.

Please note, that the line constraints are only applied in the real-time dispatch (second stage). They are not part of the first stage optimization, but are inherently taken care of in the first stage, as the expectation of second stage costs is computed with respect to the line constraints.

### 4.3 Structure Of $J^R$ – An Analytic Approach

The optimal value of the cost function for each point on a two-dimensional plane could be viewed as a third dimension. This leads to figure 4.2, as an illustrative example.
4.3. **STRUCTURE OF J^R – AN ANALYTIC APPROACH**

You can see, that $J^R$ consists of a rather simple structure, comprising five planes. Please note, that although two of those planes in the figure have almost the same slope, and hence it appears to consist of only four planes.

In fact this simple structure can be taken advantage of for lowering the computational burden. An analytic approach to solve $J^R$ is outlined in the following.

A way of making use of the prior knowledge of the structure of $J^R$ is to solve all possible outcomes of the LP in advance in an analytical way. Because of the existence of only a small set of different regions and hence different values for $J^R$, and because an actual LP solver is not needed anymore, the computational effort is reduced enormously.

For the following, assume without loss of generality, that $\alpha_1^R < \alpha_2^R$. Then we define a change of variables

$$y_1^R = d - x_1,$$
$$y_2^R = d - x_2,$$

which yields

$$g_1^R = y_1^R - f,$$
$$g_2^R = y_2^R + f.$$

To approach a solution of $J^R$, the value function is differentiated with respect
Table 4.1: All ten possible cases of arranging $y_1^R, -y_2^R, c$. Assumed $c_{21} = -c$ and $c_{12} = c$. Primed cases impose a certain degree of freedom to choose $f^*$, under which $J^R$ remains minimal.

<table>
<thead>
<tr>
<th>#</th>
<th>Cases</th>
<th>Optimal Control $(f, y_1^R, y_2^R)$</th>
<th>$J^R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$-c \leq -y_2^R \leq y_1^R \leq c$</td>
<td>($-y_2^R, y_1^R, y_2^R, 0$)</td>
<td>$\alpha_1^R(y_1^R + y_2^R)$</td>
</tr>
<tr>
<td>2'</td>
<td>$-c \leq y_1^R \leq -y_2^R \leq c$</td>
<td>($y_1^R \leq f^* \leq -y_2^R, y_1^R - f^<em>, y_2^R + f^</em>$)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$-c \leq y_1^R \leq -y_2^R \leq c$</td>
<td>($-y_2^R, y_1^R, y_2^R, 0$)</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$y_1^R, -y_2^R \leq c$</td>
<td>($-c, y_1^R + c, y_2^R - c$)</td>
<td>$\alpha_2^R(y_2^R - c)$</td>
</tr>
<tr>
<td>4</td>
<td>$y_1^R, -y_2^R \geq c$</td>
<td>($c, y_1^R - c, y_2^R + c$)</td>
<td>$\alpha_1^R(y_1^R - c)$</td>
</tr>
<tr>
<td>5</td>
<td>$-y_2^R \leq -c \leq y_1^R \leq c$</td>
<td>($-c, y_1^R + c, y_2^R - c$)</td>
<td>$\alpha_1^R(y_1^R + y_2^R) + \alpha_2^R(y_2^R - c)$</td>
</tr>
<tr>
<td>6'</td>
<td>$y_1^R \leq -c \leq -y_2^R \leq c$</td>
<td>($-c \leq f^* \leq -y_2^R, y_1^R - f^<em>, y_2^R + f^</em>$)</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>$y_1^R \leq -c \leq -y_2^R \leq c$</td>
<td>($-c, y_1^R + c, y_2^R - c$)</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>$-c \leq -y_2^R \leq c \leq y_1^R$</td>
<td>($-y_2^R, y_1^R, y_2^R, 0$)</td>
<td>$\alpha_1^R(y_1^R + y_2^R)$</td>
</tr>
<tr>
<td>8'</td>
<td>$-c \leq y_1^R \leq c \leq -y_2^R$</td>
<td>($y_1^R \leq f^* \leq c, y_1^R - f^<em>, y_2^R + f^</em>$)</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>$-c \leq y_1^R \leq c \leq -y_2^R$</td>
<td>($c, y_1^R - c, y_2^R + c$)</td>
<td>0</td>
</tr>
<tr>
<td>9'</td>
<td>$y_1^R \leq -c \leq c \leq -y_2^R$</td>
<td>($-c \leq f^* \leq c, y_1^R - f^<em>, y_2^R + f^</em>$)</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>$y_1^R \leq -c \leq c \leq -y_2^R$</td>
<td>($-c, y_1^R + c, y_2^R - c$)</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>$-y_2^R \leq -c \leq c \leq y_1^R$</td>
<td>($-c, y_1^R + c, y_2^R - c$)</td>
<td>$\alpha_1^R(y_1^R + c) + \alpha_2^R(y_2^R - c)$</td>
</tr>
</tbody>
</table>

to $x$

\[
J^R = \min_{c_{21} \leq f \leq c_{12}} \; Q(f) = \alpha_1^R(y_1^R - f)^+ + \alpha_2^R(y_2^R + f)^+ \quad (4.3)
\]

\[
\Leftrightarrow 0 = \frac{dQ(f)}{df} = \alpha_1^R 1(f \leq y_1^R) + \alpha_2^R 1(f \geq -y_2^R), \quad (4.4)
\]

where $1(\cdot)$ is the indicator function. If assuming $c_{21} = -c$ and $c_{12} = c$, there are ten possible cases of arranging $y_1^R, -y_2^R, c$. The corresponding optimal control as well as values for the objective function $J^R$ is shown in table 4.1. In figure 4.3, In view of $J^R$, this set of ten regions can be reduced to five cases with each case leading to a different cost function (cf. table 4.2 and the corresponding graphical representation in figure 4.4).

With these analytic solutions, an LP solver is not needed anymore. The optimization merely reduces to the process of looking up the solution function in a predefined table. This speeds up the computation enormously.
Figure 4.3: Two-dimensional representation of the ten different regions.

Table 4.2: Reduced set of five regions.

<table>
<thead>
<tr>
<th>#</th>
<th>Cases</th>
<th>( J^R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>( \alpha_2^R(y_2^R - c) )</td>
</tr>
<tr>
<td>B</td>
<td>5, 10</td>
<td>( \alpha_1^R(y_1^R + c) + \alpha_2^R(y_2^R - c) )</td>
</tr>
<tr>
<td>C</td>
<td>1, 7</td>
<td>( \alpha_1^R(y_1^R + y_2^R) )</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>( \alpha_1^R(y_1^R - c) )</td>
</tr>
<tr>
<td>E</td>
<td>2, 6, 8, 9</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 4.4: Two-dimensional representation of reduced set of five regions.
4.4 Probability Distribution

To compute values for the expectation $E[J(\alpha^R, d - g) | \hat{d}]$ (cf. equation 4.2), a probability distribution for the occurrence of a certain demand $d$ is to be provided. In figure 4.5 it is shown, how such a probability distribution looks like in two dimensions, i.e. for two nodes. As the demand variance is assumed to be independent (and equal) between both nodes, the distribution appears to be symmetrical. The value of the cost function is point-wise weighted by this probability distribution, as shown in 4.6. Afterwards, an expectation of the cost can be derived from 4.6, by finding the "center of mass", i.e. the weighted mean value.

4.5 Three Or More Stages

In this section, an elaborate derivation of a three-stage network risk limiting dispatch approach is given. Compared to the non-network version of the RLD problem for three stages (cf. chapter 3.6), the optimization problem now be-
Figure 4.6: Exemplary probability distribution, entry wise multiplied with value of cost function $J^R$. This is the base to determine the expected value of cost function.

\[
\min_{g^D, g^H, g^R} \mathbb{E}[\alpha^D g^D + \alpha^H g^H + \alpha^R (g^R)^+] \\
\text{s.t. } d = g^D + g^H + g^R.
\]

In the *Real-Time* stage, there is given $g^D, g^H$, and the corresponding optimization problem emerges as

\[
J^R(d_1 - g^D_1 - g^H_1, d_2 - g^D_2 - g^H_2) = \min_{g^R_1, g^R_2} \alpha^R_1 (g^R_1)^+ + \alpha^R_2 (g^R_2)^+ \\
\text{s.t. } d_1 = g^D_1 + g^H_1 + g^R_1 + f \\
d_2 = g^D_2 + g^H_2 + g^R_2 - f \\
|f| \leq c.
\]

In the *Hour-Ahead* stage, there is given $g^D, \hat{d}^H$, and optimization problem alters to

\[
J^H(\hat{d}^H_1 - g^D_1, \hat{d}^H_2 - g^D_2) = \min_{g^H_1, g^H_2} \alpha^H_1 g^H_1 + \alpha^H_2 g^H_2
\]

\[
+ \mathbb{E}[J^R(\hat{d}^H_1 + \epsilon^HR_1 - g^D_1, \hat{d}^H_2 + \epsilon^HR_2 - g^D_2 - g^H_2)].
\]

(4.5)
4.5. THREE OR MORE STAGES

Finally, in the Day-Ahead stage, there is given $\hat{d}^D$, and optimization problem is

$$J^D(\hat{d}_1^D, \hat{d}_2^D) = \min_{\hat{d}_1^D, \hat{d}_2^D} \alpha_1^D g_1^D + \alpha_2^D g_2^D + \mathbb{E}[J^H(\hat{d}_1^D + \epsilon_{1D}^D - g_1^D, \hat{d}_2^D + \epsilon_{2D}^D - g_2^D)]. \quad (4.6)$$

In the subsequent figures is shown how the expectation of the real-time cost function $\hat{J}^R$ looks like (cf. figure 4.7), and furthermore, how the hour-time cost function $J^H$, which is based on $J^R(x)$ and $J_R(x)$, can be represented (cf. figure 4.8). Lastly, for this three-stage problem, an image of the day-ahead cost function $J^D$ is shown in figure 4.9, which is based on the hour-time as well as the real-time cost function.

More Stages

For adding another stage to the setup (e.g. for a four stage implementation), usually the first stage is reproduced, with adjusting the cost parameters. The
Figure 4.8: Representation of $J^H(x)$, based on the functions $J^R(x)$ and $\hat{J}_R(x)$.

Figure 4.9: Representation of $J^D(x)$, based on the expectation $\hat{J}_H(x)$. 
4.6. **N-BUS NETWORK WITH SINGLE CONGESTED LINE**

The preceding three-stage problem is then called recursively (cf. equation 4.7).

\[
\min_{g_t} \mathbb{E} \sum_{t=1}^{T} p_t \cdot g_t + J(\alpha^R, d - x_{T+1})
\]

s.t. \( x_{t+1} = x_t + g_t \) (Accumulated power)

\( \hat{d}_{t+1} = \hat{d}_t + \epsilon_t \) (Information up to stage \( t \)) \hspace{1cm} (4.7)

---

**4.6 N-bus Network With Single Congested Line**

For tractability and possibly reducing complexity of the problem, it is a key observation that in a real network, *only a few transmission lines are congested* (cf. [20]). In fact, a (connected) network with only one congested line can be reduced to a two-node network (cf. [27]). The general procedure to collapse a complex network to a two bus network is provided subsequently.

1. Divide the network into two parts, where in each part there is no congested line.\(^{13}\)

2. Aggregate demand in one node representing each part of the network.

3. The line in between represents the congestion.

Obviously, this relies on the fact that a reduction of an uncongested network to one node is allowed, which is usually the case. Note however, that then no information on the dispatch for the various entities within these new abstract nodes is provided, as the granularity of the model becomes too simplistic for that.

Please note that collapsing to a two-node network is not always possible. One case where this simplification does not hold is if the power flow changes significantly due to the optimization, such that another line might get congested. This other congestion is not captured in the present model: Repartitioning the network with the new congested line as the connection between two collapsed nodes might lead to a congestion in the previous line.

Hence, it is assumed, that the redispatch is *small* compared to the total power flow, and does not change the congestion pattern. After collapsing to a two bus network, the previous derivations for a two-node network become valid.

\(^{13}\)Note, that the cut might go through buses or (non-congested) transmission lines, in which case nodes partly have to be counted on both sides and their contribution to production (or consumption) might have to be partitioned.
4.7 N-bus Network With Multiple Congested Lines

In this section a derivation of network risk limiting dispatch with multiple congested lines is outlined, following [20] and [27]. Generally, complex networks with \( K \) congested lines can be reduced to a network with \( K + 1 \) nodes, when the uncertainty is moderate.

Furthermore, computational simplifications by means of the small-\( \sigma \) assumption are shown and how that might reduce complexity of \( J^R \).

In particular the small-\( \sigma \) assumption implies that the congestion patterns are not expected to structurally change with the renewable penetration of today and near future. This limited congestion assumption will be utilized to simplify the dispatch calculation.

Also, back flow is included in this model. That means that a congestion between two nodes not necessarily decouples them; a power flow against the congested direction is still possible, leading to a reduction of power flow in the (before) congested direction.

Model Setup

Similar to the two-node model, the first step is executed by the TSO with load forecasts to ensure sufficient generation to meet the demand at the time of delivery. In the second step, the balancing of real-time generation and actual demand of the power system takes place. In the first stage, the net demand \( d_i = l_i - w_i \) is treated as random variable, for which some information exists, such as the forecasted demand \( \hat{d}_i \) and the historical probability distribution of it, leading to the prediction error \( e_i \). Hence, as before, it holds

\[
d_i = \hat{d}_i + e_i.
\]

The TSO decides on generation levels \( g_i \geq 0 \) in the first stage at each bus \( i \). Generation is assumed to be non-negative at each bus in the first stage, to fulfill the respective net demands \( d_i \) (or more specific the corresponding predictions \( \hat{d}_i \)), but of course not necessarily at the same bus. When the real-time realization of the net demand is known in the second stage, the TSO decides on real-time generations \( g_i^R \) at each bus, to balance out the network at time of delivery. This action might be performed by fast ramping generators, load shedding, energy import/export or forms of energy disposal. Accordingly, \( g_i^R \) is not constrained to be positive, although the possibility of selling energy (i.e. with positive financial transactions in return) is not considered here.

For the first stage the costs are assumed as \( \alpha_i g_i \) at each bus and for the second stage, at real-time, they are correspondingly \( \alpha_i^R \cdot (g_i^R)^+ \).
Problem Formulation

First, a generic DC optimal power flow problem is given, with demand \( x \geq 0 \), price \( \alpha \geq 0 \) and generation \( g \geq 0 \):

\[
J(\alpha, x) = \min_{g \geq 0} \alpha^T g \\
\text{s.t.} \quad g - \nabla^T f - x = 0 \\
K f = 0 \\
|f_{ij}| \leq c_{ij} \quad \text{(for } i = 1, \ldots, n, \text{ and } j = 1, \ldots, m) \tag{4.8}
\]

Formulating the NRLD Problem

First, the DC Optimal Power Flow (DC-OPF) problem is stated, with demand \( x \geq 0 \), price \( \alpha \geq 0 \) and generation \( g^R \geq 0 \), following [20]

\[
J(\alpha, x) = \min_{g^R} \alpha^T (g^R)^T \\
\text{s.t.} \quad g^R - x - \nabla^T f = 0 \tag{4.9} \\
K f = 0 \tag{4.10} \\
|f| \leq c \tag{4.12}
\]

with power balance constraint, the mapping \( \nabla^T \in \mathbb{R}^{n \times m} \) of \( m \) branch flows to bus injections, the Kirchhoff’s voltage law (the weighted sum of flows in a cycle equals zero), and the constrained line capacities. Especially the Kirchhoff’s voltage law leads to the fact that line flows can not be decided independently. From graph theory and fundamental flows it is known (e.g. cf. [28]) that any spanning tree of a graph defines a set of fundamental flows \((n-1)\) flows), which provide all remaining flow values by considering the Kirchhoff voltage law. This reduces the constraints 4.10 and 4.11 to

\[
g - Af - x = 0,
\]

where \( f \) now only consists of the \( n-1 \) fundamental flows. For the structure of \( A \in \mathbb{R}^{n \times (n-1)} \), please be referred to [27].

Equation 4.12 constrains the power flow to maximum capacity \( c = [c_1, c_2, \ldots, c_m]^T \) of the corresponding line.

Now, the network risk limiting dispatch (NRLD) problem can be stated, consisting of two stages (backward in time):

1. Real-Time Optimal Power Flow: Solve the RT-OPF \( J(\alpha^R, d - g) \), where \( J \) is defined in equation 4.9. The day-ahead dispatch decisions \( g \) are already made at this stage. The value of the random variables gets visible, therefore the net demand is taken as \( d - g \), and \( J(\alpha^R, d - g) \) balances the power in the network, with real-time prices \( \alpha^R \).
CHAPTER 4. NETWORK RISK LIMITING DISPATCH

2. Day-Ahead Stochastic Power Flow: Solve the DA-SPF

\[ V^*(\hat{d}) = \min_{g \geq 0} \{ \alpha^T g + \mathbb{E}_d [J(\alpha^R, d - g) \mid \hat{d}] \}. \] (4.13)

Note that the expectation over the distribution of \( d \) is taken conditionally on the forecast \( \hat{d} \) and that \( g(\hat{d}, \epsilon) \geq 0 \) limits the day-ahead decisions to purchasing only.

Denote that small \( \sigma \)-assumption is considered to hold. This follows the observation that the standard deviation of day-ahead load forecasts \( \sigma_L \) range around 1..2 % of the expected load. Wind forecasts usually impose larger \( \sigma_W \) of about 30 %, as an example for renewables. Supposed a wind penetration of 30 %, the total error would amount to 0.01 + 0.3 \cdot 0.3 = 10 % of the total load. In contrast to the financial situation, a relative forecast error of 10 % would change the overall physical operating characteristic of the network with only a small probability. That means, that if bus \( i \) would be generating power as calculated by a deterministic approach for the first stage, then, with a high probability, bus \( i \) would still be generating power in the two-stage dispatch problem. Particularly, the renewable power penetration level is assumed to be less than 100 % to assure a positive net demand \( d > 0 \). Furthermore, if we expect day-ahead to purchase power at a bus, it is not expected to shed power at this bus in real-time. Analogously, directions of congestions are assumed to not change their direction from day-ahead to real-time realization. These considerations are very useful in determining the qualitative behavior of the network [20].

Solution Method

Analytically, it is possible to write down a set of deterministic equations to retrieve \( g^* \). As the second stage problem to problem 4.13 is a linear program, the corresponding Lagrange multipliers from the dual problem show the sensitivity of the power balance equations. Taking the derivative of function 4.13 gives a set of equations to solve the problem. Note that the number of equations grows fast with the size of the network (e.g. quadratically with the number of lines, hence to the power of four with the number of nodes), leaving simulation based approaches in favor. However, with the approach of [27], large networks can be reduced to small networks, allowing an analytic solution of the problem. Basically, it relies on the assumption that whatever congestion there is in the Nominal Day-Ahead OPF problem, the congestion does not change because of decisions in subsequent stages. Only small corrections are considered in the real-time stage (small \( \sigma \)).

In the following, the solution method is given in detail, based on [20]:

1. Solve the Nominal Day-Ahead OPF, which is given in the following with price \( \alpha \geq 0 \) and demand \( \hat{d} \geq 0 \). Let \( \tilde{g} \) be the optimal solution and \( \tilde{f} \) the
corresponding flows. It assumes only one decision stage.

\[ J(\alpha, \hat{d}) = \min_{g \geq 0} \alpha^T g \]

s.t. \[ g - \nabla^T f - \hat{d} = 0 \]
\[ K f = 0 \]
\[ |f| \leq c \]

(4.14)

Take \( f_k = c_k \), if a flow on line \( k \) is at capacity \( c_k \).

2. Solve the Perturbed Two-Stage Problem, to find the solution to the original problem 4.13, by considering

\[ \min_{\Delta} \alpha^T \Delta + \mathbb{E}[\tilde{J}(\alpha^R, \epsilon - \Delta)] \]

(4.15)

with perturbation \( \Delta \), and where

\[ \tilde{J}(\alpha^R, \epsilon - \Delta) = \min_g \alpha^{RT}(g)^+ \]

s.t. \[ g - \nabla^T f - (\epsilon - \Delta) = 0 \]
\[ K f = 0 \]
\[ f_k \leq 0 \text{ if } \bar{f}_k = c_k. \]

(4.16)

The solution of the original problem 4.13 is then

\[ g^* = \bar{g} + \Delta. \]

3. The Real-Time OPF to obtain \( g^{Rs} \) can be solved as a standard LP, once the actual demand \( d \) is known, as follows (equivalent to problem 4.16)

\[ J(\alpha^R, d - g^*) = \min_{g^R} \alpha^{RT}(g^R)^+ \]

s.t. \[ g^R + g^* - \nabla^T f - d = 0 \]
\[ K f = 0 \]
\[ f_k \leq 0 \text{ if } \bar{f}_k = c_k. \]

(4.17)

The key observation is here that problem 4.16 is easier to solve than 4.13, because in the perturbed version only flow constraints, which were binding in the nominal case, are included in the second stage of the perturbed problem. Hence, the dimensionality of the perturbed problem might be orders of magnitude smaller than the original problem. The congestion information in \( \bar{f} \) (from the nominal problem) is used to reduce dimensionality in the second stage of the perturbed problem.

Formalized, the assumption to be able to do that is called the small-\( \sigma \) assumption: A power system with forecast net-demand \( \hat{d} \) and error realization \( \epsilon \), the following properties hold:
1. An uncongested line in the solution of the nominal problem remains uncongested in the real-time realization.

2. A congested line in the solution of the nominal problem will not reverse its direction in the real-time realization.

3. A bus dumping energy in the solution of the nominal problem (i.e. \( \bar{g}_i < 0 \)) will still dump energy in the real-time realization.

In practice, even Gaussian errors (which are unbounded) would lead to a very small probability of violation and can hence be ignored [29]. Following [27], a power system with \( K \) congested links can be reduced to a perturbed problem in 4.15, with at most \( K + 1 \) buses and at most \( K \) congested links. Given the small-\( \sigma \) assumption holds, then the stated solution algorithm is optimal. The algorithm to construct such a perturbed network is given there.

**Use Prior Knowledge On Structure Of \( J(\alpha^R, d - g) \)**

If no congestion is existent in the nominal problem (hence, in second step there will not show up a congestion by small-\( \sigma \) assumption), then the only constraint can be rewritten as

\[
\sum_{i=1}^{n} g_i^R = \sum_{i=1}^{n} d_i - \sum_{i=1}^{n} g_i.
\]

This implies the same marginal price \( \alpha^R \) in every node, which formally follows the subsequent reasoning. The Lagrangian (introduced in chapter 2.1.3) with \( x = [g_1^R \ g_2^R \ \ldots \ g_n^R] \) of this problem is

\[
L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)
\]

\[
= \sum_{i=1}^{n} \alpha_i^R g_i^R + \nu \left( \sum_{i=1}^{n} g_i^R + \sum_{i=1}^{n} g_i - \sum_{i=1}^{n} d_i \right).
\]

Now, we want to find the minimal value of the Lagrangian

\[
g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu).
\]

And consequently, following the KKT conditions (cf. chapter 2.1.4)

\[
\nabla L(x^*, \lambda^*, \nu^*) = \nabla f_0(x^*) + \nabla f(x^*)^T \lambda^* + \nabla h(x^*)^T \nu^* = 0
\]

\[
= \begin{bmatrix}
\alpha_1^R \\
\vdots \\
\alpha_n^R
\end{bmatrix} + \nu^* 
\begin{bmatrix}
-1 \\
\vdots \\
-1
\end{bmatrix} = 0,
\]
it holds
\[
\begin{bmatrix}
\alpha_1^R \\
\vdots \\
\alpha_n^R
\end{bmatrix}
= 
\begin{bmatrix}
\nu^* \\
\vdots \\
\nu^*
\end{bmatrix}
\]
and therefore
\[
\alpha_1^R = \cdots = \alpha_n^R = \nu^* = \alpha^R.
\]
The Lagrangian simplifies to
\[
L(x, \lambda, \nu) = 2 \sum_{i=1}^{n} \alpha_i^R g_i^R + \alpha^R \left( \sum_{i=1}^{n} g_i - \sum_{i=1}^{n} d_i \right).
\]
Which is of course at minimum, when \(g_i^R = 0\) for all \(i\). Strong duality (cf. chapter 2.1.3) holds here, and therefore the lower bound
\[
g(\lambda, \nu) = \alpha^R \left( \sum_{i=1}^{n} g_i - \sum_{i=1}^{n} d_i \right)
\]
is tight. The corresponding functional form looks like
\[
J(\alpha^R, d - g) = \alpha^R \left( \sum_{i=1}^{n} d_i - \sum_{i=1}^{n} g_i \right)^+.
\]
If one congestion is existent, that means that \(f_k \leq 0\), i.e. without loss of generality, say
\[
f_1 \leq 0,
\]
\[
\sum_{i=1}^{n} g_i^R = \sum_{i=1}^{n} d_i - \sum_{i=1}^{n} g_i.
\]
The problem can be written as (without loss of generality for two nodes, the congestion is directed from node 1 to node 2):
\[
J(\alpha^R, d - g) = \min_{\alpha^R} \alpha^R (g^R)^+ \\
= \min_{\alpha^R} \alpha_1^R \cdot (g_1^R)^+ + \alpha_2^R \cdot (g_2^R)^+ \\
s.t. \\
g_1^R + g_1 - d_1 - f_1 = 0 \\
g_2^R + g_2 - d_2 + f_1 = 0 \\
f_1 \leq 0,
\]
(4.18)
and equivalently,

\[
J(\alpha^R, d - g) = \min_{g^R} \alpha^R g^R \\
= \min_{g^R} \alpha^R_1 g^R_1 + \alpha^R_2 g^R_2 \\
s.t. \quad g^R_1 + g_1 - d_1 - f_1 \geq 0 \\
\quad g^R_2 + g_2 - d_2 + f_1 \geq 0 \\
\quad f_1 \leq 0.
\]

(4.19)

The Lagrangian with \( x = \begin{bmatrix} g^R_1 & g^R_2 \end{bmatrix} \) of this problem gets

\[
L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \\
= \alpha^R_1 g^R_1 + \alpha^R_2 g^R_2 \\
- \lambda_1 (g^R_1 + g_1 - d_1 - f_1) \\
- \lambda_2 (g^R_2 + g_2 - d_2 + f_1) \\
+ \lambda_3 f_1.
\]

(4.20)

Compute

\[
g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu)
\]

i.e.

\[
\nabla L(x^*, \lambda^*, \nu^*) = \nabla f_0(x^*) + \nabla f(x^*)^T \lambda^* + \nabla h(x^*)^T \nu^* = 0 \\
= \begin{bmatrix} \alpha^R_1 - \lambda_1 \\ \alpha^R_2 - \lambda_2 \end{bmatrix} = 0,
\]

resulting in

\[
\begin{bmatrix} \alpha^R_1 \\ \alpha^R_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \geq 0,
\]

which minimizes \( L \)

\[
g(\lambda, \nu) = \alpha^R_1 g^R_1 + \alpha^R_2 g^R_2 \\
- \alpha^R_1 (g^R_1 + g_1 - d_1 - f_1) \\
- \alpha^R_2 (g^R_2 + g_2 - d_2 + f_1) \\
+ \lambda_3 f_1 \\
= -\alpha^R_1 (g_1 - d_1 - f_1) \\
- \alpha^R_2 (g_2 - d_2 + f_1) \\
+ \lambda_3 f_1.
\]
4.7. N-BUS NETWORK WITH MULTIPLE CONGESTED LINES

The derivative of the Lagrangian \(4.20\) with respect to \(f_1\) has to equal zero, according to the KKT conditions, yielding

\[
\lambda_1 - \lambda_2 + \lambda_3 = 0 \iff \lambda_3 = \alpha^R_2 - \alpha^R_1 (\geq 0).
\] (4.21)

This relation between \(\alpha^R_1\) and \(\alpha^R_2\) makes sense, as the direction of the congestion is assumed to occur from node 1 in direction of node 2. Thus,

\[
g(\lambda, \nu) = -\alpha^R_1 (g_1 - d_1 - f_1) \\
- \alpha^R_2 (g_2 - d_2 + f_1) + \lambda_3 f_1 \\
= -\alpha^R_1 (g_1 - d_1) - \alpha^R_2 (g_2 - d_2).
\] (4.22)

Noting, that by the small-\(\sigma\) assumption, a reversed congestion is excluded, an interpretation of this plane \(4.22\) (in variables \(x_i = g_i - d_i\) for \(i = \{1, 2\}\)) is that the congestion from the first stage remains in the second stage. Therefore the price in node 2 is higher than in node 1, and all energy surplus, which is needed in node 2 has to be generated in node 2.

There is also taken care of the possibility of having a back flow, i.e. that the constraint \(f_1 \leq 0\) is not binding in the second stage. This would imply the same nodal prices in both nodes, as the flow capacity is not saturated anymore – and equation \(4.22\) holds:

\[
J(\alpha^R, d - g) = \alpha^R_1 (d_1 - g_1)^+ + \alpha^R_2 (d_2 - g_2)^+.
\] (4.23)

Basically, these equations \(4.22\) and \(4.23\) describe one plane (instead of five). This reduces complexity of \(J^R\) and results in simplified and faster computations. However, please note that this only holds for small changes of the flow (small \(\sigma\)), but still gives good insight in the structure of the problem, and is a promising basis for computations on complex networks with more than two buses.
Chapter 5

Computational Tractability

In this chapter, two computational approaches are applied in a simulation environment and their performance and tractability is analyzed. They both have been introduced theoretically in chapter 2: Dynamic Programming and Sample Average Approximation (Monte Carlo).

5.1 Dynamic Programming Based Approach

Dynamic programming is an optimization method, whose theoretical foundation was introduced earlier in chapter 2.2. It is based on the principle that an optimal subproblem also is optimal for the overall problem optimization. Also, optimal solutions to subproblems remain stored after they are computed for the first time.

Apart from that, dynamic programming algorithms are usually intractable on continuous spaces, as $x_t$ can take any value, leading to an infinite number of optimization problems. Therefore, a discretization of the state-space was introduced. Furthermore, the discrete solution tables of the minimal cost functions $J^{t+1}$ (for stage $t+1$) is reused for every computation in $J^t$. With this technique, the complexity in number of stages turns out to be (almost)$^{14}$ linear. Due to the fact that dynamic programming is a natural choice of how to approach a network risk limiting dispatch problem, the actual algorithm was introduced earlier, in chapter 4.2.

Complexity Speculation

Let $D = d^N$ be the total number of discretization points, with $d$ being the number of discretization points in each dimension, i.e. for each node. Hence,

$^{14}$Tables are growing in size at each stage, to capture the fact, that expectation computations might run out of bounds of the previous stage. Please also refer to section 5.4 for more illustrative insights on this.
with $N$ nodes\textsuperscript{15}, it can be speculated onto the complexity as follows:

\[
O(d^N) \quad \text{LPs to be solved}
+ O(d^N) \quad \text{problem evaluations}
+ O(d^N) \quad \text{multiplications}
+ O(d^N) \quad \text{summations}
+ O(d^N) \quad \text{find minimum in array of length } d^N.
\]

As most of these computational steps have to be executed for each stage $t \leq T$, the order of complexity in nodes and stages is $O(T \cdot d^N)$. Empirical results, with computation time as an indicator of complexity, are shown later in the results section, specifically in chapter 5.4.1.

\section{5.2 Sample Based Approach}

The sample based approach, which was already mentioned earlier in chapter 2.3, is applied to the problem and explained in more detail in the following by means of an example in three stages. The third, i.e. last, stage is (as usual) referred to as the real-time stage ("R"), while the second stage is the hour-ahead stage (also referred to as "HA" or "H") and the first stage takes place a day-ahead ("DA" or "D"). With linear forecast updates, the uncertainty in the DA problem can be summarized by the uncertainties of the forecast correction terms $\epsilon_{\text{DH}}$ and $\epsilon_{\text{HR}}$.

To start, simple Monte Carlo samples are used to represent the uncertainties. In particular, $N$ i.i.d. samples for $\epsilon_{\text{DH}}$, denoted by $\epsilon_{\text{DH}}(1), \ldots, \epsilon_{\text{DH}}(N)$ and $M$ i.i.d. samples for $\epsilon_{\text{HR}}$, denoted by $\epsilon_{\text{HR}}(1), \ldots, \epsilon_{\text{HR}}(M)$ can be drawn.

The DA optimization then has the form

\[
\min_{g^D, g^H} \alpha^D g^D + \frac{1}{M} \sum_{m=1}^{M} J^H \left( \hat{d}^D + \epsilon_{\text{DH}(m)} - g^D \right),
\]

where the expectation in equation 3.20 is replaced with its sample average. Noticing that the function $J^H$ is unknown at the DA market, in view of equation 3.19, the optimization in equation 5.1 can be rewritten as

\[
\min_{g^D, g^H(m), \forall m} \alpha^D g^D + \frac{1}{M} \sum_{m=1}^{M} \alpha^H g^H(m) + \frac{1}{N} \sum_{n=1}^{N} J^R \left( \hat{d}^D + \epsilon_{\text{DH}(m)} + \epsilon_{\text{HR}(n)} - g^D - g^H(m) \right),
\]

where the expectation in equation 3.19 was again replaced with its sample av-

\textsuperscript{15}N is number of congested lines + 1, if the uncongested part of the system is collapsed.
5.2. SAMPLE BASED APPROACH

Repeating the process above for $J^R$, the following optimization emerges

$$
\min_{g^D, g^H(m), g^R(m,n), \forall m,n} \alpha^D g^D + \frac{1}{M} \sum_{m=1}^{M} \left[ \alpha^H g^H(m) + \frac{1}{N} \sum_{n=1}^{N} \alpha^R \left( g^R(m,n) \right) ^+ \right] \\
\text{s.t. } \hat{d}^D + \epsilon^{DH(m)} + \epsilon^{HR(n)} - g^D - g^H(m) = g^R(m,n), \forall m, n. \quad (5.2)
$$

A similar optimization can be formed for the HA market dispatch problem:

$$
\min_{g^H, g^R(n), \forall n} \alpha^H g^H + \frac{1}{N} \sum_{n=1}^{N} \alpha^R \left( g^R(n) \right) ^+ \\
\text{s.t. } \hat{d}^H + \epsilon^{HR(n)} - g^D - g^H = g^R(n), \forall n. \quad (5.3)
$$

To conclude, this optimization roughly follows these steps:

1. Generate $s$ random samples of error terms $\epsilon$, following a given probability distribution.

2. Optimize (potentially large) LP in $O(s^{T-1})$ variables, e.g. for Day-Ahead of the form:

$$
\min_{g^D} \alpha^D g^D + \frac{1}{M} \sum_{m=1}^{M} J^H \left( \hat{d}^D + \epsilon^{DH(m)} - g^D \right)
$$

3. Observe updated state and generate random samples for it, optimize again.

![Ternary tree](image)

Figure 5.1: Ternary tree, not recombining, to illustrate growth rate. Although usually the sample trees consist of more than three sample states for each stage (e.g. $s = 50$).

The corresponding complexity speculation yields

$$
1 + s + s^2 + s^3 + \cdots + s^{T-1} = O(s^{T-1})
$$

with an order of complexity of $O(s^{T-1})$ and therefore grows a lot faster in the number of stages than the DP approach. However, opposing to the DP approach this sample strategy might lead to a complexity linear in the number of nodes. Verification of this speculation for more than two nodes goes beyond the scope of this project, but should be an interesting topic to look into in the future.
5.3 Computational Tools

For all implementations as well as simulations, Matlab 2013b was used. Optimizations, especially optimizing linear programs (LPs), were executed with the commercial solvers GUROBI (cf. [30]) as well as CVX (cf. [31]). Formulating the optimization problems for GUROBI was done with the freely available Matlab Toolbox YALMIP (cf. [32]).

YALMIP is a helpful tool. It takes for example linear or quadratic optimization problems in a high level formulation, processes these (e.g. does a convexity analysis) and is able to translate that formulation for various commercial and non-commercial solvers (e.g. CPLEX or GUROBI). Hence, YALMIP is especially interesting to avoid the error-prone process of transforming optimization problems to specific solvers. This advantage comes at the cost of reduced optimization speed, as system matrices have to be recreated at every optimization step. However, using the optimizer function minimizes this computational speed deficiency, because YALMIP then tries to warm start the used solver with the previous optimization result, and reuses the system matrices (i.e. assumes the problem structure does not change in between).

5.4 Results

In the sequel, algorithmic results on computational complexity as well as cost realizations depending on the number of discretization points or samples respectively are presented. Note that the actual cost realizations between the two approaches (cf. figures 5.6 and 5.8) can not simply be compared here. As both approaches are initialized with the same parameters (e.g. costs), the theoretical optimal cost value is the same. However, the range of possible dispatches is naturally limited for the discrete approach, because the discretization tables can not be infinitely large. Therefore the achievable optimal values are not exactly the same, and the discretization based approach hence converges slower than the sample based approach. Another difference between these two performance results is that, due to the assumingly normal distributed demand probability, the discretization tables also would have taken a rather large and impractical size. This was avoided by truncating the probability distribution at a certain point (cf. chapter 2.4) for the discrete approach. The focus here lies on tackling the computational complexity.

5.4.1 Dynamic Programming Based Approach

An empirical complexity analysis in the number of stages can be seen in figure 5.2, where complexity is represented by computational run time of the corresponding algorithm.
5.4. RESULTS

In figure 5.2 is also shown how making use of prior structural knowledge of $J^R$ and taking analytical solving approaches speeds up the computation. A visualization of the relative gain in speed is shown in figure 5.3. Note that especially for only two stages in total, the process of solving the respective LPs takes a relatively big portion of the total run time. Therefore the corresponding speed up is rather large and the analytic approach is approximately ten times faster.

Apart from that, the table of LPs, which are to be solved, gets larger in the number of stages, as greater regions of possible expected $\hat{d}$ have to be covered\(^ {16}\). This explains the super-linear growth of computation time, laid out in figure 5.2. However, if normalized with respect to the size of the tables in each stage, the computational time shows an approximately linear increase in the number of stages, as predicted in chapter 5.1 (cf. figure 5.4).

In figure 5.5 an indicator of computational complexity (i.e. run time) depending on both the number of stages as well as the discretization granularity\(^ {17}\) is shown, essentially supporting the complexity speculation in chapter 5.1.

\(^{16}\)For example, with only two stages, possibly only 20 discretization points per node might suffice, while for three stages 25 discretization points (of the same distance) per node might be needed, to account for expectation calculations. These expectation calculations might be based on table regions which are beyond previously set borders (for smaller number of stages).

\(^{17}\)Granularity refers to the distance between two discretization points here. Therefore it is anti-proportional to the number of discretization points in each dimension.
CHAPTER 5. COMPUTATIONAL TRACTABILITY

Figure 5.3: Relative efficiency gain factor of using the analytic solution instead of a standard LP solver (DP approach).

Figure 5.4: Normalized computation time for various numbers of stages, two nodes (DP approach).
Figure 5.5: Computation time for various numbers of stages and discretization granularities, two nodes (DP approach).
Figure 5.6 reveals the realized costs for an exemplary implementation of a two-stage and a three-stage optimization with a discretization based dynamic programming approach. Although the scatter of realized costs is rather large, the graph exhibits a clear trend towards decreasing cost realizations for smaller discretization distances, i.e. more discretization points. Hence, the gap to the optimal value gets smaller. Also the graph unveils that a three-stage implementation might achieve smaller total costs than a two-stage implementation. Implementations with more stages better exploit the fact that smaller forecast horizons tend to have better forecast accuracy (cf. figure 1.1).

5.4.2 Sample Based Approach

The Sample Average Approximation Approach shows an almost linear increase of computational run time in the number of samples, as shown in figure 5.7. This behavior indicates that the solver (CVX in this case) has approximately linear performance characteristics with respect to the size of the problem. The size of the linear optimization problem directly depends on the number of samples. Figure 5.8 presents how the realized costs decrease with an increasing number of samples. Note that this figure represents a dependency on the total number of samples, i.e. for three stages. Hence, the graph covers a dependency for up to $\sqrt{5000} \approx 17.1$ samples per stage.
Figure 5.7: Computational run time for various numbers of samples, for three stages in two nodes (sample based approach).

Figure 5.8: Mean total cost, for three stages, with exponential fit (sample based approach).
Chapter 6

Conclusion and Outlook

6.1 Conclusion

Starting from fundamental mathematical optimization concepts such as convex optimization and dynamic programming, the basics of a risk limiting dispatch (RLD) strategy were derived. RLD is an operating paradigm for smart grids, which manages the operating risk, i.e. reliability, of the power system while it minimizes the expected cost to fulfill the power demand. This is possible by recognizing that risk from current decisions can be mitigated by future decisions in a multistage power market.

A network risk limiting dispatch (NRLD) scheme was developed to account for the fact that line flow constraints cannot be captured in a standard RLD one-node model setup – and is therefore the next step on the road to practical application.

More specifically, two approaches to tackle NRLD optimization were implemented. First, a discretization based dynamic programming algorithm was derived and realized. This is the natural choice of optimization for the given nested problem structure. In particular, an analytic solution to this discretization based approach was elaborated. The analytic solution avoids using slow LP solvers, as the solutions can be precomputed due to prior structural knowledge of the corresponding optimal cost functions. The computational performance gain ranges from 2x to 10x.

The dynamic programming algorithm appears to be especially well fitting for a relatively large number of optimization stages, as complexity grows linearly in stages.

The other approach that was focused on is a sample average approximation based approach in a Monte-Carlo-like fashion. It was shown that the complexity of this approach grows exponentially in the number of stages. However, it might be suitable for a larger number of nodes.

Moreover, it was outlined, that the gap to the (theoretically) optimal dispatch
depends on the available computational power, i.e. both approaches exhibit accuracy improvements with increasing computational effort.

Furthermore it was presented in this thesis how NRLD works for more than two stages. Additionally, algorithmic implementations for arbitrary numbers of stages were created and analyzed. In previous research, only a setup with two stages and one node was practically simulated.

### 6.2 Future Research

Future research related to this topic might go into different directions. Showing feasibility of the derived algorithms for more than one congestion, i.e. for more than two nodes, appears to be an important one. Furthermore, the implementation of storage features is another compelling direction.

The introduction of storage possibilities, e.g. by introducing specific storage nodes, between different delivery intervals and not only within a given interval might be an interesting but challenging step as it could lead to coupled optimization problems which can further increase the computational complexity.

Another important feature which might be exciting for future research is the introduction of probabilistic marginal costs or prices for each node. This helps avoiding the assumption of increasing costs with decreasing execution time horizons, which is one of the main flaws in the present project.

Furthermore, one might think about deriving additional computational approaches. One of them could lie in the field of model predictive control (MPC). Although standard MPC does not utilize stochastic optimization, i.e. does not employ information about probability distributions, there exist variants of MPC which might be adopted for NRLD. Reducing the receding horizon by one step at each stage could possibly lead to a multistage optimization with a specific delivery interval. Together with a stochastic or chance constraint MPC implementation (cf. [33]), tailored to deal with joint probability distributions, this could be a valuable, fast and elegant approach.
Appendix A

Source Code

In the following, code fragments of special interest are shown. The code is generally written in MATLAB, using a YALMIP environment.

Listing A.1: Code of solveRT.m for computing the real-time stage optimization (i.e. last stage) using an analytically predetermined table of solutions.

```matlab
function [ J, U ] = solveRT( PR,C,Y1,Y2 )

%solveRT Solves the LP describing the RT problem with LOOK UP TABLE!

if length(Y1) ~= length(Y2)
    disp('y1 does not have same length as y2')
    return
end

U = zeros(3,length(Y1));
J = zeros(length(Y1),1);

for i = 1:length(Y1)
    y1 = Y1(i);
    y2 = Y2(i);
    if length(C) ~= 1;
        c = C(i);
    else c=C;
    end

    if length(PR) ~= 2;
        pR = PR(i,:);
    else pR=PR;
    end

    J = [ J; solveRT( pR,c ) ];
    U = [ U; c ];
end
end
```

```bash
65
```
% case 3:
if (−y2 ≤ −c) && (y1 ≤ −c);
U(:,i) = [ y1+c, y2−c, −c];
J(i) = pR(2)*(y2−c);
region=3;

% case 5:
elseif (−y2 ≤ −c) && (−c ≤ y1) && (y1 ≤ c);
U(:,i) = [ y1+c, y2−c, −c];
J(i) = pR(1)*(y1+c) + pR(2)*(y2−c);
region=5;

% case 10:
elseif (−y2 ≤ −c) && (−c ≤ c) && (c ≤ y1);
U(:,i) = [y1+c, y2−c, −c];
J(i) = pR(1)*(y1+c) + pR(2)*(y2−c);
region=10;

% case 1:
elseif (−c ≤ −y2) && (−y2 ≤ y1) && (y1 ≤ c);
U(:,i) = [y1+y2, 0, −y2];
J(i) = pR(1)*(y1+y2);
region=1;

% case 7:
elseif (−c ≤ −y2) && (−y2 ≤ c) && (c ≤ y1);
U(:,i) = [y1+y2, 0, −y2];
J(i) = pR(1)*(y1+y2);
region=7;

% case 4:
elseif (c ≤ −y2) && (c ≤ y1);
U(:,i) = [y1−c, y2+c, c];
J(i) = pR(1)*(y1−c);
region=4;

% case 2:
elseif (−c ≤ y1) && (y1 ≤ −y2) && (−y2 ≤ c);
U(:,i) = [y1+y2, 0, −y2];
J(i) = 0;
region=2;

% case 6:
elseif (y1 ≤ −c) && (−c ≤ −y2) && (−y2 ≤ c);
U(:,i,:) = [y1+c, y2−c, −c];
J(i) = 0;
region=6;
Listing A.2: Code of solveRT_solve.m for the computation of the real time stage optimization (i.e. last stage) using YALMIP solving environment and GUROBI as a solver.

```matlab
function [ JR, U ] = solveRT( pR,c12,x1,x2 )
%solveRT Solves the LP describing the RT problem
% note, that x=d−g!
% pR = pR(:);
% if numel(pR) < 3
% pR = [pR;0];
% end
% if length(x1) ≠ length(x2)
% disp('x1 does not have same length as x2')
% return
% end
% yalmip('clear');
 nu= 3;
 u = sdpvar(repmat(nu,1,1),repmat(1,1,1)); % decision variable: u = [g1R, g2R, flow]
 x = sdpvar(2,1); % state variable: x = [x1 , x2]
 U = zeros(nu,length(x1));
 JR = zeros(length(x1),1);
 % case 8:
 elseif (−c ≤ y1) & & (y1 ≤ c) & & (c ≤ −y2);
 U(:,i)= [y1−c, y2+c, c];
 J(i)= 0;
 region=8;

% case 9:
 elseif (y1 ≤ −c) && (−c ≤ c) && (c ≤ −y2);
 U(:,i)= [y1+c, y2−c, −c];
 J(i)= 0;
 region=9;

else
 disp('Something went terribly wrong with the Look Up Table');
 region=0;
 end
end
```

1. function [ JR, U ] = solveRT( pR,c12,x1,x2 )
2. %solveRT Solves the LP describing the RT problem
3. % note, that x=d−g!
4. pR = pR(:);
5. if numel(pR) < 3
6. pR = [pR;0];
7. end
8. if length(x1) ≠ length(x2)
9. disp('x1 does not have same length as x2')
10. return
11. end
12. yalmip('clear');
13. nu= 3;
14. u = sdpvar(repmat(nu,1,1),repmat(1,1,1)); % decision variable: u = [g1R, g2R, flow]
15. x = sdpvar(2,1); % state variable: x = [x1 , x2]
16. U = zeros(nu,length(x1));
17. JR = zeros(length(x1),1);
APPENDIX A. SOURCE CODE

22 Objective = pR(1)*max(u(1),0) + pR(2)*max(u(2),0);
23 Constraints = [[-inf,-inf,-c12]' ≤ u ≤ [inf,inf,c12]', u(1) - u(3) == x(1), u(2) + u(3) == x(2)];
24 Options = sdpsettings('verbose',0,'solver','gurobi');
25 controller = optimizer(Constraints,Objective,Options,x,u);
26
27 for i = 1:length(x1)
28     x0= [x1(i);x2(i)];
29     [uk,errorcode] = controller(x0);
30     if errorcode
31         yalmiperror(errorcode)
32         return
33     end
34     U(:,i) = uk;
35     JR(i) = pR(1)*max(uk(1),0) + pR(2)*max(uk(2),0);
36 end

Listing A.3: Code of solveNRT.m for the computation of each stage (excluding the real time stage) in a recursive fashion (cf. Dynamic Programming”).

1 function [ JH,gH,precomp ] = solveNRT(t, T, p,c12,x1,x2,dRange,
2     gRange, h, std, trunc, precomp)
3
4     gHRange = gRange(t,:);
5     JH = zeros(length(x1),1);
6     gH = zeros(length(x1),2)';
7     [gListX, gListY] = meshgrid(gHRange(1):h:gHRange(2),gHRange(1):h:
8         gHRange(2));
9     gList = [gListX(:), gListY(:)];
10
11     for i = 1:length(x1)
12         if ~exist('precomp','var') || isempty(precomp)
13             [ JRhatList_gList, precomp ] = evalJhat(t+1,T, p,c12,x1(i)−
14                 gListX(:),x2(i)−gListY(:),dRange, gRange, h, std, trunc
15                 );
16         else
17             [ JRhatList_gList, precomp ] = evalJhat(t+1,T, p,c12,x1(i)−
18                 gListX(:),x2(i)−gListY(:),dRange, gRange, h, std, trunc
19                 , precomp);
20         end
Listing A.4: Code of evalJ.m for evaluating $J$ for any stage.

```matlab
function [ JR, precomp ] = evalJ( t,T,p,c12,x1,x2, dRange, gRange, h, std, trunc, precomp)

%UNTITLED Summary of this function goes here

% Evaluate JR at a list of locations stored in vector x1 and x2
% If the argument precomp exists, the function leveage the precomp
% results to reduce computational cost; if the argument precomp does not
% exist, the function use solveRealTimeStage to compute JR.

if ¬exist('precomp','var')
    delRange = dRange − fliplr(sum(gRange(1:T,:)));
    [delListX, delListY] = meshgrid(delRange(1):h:delRange(2),
        delRange(1):h:delRange(2));
    delList = [delListX(:), delListY(:)];
    if t == T
        [JList, gList] = solveRT( p(T,:),c12,delListX(:),delListY(:));
    % else
    %     [JList, gList, precomp] = solveNRT( t,T,p(t,:),c12,
    %         delListX(:),delListY(:),dRange, gRange, h, std, trunc);
    % end
    eval(['precomp.del',num2str(T),'range = delRange;']);
    eval(['precomp.del',num2str(T),'List = delList;']);
    eval(['precomp.J',num2str(T),'List = JList;']);
    eval(['precomp.g',num2str(T),'List = gList;']);
else
    delRange = dRange − fliplr(sum(gRange(1:t,:)));
    [delListX, delListY] = meshgrid(delRange(1):h:delRange(2),
        delRange(1):h:delRange(2));
    delList = [delListX(:), delListY(:)];
    [JList, gList, precomp] = solveNRT( t,T,p(t,:),c12,
        delListX(:),delListY(:),dRange, gRange, h, std, trunc,precomp);
    eval(['precomp.del',num2str(t),'range = delRange;']);
    eval(['precomp.del',num2str(t),'List = delList;']);
    eval(['precomp.J',num2str(t),'List = JList;']);
    eval(['precomp.g',num2str(t),'List = gList;']);
end
```

% keyboard;
JH_loop = (p(t,:)*gList')'+JRhatList_gList;
[JH(i),minIdx] = min(JH_loop);
gH(:,i) = gList(minIdx,:);
```
Listing A.5: Code of `evalJhat.m` for evaluating $\hat{J}$ (i.e. expectation of $J$) for any stage.

```matlab
function [ JR_hat, precomp ] = evalJhat(t,T, p,c12,x1,x2,dRange, gRange, h, std, trunc, precomp)
% multiplies JR with occurrence probability of x1,x2 to get the expectation
% Furthermore, it uses a nearest neighbour method to evaluate JhatList
if ~exist('precomp','var')
    [~, precomp ] = evalJ( T,T,p,c12,0,0, dRange, gRange, h, std, trunc, precomp);
end
if eval(['~isfield(precomp, ''J',num2str(t),'List'')'])
    [~, precomp ] = evalJ( t,T,p,c12,0,0, dRange, gRange, h, std, trunc, precomp);
end
if eval(['~isfield(precomp, ''Jhat',num2str(t),'List'')'])
    delList = precomp.del';
    JList = precomp.J';
    JhatList = zeros(length(delList),1);
    for i = 1:length(JhatList)
        mu = delList(i,:);
        prob =makeDiscreteProb(delList, h, mu, std(t,:), dRange, trunc);
        JhatList(i) = sum(prob.* JList);
    end
    eval(['precomp.Jhat',num2str(t),'List = JhatList;']);
end
end
```

APPENDIX A. SOURCE CODE
Listing A.6: Code of `solveHA_saa.m` for computing the hour-ahead (as an example stage) in a sample average approximation (i.e. Monte Carlo) fashion using CVX solving environment and GUROBI as a solver.

```matlab
function [ JH,gH ] = solveHA_saa(pH,pR,C,x1,x2,nsHR,stdHR)
    %UNTITLED Summary of this function goes here
    % Detailed explanation goes here

    % Input
    x = [x1; x2]; % equiv. (dH_hat - gD)!

    % Sampling
    eHR = randn(nsHR,2).*repmat(stdHR',nsHR,1);
    cvx_begin quiet
    variable gH(2)
    variable gR(2,nsHR)
    variable f(1,nsHR)

    % minimize
    minimize ( pH*gH + ...
                1/nsHR * sum(pR*pos(gR(1:2,:))))
    % such that
    gH(:) ≥ 0;
    abs(f(:)) ≤ C;
    for idHR = 1:nsHR
        %dH_hat' - gD' + eHR(idHR,:) - gH(:,idHR)' == gR(:,idHR) + [1,-1]*f(1, idHR);
        x' + eHR(idHR,:) - gH(:,idHR)' == gR(:,idHR) + [1,-1]*f(1, idHR);
    end
    cvx_end
    JH = cvx_optval;
end
```
Appendix B

DVD Contents

This section gives an overview of the content of the accompanying DVD.

FinalDocs
Contains the final presentation as well as the final report in PDF format.

Report
LATEX source code and figures of the report. The report itself is the file thesis.pdf.

Literature
Literature cited in the report available in PDF format as well as additional literature.

MatlabCode
Created MATLAB code and results of all simulations used in the report as well as figures.

Presentation
LATEX source code and figures of the presentation. The presentation itself is the file presentation.pdf.
Bibliography


