TEP4550 - Energy and Process Engineering, Specialization Project
TKP4580 - Chemical Engineering, Specialization project

Optimization of flexible renewable energy systems using stochastic programming

Petter Engblom Nordby
Mari Elise Rugland

Submission date: 16.12.2020
Supervisors: Avinash Subramanian, EPT
Truls Gundersen, EPT
Johannes Jaschke, IKP

NTNU
Department of Energy and Process Engineering

NTNU
Department of Chemical Engineering

Norwegian University of Science and Technology

Preface

Declaration of Compliance

I, Mari Elise Rugland, and I, Petter Engblom Nordby, hereby declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).

Signature:

Place and Date: Trondheim - Gløshaugen, 16^{th} of December 2020

Acknowledgements

The authors greatly acknowledge supervisors, Truls Gundersen and Johannes Jäschke, for their guidance over the course of this specialization project. Additionally, we would like to express our gratitude to our co-supervisor, Avinash Subramanian, for valuable discussions, his rigorous feedback and useful suggestions. Our supervisors' commitment to the project has been encouraging and a great motivator.

Abstract

In this project thesis, the problem of optimal design and operation of flexible renewable energy systems (RES) under uncertainty was addressed. The inherent intermittent behavior of renewable energy sources and the associated energy conversion models give rise to considerable computational challenges. A linear renewable energy system model was developed, and a two-stage stochastic programming approach was used to optimize the design and operation of this system. The resulting two-stage MILP formulation was solved using the NGBD algorithm embedded in the GOSSIP software. GOSSIP provides a C++ framework for the formulation and efficient solution of two-stage stochastic programs. The software includes links to decomposition algorithms that scale favorably with the number of scenarios. The resulting 1^{st} stage variables correspond to decisions taken before the realizations of uncertainty while 2^{nd} stage variables correspond to decisions taken after realization of uncertainty. Thus, the design decisions were 1^{st} stage, while the operational were 2^{nd} stage. The following three case studies were modeled in GOSSIP: A simple RES with one design day, a simple RES with four design days, and a RES with one design day and short-term energy storage. The aim was to minimize the overall cost of the RES, and in order to apply NGBD to solve the problem, initial design decisions were discretized by employing binary variables. Uncertainty was accounted for by generating scenarios based on sampling from a normal distribution. Results from the three case studies show that accounting for uncertainty reduces the overall operating cost by 3-15%. Future work will involve using more realistic models which would result in nonconvex MINLP formulations.

Table of Contents

Table of Contents iii

List of Tables iv

List of Figures v

Nomenclature vi

1 Introduction 1
- 1.1 Work allocation . . . 1
- 1.2 Motivation . . . 1
- 1.3 Objective and Scope . . . 3
- 1.4 Structure of the report . . . 4

2 Optimization methodologies 5
- 2.1 Introduction to optimization . . . 5
- 2.2 Mixed-Integer Programming - MIP . . . 7
- 2.3 Stochastic programming . . . 8

3 Methodology 11
- 3.1 GOSSIP software . . . 11
- 3.2 A worked out example: Farmer's Problem . . . 14

4 Renewable Energy System (RES) 19

5 Case studies 21
- 5.1 Case study 1: Simple model of uncertainty with 1 design day . . . 21
- 5.2 Case study 2: Accounting for seasonal variability using 4 design days 26
- 5.3 Case study 3: Stochastic problem with dynamic model and energy storage . . . 28

6 Results **33**
6.1 Case study 1: Simple model of uncertainty with 1 design day 34
6.2 Case study 2: Accounting for seasonal variability using 4 design days 37
6.3 Case study 3: Stochastic problem with dynamic model and energy storage 41

7 Discussion **44**

8 Conclusions and future Work **50**

A Calculations **54**

B Scenarios for case study 1 and 2 **56**
B.1 Scenarios case study 1 and 2 56

C Scenario set for case study 3 **59**

D C++ code **61**
D.1 Expected value problem 61
D.2 Case study 1: Simple model of uncertainty with 1 design day 67
D.3 Case study 2: Accounting for seasonal variability using 4 design days 73
D.4 Case study 3: Stochastic problem with dynamic model and energy storage 80

List of Tables

3.1 Data for Farmer's Problem 14
3.2 A simple scenario representation 14
3.3 Results of Farmer's Problem 17
3.4 Results of Farmer's Problem 18

5.1 Variables with bounds, units and stage number. 23
5.2 Parameters describing the reference cost and minimum area requirement of the two technologies in Equation 5.7 for capital cost. 23
5.3 Efficiencies and other parameters used in the model. 25
5.4 Summary of uncertain variables with their mean value and associated standard deviation. 25
5.5 Modified mean values to simulate seasonal variations through 4 design days. 26
5.6 Capital and operating cost for energy storage [17] 30
5.7 List of discrete values for S_{ES} calculated from Equation (5.9). 30
5.8 Storage capacity bounds and efficiencies in the energy storage model. 32

6.1 Results of the expected value problem. 33
6.2 Results for the stochastic problem. 34
6.3 Results for the flexible design under seasonal variation. 37
6.4 Results for the nominal design under seasonal variation. 38
6.5 Results for case study 3. 41

B.1 Scenarios used in case study 1 and 2. 56

List of Figures

2.1 Geometrical representation of a constrained optimization problem [9]. 6
2.2 Convex (a) and nonconvex (b) set [10]. 6
2.3 Convex (a) and nonconvex (b) function [10]. 7
2.4 Scenario tree for a two-stage stochastic program 9

3.1 A schematic of the different decomposition algorithms and the class of stochastic programs they are applicable to. 12

4.1 Simplified flowsheet of the renewable energy system with power generation and user demand. No storage technologies included. The energy balance is shown in Equation 5.10. 19
4.2 Simplified flowsheet of the renewable energy system with power generation, user demand, and battery as chosen storage technology. . . . 20

5.1 Scenario tree for a multi-stage stochastic program 28
5.2 Scenario tree for a two-stage stochastic program 29

6.1 The expected coverage of demand by renewable energy production (top) and import (bottom) for the flexible and nominal design. . . . 35
6.2 The LCOE in $/MWh produced from renewables in the nominal and flexible design. 36
6.3 The expected coverage of demand by renewable energy (top) and electricity import (bottom) under seasonal variation for the nominal and flexible design. 39
6.4 The LCOE in $/MWh produced from renewables in the seasonal stochastic problem for the nominal and flexible design. 40
6.5 The expected coverage of demand by renewable energy production (top) and import (bottom) for the nominal design, and flexible design with and without energy storage. 42
6.6 System flows and operation of battery 43

Nomenclature

Acronyms

BD	Benders Decomposition
EEV	Expectation of Expected Value Problem
GBD	Generalized Benders Decomposition
GOSSIP	Global Optimization of non-convex two-Stage Stochastic mixed-Integer Programs
IP	Integer Programming
LCOE	Levelized Cost of Energy
LP	Linear Programming
MIP	Mixed Integer Programming
MICP	Mixed Integer Convex Programming
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
MVP	Mean Value Problem
NGBD	Nonconvex Generalized Bender Decomposition
P	Programs or Programming (used interchangeably)
RES	Renewable Energy Systems
RP	Recourse Problem
SP	Stochastic Problem
VSS	Value of Stochastic Solution

Chapter 1

Introduction

1.1 Work allocation

This project thesis was made through a collaboration between Petter Engblom Nordby and Mari Elise Rugland. Mari Elise developed the model in case 1 and 2 whereas Petter developed the model in case 3. Other sections not explicitly mentioned here were written by both parties and are regarded as joint contributions.

1.2 Motivation

The global demand for energy has increased steadily since the industrialization. So far in the 21st century, the world has seen a 2.2% annual rise in energy demand, and historically, the surge in global energy intensity has been accounted for by increased use of fossil fuels [1]. Consequently, greenhouse gas emissions related to human activities are estimated to have increased the average global temperature by 1 $^{\circ}C$ compared to pre-industrial levels [2]. As the global population keeps rising and developing nations are becoming industrialized, cheap, accessible and environmentally neutral energy sources are becoming increasingly important. In modeled pathways that limit global warming to 1.5 $^{\circ}C$, renewable energy sources are projected to account for 70-85% of the global energy mix [2].

Impelled by significant investments, technological progress and economies of scale have led to a sharp cut in the levelized costs of renewable energy (LCOE) [3]. Electricity from solar photovoltaics (PV) has seen an 82% reduction in cost the past decade, undercutting the marginal cost of existing coal-fired plants [4]. Cutting the cost of projects is however not the only problem associated with renewable energy systems. Renewable energy systems face operation challenges as a result of significant uncertainties. These uncertainties include the inherent fluctuations in renewable energy sources such as solar and wind, the varying user demand, and the

volatile market spot price of electricity. These uncertainties reduce the availability of the renewable energy system, and consequently the ability to meet energy demand. One approach is to take these uncertainties into account from the design stage by implementing a flexible design. Flexible renewable energy systems (RES) can react swiftly to changes in operational circumstances, i.e. they have the ability to adjust operating conditions in order to respond to uncertainty. The result of this property makes flexible designs more robust under uncertainty.

Flexible design can be obtained by installing additional (redundant) capacity or by installing multiple renewable energy technologies that exploit different weather conditions. It should be noted that there is a trade-off between increased capital expenses associated with redundancy and increased robustness to uncertainty. For instance, without a flexible design, the cheaper RES may not be able to meet demand peaks on cold winter days. On the other hand, installing a higher capacity would increase investment costs, but allow the RES to meet demand on winter days. In addition, there would also be the potential of redundant capacity in the summer months.

Another approach to counteract the negative effects of uncertainty is through installation of energy storage. In fact, long- and short-term energy storage technologies are expected to further compensate for the power generation fluctuations and seasonal variability. With the inclusion of storage technologies, it is believed that substantial amounts of renewable energy sources can be integrated with the global energy mix [5].

The design and operation of a flexible renewable energy system can be formulated as an optimization problem. Utilizing multiple renewable energy sources and/or storage technologies are two ways to increase flexibility, and have to be considered at the design stage. Furthermore, the nature of renewable energy sources requires long time horizons to account for seasonal variations as well as hourly resolution from an operational modeling perspective. This drives up the number of decision variables and creates a complex optimization problem. In addition, the optimal choice of installed capacity of technologies, and the subsequent operation of the RES is subject to uncertain parameters such as the market spot price of electricity and a varying user demand. In this manner, future uncertainties complicate the problem by affecting optimal system design, optimal system operation, and thereby, project profitability.

In summary, optimization problems for the design and operation of flexible energy systems pose considerable computational complexity. A deterministic mixed-integer linear programming approach has been proposed for multi-energy systems due to the reasonable size of a deterministic program. Specifically, Gabrielli et al. [6] proposed a deterministic MILP approach that allows hourly resolution and account for seasonal changes. Nonetheless, as formerly discussed, a deterministic approach is a gross simplification of RES behavior.

In a study by Mavromatidis et al. [7], uncertainty was included in the problem formulation through a two-stage stochastic MILP. However, MILP formulations can entail the risk of making sub-optimal decisions when modeling non-linearities with linear relations. Consequently, a two-stage stochastic MINLP approach that can be solved in reasonable time is currently assumed to be the optimal solution.

While MINLP can be challenging to solve, they allow for using both linear and non-linear models to accurately predict system behavior and solve a wide range of relevant problems. Two-stage stochastic programs have a specific structure that can be exploited by decomposition-based approaches to allow efficient solutions. The novel optimization software GOSSIP, developed by researchers at the Process Systems Engineering Laboratory (PSEL) at MIT, exploits the structure of two-stage stochastic MINLPs in order to reduce the solver time [8]. The development of a two-stage MINLP in GOSSIP can therefore be a promising step in the direction of optimization of flexible renewable energy systems.

1.3 Objective and Scope

The objective of this thesis is to develop a renewable energy system model and find the optimal design and operation of this system under uncertainty, using the optimization software GOSSIP. A two-stage stochastic programming approach is used to model the flexible design problem, and an outline of the approach is presented.

The optimization of a simple renewable energy system is formulated as a MILP problem and solved using GOSSIP. Firstly, a deterministic problem formulation is studied for the purpose of validating the model while omitting energy storage technologies. The problem is then developed into a two-stage stochastic program through the introduction of probability distributions for selected model parameters. This is further modified to simulate seasonal variations through the implementation of design days. In the final problem formulation, energy storage is added to the model, completing the simple renewable energy system consisting of solar PV, wind, battery and an end user. Summarized, the three case studies that are investigated are, in increasing complexity:

- **Case study 1:** Design and operation of a RES with uncertainty modeled by one design day.
- **Case study 2:** Design and operation of a RES with uncertainty and seasonal variation modeled by four design days.
- **Case study 3:** Design and operation of a RES through a dynamic model to account for uncertainty through energy storage.

1.4 Structure of the report

In Chapter 2, a brief introduction to optimization and associated methodologies is presented. Thereafter, a short introduction to mixed-integer programming (MIP) formulations and stochastic programming is given in Sections 2.1, 2.2 and 2.3 respectively. The applied methodology is presented in Section 3.1 followed by an illustrative example problem in Section 3.2. In Chapter 4 the renewable energy system(s) under study is presented, with a schematic overview of components and internal dynamics. Chapter 5 present the three different case studies with their corresponding equations and parameters. In Section 5.1 the stochastic and expected value problem (SP and EVP) is explained, in Section 5.2 seasonal variation is included in the problem formulation, and in Section 5.3 the dynamic model for energy storage is introduced. Results for the three case studies are presented in chapter 6, followed by a discussion of numerical results and the performance of the different solution methods and models in Chapter 7. Lastly, concluding remarks and suggestion for future work is asserted in Chapter 8.

Chapter 2

Optimization methodologies

2.1 Introduction to optimization

An optimization problem is a search for an optimal solution expressed through an objective function you want to i) Minimize or ii) Maximize. The function gives a scalar value as output, but the value is determined by one or several variables limited by constraints. The constraints can be equalities or inequalities, and the optimization problem can be constrained or unconstrained.

Equation 2.1 is a formulation of a constrained optimization problem with the objective function, J, and the constraints, $c_i(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$, where $\boldsymbol{x}$ is a vector of decision variables. For $\boldsymbol{x}$ to be a feasible point, $\boldsymbol{x}$ must be in the set defined by $\boldsymbol{X}$ and satisfy all equality ($\boldsymbol{c}(\boldsymbol{x})$) and inequality ($\boldsymbol{g}(\boldsymbol{x})$) constraints. If the optimization problem is a convex problem, a local optimum is guaranteed to be a global optimum. If either the objective function or the feasible set is nonconvex, then a local optimum is not guaranteed to be a global optimum. Strategies involving global information are then employed.

$$\begin{aligned} \min_{\boldsymbol{x}\in\boldsymbol{X}} \quad & J(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{c}(\boldsymbol{x}) = 0 \\ & \boldsymbol{g}(\boldsymbol{x}) \leq 0 \end{aligned} \tag{2.1}$$

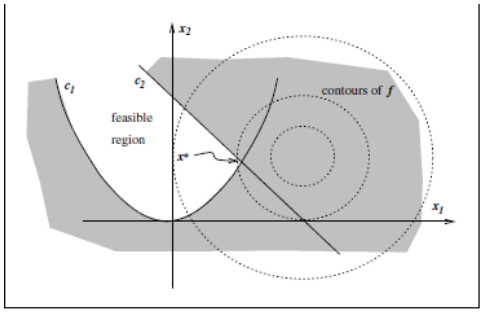


Figure 2.1: Geometrical representation of a constrained optimization problem [9].

A convex set is a set C where for every two points (x, y) in C, a line segment z, as defined in Equation 2.2 must be in the set C. In other words, for C to be a convex set, every interior point on the line segment z must also be in C [10]. A convex set C is illustrated in Figure 2.2 a).

$$z = \lambda x + (1 - \lambda y), \ \forall \lambda \ \in \ [0, 1] \tag{2.2}$$

If such a line segment z cannot be drawn, e.g. without crossing boundaries, the set is by nature non-convex, as illustrated in Figure 2.2 b).

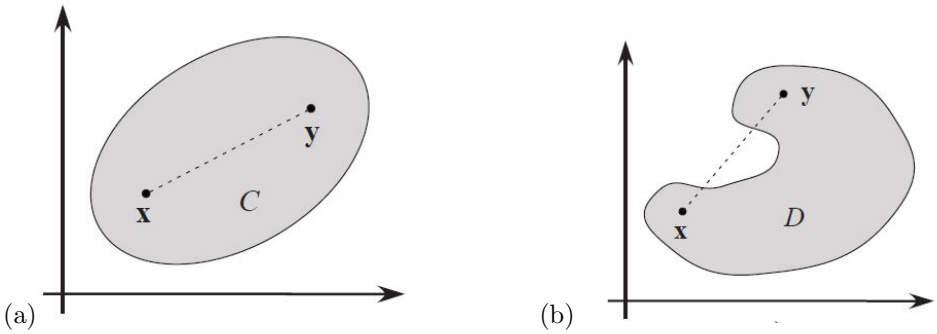


Figure 2.2: Convex (a) and nonconvex (b) set [10].

Following this, a convex function f, is a function defined on a convex domain C where for each two points $x_1, f(x_1)$ and $x_2, f(x_2)$ the line segment between x and y lies entirely above the graph of the function f as illustrated in Figure 2.3 a). In mathematical terms, a function $f : X \to \mathbb{R}$ is a convex function if X is a convex set and if:

$$f(\lambda x_1 + (1 - \lambda)x_2)) \leq \lambda f(x_1) + (1 - \lambda)f(x_2), \quad \forall \lambda \in [0, 1], \quad \forall x_1, x_2 \in X \tag{2.3}$$

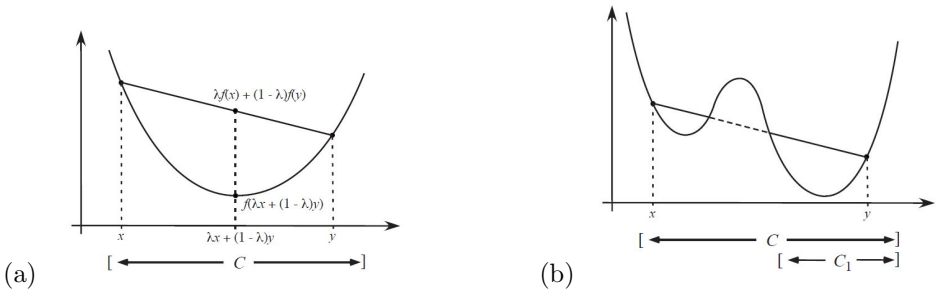


Figure 2.3: Convex (a) and nonconvex (b) function [10].

Furthermore, the optimization problem can have either or both discrete and continuous decision variables. Problems that contain both continuous and discrete variables are defined as mixed-integer programs (MIP). In addition, the parameters of the model can be either certain or uncertain. In a deterministic problem, it is assumed that none of the parameters is subject to randomness. If a model is subject to uncertainty, stochastic programming approaches are considered.

2.2 Mixed-Integer Programming - MIP

Mixed-Integer Programming (MIP) is frequently applied in industry for design and planning of production systems and optimization of energy systems. While simple energy systems can be described using a linear model, providing a realistic representation of most systems requires the use of nonlinear (and thus nonconvex) model equations. It is then considered a nonconvex mixed-integer non-linear programming problem (MINLP).

A general mixed-integer linear programming problem (MILP) formulation is given in equation 2.4,

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} \leq \boldsymbol{b} \\ & \boldsymbol{x} \quad \in \{0,1\}^p \\ & \boldsymbol{y} \quad \in \mathbb{R}^n_+ \end{aligned} \tag{2.4}$$

where $\boldsymbol{A}$ is a $m \times n$ matrix, $\boldsymbol{B}$ is a $m \times p$ matrix, and $\boldsymbol{b}$, $\boldsymbol{c}$ and $\boldsymbol{d}$ are m-,n- and p-dimensional vectors [11]. $\boldsymbol{x}$ is a n-dimensional vector with integer variables and $\boldsymbol{y}$ is a p-dimensional vector of binary variables.

A general MINLP formulation is given in 2.5.

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & f(\boldsymbol{x},\boldsymbol{y}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq \mathbf{0}, \\ & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = \mathbf{0} \\ & \boldsymbol{x} \quad \in \{0,1\}^p \\ & \boldsymbol{y} \quad \in \mathbb{R}^n_+ \end{aligned} \tag{2.5}$$

$\boldsymbol{x}$ and $\boldsymbol{y}$ are discrete and continuous variables, respectively, and at least one of the $f(\boldsymbol{x},\boldsymbol{y})$, $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y})$ and $\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y})$ functions contains non-linear terms with respect to the decision variables [12].

2.3 Stochastic programming

Stochastic programming provides an approach to account for uncertainty related to the parameters of a model. Contrary to deterministic approaches, stochastic programs incorporate random variables into the problem formulation in order to capture the uncertain nature of the given optimization problem. The goal of stochastic programming is to reduce the risk of undertaking sub-optimal decisions. The parametric uncertainty can be described by assuming either probability distributions or patterns from historical data. Uncertainty makes the decision-making process more complex as larger optimization problems are required. An approach for handling optimization under uncertainty is through two-stage stochastic programming with scenario generation.

The number of scenarios, S, is a function of the number of uncertain variables and number of possible realizations of these variables. In the first stage of a stochastic program, a set of immediate decisions have to be made prior to the realization of the uncertain parameters. In the second stage, corrective actions are made to compensate for the realizations of uncertainty. Stochastic programming involves making design and operational decisions that minimize the sum of the objective value of the first stage and the expected value of the second stage across all scenarios. Finding the optimal solution of the second stage is called the recourse problem. The first stage decision variables of a two-stage stochastic program are represented by a vector $\boldsymbol{x}$, while the second stage variables are represented by a vector $\boldsymbol{y}$. A general mathematical formulation of a two-stage stochastic program is shown in equation 2.6,

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) + E_{\boldsymbol{\xi}}[Q(\boldsymbol{x},\boldsymbol{\xi})], \\ \text{s.t.} \quad & \mathbf{g}^{(1)}(\boldsymbol{x}) \leq 0 \\ & \mathbf{h}^{(1)}(\boldsymbol{x}) = 0 \\ & \boldsymbol{x} \in \boldsymbol{X} \end{aligned} \tag{2.6}$$

where $f(\boldsymbol{x})$ is some function of $\boldsymbol{x}$, and $\boldsymbol{\xi}$ forms the vector of the uncertain parameters. The functions $\mathbf{g}^{(1)}$ and $\mathbf{h}^{(1)}$ are constraints on the design variables, and $\boldsymbol{X}$ defines which values the variables can take. $E_{\boldsymbol{\xi}}$ is the expected value of the function $\mathrm{Q}(\boldsymbol{x},\boldsymbol{\xi})$, which is the optimal value of the recourse problem for each scenario, given by equation 2.7.

$$\begin{aligned} \min_{\boldsymbol{y}} \quad & q(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\xi}) \\ \text{s.t.} \quad & \mathbf{g}^{(2)}(\boldsymbol{y}_s,\boldsymbol{x},\boldsymbol{\xi}_s) \leq 0 \qquad \forall\ \mathrm{s} \in \mathrm{S} \\ & \mathbf{h}^{(2)}(\boldsymbol{y}_s,\boldsymbol{x},\boldsymbol{\xi}_s) = 0 \qquad \forall\ \mathrm{s} \in \mathrm{S} \\ & \boldsymbol{y}_s \in \boldsymbol{Y} \end{aligned} \tag{2.7}$$

The functions $\mathbf{g}^{(2)}$ and $\mathbf{h}^{(2)}$ are operational constraints, and $\boldsymbol{Y}$ defines which values the second stage variables can take. One approach to lower the complexity of the optimization problem is to employ a scenario representation of the problem. In this approach, the uncertain parameters are assumed to take on values from a finite number of realizations, each with an associated probability. The sum of the constructed scenarios s, each with its own probability p_s and parameter values ξ_s make up the recourse problem given in Equation 2.8. A typical scenario tree for a two-stage stochastic program is illustrated in Figure 2.4.

$$E_{\boldsymbol{\xi}}[Q(\boldsymbol{x},\boldsymbol{\xi})] = \sum_{s=1}^{S} p_s Q(\boldsymbol{x},\xi_s) \tag{2.8}$$

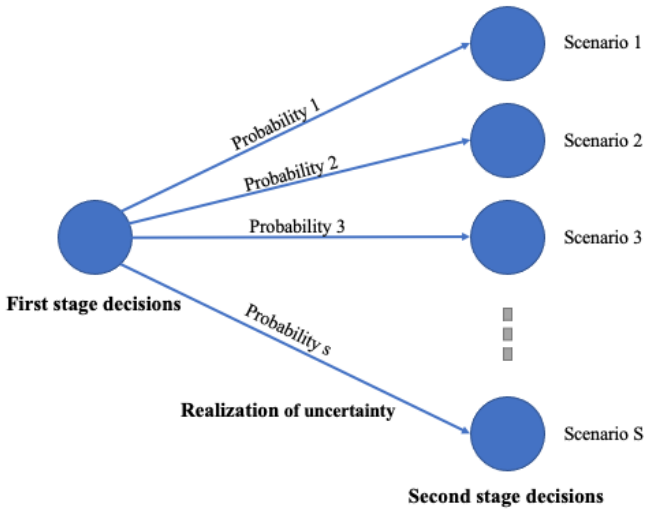


Figure 2.4: Scenario tree for a two-stage stochastic program

The single-level formulation of the stochastic program can then be formulated

by equation 2.9 by combining equation 2.6, 2.7 and 2.8.

$$
\begin{aligned}
\underset{\boldsymbol{x},\boldsymbol{y}_s}{max} \quad & f^{(1)}(\boldsymbol{x}) + \sum_{s=1}^{S} p_s f^{(2)}(\boldsymbol{y}_s, \boldsymbol{x}, \boldsymbol{\xi}_s) \\
\text{s.t.} \quad & \mathbf{g}^{(1)}(\boldsymbol{x}) \leq 0 \\
& \mathbf{h}^{(1)}(\boldsymbol{x}) = 0 \\
& \boldsymbol{x} \in \boldsymbol{X} \\
& \mathbf{g}^{(2)}(\boldsymbol{y}_s, \boldsymbol{x}, \boldsymbol{\xi}_s) \leq 0 \qquad \forall \text{ s} \in \text{S} \\
& \mathbf{h}^{(2)}(\boldsymbol{y}_s, \boldsymbol{x}, \boldsymbol{\xi}_s) = 0 \qquad \forall \text{ s} \in \text{S} \\
& \boldsymbol{y}_s \in \boldsymbol{Y}
\end{aligned}
\tag{2.9}
$$

Chapter 3

Methodology

3.1 GOSSIP software

GOSSIP is a software framework for modeling and solving two-stage stochastic nonconvex MINLPs. It is embedded on a C++ platform and an overview of functionalities and worked out examples for solving two-stage stochastic programs in GOSSIP can be found in the documentation [13]. The GOSSIP software is, under certain requirements, guaranteed to determine the global optimum of a non-convex two-stage stochastic problem. However, GOSSIP can also be used to solve convex two-stage stochastic programs (MICP) as well as large-scale MILP. Four different solution methods are implemented in GOSSIP, but only the NGBD method is applied in this project.

Various decomposition approaches have been developed to handle different classes of stochastic programming problems as illustrated in Figure 3.1. The earliest approach was termed 'Benders decomposition (BD)' and was only applicable to the class of two-stage stochastic MILPs. BD was then extended to give Generalized Benders Decomposition (GBD) which could solve the class of two-stage stochastic Mixed-Integer Convex Programs (MICPs). Finally, GBD was extended for the class of two-stage nonconvex MINLPs with the Nonconvex Generalized Benders Decomposition (NGBD) algorithm. We note that NGBD reduces to the GBD algorithm for convex problems and to the BD algorithm for linear problems.

Next, a brief overview of the GBD and NGBD algorithm is presented. Complete details are presented in [10]. GBD is a method for solving two-stage stochastic MICPs. The GBD strategy involves constructing an equivalent dual representation of the original problem with a large but finite number of constraints. A relaxation of the dual representation is then constructed by only including a small subset of the constraints. The solution of this relaxed problem yields the lower bound on the solution to the original problem. Due to the strong duality for convex problems, the

solution to the dual problem itself yields the upper bound to the original problem. These two steps are done in an iterative manner until a global solution is found [10].

The NGBD extension strategy involves convexifying the MINLP and then applying GBD to give a lower bound to the problem. The upper bound is found by solving the MINLP using a local solver. This procedure is done in an iterative manner shrinking the gap between the lower and upper bound until convergence to a global optimum.

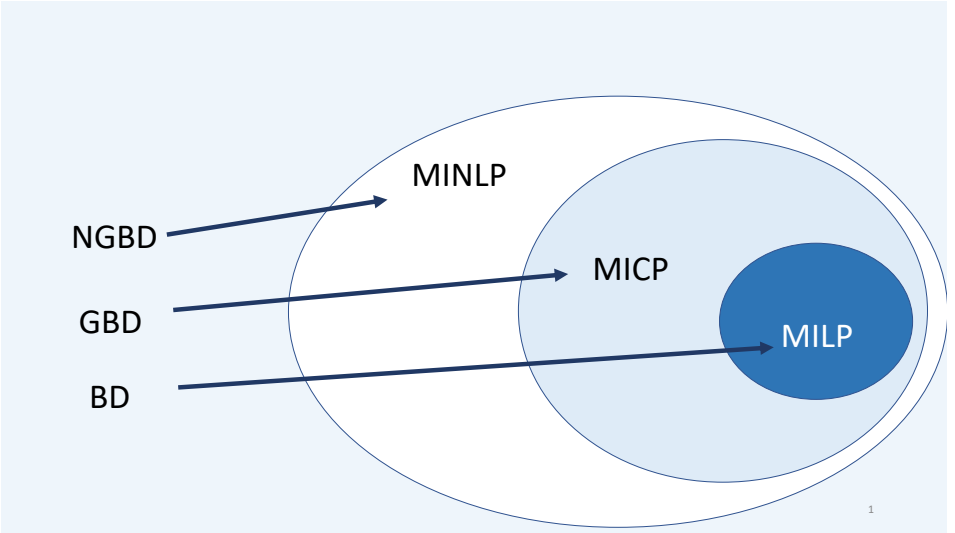


Figure 3.1: A schematic of the different decomposition algorithms and the class of stochastic programs they are applicable to.

More specifically, the NGBD algorithm decomposes two-stage stochastic problems into smaller sub-problems, e.g. one for each scenario, and thus provides efficient scaling of solution time with increasing number of scenarios considered. The aforementioned methods make NGBD a strong tool for global optimization of MINLP and other non-linear programming problems. However, the NGBD algorithm is only guaranteed to converge for discrete 1^{st} stage variables. Consequently, the $\boldsymbol{x}$-vector in Equation 2.6 and 2.7 can only contain variables from a discrete set of integer values. A potential workaround (used in this project) is to discretize each continuous 1^{st} stage variable by assuming it can only take on a fixed number of values, x_j, within its interval bounds as defined in Equation 3.1.

$$x_j^{discrete} = x^{LBD} + \frac{j-1}{n-1} \cdot (x^{UBD} - x^{LBD}), \qquad \forall \text{ j} \in \{1,....,\text{n}\} \tag{3.1}$$

$x_j^{discrete}$ denotes the discretized value in the j^{th} interval, n number of intervals, and x^{LBD} and x^{UBD} the lower and upper bound of the interval, respectively. A

set of binary variables, $\boldsymbol{x}^{binary}$, are implemented to ensure that x only take on one of the fixed variables discretized above. This implies that the first stage variables in a two-stage program, x, needs to be included through a sum of the product of the binary variables and their corresponding fixed value, as shown in equation 3.2. Including equation 3.3 in the program ensures that only one of the binary variables are selected.

$$x = \sum_{j=0}^{n} x_j^{binary} \cdot x_j^{discrete} \tag{3.2}$$

$$\sum_{j=0}^{n} x_j^{binary} - 1 = 0 \tag{3.3}$$

3.2 A worked out example: Farmer's Problem

To illustrate the value of stochastic programming and how GOSSIP can be applied to optimization problems, the stochastic Farmer's Problem was solved. In the problem, a farmer aims to maximize his expected profit by taking uncertainty into account when deciding how much land to devote to each crop.

Physical formulation and input

In Table 3.1, the input data for the Farmer's Problem problem is given. The farmer has 500 acres of land available and can raise wheat, corn and sugar beets, however with a minimum requirement for wheat and corn for own consumption. This minimum requirement can either be from crops raised on his land or bought from the market. The market price is 40% higher than what the farmer can sell his crops for. There is no minimum requirement for sugar beets, but there is an assumed quota for sugar beet production. Any production above this quota can be sold, however at a significantly lower price.

Prices are assumed to be known and constant, while the yield from each acre of land is uncertain. The expected yield is known from historical data, whereas the yearly yield depends on the weather conditions. To formulate a two-stage stochastic program, probability distributions or the probability of all scenarios must be known. For simplicity, the Farmer's Problem was at first formulated with three equally likely scenarios (i.e., uniform distribution), as shown in Table 3.2. Thereafter, scenario generation from a normal distribution with the expected yield as the mean and a standard deviation of 20% was investigated.

Table 3.1: Data for Farmer's Problem

	Unit	Wheat	Corn	Sugar Beets
Planting Cost	\$/acre	150	230	260
Selling Price	\$/T	170	150	36 under 6000T
				10 above 6000T
Purchase Price	\$/T	238	210	-
Minimum Requirement	T	200	240	-
Mean Yield	T/acre	2.5	3	20
Total Available Land:	500 acres			

Table 3.2: A simple scenario representation

Scenario	Unit	Wheat	Corn	Sugar Beets	Probability
Above Average Yield	T/acre	3	3.6	24	1/3
Expected Yield	T/acre	2.5	3	20	1/3
Below Average Yield	T/acre	2	2.4	16	1/3

Mathematical formulation

The uncertainty implies that there is no single decision which is optimal in every scenario. Instead, the farmer has two sets of decisions to make, where the first decisions have to be made before he has access to full information. In the winter, he has to decide how to allocate the land in order to maximise his expected profit in the fall, regardless of the actual yield. Consequently, area allocations are the 1^{st} stage decision variables for the program, represented by the vector $\boldsymbol{x}$. The vector $\boldsymbol{x}$ will have one entry for each crop, resulting in three entries in total.

When the exact yield is revealed, the farmer can decide the amount of each crop to sell, and potentially, how much he must buy to cover his own requirements. Thus, the sale and purchase of each crop are categorized as 2^{nd} stage decision variables. These corrective actions for each scenario, s, are represented by the vectors $\boldsymbol{y}_s$ and $\boldsymbol{w}_s$, where $\boldsymbol{y}_s$ is the amount of tonnes of each crop sold and $\boldsymbol{w}_s$ is the amount of tonnes of each crop purchased. As a result, the objective function of the optimization problem consist of terms representing the costs related to both 1^{st} and 2^{nd} stage decisions.

$$Obj_{min} = \sum_{i=1}^{3} PlantingCost_i \cdot x_i + \sum_{s=1}^{S} p_s \cdot RP_s \tag{3.4}$$

The objective function in Equation 3.4 aims to minimize the sum of planting costs for each crop, i, and the expected value of the recourse problem. The expected value of the recourse problem is the net profit from purchases and sales for each scenario, RP_s, multiplied with the associated probability, p_s. Equation 3.5 formulates the profit function, where $SellingPrices$ will have four entries as sugar beets can be sold at both a favourable and unfavourable price. Purchasing prices only contains the purchase prices for wheat and corn as there is no minimum requirement for own consumption of sugar beets.

$$RP_s = \sum_{j=1}^{2} PurchasingPrice_j \cdot w_{s,j} - \sum_{j=1}^{4} SellingPrice_j \cdot y_{s,j} \tag{3.5}$$

Problem constraints are derived from the description of the physical problem and are formulated in Equation 3.6, 3.8, 3.9 and 3.10. Firstly, the 1^{st} stage decision problem is constrained by equation 3.6 and 3.7, representing the sum of land devoted to crops constrained by land available.

$$\sum_{i=1}^{3} x_i \leq AvailableLand \tag{3.6}$$

$$x_i \geq 0 \qquad i \in \{Wheat, Corn, SugarBeets\} \tag{3.7}$$

The constraints for the recourse problem and the 2^{nd} stage decision variables is defined for each scenario s. Equation 3.8 ensure that yields ($yield_{s,i}$), sales ($y_{s,i}$)

and purchases ($w_{s,i}$) of crops cover the minimum requirements. Equation 3.9 limit the amount of sugar beets sold at a favorable price and Equation 3.10 ensure that the amount of sugar beets sold does not exceed the amount of sugar beets produced. Equation 3.11 ensures non-negative sales and purchases for all selling and purchase prices in $Prices$, where the set $Prices$ consists of wheat, corn and sugar beets at the favourable, and unfavourable price.

$$yield_{s,i} \cdot x_i - y_{s,i} + w_{s,i} \geq MinReq_i \quad \forall \text{ s}\in \text{ S, i} \in \{Wheat, Corn\} \tag{3.8}$$

$$y_{s,3} \leq 6000 \quad \forall \text{ s} \in \text{S} \tag{3.9}$$

$$y_{s,3} + y_{s,4} \leq yield_{s,3} \quad \forall \text{ s} \in \text{ S} \tag{3.10}$$

$$y_{s,j}, w_{s,j} \geq 0 \quad \forall \text{ s} \in \text{S, } j \in Prices \tag{3.11}$$

Expected value problem and value of stochastic solution

Next, a few relevant concepts are introduced in order to evaluate the stochastic program.

- The expected value problem (EVP) is an optimization problem in which all the uncertain parameters are assumed to take on their expected values. Thus, the EVP can be viewed as a deterministic problem with one scenario and where all uncertain parameters are set to their mean value. The first stage decision variables in this problem will therefore be selected to yield the optimal value of the objective function for this scenario. The selection of the first stage variables is referred to as the nominal design.
- The expectation of expected value problem (EEV) is then defined as the expected value of the objective function for the nominal design subject to uncertainty.
- The value of the stochastic solution (VSS) is the additional value obtained by taking uncertainty into account when selecting the first stage variables. This extra value is defined as the difference between the optimal objective value of the stochastic program (SP) and the EEV, shown below in equation 3.12.

$$VSS = EEV - SP \tag{3.12}$$

Results of Farmer's problem

Table 3.3 show the results for the expected value and stochastic problem, where the stochastic solution is obtained by running the model with the three scenarios previously presented. The solution for the EVP is calculated from the average yield scenario, thus the crop allocation is optimized for the mean values. The

Table 3.3: Results of Farmer's Problem

		Stochastic Problem	**Nominal design**
Number of scenarios		3	3
Probability distribution		Uniform	Uniform
Land allocation (acres):			
Wheat		170	120
Corn		80	80
Sugar Beets		250	300
Sales (tonnes):			
Low yield	Wheat	140	40
	Corn	-48 (purchase)	-48 (purchase)
	Sugar beets	4000	4800
Mean yield	Wheat	225	100
	Corn	-	-
	Sugar beets	5000	6000
High yield	Wheat	310	160
	Corn	48	48
	Sugar beets	6000	7200
Profit for each scenario($):			
Low yield		48,820	55,120
Mean yield		109,350	118,600
High yield		167,000	148,000
Expected Profit ($)		108,390	107,240
VSS ($)		1150	-

two methods produce different results where the EVP generates the largest profit for the average yield scenario, whereas the stochastic solution produce the largest expected profit. The results show that the optimal solution when using expected values will cause the sales of sugar beets to surpass the quota for the high yield scenario, meaning that the farmer will have to sell his sugar beets at an unfavourable price. The stochastic program concludes that it will be more beneficial to ensure the sugar beet quota is not exceeded in any of the three scenarios. As a result, area is reallocated and the farmer produces more wheat than in the EVP.

The value of the stochastic solution is the additional expected profit the farmer obtains by taking uncertainty into account. Calculations show that the VSS of the Farmers Problem is $1150, substantiating the value of incorporating uncertainty in an optimization model.

The number of scenarios in the program can easily be increased by generating scenarios from a normal distribution. The resulting land allocation and expected profit are presented in Table 3.4. With a normal probability distribution, the crop allocation differs from what it did with 3 scenarios. The value of the stochastic solution is shown to have increased due to the increased number of scenarios.

Table 3.4: Results of Farmer's Problem

	Stochastic Problem	**Nominal Design**
Number of scenarios	15	15
Probability distribution	Normal	Normal
Land allocation (acres):		
Wheat	136	120
Corn	87	80
Sugar Beets	277	300
Expected Profit ($)	106,792	105,040
Value of stochastic solution ($)	1752	-

Chapter 4

Renewable Energy System (RES)

The system under study consists of two different renewable energy sources, namely solar and wind, an electric grid on the supply side, and the electricity requirement of a medium-sized industrial plant on the demand side. Solar panels (PV) and offshore wind turbines (WT) can be used for generating electricity for the renewable energy system (RES). A schematic overview of the system is illustrated in Figure 4.1. If user demand is satisfied, excess electricity can be exported to the grid at a constant feed-in tariff (FiT). On the other hand, if renewable energy production cannot satisfy demand, electricity can be imported from the grid at cost OC_{grid}.

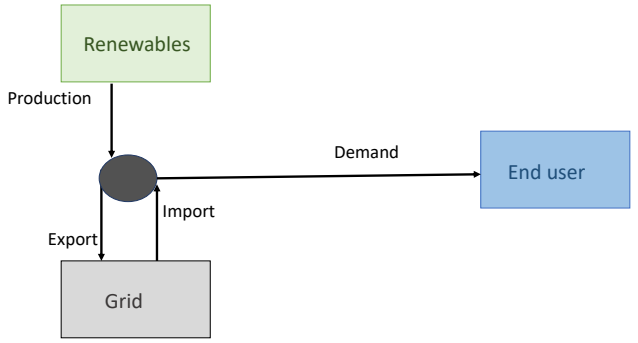


Figure 4.1: Simplified flowsheet of the renewable energy system with power generation and user demand. No storage technologies included. The energy balance is shown in Equation 5.10.

Area available for installation of each technology as well as area required per unit of technology is fixed. The area in m^2 required per m^2 of photo-voltaic panels is set at a ratio of 1:1, whereas the area required for a wind turbine is a function of rotor-blade diameter where the rotor-blade diameter d_r in Equation 4.1 is set to 164 metres. [14].

$$A_{req,WT}(d_r) = d_r \cdot d_{row} \cdot d_{col} \tag{4.1}$$

d_{row} and d_{col} is the distance between two turbines in a row and column, and set to 5 and 7 times the rotor-blade diameter, respectively.

The objective is to design the RES optimally under uncertainty. The intermittent nature of both wind speed and solar intensity as well as the volatile market spot price for electricity introduces uncertainty into the system. Another complicating factor is the variation in demand with peak hours and seasonal changes.

The study of the RES has been split into three cases which each attempts to account for uncertainty and volatility through different approaches. The three case studies also represent increasing levels of complexity. The first case handles uncertainty by using one design day and is formulated in Section 5.1. In the second section (5.2), seasonal variability is introduced through the implementation of four design days, and in the last section (5.3), battery storage is included, thereby exploiting peak production hours and potentially satisfying peak demand. For instance, if production cannot satisfy demand, one can discharge the battery, or, conversely, if production surpasses demand, charge the battery, increasing the reliability of the system. The RES with storage technologies included is illustrated in Figure 4.2.

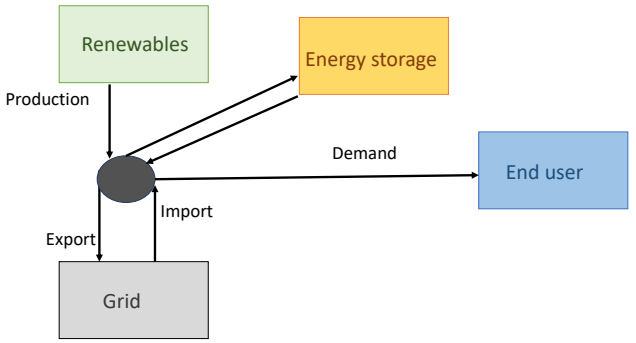


Figure 4.2: Simplified flowsheet of the renewable energy system with power generation, user demand, and battery as chosen storage technology.

Chapter 5

Case studies

In the following case studies, uncertainty is assumed to be linked with and limited to the electric grid price, OC_{grid}, wind speed, W, solar intensity, I, and demand, f^{demand}, whose mean values are presented in Table 5.3. To quantify the value of the stochastic approaches in Sections 5.1, 5.2 and 5.3, the expected value problem (EVP) is solved to obtain the expectation of expected value (EEV), and subsequently calculate the value of the stochastic solution (VSS).

In the expected value problem the program uses the expected values of the uncertain parameters to determine the optimal solution. Consequently, only one scenario is generated and used throughout the system's lifetime. Thus, the expected value problem can be viewed to be a deterministic optimization approach as no parameters are considered uncertain. Furthermore, because the program has access to perfect information there is only one decision stage, meaning design and operational decisions are made simultaneously. In the expected value problem all variables listed in Table 5.1 are denoted as 1^{st} stage variables.

5.1 Case study 1: Simple model of uncertainty with 1 design day

The stochastic program accounts for uncertainty by scenario representations as described in Section 2.3. Decision variables for the design (size) of the technologies belong to the first stage, whereas operational decisions belong to the second stage.

Objective function

The objective is to minimize the overall cost of the RES as defined by Equation 5.1. This can be split into the capital costs, CAP, and the expected operational costs over the different scenarios, where OP_s stands for the operational costs in

scenario s. The capital costs are determined in the 1^{st} stage, and operational costs are incurred in the 2^{nd} stage.

$$Obj_{min} = CAP(\boldsymbol{x}) + \sum_{s} p_s \cdot OP_s(\boldsymbol{y}) \qquad \forall \text{ s} \in \text{S} \tag{5.1}$$

Here $\boldsymbol{x}$ is a vector of binary decision variables in the 1^{st} stage, p_s is probability of scenario s and $\boldsymbol{y}$ is a vector of decision variables in the 2^{nd} stage. The capital cost term can be written as,

$$CAP(\boldsymbol{x}) = FC_{PV}(x_{PV}) + FC_{WT}(x_{WT}) \tag{5.2}$$

where FC_i is fixed cost for technology i, where the size of technology i is determined in the first stage. Subscript PV indicate photovoltaic, WT wind turbine and t time interval. The operational costs in the second term of Equation 5.1 must in each scenario be summed up from t_0 to t_{final}. More specifically, it is the expected operation cost over the entire project lifetime T and is formulated in Equation 5.3.

$$OP_s(\boldsymbol{y}) = \sum_{t=0}^{T} f_{s,t}^{grid} \cdot OC_{grid,s} - \sum_{t=0}^{T} f_{s,t}^{export} \cdot FiT \qquad \forall \text{ s} \in \text{S} \tag{5.3}$$

Energy flow $f_{s,t}^{grid}$ is electricity imported from the grid and $f_{s,t}^{export}$ is renewable electricity exported to the grid, all denoted by scenario s and time t. $\text{OC}_{grid,s}$ is price of imported electricity in scenario s and FiT is the feed-in tariff for exported electricity back to the grid.

Capital cost estimation

The total fixed cost of technology i consists of an investment cost incurred at the start of project lifetime, $J_{c,i}$, and the annual maintenance costs $J_{m,i}$, denoted by year yr.

$$FC_i(x_i) = J_{c,i} + \sum_{yr=1}^{Y} J_{m,i} \qquad \forall \text{ i} \in \{\text{PV,WT}\} \tag{5.4}$$

As mentioned in Section 3.1 the 1^{st} stage variables must be limited to sets of discrete sizes to be guaranteed convergence of the NGBD solver. Consequently, the investment cost, $J_{c,i}$ is a function of the binary decision variable $z_{i,j}$ and a discrete capacity dependent cost function, $C_{i,j}$ where subscript j denote relative size of technology i.

$$J_{c,i} = C_{i,j} \cdot z_{i,j} \qquad \forall \text{ i} \in \{\text{PV,WT}\} \tag{5.5}$$

To ensure that only one size, $C_{j,i}$, is selected for each technology i, the variable constraint formulated in Equation 5.6 is imposed.

$$\sum_{j=0}^{d} z_{i,j} = 1 \qquad \forall \text{ i} \in \{\text{PV,WT}\} \tag{5.6}$$

The cost function, $C_{i,j}$, is calculated from Equation (5.7) where $C_{i,0}$ is cost of reference capacity, $S_{i,0}$, and the constant sfi is the economy of scale factor listed in Table 5.2.

$$C_{i,j} = C_{i,0} \cdot (\frac{S_{i,j}}{S_{i,0}})^{sfi}, \quad \forall \text{ i} \in \{\text{PV,WT}\}, \forall \text{ j} \in \{1,....,\text{d}\} \tag{5.7}$$

The maintenance cost, $J_{m,i}$ is simply a fraction, ξ, of the annual investment cost, $J_{c,i}$.

$$J_{m,i} = J_{c,i} \cdot \xi \quad \forall \text{ i} \in \{\text{PV,WT}\} \tag{5.8}$$

The value $S_{i,j}$ is determined from the discontinuous function (5.9) with bounds S_i^{UBD} and S_i^{LBD} listed in Table 5.1. The approximation was generated by Chen et al. [15]. Area (m^2) and number of wind turbines is used to indicate the size of installed solar and wind capacity, respectively.

$$S_{i,j} = S_i^{LBD} + \frac{j-1}{d-1} \cdot (S_i^{UBD} - S_i^{LBD}), \quad \forall \text{ i} \in \{\text{PV,WT}\}, \forall \text{ j} \in \{1,....,\text{d}\} \tag{5.9}$$

Where i denotes technology and j size interval of in total d_n number of discrete intervals.

Table 5.1: Variables with bounds, units and stage number.

Variable	Lower	Upper	Unit	Stage
S_{PV}	0	1 200 000	m^2	1
S_{WT}	0	14	[-]	1
$f_{s,t}^{grid}$	0	$f_{s,t}^{demand}$	MWh/day	2
$f_{s,t}^{export}$	0	∞ [1]	MWh/day	2

Table 5.2: Parameters describing the reference cost and minimum area requirement of the two technologies in Equation 5.7 for capital cost.

Symbol Unit	C_0 [$]	S_0 [MWh/day]	A_0 [m^2]	ξ [-]	sfi [-]
Solar PV	27 000	1	180	0.1025	0.7
WT	49 840 000	192	94 136	0.1025	0.7

System constraints

The system constraints consist of an energy balance as well as a demand and two capacity constraints. The energy balance for scenario s at time t is formulated in Equation 5.10, where energy flow $f_{s,t}^h$ ($h \in$ {export,grid,demand}) is energy flow

to or from sink/source h, and $P^i_{s,t}$ is power output from technology i in scenario s at time t.

$$f^{demand}_{s,t} + f^{export}_{s,t} = P^{PV}_{s,t} + P^{WT}_{s,t} + f^{grid}_{s,t} \quad \forall \text{ s} \in \text{S}, \quad \forall \text{ t} \in [0,\text{T}] \tag{5.10}$$

The power functions are formulated in Equation 5.16 and 5.15. The demand constraint is formulated in Equation 5.11.

$$P^{PV}_{s,t} + P^{WT}_{s,t} + f^{grid}_{s,t} \geq f^{demand}_{s,t} \quad \forall \text{ s} \in \text{S}, \quad \forall \text{ t} \in [0,\text{T}] \tag{5.11}$$

The individual capacity throughput constraints are formulated in Equation 5.13 and 5.12 where S_{PV} and S_{WT} are constrained by Equation 5.14.

$$P^{PV}_{s,t} \leq S_{PV} \quad \forall \text{ s} \in \text{S}, \quad \forall \text{ t} \in [0,\text{T}] \tag{5.12}$$

$$P^{WT}_{s,t} \leq S_{WT} \quad \forall \text{ s} \in \text{S}, \quad \forall \text{ t} \in [0,\text{T}] \tag{5.13}$$

$$S_i = \sum_{j=0}^{d} S_i \cdot z_{i,j} \quad \forall \text{ i} \in \{\text{PV,WT}\} \tag{5.14}$$

These constraints ensure that the production cannot surpass the capacity at any time interval in any scenario.

Energy models

The power output, $P_{PV\,s,t}$, from the solar panels is modelled after Equation 5.16,

$$P^{PV}_{s,t} < \eta_{PV}(I_{s,t}, \theta_{s,t}) \cdot I_{s,t} \cdot A_{PV} \tag{5.15}$$

where A_{PV} is installed solar PV area, $I_{s,t}$ is solar radiation per unit area, and, for simplicity, the conversion efficiency η_{PV} is assumed constant [15].

The power output, $P^{WT}_{s,t}$, from the wind turbines is modelled after Equation 5.16.

$$P^{WT}_{s,t} \leq \eta_{WT} \cdot q_{s,t} \cdot Z \tag{5.16}$$

The efficiency η_{WT} is also assumed constant, whereas $q_{s,t}$ is a function of wind speed and parameters specific for the model [16]. Z is number of wind turbines

$$q_{s,t} = \begin{cases} 0, & \text{if } W_{s,t} \leq W_{min} \\ q_d \cdot \frac{W^3_{s,t} - W^3_{min}}{W^3_d - W^3_{min}}, & \text{if } W_{min} \leq W_{s,t} < W_d \\ q_d, & \text{if } W_d \leq W_{s,t} < W_{max} \\ 0, & \text{if } W_{s,t} \geq W_{max} \end{cases} \tag{5.17}$$

Performance coefficients for the wind turbines can be found in Table 5.3. In the deterministic case, the wind speed will be constant at the average value for the south-west coast of Norway.

Power model efficiencies and other relevant parameters are summarized in Table 5.3. The uncertain variables with their associated mean value and standard deviation are listed in Table 5.4. A normal distribution is assumed for the uncertain variables.

Table 5.3: Efficiencies and other parameters used in the model.

Symbol	Value	Unit
W_{min}	3.5	m/s
W_{max}	25	m/s
W_d	13	m/s
q_d	8	MW
η_{PV}	0.15	-
η_{WT}	0.85	-
FiT	1.45	\$/MWh

Table 5.4: Summary of uncertain variables with their mean value and associated standard deviation.

Variable	Mean value	Standard deviation	Unit
W	7.5	1.05	m/s
I	$1.5e^{-4}$	$2.25e^{-5}$	MW/m^2
$f_{s,t}^{demand}$	100	15	MWh
$OC_{grid,s}$	15	2.25	\$/MWh

Summary of case 1

- The objective is to minimize the overall cost, the sum of capital and operational costs, formulated in Equation 5.1.
- The operational cost model is formulated in Equations 5.3.
- The capital cost model is formulated in Equations 5.4 to 5.9.
- The system energy balance is formulated in Equation 5.10.
- The primary energy conversion models are formulates in Equations 5.15, 5.16 and 5.17.
- The demand constraint is formulated in Equation 5.11.
- The capacity (linking) constraints are formulated in Equations 5.12 and 5.13.

5.2 Case study 2: Accounting for seasonal variability using 4 design days

Modelling seasonal variations through design days

Both solar radiation and wind are expected to change notably through the season, thus a mean value with standard deviation does not necessarily replicate the actual weather conditions for an entire year. Consequently, implementation of design days, as suggested in [6], is investigated. Seasonal variation is imposed by shifting the mean wind speed and solar intensity. In this manner, the uncertainty for the different seasons is represented by one common scenario tree, as shown previously in Figure 2.4. Table 5.5 summarize the different seasonal factors for solar intensity and wind speed.

Table 5.5: Modified mean values to simulate seasonal variations through 4 design days.

Season	Solar intensity mean $\overline{I}$	Wind speed mean $\overline{W}$
Spring	$\overline{I}$	$1.3{\cdot}\overline{W}$
Summer	$1.2{\cdot}\overline{I}$	$0.8{\cdot}\overline{W}$
Fall	$\overline{I}$	$\overline{W}$
Winter	$0.8\cdot\overline{I}$	$\overline{W}$

In case study 2, some of the model equations change. For instance, instead of summing from 0 to T in Equation 5.3, you sum over seasonal days meaning first winter days (t_w), then spring (t_s), summer (t_{su}) and lastly fall (t_f). Thereafter the function is scaled with the number of years, n_y. The modified expression is formulated in Equation 5.18. Conversely, the variable bounds in Table 5.1 with corresponding discrete intervals is identical in case study 1 and 2 along with the capital cost estimation in Equations 5.4 to 5.9.

$$\begin{aligned} OP_s(\boldsymbol{y}) = n_y \cdot \Bigg[& \sum_{t=0}^{t_w} f_{s,t}^{grid} \cdot OC_{grid,s} - \sum_{t=0}^{t_w} f_{s,t}^{export} \cdot FiT + \sum_{t=0}^{t_s} f_{s,t}^{grid} \\ & \cdot OC_{grid,s} - \sum_{t=0}^{t_s} f_{s,t}^{export} \cdot FiT + \sum_{t=0}^{t_{su}} f_{s,t}^{grid} \cdot OC_{grid,s} - \sum_{t=0}^{t_{su}} f_{s,t}^{export} \\ & \cdot FiT + \sum_{t=0}^{t_f} f_{s,t}^{grid} \cdot OC_{grid,s} - \sum_{t=0}^{t_f} f_{s,t}^{export} \cdot FiT \Bigg] \quad \forall \text{ s} \in \text{S} \end{aligned} \tag{5.18}$$

Summary of case 2

- The objective is to minimize the overall cost, the sum of capital and operational costs, formulated in Equation 5.1
- The modified operational cost model is formulated in Equations 5.18.
- The capital cost model is formulated in Equations 5.4 to 5.9.
- The system energy balance is formulated in Equation 5.10.
- The primary energy conversion models are formulates in Equations 5.15, 5.16 and 5.17.
- The demand constraint is formulated in Equation 5.11.
- The capacity (linking) constraints are formulated in Equations 5.12 and 5.13.

5.3 Case study 3: Stochastic problem with dynamic model and energy storage

With the introduction of energy storage technology, the energy balance and constraints are modified, and variables for charge, discharge and state-of-charge must be defined. The corresponding parameters and constraints for the battery dynamics must also be declared. Furthermore, energy and capital cost models as well as the objective function are expanded to accommodate the storage functionality. The battery is included such that on days with surplus production of power, energy can be stored, and discharged to compensate for energy production deficit on later days.

Scenario generation when modeling with timesteps

Keeping track of the state of charge of the battery requires a different approach for scenario generation than in Section 5.1 and 5.2. Previously, sequential days were entirely independent of each other, but with the inclusion of a battery, the optimal operation is dependent on earlier states of the system. Consequently, the decision-making process can be represented by a multi-stage stochastic program, in which each stage represents a time step.

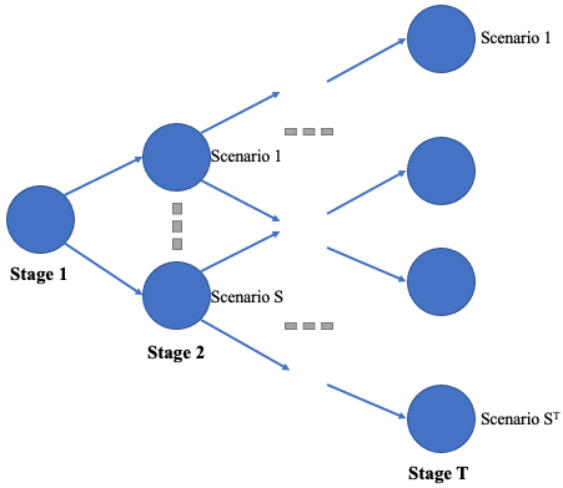


Figure 5.1: Scenario tree for a multi-stage stochastic program

As can be seen in Figure 5.1, the number of scenarios increase exponentially with the number of time steps when modeling dynamic behavior. For instance, in a stochastic optimization problem with 12 scenarios each day, the result is approximately 35 million (12^7) scenarios when simulating one week with daily resolution. As noted in Section 3.1, GOSSIP requires an input on the form of a two-stage stochastic program. Moreover, multi-stage stochastic programming is widely ac-

knowledged to be computationally intractable [13]. Instead of the multi-stage formulation shown in Figure 5.1, the two-stage multi-period formulation, with corresponding scenario tree in Figure 5.2 is used. With this approach it is assumed that the uncertain parameters can be aggregated over the entire period considered. In other words, for a given scenario branch, all uncertain parameters for the entire period are assumed to be known. As a result, all operating decisions can be made optimally at the first time-step.

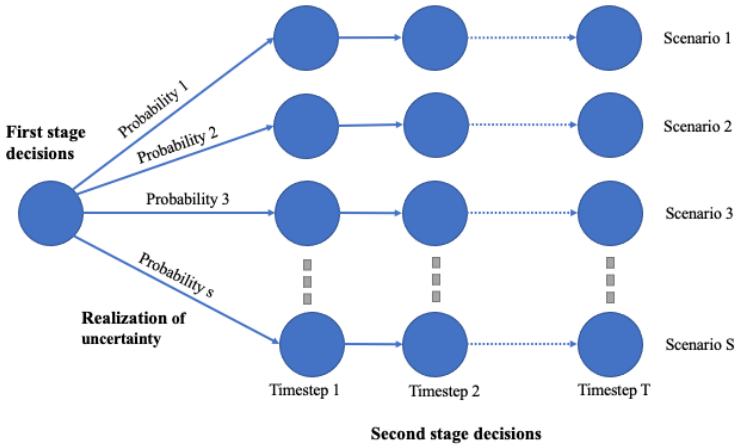


Figure 5.2: Scenario tree for a two-stage stochastic program

The scenarios are constructed for the period of 7 consecutive days, where the parameter realizations for each day is generated arbitrarily from the same mean values and standard deviations given previously in Table 5.3. The associated probability for each scenario is then given relative to the probability of any of the scenarios in the scenario tree, given by equation 5.19.

$$w_s = \frac{\prod_{t=1}^{7} p_{t,s}}{\sum_{s=1}^{S} \prod_{t=1}^{7} p_{t,s}} \tag{5.19}$$

Here w_s is the weighted probability of each scenario and $p_{t,s}$ is the actual probability of the parameter realization for each time-step.

Objective function

The objective function formulation in Equation 5.20 is the same as in case 1 and 2, but the terms for capital and operating cost (Equation 5.21 and 5.22) are expanded to include energy storage expenditures.

$$Obj_{min} = CAP(\boldsymbol{x}) + \sum_{s} p_s \cdot OP_s(\boldsymbol{y}) \tag{5.20}$$

$$CAP(\boldsymbol{x}) = FC_{PV}(x_{PV}) + FC_{WT}(x_{WT}) + FC_{ES}(x_{ES}) \tag{5.21}$$

$FC_{ES}(x_{ES})$ is fixed cost related to the energy storage, where x_{ES} is a first stage decision variable determining capacity of storage technology installed. In Equation 5.22 the variables $f_{s,t}^{grid}$ and $f_{s,t}^{export}$ are now functions of charge and discharge of the battery as well as the energy production from solar PV and wind ($P_{s,t}^{PV}$ and $P_{s,t}^{WT}$). Furthermore, the operational costs in Equation 5.22 are scaled with number of weeks in a year n_w (52) and the plant lifetime n_y (30 years).

$$OP_s(\boldsymbol{y}) = n_y \cdot n_w \cdot \left[\sum_{t=0}^{7} f_{s,t}^{grid} \cdot OC_{grid,s} - \sum_{t=0}^{7} f_s^{export} \cdot FiT \right] \quad \forall \text{ s} \in \text{S} \tag{5.22}$$

Capital cost estimation

The capital cost for energy storage is calculated in the same manner as solar PV and wind, and bound by the same constraints (Equation 5.6 and 5.14). The cost calculation parameters for energy storage are summarized in Table 5.6.

Table 5.6: Capital and operating cost for energy storage [17]

Symbol Unit	C_0 [$]	S_0 [MWh/day]	A_0 [m^2]	ξ [-]	sfi [-]
Energy storage	50000	1	-	-	0.6

Table 5.7: List of discrete values for S_{ES} calculated from Equation (5.9).

Capacity Unit	S_{ES} [MW]
1	0
2	18
3	36
4	54
5	72
6	90
7	108
8	126
9	144
10	162
11	180
12	200

Constraints

The constraints in the system consist of an energy balance as well as demand and capacity constraints. The energy balance for the overall system in scenario s at time t is formulated in Equation 5.23.

$$f_{s,t}^{demand} + f_{s,t}^{charge} = P_{s,t}^{PV} + P_{s,t}^{WT} + f_{s,t}^{import} + \frac{f_{s,t}^{discharge}}{\eta_{ES}^{discharge}} \quad \forall \text{ s}\in \text{S}, \forall \text{ t}\in[0,\text{T}] \quad (5.23)$$

Net energy exported to end user and grid ($f_{s,t}^{export}$), defined in Equation 5.22, is formulated in Equation 5.24.

$$f_{s,t}^{export} = P_{s,t}^{PV} + P_{s,t}^{WT} + \frac{f_{s,t}^{discharge}}{\eta_{ES}^{discharge}} \qquad \forall \text{ s} \in \text{S}, \quad \forall \text{ t} \in [0,\text{T}] \qquad (5.24)$$

In addition to the production capacity constraints in Equation 5.12 and 5.13, charge, discharge and state of charge capacity constraints for the battery is formulated in Equation 5.25, 5.26 and 5.27.

$$f_{s,t}^{charge} \leq S_{ES} - ES_{s,t}^{SoC} \quad \forall \text{ s} \in \text{S}, \quad \forall \text{ t} \in [0,\text{T}] \qquad (5.25)$$

$$f_{s,t}^{discharge} \leq ES_{s,t}^{SoC} \quad \forall \text{ s} \in \text{S}, \quad \forall \text{ t} \in [0,\text{T}] \qquad (5.26)$$

$$ES_{s,t}^{SoC} \leq S_{ES} \quad \forall \text{ s} \in \text{S}, \quad \forall \text{ t} \in [0,\text{T}] \qquad (5.27)$$

$ES_{s,t}^{SoC}$ is the battery state of charge, which cannot surpass the maximum capacity of the battery, S_{ES}. Equation 5.25 and 5.26 state that charge and discharge cannot be higher than available capacity and stored capacity, respectively.

Battery model

In addition to the energy balance in Equation 5.23, the battery energy balance for scenario s at time t is formulated in Equation 5.28. A dynamic model is required because the state of charge for the battery ($ES_{s,t}^{Soc}$) depend on previous states of the system. This can bee seen from Equation 5.28 which contain terms from both the current and previous time-step. Parameters associated with the battery model are summarized in Table 5.8.

$$ES_{s,t}^{SoC} - (ES_{s,t-1}^{SoC} \cdot \eta_{ES}^{SelfDischarge}) = (f_{s,t}^{charge} \cdot \eta_{ES}^{charge}) - (\frac{f_{s,t}^{discharge}}{\eta_{ES}^{discharge}}) \qquad (5.28)$$

Table 5.8: Storage capacity bounds and efficiencies in the energy storage model.

Symbol	Value	Unit
ES_{min}	0	MW
ES_{max}	200	MW
$\eta_{ES}^{discharge}$	0.99	-
η_{ES}^{charge}	0.99	-
$\eta_{ES}^{efficiency}$	0.99	-

Summary of case 3

- The objective is to minimize the overall cost, the sum of capital and operational costs, formulated in Equation 5.20
- The modified operational cost model is formulated in Equation 5.18.
- The capital cost model is formulated in Equations 5.4 to 5.9. The specific capital cost equation for the battery model is formulated in Equation 5.21.
- The system and battery energy balances are formulated in Equations 5.23 and 5.28
- The primary energy conversion models are formulated in Equations 5.15, 5.16 and 5.17.
- The overall demand constraint is formulated through the energy balance in Equation 5.23.
- The capacity (linking) constraints are formulated in Equations 5.12, 5.13, 5.25, 5.26 and 5.27.

Chapter 6

Results

Results for the three case studies in Chapter 5 are presented in Sections 6.1, 6.2 and 6.3.

The result from running the expected value problem is presented in Table 6.1. The optimal 1^{st} stage variables for the expected value problem in Table 6.1, the nominal design, is used to calculate the expectation of expected value (EEV) and subsequently the value of stochastic solution (VSS) in Sections 6.1, 6.2 and 6.3. The calculations of EEV and VSS is found in Appendix A.

Table 6.1: Results of the expected value problem.

Expected value problem	
Number of scenarios	1
Design Decisions	
Area of Solar PV (m^2)	890,000
Number of Wind Turbines (-)	0
Maximum energy production (MWh/day	
Solar PV	120.15
Wind	0
Expected energy production (MWh/day)	
Solar PV	100.25
Wind	0
Expected energy supply (MWh/day)	
Grid import	0

6.1 Case study 1: Simple model of uncertainty with 1 design day

The result from running the stochastic problem in Section 5.1 is summarized in Table 6.2. The 81 scenarios are a result of setting the number of realizations of each of the four uncertain parameters to 3. As for the cost calculations in Table 6.2, only a basic economic model was used: Depreciation of the capital costs as well as discounting of future cash flows were not considered. These will be studied in future work.

Table 6.2: Results for the stochastic problem.

	Flexible design	**Nominal design**
Number of scenarios	81	81
Design Decisions		
Area of solar PV (m^2)	1,070,000	890,000
Number of wind turbines	0	0
Maximum energy capacity (MWh/day)		
Solar PV	144.45	120.15
Wind	0	0
Expected energy production (MWh/day)		
Solar PV	120.38	100.13
Wind	0	0
Expected energy import (MWh/day)		
Grid	1.00	5.31
Expected cost ($)		
Total cost	9,560,000	10,037,000
Annual cost	319,000	335,000
Average cost per MWh produced	7.25	9.15
VSS ($)		
Total VSS	482, 000	-

Security of clean energy supply

The expected % of demand covered by renewable energy for case 1 in Section 5.1 is illustrated in the upper plot of Figure 6.1. The expected coverage from renewables and import in Figure 6.1 were calculated from Equation A.4 in Appendix A.

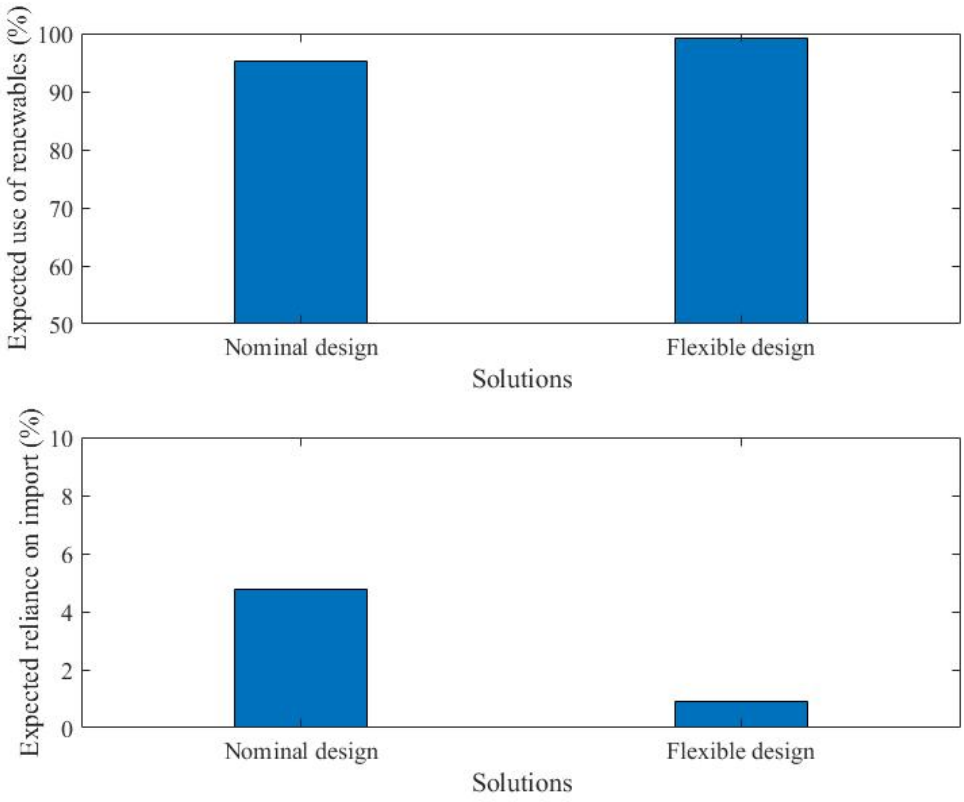


Figure 6.1: The expected coverage of demand by renewable energy production (top) and import (bottom) for the flexible and nominal design.

Expected levelized cost of energy (LCOE)

The expected levelized cost of energy in $/MWh produced from renewables in the flexible and nominal design is illustrated in Figure 6.4. The values were calculated as explained in Equation A.3 in appendix A with a discount rate r_i of 10%.

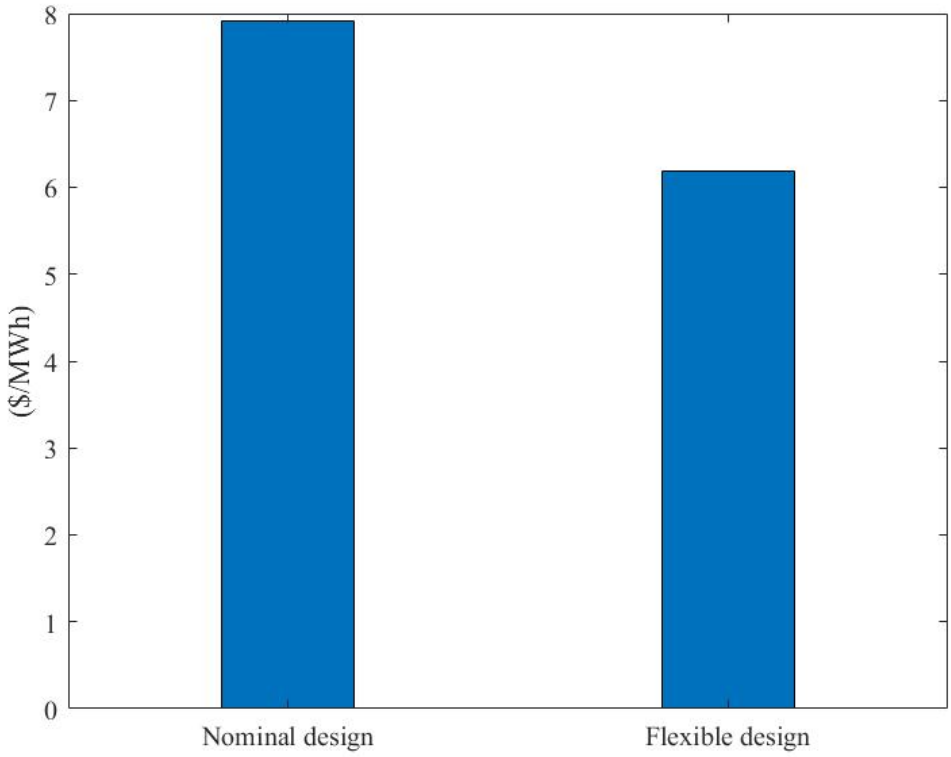


Figure 6.2: The LCOE in $/MWh produced from renewables in the nominal and flexible design.

6.2 Case study 2: Accounting for seasonal variability using 4 design days

The result from running the optimization problem in Section 5.2 with seasonal variation is summarized in Table 6.3. The results from the nominal design[1] is summarized in Table 6.4. The 81 scenarios are a result of setting the number of realizations of each of the four uncertain parameters to 3. As described in Section 5.2, there is still only one scenario tree, but the mean values are shifted to represent the different seasons. The cost calculations in Table 6.3 were as mentioned in Section 6.2 computed with a basic economic model.

Table 6.3: Results for the flexible design under seasonal variation.

Flexible design				
Season	Winter	Spring	Summer	Fall
Number of scenarios	81	81	81	81
Design Decisions				
Area of solar PV (m^2)	1,110,000			
Number of wind turbines	0			
Maximum energy capacity (MWh/day)				
Solar PV	119.88	149.85	179.82	149.85
Wind	0	0	0	0
Expected energy production (MWh/day)				
Solar PV	99.90	137.90	162.86	137.90
Wind	0	0	0	0
Expected energy import (MWh/day)				
Grid	6.21	0.52	0	0.52
Expected cost ($)				
Total cost	9,290,000			
Yearly cost	310,000			
Average cost per MWh produced	6.80			
VSS ($)				
Total VSS	708,000			

[1] Same nominal design as presented in case 1, Section 6.1

Table 6.4: Results for the nominal design under seasonal variation.

Nominal design				
Season	Winter	Spring	Summer	Fall
Number of scenarios	81	81	81	81
Design Decisions				
Area of solar PV (m^2)	890,000			
Number of wind turbines	0			
Maximum energy capacity (MWh/day)				
Solar PV	96.12	120.15	144.18	120.15
Wind	0	0	0	0
Expected energy production (MWh/day)				
Solar PV	80.10	100.13	120.15	100.13
Wind	0	0	0	0
Expected energy import (MWh/day)				
Grid	20.43	5.13	0.50	5.13
Expected cost ($)				
Total cost	10,960,000			
Yearly cost	365,000			
Cost per MWh produced	9.70			

Security of clean energy supply

The expected % of demand covered by renewable energy for case 2 in Section 5.2 is illustrated in the upper plot in Figure 6.3. The expected coverage from renewables and import in Figure 6.1 were calculated from Equation A.4 in Appendix A.

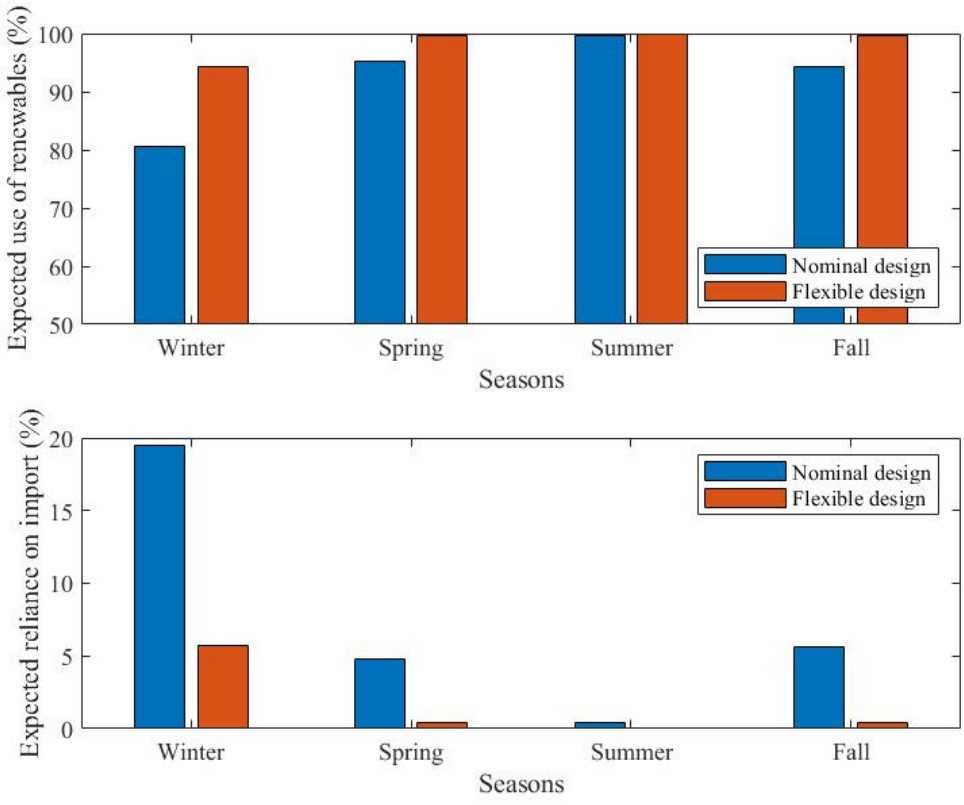


Figure 6.3: The expected coverage of demand by renewable energy (top) and electricity import (bottom) under seasonal variation for the nominal and flexible design.

Expected levelized cost of energy (LCOE)

The expected LCOE in $/MWh produced from renewables in the flexible and nominal design under seasonal variability is illustrated in Figure 6.4. The values were calculated as explained in Equation A.3 in appendix A with a discount rate r_i of 10%.

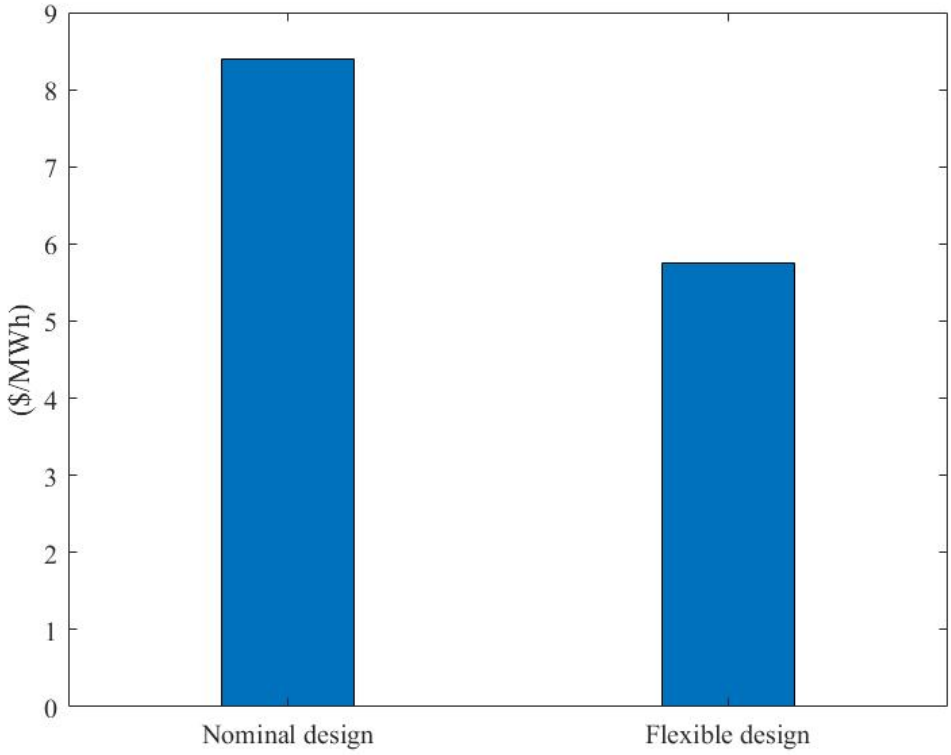


Figure 6.4: The LCOE in $/MWh produced from renewables in the seasonal stochastic problem for the nominal and flexible design.

6.3 Case study 3: Stochastic problem with dynamic model and energy storage

The results from running the model in case 3, Section 5.3 are presented in Table 6.5. The results are based on selecting scenario samples over the course of 7 days, which can be found in Appendix C. Design decisions, operational results, and the objective values are shown for the nominal design, and the flexible design with and without battery.

Table 6.5: Results for case study 3.

Design	**Nominal**	**Flexible**	
	w/o battery	w/o battery	w/battery
Number of scenarios	10	10	10
Number of days	7	7	7
Probability distribution	Sampling	Sampling	Sampling
Design Decisions			
Area of solar PV (m^2)	890,000	890,000	890,000
Number of wind turbines (-)	0	0	0
Battery (MWh)	-	-	40
Expected energy production (MWh/day)			
Solar PV	101.18	101.18	101.18
Wind	0	0	0
Expected energy import (MWh/day)			
Grid (import)	9,97	9,97	4.95
Grid (export)	7.01	7.01	2.67
From battery	-	-	6.84
Expected state of charge (%)	-	-	36.37
Expected cost ($)			
Total cost	11,300,000	11,300,000	10,974,000
Yearly cost	377,000	377,000	366,000
Value of Battery ($)			
Total	-	-	326,000
Yearly	-	-	10,867

Security of clean energy supply

The expected % of demand covered by renewable energy in case 3 for the nominal design and flexible design with and without battery, is presented in the upper plot of Figure 6.5. The lower plot shows the expected amount of energy required from import. The values in Figure 6.5 were calculated from Equation A.4 in Appendix A.

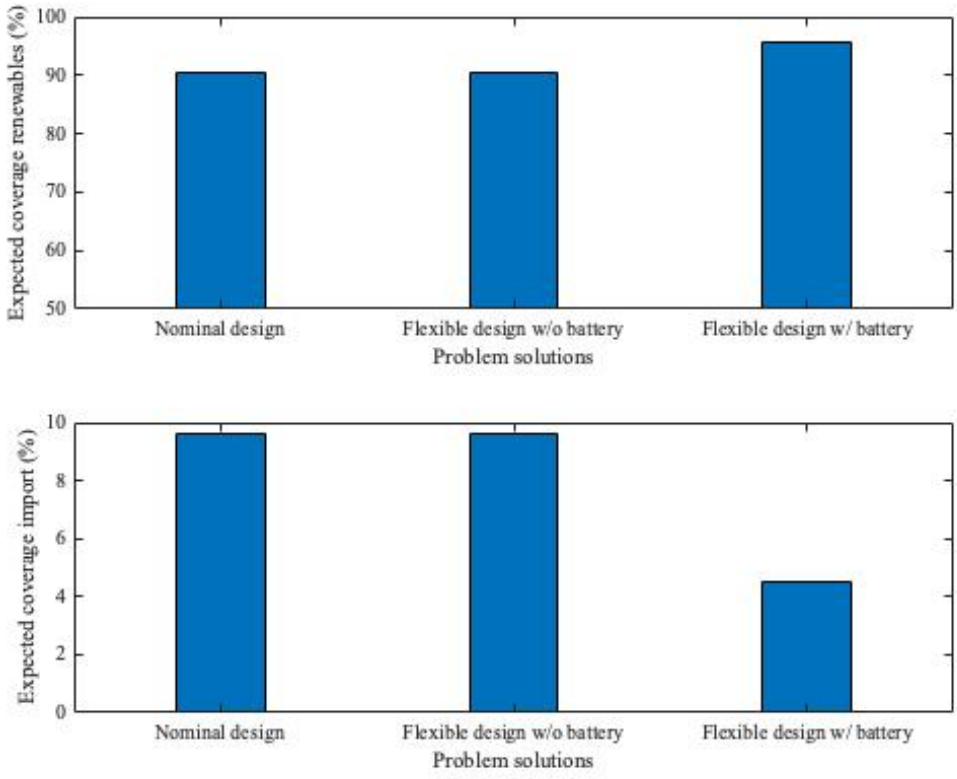
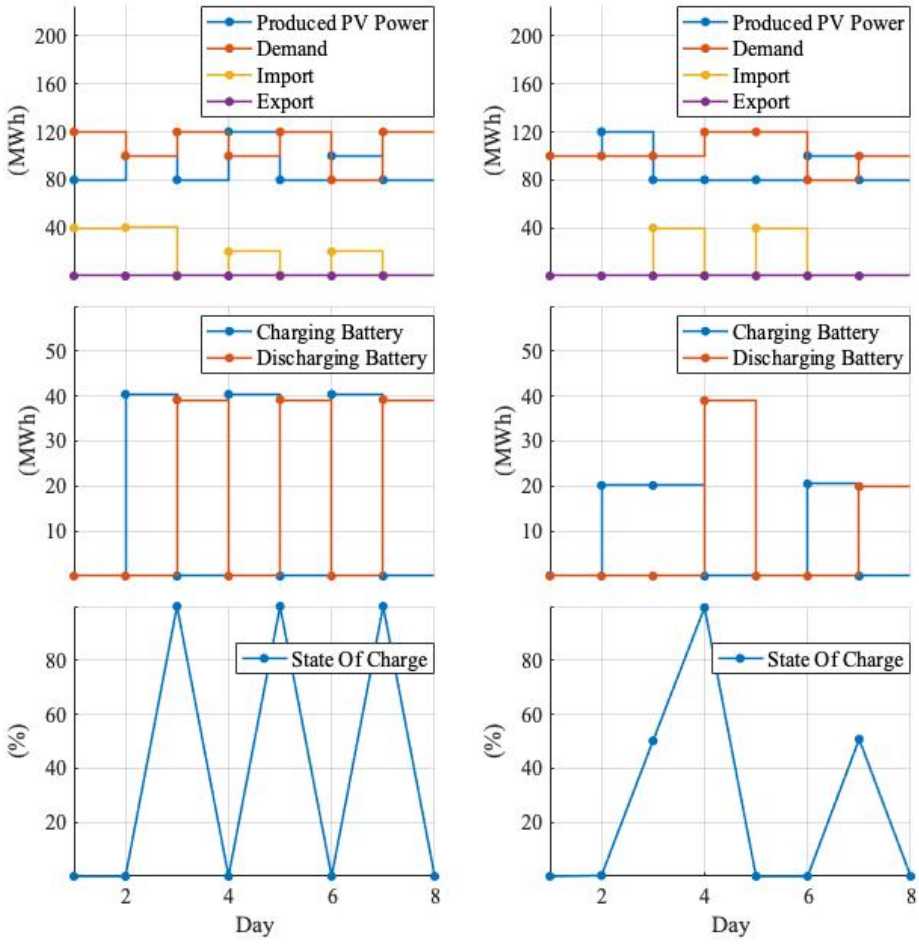


Figure 6.5: The expected coverage of demand by renewable energy production (top) and import (bottom) for the nominal design, and flexible design with and without energy storage.

Battery operation

Two example scenarios from the scenario set in Appendix C have been chosen to illustrate the operational pattern of the battery with respect to the rest of the system. In Figure 6.6a and 6.6b the flows in the system and state of charge for the battery is illustrated for example scenario 9 and 6 respectively.



(a) Scenario 9 **(b)** Scenario 6

Figure 6.6: System flows and operation of battery

Chapter 7

Discussion

Case 1 and 2

In case study 1 and 2, it is attempted to account for uncertainty by the inclusion of all generated scenarios, but the opportunity for energy storage is not incorporated.

System design and performance

The results in Table 6.2, 6.3 and 6.4 show that compared to the nominal design, a larger solar capacity is installed in the flexible solutions. This is in accordance with what is expected as variation in demand, and the varying energy output per m^2 of the solar PV-panels under varying solar intensity is accounted for. Moreover, the installed solar capacity is increased with the inclusion of seasonal variation in the problem formulation. This is expected as seasonal variation in practice further increases the volatility of solar intensity. A larger installed capacity makes up for potentially lower output per m^2 of solar panels. A result of this is the increased security of clean energy supply illustrated in Figures 6.1 and 6.3, and consequently the lowered LCOE illustrated in Figure 6.2 and 6.4.

Market conditions

There is a trade-off between the capital expenses associated with a larger plant, and the economic penalty from importing electricity. Locating this trade-off optimum is in part what yields the value of stochastic solution in Table 6.2 and 6.3. The constant feed-in-tariff is a simplification of real market conditions as surplus production often results in a decrease in electricity price. Thus, the constant feed-in-tariff rewards surplus production. As a result, the risk of installing a slightly too large energy plant is substantial. Adding a constraint to limit production at favorable feed-in-tariff is expected to affect capacity installed, meaning the aforementioned trade-off optimum is shifted towards a smaller plant. Similarly, there

is no seasonal variation in demand and market spot price. Normally, energy production from solar PV is reduced during the winter months whereas user heating demand increases. This is turn affects the market spot price for electricity as it is heavily dependent on the supply and demand relationship between producer and end-user. Including seasonal variation in market spot price and demand is expected to aggravate the EEV, thus increasing the VSS of the seasonal model. In summary, the optimal solution minimizes the expected total cost for the system under varying weather and market conditions. With the present simplifications, the main trade-off is between increased capital cost and the economic penalty associated with deficit production.

Modelling approach

In case 1 and 2 sequential days are modeled separately and scaled up in the objective function to simulate the plant lifetime. Thus, there is no correlation between the weather and/or market conditions today and tomorrow, imposing a significant simplification on the model behavior. Moreover, the scenario generation in the seasonal model is simplified to consist of one scenario tree, however with 4 shifted means, each representing one season. Together with the disjoint modeling of consecutive days, this simplification is believed to reduce the cogency of the resulting solution.

Case 3

The battery was introduced to combat the volatile renewable energy production and was thought to affect the optimal design decision. A change in modeling approach was required as the state of charge of a battery depends on consecutive days. To calculate the value of the stochastic solution (VSS), and the value of adding a battery, the model was run for a given scenario set with the nominal solution, as a stochastic problem, and as a stochastic problem with the possibility of energy storage.

Value of energy storage

The results summarized in Table 6.5 show that even though the nominal design and the stochastic problem (SP) yield the same results, the inclusion of the battery is expected to add value to the energy system. Comparatively to the nominal design and the stochastic solution without the battery, the expected cost of the system is reduced without a change in the 1^{st} stage decision variables.

The size of the installed battery in the optimal solution is relatively high with a capacity of about 40% of average demand. This means that the system can offset a fluctuating power supply for multiple different scenarios, covering both high surplus and deficit, as well as sequential days of moderate surplus and deficit. In other words, renewable energy is more evenly distributed through charge and discharge

of the battery on days with surplus and deficit, respectively. In this manner, the expected amount of imported electricity to cover demand is substantially reduced.

As seen in Figure 6.5, the battery allows for a larger fraction of the renewable power produced to be utilized to cover demand compared to the solution without a battery. This adds value to the stochastic solution as the system's intended purpose is to cover demand solely with renewable power.

A possible explanation for the identical 1^{st} stage decisions is that the expected production is marginally larger than the expected demand for the stochastic problem without a battery, meaning that the program entails a risk of importing a larger amount of energy if it would attempt to cut capital cost by installing less solar PV panels.

Modelling approach

Figure 5.2 in Section 5.3 show how the introduction of time-dependence, and thus the battery, affect the scenario generation. The explosion in the number of scenarios was attempted to be dealt with by arbitrarily selecting a sample set of 10 scenarios from the original tree. However, since the total number of scenarios generated is 81^d, where d is the number of days, the risk of choosing an unrepresentative set of scenarios when only choosing 10 different ones is considerable. Such an erroneous decision could have affected the solution to the stochastic problem in Table 6.5. Specifically, it is expected that this could explain why the nominal and flexible design is identical. If a more representative set of scenarios had been selected, a more valuable solution compared to the nominal design would be expected.

Operation of battery

In Figures 6.6a and 6.6b, two example scenarios were chosen to illustrate the operation of the battery. As can be seen from the plots in Figure 6.6a and 6.6b, the battery is charged whenever there are surplus production and available capacity, and discharged when there is a production deficit. This illustrates how renewable energy is stored at one point and consumed at a later point to reduce the amount of imported energy. However, an effect of representing the actual multi-stage scenario tree in Figure 5.2 by a two-stage scenario tree in Figure 2.4 becomes apparent. From the plots in Figures 6.6a and 6.6b it can be seen that in certain time-steps the program picks solutions that imports electricity from the grid despite zero deficit between production and demand. By simplifying the multi-stage tree to a two-stage tree, the uncertainty is only revealed once. In a multi-stage scenario tree, the uncertainty of one stage is not revealed before the decisions in the previous stage have been made. For the scenario approach used in the two-stage stochastic program, decisions for different days can be made simultaneously. The battery operation will therefore work as a wait-and-see program, meaning that the program can wait and see what happens at the final time-step before making a decision in the first time-step. In this way, the program can accurately predict the grid price,

the demand, and how much electricity it can produce in every single time-step.

Therefore, in addition to storing surplus energy, thus coping with fluctuating energy sources, the battery can be used to store purchased electricity from the grid whenever it is profitable. For instance, if the grid import price today is lower than the average, and the program knows that there will be a deficit and higher import price on a future day, it makes sense to import the energy today, given available storage capacity. This behavior can be seen in Figure 6.6a at day 2, day 4 and day 6, as well as the third day in Figure 6.6b, where the program finds it beneficial to import and store energy in the battery despite charge and discharge losses. This might also partly explain why the original design decisions are left unchanged despite the installation of the battery. If the program can mitigate the negative effect of importing electricity by lowering the effective cost of import, the solutions that give a higher fraction of import electricity become less unfavorable. This is not optimal as the future always brings uncertainty. Nonetheless, if one can assume the weather forecasts to be relatively certain a week forward in time, this simplified approach could prove to be valuable.

Further remarks

Technology selection

The results in Sections 6.1, 6.2 and 6.3 show that wind turbines are not installed in any of the three cases. This could be a result of the higher capital cost per MW installed of offshore wind (Table 5.2), and the fact that the installed solar capacity can supply 100% of demand under nominal weather conditions. Thus, there is no need for additional capacity from wind turbines. Furthermore, while the effective capacity from solar PV is limited solely by the solar intensity, the wind turbines are turned off if the wind speed surpasses W_{max} (Table 5.3). This makes solar PV more valuable under "extreme" weather conditions. Lastly, the economies of scale have an impact on the capital expenses, and the effect is less substantial if two smaller and different technologies are installed compared to the installation of one large solar PV plant. It is therefore in many cases beneficial to choose one technology at a lower cost per MW installed instead of two technologies and obtain higher flexibility with regards to weather conditions.

Financial policy

The program entails a significant risk of overestimating the economic value of renewable power generation from both solar PV and wind as the objective function does not take the time value of money into account. In real life, the present value of future revenue is considered to be less valued. The exclusion of this concept in the problem formulation leads to a significant underestimation of the capital costs associated with the renewable energy system. In effect, larger system capacities appear more favorable than what is truly the case.

Applicability of renewables

The issue of hourly fluctuation is not addressed as the uncertain parameters are set constant for the duration of an entire day. The renewable system's ability to cover daily fluctuations is therefore not evaluated. This simplification can result in an erroneous estimate of amount of power produced and required, thereby posing the risk of producing non-optimal decisions. The daily average approach is expected to have affected both the value of renewable sources and the value of technologies installed. Hourly fluctuations make renewable energy less applicable to cover a constant demand. Lastly, the limited hours of sun compared to wind would affect the applicability of solar PV with hourly changes (resolution).

Value of renewables

The simplified energy conversion models may have affected the valuation of the renewable systems. Conversion of both wind and solar to electric power is modeled with a linear relationship that does not change with time. Reduction in the capacity and efficiency of the battery is also assumed to be negligible. In a real-life system, degradation over time would have affected the components' capability. In addition to a decrease in battery capacity, the efficiency of the solar panels and wind turbines would drop. Consequently, the program entails the risk of choosing a too small system caused by an overestimation of the system's ability to deliver clean renewable energy in the later years.

Another way to look at the problem formulation with and without energy storage is to view the concepts as two different approaches to dealing with uncertainty. As shown in the results for case study 1 and 2 presented in Section 6.1 and Section 6.2, the program handles uncertainty, and adds value by increasing the amount of power produced at all times, reducing the amount of imported electricity. While the key logic there is to reduce the amount of imported power through larger production capacity, the battery adds value by distributing generated power more evenly. In general, it is expected that an energy storage system can benefit a stochastic solution by allowing a reduction in installed capacity while retaining high security of energy supply. In this thesis, an attempt was made to handle short-term uncertainty and variation through the inclusion of day to day storage. However, seasonal variation was not imposed on the system, and the effect on long-term variability was therefore not tested. The solution to counteract seasonal variation in Section 6.2 was to install more solar panels, thus it is expected that there is a potential value of long-term energy storage.

Linear approximation

Although the linear approximations impose the risk of sub-optimal results, they ensure a convex optimization problem. This improves solver time and guarantees that a local optimum is a global optimum. The inclusion of nonlinear degradation equations is expected to result in a significant increase in both modeling and

computational complexity as the optimization problem could become non-convex. However, this increased complexity might prove to be crucial for the robust optimization of flexible renewable energy systems.

Chapter 8

Conclusions and future Work

Conclusions

Through a two-stage stochastic MILP, the optimal design and operation of a flexible renewable energy system were investigated. The result from the three case studies gave valuable insight into the different aspects and problems associated with the optimization of flexible renewable energy systems.

Firstly, the case study based on one design day illustrated how the stochastic program coped with uncertainty by increasing the flexibility of the system through increased capacity of power generation. Secondly, the inclusion of seasonal variation through four design days in case study two then showed how the stochastic program further increased the installed capacity to cope with the increased volatility of the uncertain parameters. The main trade-off in the first two case studies was shown to be between the economic perspective of design and operation, where the stochastic solution added value by investing more upfront in order to avoid later economic penalties from not meeting energy demand. Lastly, the dynamic model in the third case study illustrated how energy storage can reduce the expected amount of imported power, thereby increasing the flexibility of the renewable energy system. With the installation of a battery, the renewable energy produced was more evenly distributed over the course of a week, resulting in a smaller capacity of solar PV installed compared to the first two cases.

Overall, total expenses were reduced and the % of demand covered by renewable energy increased for the flexible design compared to the nominal design. In conclusion, the case studies showed that stochastic programming in optimization of RES is a novel method to account for uncertainty. In addition, case study three showed that the implementation of a battery decreased overall costs, thus illustrating how uncertainty can be counteracted through the inclusion of energy storage. All three cases were modeled and solved using GOSSIP, demonstrating the software's applicability to optimization problems in flexible renewable energy

systems.

Future work

In this thesis, energy conversion was modeled by linear approximations. This means that degradation effects and non-linear relationships between model variables were neglected and omitted from the program. A logical next step would be to use a more rigorous approach to the energy modeling in the system to achieve a more realistic model. For instance, account for the drop in efficiency for both solar PV panels and wind turbines over time.

Furthermore, to cope with the exponential increase of scenarios in the dynamic model in case 3, scenarios were generated by random sampling from a discrete normal distribution. This is a simplification of the situation, thus future work for the dynamic model should include methods for finding representative scenario sets.

Another area of improvement involves looking at ways to increase the resolution of the model. In this work, uncertainty was simplified by using daily average values for each of the uncertain parameters, meaning the model was limited to capture variations from one day to the next. A possible improvement is to incorporate hourly resolutions to more accurately model the volatile behavior of the major uncertainties associated with a renewable energy system.

Bibliography

[1] R. West and B. Fattouh. *The Energy Transition and Oil Companies' hard choices*. Tech. rep. The Oxford Instiute for Energy Studies, 2019.

[2] IPCC. *Global Warming of 1.5C. An IPCC Special Report on the impacts of global warming of 1.5C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty*. Tech. rep. Intergovernmental Panel on Climate Change, 2018.

[3] Rina Zeller, Agustin Delgado, Gerard Reid, and Kirsten Panerali. "Wind and solar PV will keep taking the lead". In: *Global Future Council on Energy Technologies* August 202 (2020), p. 5.

[4] IRENA. *Renewable Power Generation Costs in 2019*. Tech. rep. International Renewable Energy Agency, 2020.

[5] Omar J. Guerra, Jiazi Zhang, Joshua Eichman, Paul Denholm, Jennifer Kurtz, and Bri-Mathias Hodge. "The value of seasonal energy storage technologies for the integration of wind and solar power". In: *Energy Environ. Sci.* 13 (7 2020), pp. 1909–1922. DOI: 10.1039/D0EE00771D.

[6] Paolo Gabrielli, Florian Fürer, Georgios Mavromatidis, and Marco Mazzotti. "Robust and optimal design of multi-energy systems with seasonal storage through uncertainty analysis". In: *Applied Energy* 238 (Mar. 2019), pp. 1192–1210. DOI: 10.1016/j.apenergy.2019.01.064.

[7] Georgios Mavromatidis, Kristina Orehounig, and Jan Carmeliet. "Design of distributed energy systems under uncertainty: A two-stage stochastic programming approach". In: *Applied Energy* 222 (2018), pp. 932–950. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2018.04.019. URL: http://www.sciencedirect.com/science/article/pii/S0306261918305580.

[8] R. Kannan. “Algorithms, analysis and software for the global optimization of two-stage stochastic programs”. PhD thesis. Massachusetts Institute of Technology, 2018.

[9] Stephen M Robinson. *Numerical Optimization*. 2006. ISBN: 9780387303031. DOI: 10.1007/978-0-387-40065-5.

[10] Paul I. Barton. “Mixed-Integer and Nonconvex Optimization”. In: *Notes* (2007).

[11] NTNU. URL: https://www.itk.ntnu.no/_media/emner/fordypning/ttk16/introductiontomip2015.pdf.

[12] Eduardo Camponogara. 2016. URL: https://www.itk.ntnu.no/_media/emner/fordypning/l17-minlp-intro.pd.

[13] P. I. Barton R. Kannan. “GOSSIP documentation”. In: (2018).

[14] IRENA. *Wind power spatial planning techniques*. Tech. rep. IRENA, 2014.

[15] Y. Chen, X. Li, T. A. Adams II, and P. I. Barton. “Decomposition strategy for the global optimization of flexible energy polygeneration systems”. In: *AIChe* 58 (2012), pp. 3080–3095. DOI: https://doi.org/10.1002/aic.13708.

[16] Oersted. *Our offshore wind capabilitites*. Oct. 2020.

[17] NREL. Tech. rep. National Renewable Energy Laboratory, 2018. URL: https://www.nrel.gov/docs/fy19osti/72399.pdf.

Appendix A

Calculations

Expected values

Expectation values are calculated from equation A.1 where x_n is the value of interest in scenario n and ρ_{x_n} is the associated probability of the occurrence of value x_n.

$$E(x) = \sum_{n}^{N} x_n \cdot \rho_{x_n} \tag{A.1}$$

LCOE - Net present value

The LCOE is calculated as the net present value (NPV) of the project divided by MWh produced in project lifetime. The general equation for calculation of NPV is formulated in Equation A.2.

$$NPV = \sum_{n}^{N} \frac{CF_n}{(1+r_i)^n} - I_0 \quad \forall \text{ n} \in \{1.....\text{N}\} \tag{A.2}$$

n denotes year in project lifetime, CF_n cash flow in year n, r_i the discount rate and I_0 the initial investment in year zero. The formula for the LCOE is formulated in Equation A.3,

$$LCOE = \frac{NPV}{n_{MWh}} \tag{A.3}$$

where n_{MWh} is number of MWh produced from the renewable sources.

Security of clean energy supply

The general expression for calculation of security of clean energy supply (SCES) in Section 5.1, 5.2 and 5.3 is formulated in Equation A.4.

$$SCES = \sum_{s}^{S} \frac{f_s^{export}}{f_s^{demand}} \cdot \rho_s \tag{A.4}$$

f_s^{export} and f_s^{demand} are the previously defined flows, each denoted by scenario s, and ρ_s is probability of scenario s. The subscript t for time interval is omitted here for the purpose of generating a general expression for all case studies.

Appendix B

Scenarios for case study 1 and 2

B.1 Scenarios case study 1 and 2

The generated scenarios with associated probabilities are listed in Table B.1.

Table B.1: Scenarios used in case study 1 and 2.

Scenario	Probability	Solar intensity	Wind speed	Grid import price	Demand
-	-	MW/m^2	m/s	$/MWh	MWh/day
1	0,000634	0,00012	6	12	80
2	0,002726	0,00015	6	12	80
3	0,000634	0,00018	6	12	80
4	0,002726	0,00012	7,5	12	80
5	0,011732	0,00015	7,5	12	80
6	0,002726	0,00018	7,5	12	80
7	0,000634	0,00012	9	12	80
8	0,002726	0,00015	9	12	80
9	0,000634	0,00018	9	12	80
10	0,002726	0,00012	6	15	80
11	0,011732	0,00015	6	15	80
12	0,002726	0,00018	6	15	80
13	0,011732	0,00012	7,5	15	80
14	0,050481	0,00015	7,5	15	80
15	0,011732	0,00018	7,5	15	80
16	0,002726	0,00012	9	15	80
17	0,011732	0,00015	9	15	80
18	0,002726	0,00018	9	15	80
19	0,000634	0,00012	6	18	80

20	0,002726	0,00015	6	18	80
21	0,000634	0,00018	6	18	80
22	0,002726	0,00012	7,5	18	80
23	0,011732	0,00015	7,5	18	80
24	0,002726	0,00018	7,5	18	80
25	0,000634	0,00012	9	18	80
26	0,002726	0,00015	9	18	80
27	0,000634	0,00018	9	18	80
28	0,002726	0,00012	6	12	100
29	0,011732	0,00015	6	12	100
30	0,002726	0,00018	6	12	100
31	0,011732	0,00012	7,5	12	100
32	0,050481	0,00015	7,5	12	100
33	0,011732	0,00018	7,5	12	100
34	0,002726	0,00012	9	12	100
35	0,011732	0,00015	9	12	100
36	0,002726	0,00018	9	12	100
37	0,011732	0,00012	6	15	100
38	0,050481	0,00015	6	15	100
39	0,011732	0,00018	6	15	100
40	0,050481	0,00012	7,5	15	100
41	0,217217	0,00015	7,5	15	100
42	0,050481	0,00018	7,5	15	100
43	0,011732	0,00012	9	15	100
44	0,050481	0,00015	9	15	100
45	0,011732	0,00018	9	15	100
46	0,002726	0,00012	6	18	100
47	0,011732	0,00015	6	18	100
48	0,002726	0,00018	6	18	100
49	0,011732	0,00012	7,5	18	100
50	0,050481	0,00015	7,5	18	100
51	0,011732	0,00018	7,5	18	100
52	0,002726	0,00012	9	18	100
53	0,011732	0,00015	9	18	100
54	0,002726	0,00018	9	18	100
55	0,000634	0,00012	6	12	120
56	0,002726	0,00015	6	12	120
57	0,000634	0,00018	6	12	120
58	0,002726	0,00012	7,5	12	120
59	0,011732	0,00015	7,5	12	120
60	0,002726	0,00018	7,5	12	120
61	0,000634	0,00012	9	12	120
62	0,002726	0,00015	9	12	120
63	0,000634	0,00018	9	12	120
64	0,002726	0,00012	6	15	120

65	0,011732	0,00015	6	15	120
66	0,002726	0,00018	6	15	120
67	0,011732	0,00012	7,5	15	120
68	0,050481	0,00015	7,5	15	120
69	0,011732	0,00018	7,5	15	120
70	0,002726	0,00012	9	15	120
71	0,011732	0,00015	9	15	120
72	0,002726	0,00018	9	15	120
73	0,000634	0,00012	6	18	120
74	0,002726	0,00015	6	18	120
75	0,000634	0,00018	6	18	120
76	0,002726	0,00012	7,5	18	120
77	0,011732	0,00015	7,5	18	120
78	0,002726	0,00018	7,5	18	120
79	0,000634	0,00012	9	18	120
80	0,002726	0,00015	9	18	120
81	0,000634	0,00018	9	18	120

Appendix C

Scenario set for case study 3

	Scenario	1	2	3	4	5	6	7	8	9	10
0	Prob	0.056	0.240	0.000	0.240	0.056	0.056	0.240	0.001	0.056	0.056
$I \cdot 10^3$	t1	0.12	0.18	0.15	0.15	0.18	0.12	0.18	0.18	0.15	0.12
$I \cdot 10^3$	t2	0.12	0.12	0.15	0.15	0.18	0.15	0.15	0.12	0.18	0.15
$I \cdot 10^3$	t3	0.12	0.18	0.18	0.12	0.18	0.12	0.18	0.18	0.12	0.15
$I \cdot 10^3$	t4	0.15	0.18	0.12	0.18	0.15	0.18	0.15	0.12	0.12	0.18
$I \cdot 10^3$	t5	0.18	0.18	0.18	0.12	0.15	0.12	0.12	0.12	0.12	0.18
$I \cdot 10^3$	t6	0.15	0.12	0.12	0.15	0.12	0.15	0.12	0.12	0.15	0.18
$I \cdot 10^3$	t7	0.15	0.18	0.18	0.15	0.18	0.12	0.15	0.15	0.12	0.18
W	t1	6	7.5	9	7.5	6	6	9	6	7.5	9
W	t2	7.5	7.5	9	6	6	7.5	7.5	6	9	7.5
W	t3	6	9	9	9	7.5	7.5	7.5	9	9	6
W	t4	6	7.5	9	7.5	7.5	9	6	9	9	7.5
W	t5	9	9	7.5	9	7.5	9	7.5	9	6	9
W	t6	7.5	9	7.5	6	7.5	7.5	9	6	9	7.5
W	t7	7.5	7.5	7.5	6	6	7.5	6	6	7.5	6
OC_grid	t1	12	18	18	12	18	18	15	18	15	15
OC_grid	t2	18	15	18	18	18	12	15	18	12	15
OC_grid	t3	15	12	18	12	15	15	18	18	12	15
OC_grid	t4	18	18	18	18	12	12	12	15	15	15
OC_grid	t5	12	15	12	15	15	18	12	18	12	18
OC_grid	t6	15	15	15	15	12	15	15	15	12	18
OC_grid	t7	15	18	18	15	12	18	12	15	18	12
f_dem	t1	100	100	80	100	100	120	120	120	100	120
f_dem	t2	120	120	120	80	100	100	100	120	100	80
f_dem	t3	120	120	120	100	120	120	80	80	100	120
f_dem	t4	80	100	120	120	120	100	80	100	120	80
f_dem	t5	120	100	80	80	120	120	100	100	120	80
f_dem	t6	80	120	120	120	80	80	80	80	80	120
f_dem	t7	80	100	80	120	120	120	120	100	100	100
f_dem	t6	80	120	120	120	80	80	80	80	80	120
f_dem	t7	80	100	80	120	120	120	120	100	100	100

Appendix D

C++ code

D.1 Expected value problem

```
#include <iostream>
#include <string>
#include <vector>
#include "definitions.hpp"
#include "CompGraph.hpp"
#include "GenerateScenarios.hpp"

#include "inputmodel.hpp"

using namespace std;

int inputmodel(vector<double> &weights)
{
  cout<<"This is the deterministic PV_wind_model: "<<endl;


// Defining input parameters
  int techTypes=2;
  //Nominal power model params
  double Inom=0.00015; //[MW/M ] solar radiance intensity
  double Wnom=7.5; //[m/s] wind speed

  //Wind power model params
  double Wmin=3.5; // [m/s]
  double Wmax=25;
  double Wd=13;
  double qd=8; //[MW] for one wind turbine

  //Efficiencies
  double etaPV=0.15;//[-]
  double etaWT=0.85;

  //Grid parameters
  double FiT=1.45; //feed-in tariff [$/MWh]
```

```
35  double OCgridNom=15; //price imported el from grid [$/MWh]
36  double demandNom=100; //MWh/day
37
38  //Technology cost parameteres ={PV,WT}
39  vector<double> C0={27000,49840000}; //$
40  vector<int> Smax={1200000,15}; //Area PV and number of wind turbines
      ;
41  vector<int> S0={180,1}; //Reference value for cost function
42  vector<double> xi={0.05,0.05}; //Maintenance cost (factor)
43  vector<double> sfi={0.7,0.7}; //Cost scaling factor
44
45  bool automaticallyGenerateScenarios=true;
46
47  int NumScen=0; //number of scenarios
48  int NumUncert=4; //solar intensity, wind speed, cost of el-import,
      demand
49  int NumDays=1; //number of days
50  int NumDiscrete=15;  //number of discrete intervals
51
52  //Convert binary 1st stage decision vars to discrete 2nd stage vars
53  vector<vector<double>> S(techTypes,vector<double>(NumDiscrete)); //[
      m  ] and [-] units of wind turbines
54  cout<<"The discrete size values are:"<<'\n';
55  for(int i=0;i<techTypes;++i)
56  {
57    cout<<"Technology "<<i+1<<'\n';
58    for(int n=0;n<NumDiscrete;++n)
59    {
60      S[i][n]=(n*Smax[i])/(NumDiscrete-1); //S_lower=0 and ommited
      here
61      cout<<S[i][n] << '\n'; //printing to check values
62    }
63  }
64  //Calculate capital cost for given discrete size value
65  vector<vector<double>> C(techTypes,vector<double>(NumDiscrete));//[$
      ]
66  cout<<"The corresponding capital costs are:"<<'\n';
67  for(int i=0;i<techTypes;++i)
68  {
69    for(int n=0;n<NumDiscrete;++n)
70    {
71      C[i][n]=(1+xi[i])*C0[i]*pow(S[i][n]/S0[i],sfi[i]);
72      cout<<C[i][n]<<'\n';//printing to check values
73    }
74  }
75
76  vector<vector<vector<double>>> StochParams(NumUncert);//Vector to
      contain uncertain vars realizations
77
78  if (automaticallyGenerateScenarios)
79  {
80    int numUncertainParams=NumUncert;
81    vector<double> UncertainParamsMean(numUncertainParams);
82    vector<double> UncertainParamsDev(numUncertainParams);
83    vector<int> numParamRealisations(numUncertainParams);
84
85    for(int d=0;d<NumDays;++d)
```

```
86      {
87
88        UncertainParamsMean[0]=Inom; //solar intensity
89        UncertainParamsMean[1]=Wnom; //wind speed
90        UncertainParamsMean[2]=OCgridNom;  //import price
91        UncertainParamsMean[3]=demandNom; //demand
92
93        UncertainParamsDev[0]=0;
94        UncertainParamsDev[1]=0;
95        UncertainParamsDev[2]=0;
96        UncertainParamsDev[3]=0;
97
98        numParamRealisations[0]=1;
99        numParamRealisations[1]=1;
100       numParamRealisations[2]=1;
101       numParamRealisations[3]=1;
102
103       vector<double> uncertainParamRealizations;
104
105       NumScen=decomposition::generateScenarios(decomposition::NORMAL,
106                             numUncertainParams,
107                             numParamRealisations,
108                             UncertainParamsMean,
109                             UncertainParamsDev,
110                             weights,
111                             uncertainParamRealizations);
112       for(int i=0;i<NumUncert;++i)
113       {
114         StochParams[i].resize(NumDays);
115         for(int d=0; d<NumDays;++d)
116         {
117           StochParams[i][d].resize(NumScen);
118         }
119       }
120
121       for(int d=0;d<NumDays;++d)
122       {
123         for(int s=0; s<NumScen;++s)
124         {
125           StochParams[0][d][s]=uncertainParamRealizations[s]; //solar
    intensity [-]
126           StochParams[1][d][s]=uncertainParamRealizations[NumScen+s];
    //wind speed [m/s]
127           StochParams[2][d][s]=uncertainParamRealizations[NumScen*2+s
    ]; //cost EL-import [$/MWh]
128           StochParams[3][d][s]=uncertainParamRealizations[NumScen*3+s
    ]; //demand [MWh/day]
129           cout<<StochParams[0][d][s]<<" -------"<<StochParams[1][d][s
    ]<<" -------"<<StochParams[2][d][s]<<" -------"<<StochParams[3][d
    ][s]<<'\n';
130         }
131       }
132     }
133   }
134   else
135   {
136     //generate by hand?
```

```
137   }
138   //1st stage variables
139   vector<vector<Variables>> z(techTypes,vector<Variables>(NumDiscrete)
        ); //Binary variable,
140                                       //on/off for size interval z to
        capacity size S and capcost C
141   int varcount = -1;
142   int concount=-1;
143   char clabel[30];
144
145   for(int i=0;i<techTypes;++i)
146   {
147     for(int n=0;n<NumDiscrete;++n)
148
149     {
150       sprintf(clabel,"z[%d][%d]", i+1,n+1);
151           z[i][n].setIndependentVariable( ++varcount,
152                 compgraph::BINARY,
153                 I(0,1),
154                 0.,
155                 -1,  //first stage variable, does not belong to
        specific scenario
156                 clabel);
157         }
158   }
159
160   //2nd stage variables
161   //Set export and import of electricity, 2nd stage variables
162   vector<vector<Variables>> f_import(NumDays,vector<Variables>(NumScen
        )); //[MWh/day]
163   vector<vector<Variables>> fPV_export(NumDays,vector<Variables>(
        NumScen)); //[MWh/day]
164   vector<vector<Variables>> fWT_export(NumDays,vector<Variables>(
        NumScen));//[MWh/day]
165
166   for(int d=0;d<NumDays;++d)
167   {
168     for(int s=0;s<NumScen;++s)
169     {
170       sprintf(clabel,"f_import[%d][%d]", d+1, s+1);
171       f_import[d][s].setIndependentVariable(++varcount,compgraph::
        CONTINUOUS,I(0,1000),0.,s+1,clabel);
172       sprintf(clabel,"fPV_export[%d][%d]", d+1, s+1);
173       fPV_export[d][s].setIndependentVariable(++varcount,compgraph::
        CONTINUOUS,I(0,1000),0.,s+1,clabel);
174       sprintf(clabel,"fWT_export[%d][%d]", d+1, s+1);
175       fWT_export[d][s].setIndependentVariable(++varcount,compgraph::
        CONTINUOUS,I(0,1000),0.,s+1,clabel);
176     }
177   }
178   //Constraint involving 1st stage variables, make sure only one size
        intercal zi is chosen per technology
179   vector<Constraints> zLim(techTypes);
180
181   for(int i=0;i<techTypes;++i)
182   {
183     zLim[i]=-1;
```

```
184     for(int n=0;n<NumDiscrete;++n)
185     {
186       zLim[i]+=z[i][n];
187     }
188     zLim[i].setDependentVariable(++concount,compgraph::EQUALITY,false
        ,-1);
189   }
190
191   //Adding throughoutput capacity constraints
192   vector<vector<Constraints>> PVprod(NumDays,vector<Constraints>(
        NumScen)); //[MWh/day]
193   vector<vector<Constraints>> WTprod(NumDays,vector<Constraints>(
        NumScen));//[MWh/day]
194   //Demand constraint, make sure imported el <= demand
195   vector<vector<Constraints>> demand(NumDays,vector<Constraints>(
        NumScen));//[MWh/day]
196
197   for(int d=0;d<NumDays;++d)
198   {
199     for(int s=0;s<NumScen;++s)
200     {
201       PVprod[d][s]=0;
202       WTprod[d][s]=0;
203       for(int n=0;n<NumDiscrete;++n)
204       {
205         PVprod[d][s]+=etaPV*StochParams[0][d][s]*S[0][n]*z[0][n]*5;
206         if( StochParams[1][d][s]>Wmin && StochParams[1][d][s]<Wd)
207         {
208           WTprod[d][s]+=etaWT*qd*((pow(StochParams[1][d][s],3)-pow(
        Wmin,3))/(pow(Wd,3))-pow(Wmin,3))*S[1][n]*z[1][n]*24;
209         }
210         else if(StochParams[1][d][s]>=Wd && StochParams[1][d][s]<=Wmax
        )
211         {
212           WTprod[d][s]+=qd*etaWT*S[1][n]*z[1][n]*24;
213         }
214         else //if(StochParams[1][d][s]<Wmin || StochParams[1][d][s]>
        Wmax)
215         {
216           WTprod[d][s]+=0;
217         }
218       }
219       WTprod[d][s]-=fWT_export[d][s];
220       WTprod[d][s].setDependentVariable(++concount,compgraph::EQUALITY
        ,true,s+1);
221       PVprod[d][s]-=fPV_export[d][s];
222       PVprod[d][s].setDependentVariable(++concount,compgraph::EQUALITY
        ,true,s+1);
223
224       demand[d][s]=StochParams[3][d][s]-fPV_export[d][s]-fWT_export[d
        ][s]-f_import[d][s];
225       demand[d][s].setDependentVariable(++concount,compgraph::LEQ,true
        ,s+1);
226     }
227   }
228
229   //Declaring objective function, one for each scenario s
```

```
230   vector<Objective> obj(NumScen);//[$]
231
232   for(int s=0;s<NumScen;++s)
233   {
234     obj[s]=0;
235     for(int i=0;i<techTypes;++i)
236     { //Adding CAPEX for technologi i
237       for(int n=0;n<NumDiscrete;++n)
238       {
239         obj[s]+=C[i][n]*z[i][n]*(1+xi[i]); //[$]
240       }
241
242       for(int d=0;d<NumDays;++d)
243       {
244         obj[s]+=f_import[d][s]*StochParams[2][d][s]*365*30; //[$]
245         obj[s]-=(fPV_export[d][s]+fWT_export[d][s])*FiT*365*30; //[$]
246       }
247     }
248     obj[s].setDependentVariable(++concount,compgraph::OBJ,true,s+1);
249   }
250   return NumScen;
251
252 }
```

D.2 Case study 1: Simple model of uncertainty with 1 design day

```
#include <iostream>
#include <string>
#include <vector>
#include "definitions.hpp"
#include "CompGraph.hpp"
#include "GenerateScenarios.hpp"

#include "inputmodel.hpp"

using namespace std;

int inputmodel(vector<double> &weights)
{
  cout<<"This is the stochastic PV_wind_model without design days,
    just random variation around expected values: "<<endl;

  bool automaticallyGenerateScenarios=true;
  bool findEEVP = false;

// Defining input parameters
  int techTypes=2;
  //Nominal power model params
  double Inom=0.00015; //[MW/M ] solar radiance intensity
  double Wnom=7.5; //[m/s] wind speed

  //Wind power model params
  double Wmin=3.5; // [m/s]
  double Wmax=25;
  double Wd=13;
  double qd=8; //[MW] for one wind turbine

  //Efficiencies
  double etaPV=0.15;//[-]
  double etaWT=0.85;

  //Grid parameters
  double FiT=1.45; //feed-in tariff [$/MWh]
  double OCgridNom=15; //price imported el from grid [$/MWh]
  double demandNom=100; //MWh/day

  //Technology cost parameteres ={PV,WT}
  vector<double> C0={27000,49840000}; //$
  vector<int> Smax={1200000,14}; //Area PV and number of wind turbines
    ;
  vector<int> S0={180,1}; //Reference value for cost function
  vector<double> xi={0.05,0.05}; //Maintenance cost (factor)
  vector<double> sfi={0.7,0.7}; //Cost scaling factor

  int NumScen=0; //number of scenarios
  int NumUncert=4; //solar intensity, wind speed, cost of el-import,
    demand
  int NumDays=1; //number of days
  int NumDiscrete=15;  //number of discrete intervals
```

```
51
52  //Convert binary 1st stage decision vars to discrete 2nd stage vars
53  vector<vector<double>> S(techTypes,vector<double>(NumDiscrete)); //[
      m ] and [-] units of wind turbines
54  cout<<"The discrete size values are:"<<'\n';
55  for(int i=0;i<techTypes;++i)
56  {
57    cout<<"Technology "<<i+1<<'\n';
58    for(int n=0;n<NumDiscrete;++n)
59    {
60      S[i][n]=(n*Smax[i])/(NumDiscrete-1); //S_lower=0 and ommited
        here
61      cout<<S[i][n] << '\n'; //printing to check values
62    }
63  }
64  //Calculate capital cost for given discrete size value
65  vector<vector<double>> C(techTypes,vector<double>(NumDiscrete));//[$
      ]
66  for(int i=0;i<techTypes;++i)
67  { cout<<"Cost technology " << i <<'\n';
68    for(int n=0;n<NumDiscrete;++n)
69    {
70      C[i][n]=(1+xi[i])*C0[i]*pow(S[i][n]/S0[i],sfi[i]);
71      cout<<C[i][n]<<'\n';//printing to check values
72    }
73  }
74  vector<vector<vector<double>>> StochParams(NumUncert);//Vector to
      contain uncertain vars realizations
75
76  if (automaticallyGenerateScenarios)
77  {
78    int numUncertainParams=NumUncert;
79    vector<double> UncertainParamsMean(numUncertainParams);
80    vector<double> UncertainParamsDev(numUncertainParams);
81    vector<int> numParamRealisations(numUncertainParams);
82
83    UncertainParamsMean[0]=Inom; //solar intensity
84    UncertainParamsMean[1]=Wnom; //wind speed
85    UncertainParamsMean[2]=OCgridNom;  //import price
86    UncertainParamsMean[3]=demandNom; //demand
87
88    UncertainParamsDev[0]=0.15*Inom;
89    UncertainParamsDev[1]=0.15*Wnom;
90    UncertainParamsDev[2]=0.15*OCgridNom;
91    UncertainParamsDev[3]=0.15*demandNom;
92
93    numParamRealisations[0]=6;
94    numParamRealisations[1]=6;
95    numParamRealisations[2]=6;
96    numParamRealisations[3]=6;
97
98
99    vector<double> uncertainParamRealisations;
100
101   NumScen=decomposition::generateScenarios(decomposition::NORMAL,
102                         numUncertainParams,
103                         numParamRealisations,
```

```
104                             UncertainParamsMean,
105                             UncertainParamsDev,
106                             weights,
107                             uncertainParamRealisations);
108
109       for(int n=0;n<(NumUncert);++n)
110       {
111         StochParams[n].resize(NumDays);
112         for(int d=0; d<NumDays;++d)
113         {
114           StochParams[n][d].resize(NumScen);
115         }
116       }
117       ofstream outfile("Scenarios.txt");
118       outfile << "Scenario;Probability;solarIntensity;WindSpeed;
    GridPrice;Demand \n";
119       for(int d=0;d<NumDays;++d)
120       {
121         cout <<"Day: "<< d<<endl;
122         for(int s=0; s<NumScen;++s)
123         {
124           StochParams[0][d][s]=uncertainParamRealisations[s]; //solar
    intensity [-]
125           StochParams[1][d][s]=uncertainParamRealisations[NumScen+s];
    //wind speed [m/s]
126           StochParams[2][d][s]=uncertainParamRealisations[NumScen*2+s
    ]; //grid import price
127           StochParams[3][d][s]=uncertainParamRealisations[NumScen*3+s
    ]; //demand
128           cout<<StochParams[0][d][s]<<" -------"<<StochParams[1][d][s
    ]<<" -------"<<StochParams[2][d][s]<<" -------"<<StochParams[3][d
    ][s]<<'\n';
129
130           outfile << s+1<<";"<<weights[s]<<";"<<StochParams[0][d][s]<<
    ";"<<StochParams[1][d][s]<<";"<<StochParams[2][d][s]<< ";"<<
    StochParams[3][d][s]<<'\n';
131         }
132       }
133       outfile.close();
134     }
135
136   else
137   {
138     //generate by hand?
139   }
140   //1st stage variables
141   vector<vector<Variables>> z(techTypes,vector<Variables>(NumDiscrete)
    ); //Binary variable,
142                                       //on/off for size interval z to
    capacity size S and capcost C
143   int varcount = -1;
144   int concount=-1;
145   char clabel[30];
146
147   for(int i=0;i<techTypes;++i)
148   {
149     for(int n=0;n<NumDiscrete;++n)
```

```
150
151     {
152       sprintf(clabel,"z[%d][%d]", i+1,n+1);
153           z[i][n].setIndependentVariable( ++varcount,
154                 compgraph::BINARY,
155                 I(0,1),
156                 0.,
157                 -1,  //first stage variable, does not belong to
    specific scenario
158                 clabel);
159         }
160   }
161
162   //2nd stage variables
163   //Set export and import of electricity, 2nd stage variables
164   vector<vector<Variables>> f_import(NumDays,vector<Variables>(NumScen
      )); //[MWh/day]
165   vector<vector<Variables>> f_PVexport(NumDays,vector<Variables>(
      NumScen)); //[MWh/day]
166   vector<vector<Variables>> f_WTexport(NumDays,vector<Variables>(
      NumScen));//[MWh/day]
167
168   for(int d=0;d<NumDays;++d)
169   {
170     for(int s=0;s<NumScen;++s)
171     {
172       sprintf(clabel,"f_import[%d][%d]", d+1, s+1);
173       f_import[d][s].setIndependentVariable(++varcount,compgraph::
      CONTINUOUS,I(0,1000),0.,s+1,clabel);
174       sprintf(clabel,"f_PVexport[%d][%d]", d+1, s+1);
175       f_PVexport[d][s].setIndependentVariable(++varcount,compgraph::
      CONTINUOUS,I(0,1000),0.,s+1,clabel);
176       sprintf(clabel,"f_WTexport[%d][%d]", d+1, s+1);
177       f_WTexport[d][s].setIndependentVariable(++varcount,compgraph::
      CONTINUOUS,I(0,1000),0.,s+1,clabel);
178     }
179   }
180   //Constraint involving 1st stage variables, make sure only one size
      intercal zi is chosen per technology
181   vector<Constraints> zLim(techTypes);
182
183   for(int i=0;i<techTypes;++i)
184   {
185     zLim[i]=-1;
186     for(int n=0;n<NumDiscrete;++n)
187     {
188       zLim[i]+=z[i][n];
189     }
190     zLim[i].setDependentVariable(++concount,compgraph::EQUALITY,false
      ,-1);
191   }
192
193   //Adding throughoutput capacity constraints
194   vector<vector<Constraints>> PVprod(NumDays,vector<Constraints>(
      NumScen)); //[MWh/day]
195   vector<vector<Constraints>> WTprod(NumDays,vector<Constraints>(
      NumScen));//[MWh/day]
```

```
196 //Demand constraint, make sure imported el <= demand
197 vector<vector<Constraints>> demand(NumDays,vector<Constraints>(
      NumScen));//[MWh/day]
198
199 for(int d=0;d<NumDays;++d)
200 {
201   for(int s=0;s<NumScen;++s)
202   {
203     PVprod[d][s]=0;
204     WTprod[d][s]=0;
205     for(int n=0;n<NumDiscrete;++n)
206     {
207       PVprod[d][s]+=etaPV*StochParams[0][d][s]*S[0][n]*z[0][n]*5; //
      approx 5 hours of sun each day
208       if( StochParams[1][d][s]>Wmin && StochParams[1][d][s]<Wd)
209       {
210         WTprod[d][s]+=etaWT*qd*((pow(StochParams[1][d][s],3)-pow(
      Wmin,3))/(pow(Wd,3))-pow(Wmin,3))*S[1][n]*z[1][n]*24; //24 hours
      of wind each day
211       }
212       else if(StochParams[1][d][s]>=Wd && StochParams[1][d][s]<=Wmax
      )
213       {
214         WTprod[d][s]+=qd*etaWT*S[1][n]*z[1][n]*24; //24 hours of
      wind each day
215       }
216       else //if(StochParams[1][d][s]<Wmin || StochParams[1][d][s]>
      Wmax)
217       {
218         WTprod[d][s]+=0;
219       }
220     }
221     WTprod[d][s]-=f_WTexport[d][s];
222     WTprod[d][s].setDependentVariable(++concount,compgraph::EQUALITY
      ,true,s+1);
223     PVprod[d][s]-=f_PVexport[d][s];
224     PVprod[d][s].setDependentVariable(++concount,compgraph::EQUALITY
      ,true,s+1);
225
226     demand[d][s]=StochParams[3][d][s]-f_PVexport[d][s]-f_WTexport[d
      ][s]-f_import[d][s];
227     demand[d][s].setDependentVariable(++concount,compgraph::LEQ,true
      ,s+1);
228   }
229 }
230 //
      ----------------------------------------------------------------------------------------

231 //EEVP
232
233     if (findEEVP){
234        vector<Constraints>xEVP(techTypes);
235       xEVP[0]= S[0][9];  xEVP[1]= 0;
236       for (int i=0;i<techTypes;i++){
237         for (int j=0;j<NumDiscrete;j++){
238            xEVP[i] += z[i][j]*S[i][j];
239          }
```

```
240             xEVP[i].setDependentVariable(++concount,compgraph::EQUALITY
    ,false,-1);
241
242         }
243       }
244    //
    ----------------------------------------------------------------------------------------

245   //Declaring objective function, one for each scenario s
246   vector<Objective> obj(NumScen);//[$]
247
248   for(int s=0;s<NumScen;++s)
249   {
250     obj[s]=0;
251     for(int i=0;i<techTypes;++i)
252     { //Adding CAPEX for technologi i
253       for(int n=0;n<NumDiscrete;++n)
254       {
255         obj[s]+=C[i][n]*z[i][n]*(1+xi[i]); //[$]
256       }
257       //Adding opex pre dy
258       for(int d=0;d<NumDays;++d)
259       {
260         obj[s]+=f_import[d][s]*StochParams[2][d][s]*365*30; //[$]
261         obj[s]-=(f_PVexport[d][s]+f_WTexport[d][s])*FiT*365*30; //[$]
262       }
263     }
264     obj[s].setDependentVariable(++concount,compgraph::OBJ,true,s+1);
265   }
266   return NumScen;
267 }
```

D.3 Case study 2: Accounting for seasonal variability using 4 design days

```
#include <iostream>
#include <string>
#include <vector>
#include "definitions.hpp"
#include "CompGraph.hpp"
#include "GenerateScenarios.hpp"

#include "inputmodel.hpp"

using namespace std;

int inputmodel(vector<double> &weights)
{
  cout<<"This is the stochastic PV_wind_model with Design Days
    implemented: "<<endl;
  int techTypes=2;
  //Set to true if you want seasonal weather changes for wind speed
    and solar intensity
  bool seasonalVariability=true;
  bool automaticallyGenerateScenarios=true;
  bool findEEVP = false;

// Defining input parameters
  //Nominal power model params
  double Inom=0.00015; //MW/m   solar radiance intensity
  double Wnom=7.5; //[m/s] wind speed

  //Wind power model params
  double Wmin=3.5; // [m/s]
  double Wmax=25; // [m/s]
  double Wd=13; // [m/s]
  double qd=8; //[MW] for one wind turbine

  //Seasonal variability params
  double minSun=0; double medSun=0; double maxSun=0; double minWind=0;
     double medWind=0; double maxWind=0;

  //Efficiencies
  double etaPV=0.15;//[-]
  double etaWT=0.85;

  //Grid parameters
  double FiT=1.5; //feed-in tariff [$/MWh]
  double OCgridNom=15;//price imported el from grid [$/MWh]
  double demandNom=100; //MWh/day

  //Technology cost parameteres ={PV,WT}
  vector<double> C0={27000,49840000}; //$
  vector<int> Smax={1200000,14}; //Area PV and number of wind turbines
    ;
  vector<int> S0={180,1}; //Reference value for cost function
  vector<double> xi={0.05,0.05}; //Maintenance cost (factor)
  vector<double> sfi={0.7,0.7}; //Cost scaling factor
```

```
50
51  int NumScen=0; //number of scenarios
52  int NumUncert=4; //solar intensity, wind speed, cost of el-import,
      demand
53  int NumDesignDays=1; //number of design days, one for each season:
      spring, summer, fall, winter
54  int designDayCount=90; //Number of days for each design day
55  int NumDiscrete=15;  //number of discrete intervals
56
57  if(seasonalVariability)
58  {
59    NumDesignDays=4;
60
61    minSun     = -0.2*Inom;
62    medSun     = 0*Inom;
63    maxSun     = 0.2*Inom;
64    minWind    = -0.2*Wnom;
65    medWind    = 0*Wnom;
66    maxWind    = 0.3*Wnom;
67  }
68
69  struct season
70  {
71    double sun;
72    double wind;
73    int count;
74  };
75
76  struct season spring={medSun,maxWind,designDayCount}; //SPRING -
      medium sun, much wind
77  struct season summer={maxSun,minWind,designDayCount};//SUMMER - much
       sun, medium wind
78  struct season fall={medSun,medWind,designDayCount}; //FALL - medium
      sun, much wind
79  struct season winter={minSun,medWind,designDayCount};//WINTER -
      little sun, medium wind
80
81  vector<struct season> AllSeasons={winter,spring,summer,fall};
82
83  //Convert binary 1st stage decision vars to discrete 2nd stage vars
84  vector<vector<double>> S(techTypes,vector<double>(NumDiscrete)); //[
      m  ] and [-] units of wind turbines
85  cout<<"The discrete size values are:"<<'\n';
86  for(int i=0;i<techTypes;++i)
87  {
88    cout<<"Technology "<<i+1<<'\n';
89    for(int n=0;n<NumDiscrete;++n)
90    {
91      S[i][n]=(n*Smax[i])/(NumDiscrete-1); //S_lower=0 and ommited
      here
92      cout<<S[i][n] << '\n'; //printing to check values
93    }
94  }
95  //Calculate capital cost for given discrete size value
96  vector<vector<double>> C(techTypes,vector<double>(NumDiscrete));//[$
      ]
97  for(int i=0;i<techTypes;++i)
```

```
98  { cout<<"Cost technology " << i <<'\n';
99    for(int n=0;n<NumDiscrete;++n)
100   {
101     C[i][n]=(1+xi[i])*C0[i]*pow(S[i][n]/S0[i],sfi[i]);//r*pow(1+r,
      LifeTime)/(pow(1+r,LifeTime)-1);
102     cout<<C[i][n]<<'\n';//printing to check values
103   }
104 }
105
106 vector<vector<vector<double>>> StochParams(NumUncert);//Vector to
      contain uncertain vars realizations
107
108 if (automaticallyGenerateScenarios)
109 {
110   int numUncertainParams=NumUncert;
111   vector<double> UncertainParamsMean(numUncertainParams);
112   vector<double> UncertainParamsDev(numUncertainParams);
113   vector<int> numParamRealisations(numUncertainParams);
114
115   UncertainParamsMean[0]=Inom; //solar intensity
116   UncertainParamsMean[1]=Wnom; //wind speed
117   UncertainParamsMean[2]=OCgridNom;  //import price
118   UncertainParamsMean[3]=demandNom; //demand
119
120   UncertainParamsDev[0]=Inom*0.15;
121   UncertainParamsDev[1]=Wnom*0.15;
122   UncertainParamsDev[2]=OCgridNom*0.15;
123   UncertainParamsDev[3]=demandNom*0.15;
124
125   numParamRealisations[0]=3;
126   numParamRealisations[1]=3;
127   numParamRealisations[2]=3;
128   numParamRealisations[3]=3;
129
130
131   vector<double> uncertainParamRealisations;
132
133   NumScen=decomposition::generateScenarios(decomposition::NORMAL,
134                         numUncertainParams,
135                         numParamRealisations,
136                         UncertainParamsMean,
137                         UncertainParamsDev,
138                         weights,
139                         uncertainParamRealisations);
140
141     for(int n=0;n<(NumUncert);++n)
142     {
143       StochParams[n].resize(NumDesignDays);
144       for(int d=0; d<NumDesignDays;++d)
145       {
146         StochParams[n][d].resize(NumScen);
147       }
148     }
149     for(int d=0;d<NumDesignDays;++d)
150     {
151       cout <<"Design day: "<< d+1<<endl;
152       for(int s=0; s<NumScen;++s)
```

```
153         {
154           StochParams[0][d][s]=uncertainParamRealisations[s]+
      AllSeasons[d].sun; //solar intensity [-]
155           StochParams[1][d][s]=uncertainParamRealisations[NumScen+s]+
      AllSeasons[d].wind; //wind speed [m/s]
156           StochParams[2][d][s]=uncertainParamRealisations[NumScen*2+s
      ]; //grid import price
157           StochParams[3][d][s]=uncertainParamRealisations[NumScen*3+s
      ]; //demand
158           cout<<StochParams[0][d][s]<<" -------:"<<StochParams[1][d][s
      ]<<" -------:"<<StochParams[2][d][s]<<" -------:"<<StochParams[3][
      d][s]<<'\n';
159         }
160       }
161     }
162
163   else
164   {
165     //generate by hand?
166   }
167   //1st stage variables
168   vector<vector<Variables>> z(techTypes,vector<Variables>(NumDiscrete)
      ); //Binary variable,
169                                       //on/off for size interval z to
      capacity size S and capcost C
170   int varcount = -1;
171   int concount=-1;
172   char clabel[30];
173
174   for(int i=0;i<techTypes;++i)
175   {
176     for(int n=0;n<NumDiscrete;++n)
177
178     {
179       sprintf(clabel,"z[%d][%d]", i+1,n+1);
180           z[i][n].setIndependentVariable( ++varcount,
181                 compgraph::BINARY,
182                 I(0,1),
183                 0.,
184                 -1,  //first stage variable, does not belong to
      specific scenario
185                 clabel);
186         }
187   }
188
189   //2nd stage variables
190   //Set export and import of electricity, 2nd stage variables
191   vector<vector<Variables>> f_import(NumDesignDays,vector<Variables>(
      NumScen)); //[MWh/day]
192   vector<vector<Variables>> f_PVexport(NumDesignDays,vector<Variables
      >(NumScen)); //[MWh/day]
193   vector<vector<Variables>> f_WTexport(NumDesignDays,vector<Variables
      >(NumScen));//[MWh/day]
194
195   for(int d=0;d<NumDesignDays;++d)
196   {
197     for(int s=0;s<NumScen;++s)
```

```
198     {
199       sprintf(clabel,"f_import[%d][%d]", d+1, s+1);
200       f_import[d][s].setIndependentVariable(++varcount,compgraph::
      CONTINUOUS,I(0,1000),0.,s+1,clabel);
201       sprintf(clabel,"f_PVexport[%d][%d]", d+1, s+1);
202       f_PVexport[d][s].setIndependentVariable(++varcount,compgraph::
      CONTINUOUS,I(0,1000),0.,s+1,clabel);
203       sprintf(clabel,"f_WTexport[%d][%d]", d+1, s+1);
204       f_WTexport[d][s].setIndependentVariable(++varcount,compgraph::
      CONTINUOUS,I(0,1000),0.,s+1,clabel);
205     }
206   }
207   //Constraint involving 1st stage variables, make sure only one size
      intercal zi is chosen per technology
208   vector<Constraints> zLim(techTypes);
209
210   for(int i=0;i<techTypes;++i)
211   {
212     zLim[i]=-1;
213     for(int n=0;n<NumDiscrete;++n)
214     {
215       zLim[i]+=z[i][n];
216     }
217     zLim[i].setDependentVariable(++concount,compgraph::EQUALITY,false
      ,-1);
218   }
219
220   //Adding throughoutput capacity constraints
221   vector<vector<Constraints>> PVprod(NumDesignDays,vector<Constraints
      >(NumScen)); //[MWh/day]
222   vector<vector<Constraints>> WTprod(NumDesignDays,vector<Constraints
      >(NumScen));//[MWh/day]
223   //Demand constraint, make sure imported el <= demand
224   vector<vector<Constraints>> demand(NumDesignDays,vector<Constraints
      >(NumScen));//[MWh/day]
225
226   for(int d=0;d<NumDesignDays;++d)
227   {
228     for(int s=0;s<NumScen;++s)
229     {
230       PVprod[d][s]=0;
231       WTprod[d][s]=0;
232       for(int n=0;n<NumDiscrete;++n)
233       {
234         PVprod[d][s]+=etaPV*StochParams[0][d][s]*S[0][n]*z[0][n]*5;
235         //WTprod[d][s]+=etaWT*qd*((pow(StochParams[1][d][s],3)-pow(
      Wmin,3))/(pow(Wd,3)-pow(Wmin,3)))*S[1][n]*z[1][n]*24*3600;
236         //WTprod[d][s]+=etaWT*qd*((StochParams[1][d][s]-Wmin)/(Wd-Wmin
      ))*S[1][n]*z[1][n]*24*3600;
237         if( StochParams[1][d][s]>Wmin && StochParams[1][d][s]<=Wd)
238         {
239           WTprod[d][s]+=etaWT*qd*((pow(StochParams[1][d][s],3)-pow(
      Wmin,3))/(pow(Wd,3)-pow(Wmin,3)))*S[1][n]*z[1][n]*24;
240         }
241         else if(StochParams[1][d][s]>Wd && StochParams[1][d][s]<=Wmax)
242         {
243           WTprod[d][s]+=etaWT*qd*S[1][n]*z[1][n]*24;
```

```
244         }
245         else //if(StochParams[1][d][s]<Wmin || StochParams[1][d][s]>
      Wmax)
246         {
247           WTprod[d][s]+=0;
248         }
249       }
250       PVprod[d][s]-=f_PVexport[d][s];
251       PVprod[d][s].setDependentVariable(++concount,compgraph::EQUALITY
      ,true,s+1);
252       WTprod[d][s]-=f_WTexport[d][s];
253       WTprod[d][s].setDependentVariable(++concount,compgraph::EQUALITY
      ,true,s+1);
254
255       demand[d][s]=StochParams[3][d][s]-f_PVexport[d][s]-f_WTexport[d
      ][s]-f_import[d][s];
256       demand[d][s].setDependentVariable(++concount,compgraph::LEQ,true
      ,s+1);
257     }
258   }
259
260   //
        ----------------------------------------------------------------------------------

261   //EEVP
262       if (findEEVP){
263         vector<Constraints>xEVP(techTypes);
264         xEVP[0]= S[0][10];  xEVP[1]= 0;
265         for (int i=0;i<techTypes;i++){
266           for (int j=0;j<501;j++){
267              xEVP[i] += z[i][j]*S[i][j];
268            }
269            xEVP[i].setDependentVariable(++concount,compgraph::EQUALITY
      ,false,-1);
270
271         }
272       }
273    //
        ----------------------------------------------------------------------------------

274   //Declaring objective function, one for each scenario s
275   vector<Objective> obj(NumScen);//[$]
276
277   for(int s=0;s<NumScen;++s)
278   {
279     obj[s]=0;
280     for(int i=0;i<techTypes;++i)
281     { //Adding CAPEX for technologi i
282       for(int n=0;n<NumDiscrete;++n)
283       {
284         obj[s]+=C[i][n]*z[i][n]; //[$]
285       }
286       //Adding opex pre dy
287       for(int d=0;d<NumDesignDays;++d)
288       {
289         obj[s]+=AllSeasons[d].count*f_import[d][s]*StochParams[2][d][s
      ]*30; //[$]
```

```
290         obj[s]-=AllSeasons[d].count*(f_PVexport[d][s]+f_WTexport[d][s
    ])*FiT*30; //[$]
291       }
292     }
293     obj[s].setDependentVariable(++concount,compgraph::OBJ,true,s+1);
294   }
295   return NumScen;
296 }
```

D.4 Case study 3: Stochastic problem with dynamic model and energy storage

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <sstream>
#include "definitions.hpp"
#include "CompGraph.hpp"
#include "GenerateScenarios.hpp"

#include "inputmodel.hpp"

using namespace std;

int inputmodel(vector<double> &weights)
{
  //Controls
    //bool meanvalueproblem = false;
    bool findEEVP = false; bool battery = true;
    bool importFile = true; bool printFile = true; string filepath = "
    ScenarioSet3.txt";

  //
    ------------------------------------------------------------------------------------

  //Defining Model Parameters
    int techTypes=2;          //Power generation methods: Solar and
    Wind
    int numDiscrete   = 15;       // number of discrete intervals

    //Solar power parameters
    double PV_eff    = 0.15;     // Efficiency [-]
    int solarHours    = 5;       // Solar hours per day [h]

    //Wind power parameters
    double Wmin      = 3.5;       // Minimum wind speed [m/s]
    double Wmax      = 25;        // Maximal wind speed [m/s]
    double Wd        = 13;      //
    double qd        = 8;       // Power gen per wind turbine [MW/
    turbine]
    double WT_eff    = 0.85;      // Efficiency
    int windHours    = 24;      // Windy hours per day [h]

    //Grid and end user parameters
    double FiT       = 1.45;      // Feed-in-Tariff [$/MWh]
    double elPrice    = 1.45;      // Price recieved from covering
    demand

    //Energy Storage Parameters; https://www.nrel.gov/docs/fy19osti
    /73222.pdf
    double batteryCost = 100000; // $/MWh (Optimistic estimate, 100$/
    kwh)
    //double batteryCost = 0; // $/MWh (Optimistic estimate, 100$/kwh)
    //Lifetime?
```

```
46      //Fixed and variable maintenance costs?
47      double ES_max     = 200;      // Capacity of Energy storage
48      double ES_eff_storage = 0.99;   // Representing eergy storage self
         discharge losses
49      double ES_eff_ch    = 0.99;   // Energy storage charging losses
50      double ES_eff_dis   = 0.99;   // Energy storage discharging losses
51
52      //Cost Parameters
53      vector<double> C0 = {27000,49840000}; // Cost of first
        installation [$]
54      vector<int> Smax  = {1200000,14};   // Max area PV and max number
        of wind turbines;
55      vector<int> S0    = {180,1};      // Reference value for cost
        function
56      vector<double> xi = {0.05,0.05};    // Maintenance cost factor
57      vector<double> sfi  = {0.7,0.7};    // Cost scaling factor
58      int lifetime    = 30;       // Assumed lifetime [years]
59      double r      = 0.1;        // Annuity factor
60
61    //
        ------------------------------------------------------------------------------
62    // Scenario Generation
63      int numScen;      // Number of scenarios
64      int numUncertainParams; // solar int, wind speed, cost of el-
        import, demand
65      int numDays;      // Number of timesteps/days
66      vector<vector<vector<double>>>stochParams(1);
67      /*if(meanvalueproblem)
68  //MVP   {
69        numScen=1; numUncertainParams=4;numDays=4;
70        stochParams.resize(numUncertainParams);
71        vector<vector<double>>p1      =
        {{0.001},{0.0015},{0.002},{0.0015}};  // solar
72        vector<vector<double>>p2      = {{11.25},{15},{2.5},{11.25}};
          // wind
73        vector<vector<double>>p3      = {{5},{4},{3},{4}};          //
        price
74        vector<vector<double>>p4      = {{600},{400},{400},{600}};
        // demand
75        stochParams = {p1,p2,p3,p4};    // Parameters
76        weights.push_back(1);       // Determinstic -> 1 scenario, 100%
        probability
77      }
78      else*/ if (importFile) //stochastic
79      {
80          // Initializing indices
81          int p = 0; int t = 0; int s = 0;
82          string cell; string line;
83
84          //Counters for col and row in txt file
85          int row=0; int col=0;//
86
87          //Opening file
88          ifstream file;
89          file.open(filepath);
90          if (!file.is_open()){
```

```
91                cout<<"Error opening file"<<endl;
92                return -1;
93            }else{
94              cout <<"CSV input file opened successfully"<<endl;
95
96            }
97        if (printFile){
98            cout << "Printing file.. " << endl;
99        }
100       //File operations
101       while (getline(file,line)){
102           if (printFile){
103               cout <<line<<endl;
104           }
105           if (row <= 3) {
106               if (row == 1) {
107                   stringstream ss(line);
108                   while (getline(ss, cell, '\t')) {
109                       if (col == 0) { numUncertainParams = stoi(cell
    ); }
110                       if (col == 1) { numDays = stoi(cell); }
111                       if (col == 2) { numScen = stoi(cell); }
112                       col++;
113                   }
114                   stochParams.resize(numUncertainParams);
115                   for (int p=0;p<numUncertainParams;p++)
116                   {
117                     stochParams[p].resize(numDays);
118                     for (int t=0;t<numDays;t++)
119                     {
120                       stochParams[p][t].resize(numScen);
121                     }
122                   }
123                   weights.resize(numScen);
124               }
125               row++;
126               continue;
127           }
128           p = 0; t = 0; col = 0;
129           stringstream ss(line);
130           while (getline(ss,cell,'\t')){
131               if(col!=0){
132                   stochParams[p][t][s]=stod(cell);
133                   t++;
134                   if(t%numDays==0){
135                       p++;
136                       t=0;
137                   }
138               }else{
139                   weights[s]=stod(cell);
140               }
141               col++;
142           }
143           s++;
144       }
145       file.close();
146
```

```
147     }
148
149     //Printing Values of scenarios
150     for(int s=0; s<numScen;++s)
151     {
152         cout<<"Scenario: "<<s+1 <<" \t probability:"<< weights[s]<<
        endl;
153       for(int t=0;t<numDays;t++)
154       {
155         cout <<"Time step: "<< t+1<<"\t";
156         cout<<stochParams[0][t][s]<<"\t "<<stochParams[1][t][s]<<"\t "
        <<stochParams[2][t][s]<< "\t "<<stochParams[3][t][s]<<endl;
157       }
158     }
159     cout <<"
        ------------------------------------------------------------------------
        "<<endl;
160
161   //
        ------------------------------------------------------------------------------------
162   // Creation of a discrete set of variables for 1st stage decision
        variables
163
164
165     //Power Generation
166     vector<vector<double>> z_d(techTypes,vector<double>(numDiscrete));
         //[m  ] and [-] units of wind turbines
167     cout<<"The discrete size values are:"<<'\n';
168     for(int i=0;i<techTypes;++i)
169     {
170       cout<<"Technology "<<i+1<<'\n';
171       for(int n=0;n<numDiscrete;++n)
172       {
173         z_d[i][n] = (n*Smax[i])/(numDiscrete-1); //S_lower=0 and
        ommited here
174         cout<<z_d[i][n] << '\n';          //printing to check values
175       }
176     }
177
178     //Energy Storage
179     vector<double>ES_d(numDiscrete);
180     if(battery){
181       cout <<"Battery"<<endl;
182       for (int n=0;n<numDiscrete;n++){
183         ES_d[n]   = (n*ES_max)/(numDiscrete-1);
184         cout<< ES_d[n]<<endl;
185       }
186     }
187     //Capital cost related to the discrete sets of variables.
188     //Power generation
189     vector<vector<double>> C(techTypes,vector<double>(numDiscrete));//
        [$]
190       for(int i=0;i<techTypes;++i)
191       { cout<<"Capital cost of technology " << i <<'\n';
192         for(int n=0;n<numDiscrete;++n)
193         {
```

```
194         C[i][n]    =    (1+xi[i])*C0[i]*pow(z_d[i][n]/S0[i],sfi[i]);//
    r*pow(1+r,LifeTime)/(pow(1+r,LifeTime)-1);
195         cout<<C[i][n]<<'\n';                    //printing to check
    values
196       }
197     }
198
199     //Energy storage
200     vector<double>C_ES(numDiscrete);
201     if(battery){
202       cout << "Capital Cost of Battery "<<endl;
203       for (int n=0;n<numDiscrete;n++){
204         C_ES[n]    = batteryCost*pow(ES_d[n]/1,0.6);
205         cout <<C_ES[n]<<endl;
206       }
207     }
208     cout <<"
    ------------------------------------------------------------------------
    "<<endl;
209
210   //
    ------------------------------------------------------------------------------------
211   // 1st Stage Variables; continuous variables as a discrete set using
       binary variables
212     //Initializng Variables
213     int varcount  = -1;
214     int concount  = -1;
215     char clabel[30];
216
217     // Power generation
218     vector<vector<Variables>> z_b(techTypes,vector<Variables>(
    numDiscrete));
219     for(int i=0;i<techTypes;++i)
220     {
221       for(int n=0;n<numDiscrete;++n)
222       {
223         sprintf(clabel,"z_b[%d][%d]", i+1,n+1);
224             z_b[i][n].setIndependentVariable(++varcount,compgraph::
    BINARY,I(0,1),0.,-1, clabel);
225         }
226     }
227
228     //Energy Storage
229     vector<Variables>ES_b(numDiscrete);
230     if(battery){
231       for (int n=0;n<numDiscrete;n++){
232         sprintf(clabel, "ES_b[%d]",n+1);
233         ES_b[n].setIndependentVariable(++varcount,compgraph::BINARY,I
    (0,1),0.,-1,clabel);
234       }
235     }
236
237
238   //
    ------------------------------------------------------------------------------------
```

```
239  // 2nd Stage Variables: Variables needed for energy balances
240    // Power generation flows
241    vector<vector<Variables>> f_PV(numDays,vector<Variables>(numScen))
       ;     //[MWh/day]
242    vector<vector<Variables>> f_WT(numDays,vector<Variables>(numScen))
       ;     //[MWh/day]
243    // Export and import flows
244    vector<vector<Variables>> f_surplus(numDays,vector<Variables>(
       numScen));    //[MWh/day]
245    vector<vector<Variables>> f_deficit(numDays,vector<Variables>(
       numScen));  //[MWh/day]
246    // Energy storage: Mean daily charge and discharge flows
247    vector<vector<Variables>>f_charge(numDays,vector<Variables>(
       numScen));    //[MWh/day]
248    vector<vector<Variables>>f_discharge(numDays,vector<Variables>(
       numScen)); //[MWh/day]
249    // Energy storage: State of charge
250    vector<vector<Variables>>SoC(numDays,vector<Variables>(numScen));
           //[MWh]
251
252    for(int t=0;t<numDays;++t)
253    {
254      for(int s=0;s<numScen;++s)
255      {
256        sprintf(clabel,"f_PV[%d][%d]", t+1, s+1);
257        f_PV[t][s].setIndependentVariable(++varcount,compgraph::
       CONTINUOUS,I(0,10000),0.,s+1,clabel);
258        sprintf(clabel,"f_WT[%d][%d]", t+1, s+1);
259        f_WT[t][s].setIndependentVariable(++varcount,compgraph::
       CONTINUOUS,I(0,10000),0.,s+1,clabel);
260        sprintf(clabel,"f_surplus[%d][%d]", t+1, s+1);
261        f_surplus[t][s].setIndependentVariable(++varcount,compgraph::
       CONTINUOUS,I(0,10000),0.,s+1,clabel);
262        sprintf(clabel,"f_deficit[%d][%d]", t+1, s+1);
263        f_deficit[t][s].setIndependentVariable(++varcount,compgraph::
       CONTINUOUS,I(0,10000),0.,s+1,clabel);
264        if(battery){
265          sprintf(clabel,"f_charge[%d][%d]", t+1, s+1);
266          f_charge[t][s].setIndependentVariable(++varcount,compgraph::
       CONTINUOUS,I(0,10000),0.,s+1,clabel);
267          sprintf(clabel,"f_discharge[%d][%d]", t+1, s+1);
268          f_discharge[t][s].setIndependentVariable(++varcount,
       compgraph::CONTINUOUS,I(0,10000),0.,s+1,clabel);
269          sprintf(clabel,"State of charge[%d][%d]", t+1, s+1);
270          SoC[t][s].setIndependentVariable(++varcount,compgraph::
       CONTINUOUS,I(0,10000),0.,s+1,clabel);
271        }
272      }
273    }
274
275  //
       ------------------------------------------------------------------------------------------
276  // Constraints necessary for binary variables: Only one positive per
       set of discrete variables
277
278    // Power Generation
```

```
279     vector<Constraints> zLim(techTypes);
280
281     for(int i=0;i<techTypes;++i)
282     {
283       zLim[i]=-1;
284       for(int n=0;n<numDiscrete;++n)
285       {
286         zLim[i]   += z_b[i][n];
287       }
288       zLim[i].setDependentVariable(++concount,compgraph::EQUALITY,
    false,-1);
289     }
290
291     // Energy Storage
292     Constraints ES_lim;
293     if(battery)
294     {
295       ES_lim = -1;
296       for (int n=0;n<numDiscrete;n++){
297         ES_lim += ES_b[n];
298       }
299       ES_lim.setDependentVariable(++concount,compgraph::EQUALITY,false
    ,-1);
300     }
301   //
      ------------------------------------------------------------------------------

302   //EEVP
303       if (findEEVP){
304         vector<Constraints>xEVP(techTypes);
305         xEVP[0]= z_d[0][5];  xEVP[1]= 0;
306         for (int i=0;i<techTypes;i++){
307           for (int j=0;j<501;j++){
308              xEVP[i] += z_b[i][j]*z_d[i][j];
309            }
310            xEVP[i].setDependentVariable(++concount,compgraph::EQUALITY
    ,false,-1);
311
312         }
313       }
314   //
      ------------------------------------------------------------------------------

315   // Model Constraints
316     //Constraints related to installation capacities
317       // Power generation by PV
318       vector<vector<Constraints>> PVprod(numDays,vector<Constraints>(
    numScen)); //[MWh/day]
319
320       for(int t=0;t<numDays;t++)
321       {
322         for(int s=0;s<numScen;++s)
323         {
324           PVprod[t][s]=0;
325           for(int n=0;n<numDiscrete;++n)
326           {
327             PVprod[t][s]   += PV_eff*stochParams[0][t][s]*z_d[0][n]*z_b
```

```
[0][n]*solarHours;
328         }
329         PVprod[t][s]  -= f_PV[t][s];
330         PVprod[t][s].setDependentVariable(++concount,compgraph::
    EQUALITY,true,s+1);
331       }
332     }
333
334     // Power generation by Wind
335     vector<vector<Constraints>> WTprod(numDays,vector<Constraints>(
    numScen)); //[MWh/day]
336
337     for(int t=0;t<numDays;t++)
338     {
339       for(int s=0;s<numScen;++s)
340       {
341         WTprod[t][s]=0;
342         for(int n=0;n<numDiscrete;++n)
343         {
344           if( stochParams[1][t][s]>Wmin && stochParams[1][t][s]<Wd)
345           {
346             WTprod[t][s]  += WT_eff*qd*(pow(stochParams[1][t][s],3)-
    pow(Wmin,3))/(pow(Wd,3)-pow(Wmin,3))*z_d[1][n]*z_b[1][n]*windHours
    ;
347           }
348           if(stochParams[1][t][s]>=Wd && stochParams[1][t][s]<=Wmax)
349           {
350             WTprod[t][s]  += qd*WT_eff*z_d[1][n]*z_b[1][n]*windHours
    ;
351           }
352           else //if(stochParams[1][t][s]<Wmin || stochParams[1][t][s
    ]>Wmax)
353           {
354             WTprod[t][s]  += 0;
355           }
356         }
357
358         WTprod[t][s]  -= f_WT[t][s];
359         WTprod[t][s].setDependentVariable(++concount,compgraph::
    EQUALITY,true,s+1);
360       }
361     }
362
363     // Battery State of Charge, charging and discharging
364     vector<vector<Constraints>> ES_SoC(numDays,vector<Constraints>(
    numScen));
365       vector<vector<Constraints>> ES_charging(numDays,vector<
    Constraints>(numScen));
366       vector<vector<Constraints>> ES_discharging(numDays,vector<
    Constraints>(numScen));
367       vector<vector<Constraints>> ES_charging2(numDays,vector<
    Constraints>(numScen));
368       vector<vector<Constraints>> ES_discharging2(numDays,vector<
    Constraints>(numScen));
369
370     if(battery){
371       for(int t=0;t<numDays;t++)
```

```
372         {
373           for(int s=0;s<numScen;s++)
374           {
375             ES_SoC[t][s]      = 0;
376             ES_charging[t][s]   = 0;
377             ES_discharging[t][s]  = 0;
378             ES_charging2[t][s]    = 0;
379             ES_discharging2[t][s] = 0;
380             for(int n=0;n<numDiscrete;n++){
381               ES_SoC[t][s]      -= ES_b[n]*ES_d[n];
382               ES_charging[t][s]   -= ES_b[n]*ES_d[n];
383             }
384             if(t!=0){
385               ES_charging[t][s]   += SoC[t-1][s];
386               ES_discharging[t][s]  -= SoC[t-1][s];
387             }
388             ES_SoC[t][s]      += SoC[t][s];
389             ES_charging[t][s]   += f_charge[t][s]*ES_eff_ch;
390             ES_discharging[t][s]  += f_discharge[t][s]/ES_eff_dis;
391             ES_charging2[t][s]    -= f_surplus[t][s];
392             ES_discharging2[t][s] -= f_deficit[t][s];
393             ES_charging2[t][s]    += f_charge[t][s];
394             ES_discharging2[t][s] += f_discharge[t][s];
395
396             ES_SoC[t][s].setDependentVariable(++concount,compgraph::
    LEQ,true,s+1);
397             ES_charging[t][s].setDependentVariable(++concount,
    compgraph::LEQ,true,s+1);
398             ES_discharging[t][s].setDependentVariable(++concount,
    compgraph::LEQ,true,s+1);
399             ES_charging2[t][s].setDependentVariable(++concount,
    compgraph::LEQ,true,s+1);
400             ES_discharging2[t][s].setDependentVariable(++concount,
    compgraph::LEQ,true,s+1);
401           }
402         }
403       }
404
405     //Energy balance to satisfy demand
406     vector<vector<Constraints>> energyBalance(numDays,vector<
    Constraints>(numScen));  //[MWh/day]
407
408     for(int t=0;t<numDays;++t)
409     {
410       for(int s=0;s<numScen;++s)
411       {
412         energyBalance[t][s]    = 0;
413         energyBalance[t][s]    += stochParams[3][t][s];
414         energyBalance[t][s]    -= f_PV[t][s];
415         energyBalance[t][s]    -= f_WT[t][s];
416         energyBalance[t][s]    -= f_deficit[t][s];
417         energyBalance[t][s]    += f_surplus[t][s];
418         energyBalance[t][s].setDependentVariable(++concount,compgraph
    ::EQUALITY,true,s+1);
419       }
420     }
421
```

```
422     //Energy balance for Battery
423     vector<vector<Constraints>> energyBalance_Battery(numDays,vector<
        Constraints>(numScen));
424
425     if(battery){
426       for(int t=0;t<numDays;++t)
427       {
428         for(int s=0;s<numScen;++s)
429         {
430           energyBalance_Battery[t][s]   = 0;
431           energyBalance_Battery[t][s]   += SoC[t][s];
432           if(t!=0)
433           {
434             energyBalance_Battery[t][s]   -= SoC[t-1][s]*
        ES_eff_storage;
435           }
436           energyBalance_Battery[t][s]   -= f_charge[t][s]*ES_eff_ch;
437           energyBalance_Battery[t][s]   += f_discharge[t][s]*(1/
        ES_eff_dis);
438           energyBalance_Battery[t][s].setDependentVariable(++concount,
        compgraph::EQUALITY,true,s+1);
439         }
440       }
441     }
442
443   //
        ------------------------------------------------------------------------------------

444   // Objective function
445     vector<Objective> obj(numScen);//[$]
446
447     for(int s=0;s<numScen;++s)
448     {
449       obj[s]=0;
450       for(int i=0;i<techTypes;++i)
451       {
452         for(int n=0;n<numDiscrete;++n)
453         {
454           obj[s]    += C[i][n]*z_b[i][n]; // CapEx[$]
455         }
456       }
457       if(battery){
458         for (int n=0;n<numDiscrete;n++)
459         {
460           obj[s]    += C_ES[n]*ES_b[n];
461         }
462       }
463       for(int t=0;t<numDays;++t)
464       {
465         obj[s]-=(stochParams[3][t][s])*elPrice*52*30;             //
        OpEx: Earnings for covering demand[$]
466         if(battery){
467           obj[s]+=(f_deficit[t][s]-f_discharge[t][s])*stochParams[2][t
        ][s]*52*30;   // OpEx: Cost for energy deficit[$]
468           obj[s]-=(f_surplus[t][s]-f_charge[t][s])*elPrice*52*30;   //
         OpEx: Earnings for energy surplus[$]
469         }else{
```

```
470             obj[s]+=f_deficit[t][s]*stochParams[2][t][s]*52*30;   //
    OpEx: Cost for energy deficit[$]
471             obj[s]-=f_surplus[t][s]*elPrice*52*30;
472           }
473           } //f_import=f_deficit-F_discharge    f_export =f_surplus-
    f_charge
474         obj[s].setDependentVariable(++concount,compgraph::OBJ,true,s+1);
475       }
476
477   return numScen;
478
479 }
```